TECHNISCHE UNIVERSITÄT DRESDEN

Future-Proof Software-Systems (FPSS)

Part 1

Lecture WS 2019/20: Prof. Dr. Frank J. Furrer

Version 1.0/10.10.2019

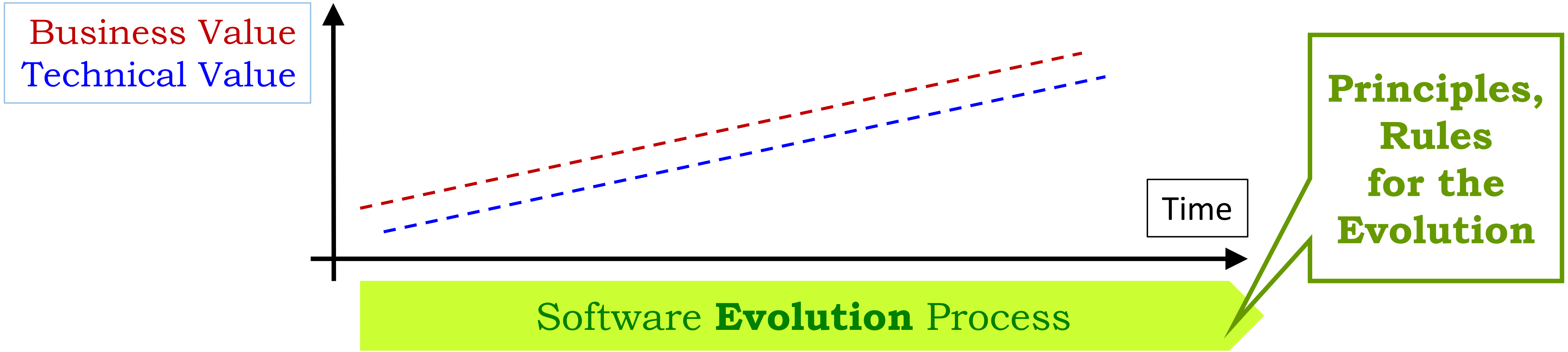
Lecture Navigation

Preliminary Definition

https://hcldr.files.wordpress.com

Future-Proof Software-Systems =

Software-Systems which are built and evolved in such a way, that their **business value** and **technical value** increases over time



Lecture Structure:

Part 1: Introduction

Part 2: Managed Evolution Strategy for Software Systems

Part 3: Architecting for Changeability

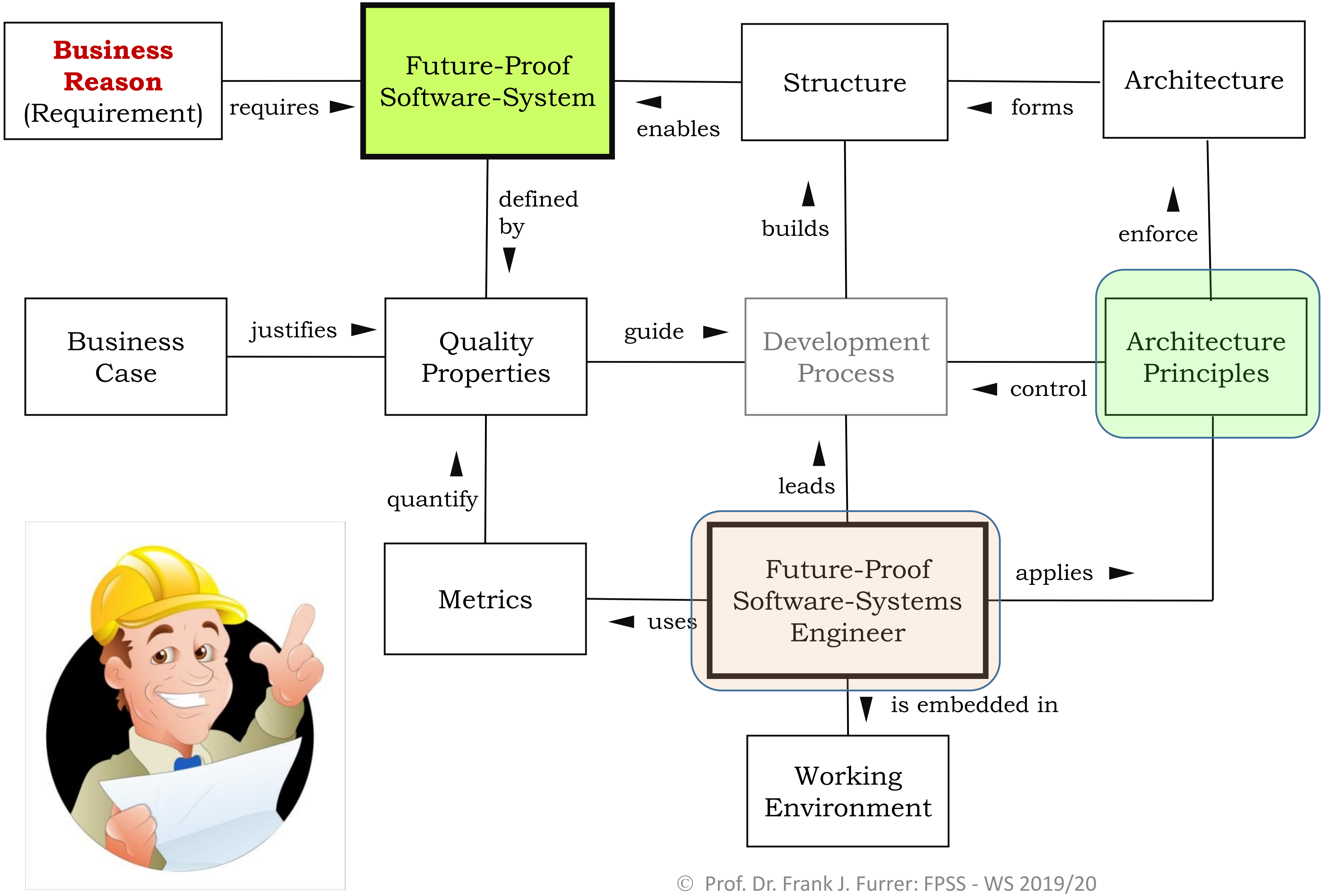
Part 4: Architecting for Dependability

Part 5: Skills of a Future-Proof Software Engineer

Slides + Additional Information:

http://st.inf.tu-dresden.de/teaching/fps

Future-Proof Software-Systems [Part 1]



http://images.all-free-download.com

Conceptual Context: (Lecture Map)

Future-Proof Software-Systems [Part 1]

Our journey:

«Software everywhere» → Systems & Software Engineering → Three devils of Systems Engineering → Future-Proof Software-Systems → Technial Debt Architecture Erosion → Managed Evolution Strategy → Architecture → Architecting for Changeability → Special Topics → Architecting for Dependability → The Future-Proof Software-Systems Engineer

**“Tell me and I forget,
teach me and I may remember,
involve me and I learn.”**

-Benjamin Franklin

http://blog.westinteractive.com

... please be interactive !

Contents Part 1:

https://www.techwell.com

www.123rf.com

«Software Everywhere»

2020:
It is difficult to find a product or service which is not controlled by **software**

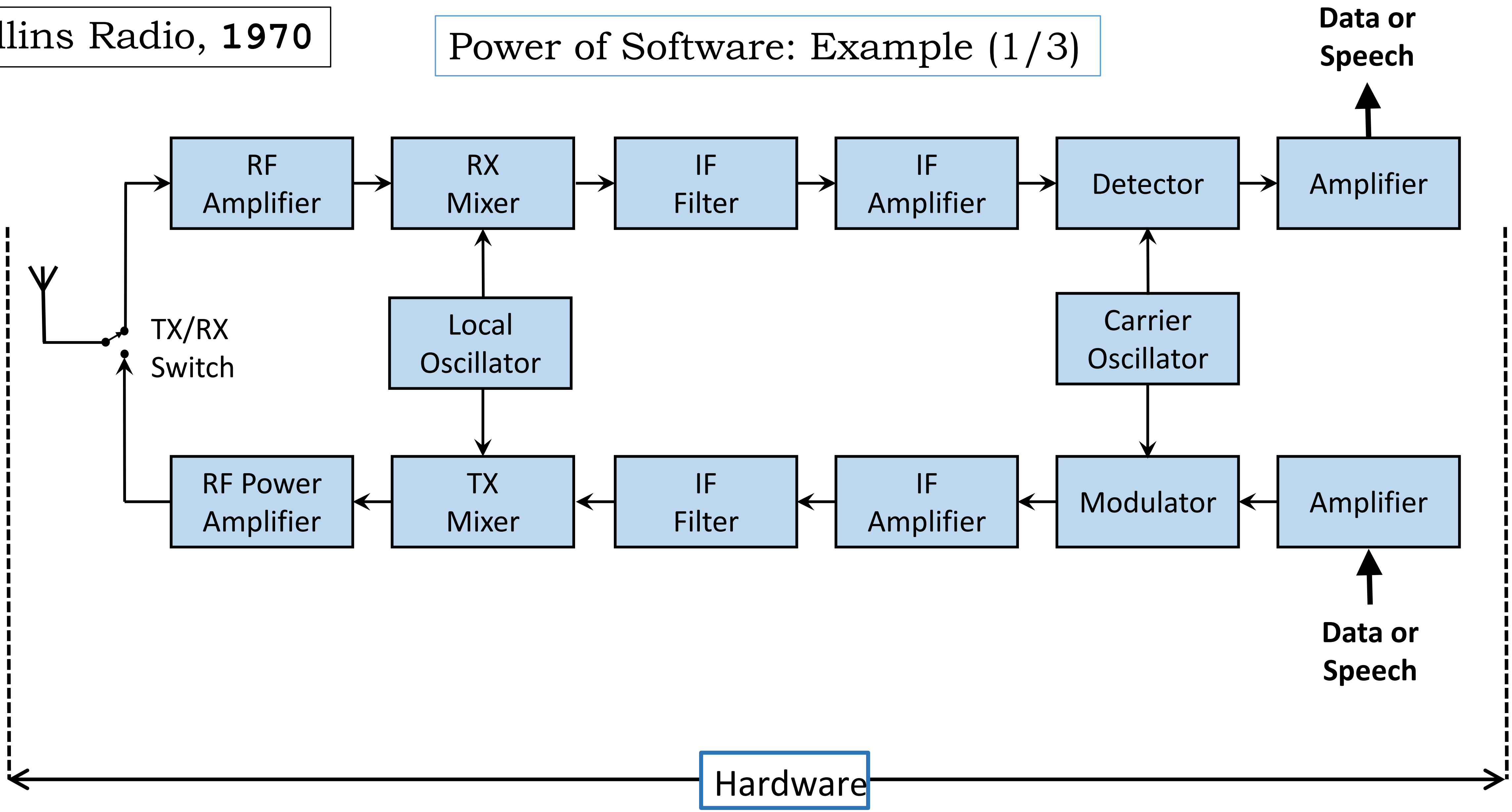
Future:
The software is **conquering** more and more of our life and work

Example:
COLLINS KWM-2
Short Wave Transceiver

https://rigreference.com

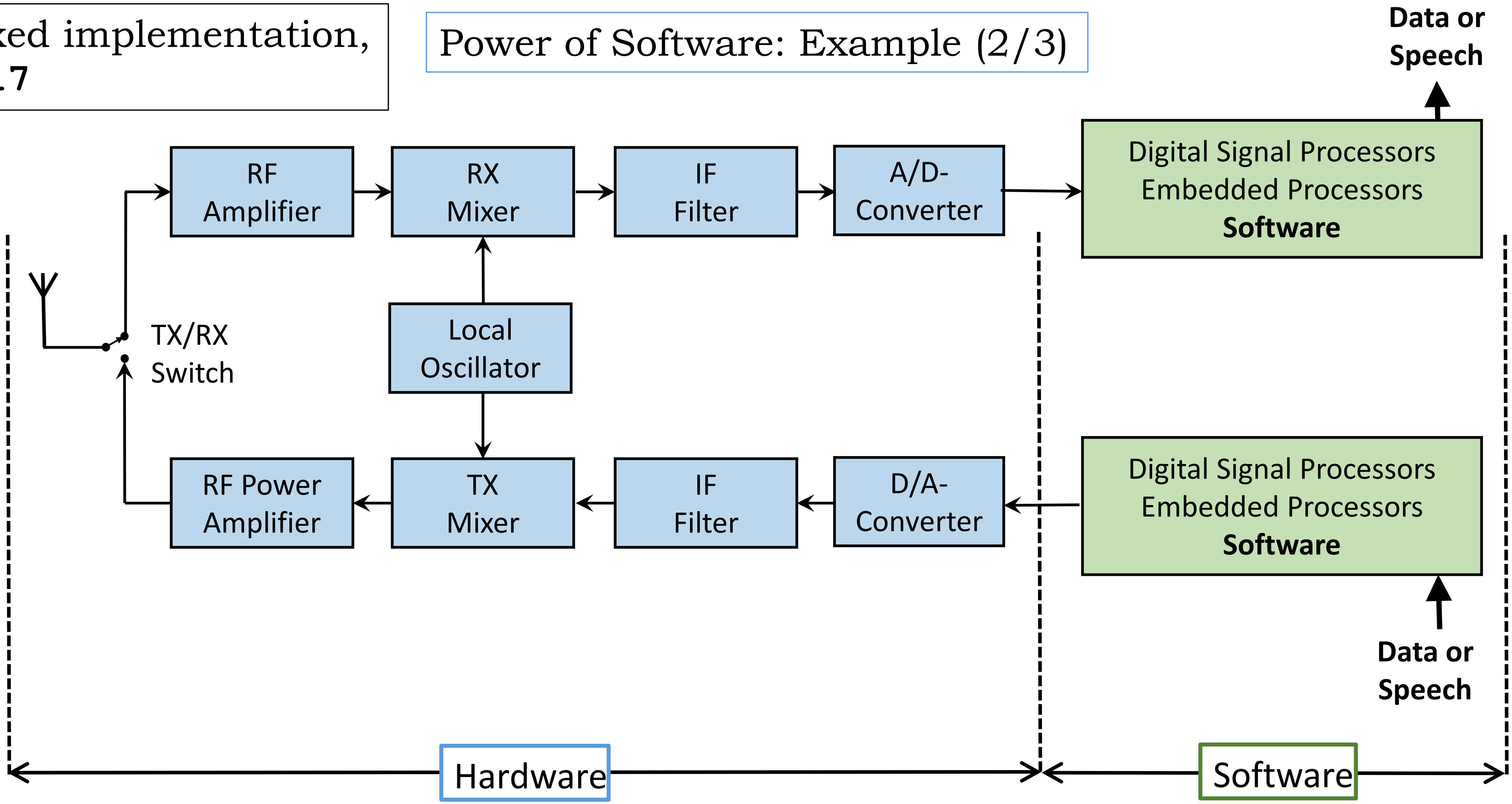
Collins Radio, **1970**

Power of Software: Example (1/3)

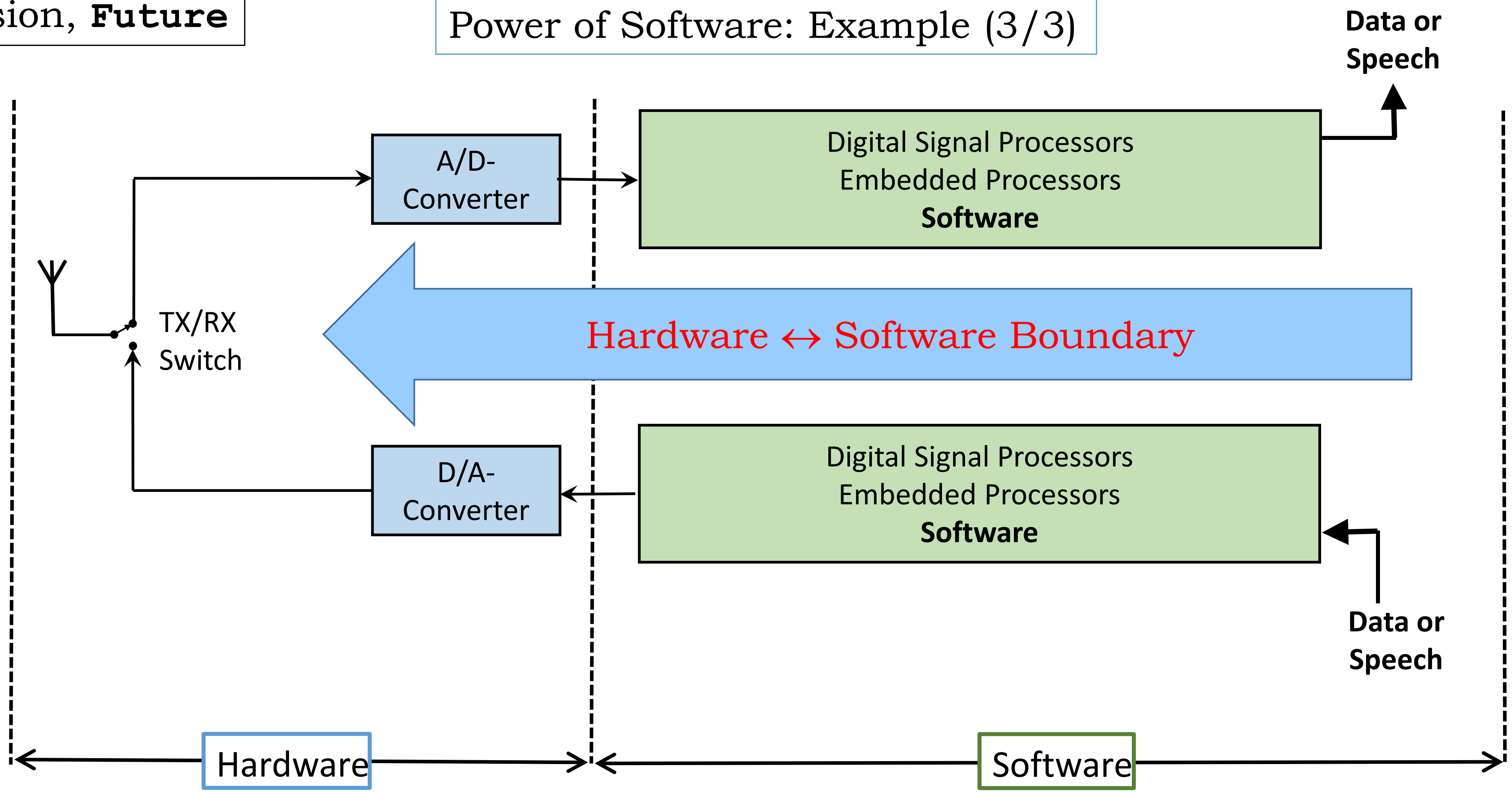


Mixed implementation, 2017

Power of Software: Example (2/3)



Power of Software: Example (3/3)



«Software Everywhere»

Enterprise Software

https://guernseycheshirehome.org

https://www.techbook.de

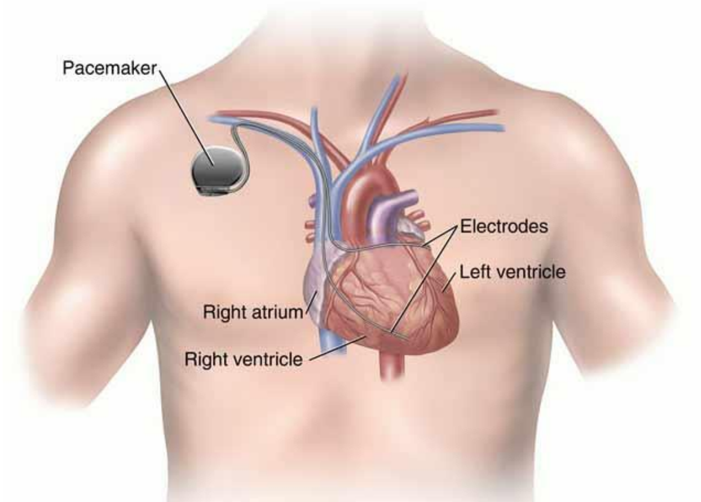
https://www.kimley-horn.com

https://www.produktion.de

https://today.uconn.edu

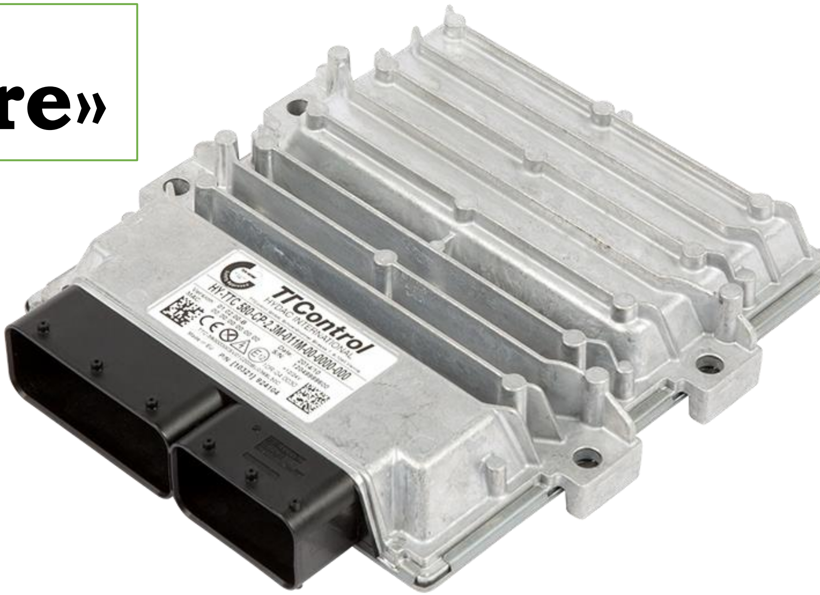
«Software Everywhere»

Embedded Software



http://www.heartrx.com

https://www.networkworld.com



https://www.hydac.com

https://www.cnet.com

https://www.which.co.uk

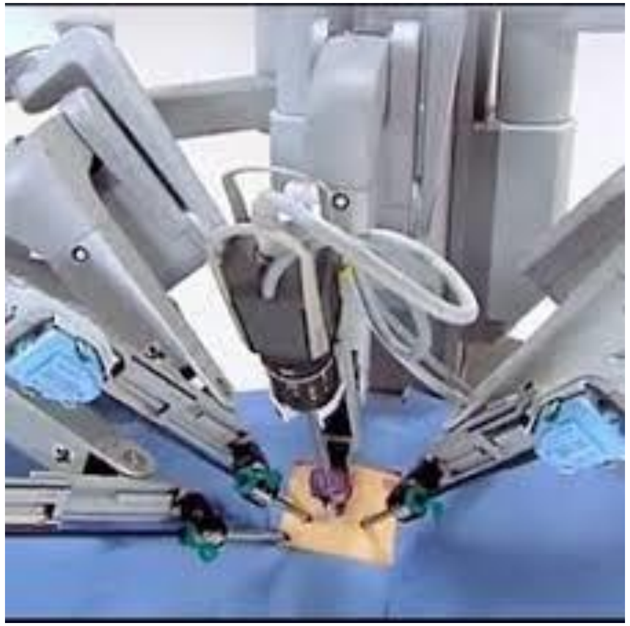
«Software Everywhere»

Cyber-Physical Systems (CPS)

https://www.indiamart.com

https://www.instarmac.co.uk

http://www.marina-fallenbach.ch



https://amazingherald.com

http://hotelmagazine.co.nz

http://footage.framepool.com

1.1

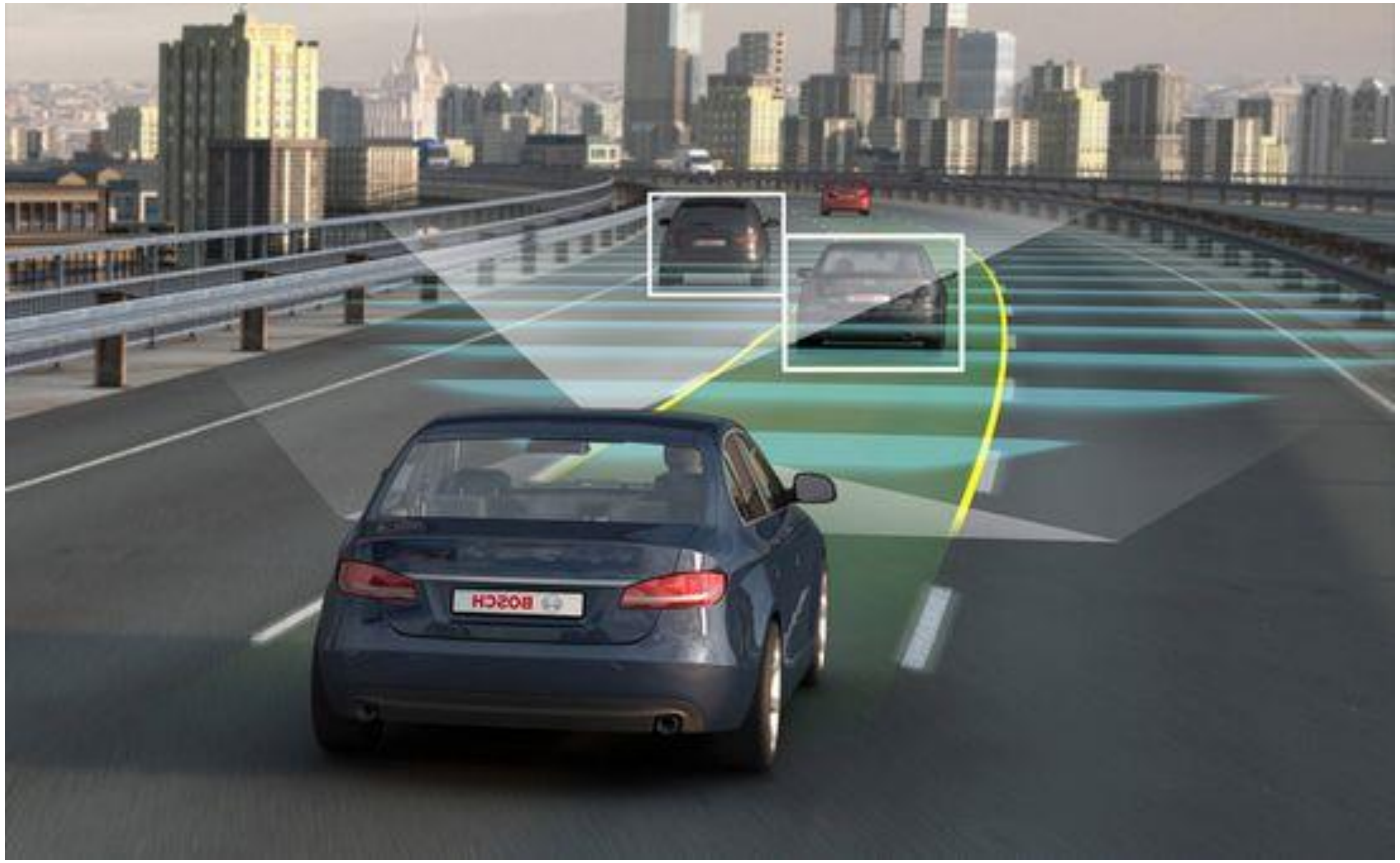
«Software Everywhere»

- Success Stories
- Failure Stories

… Some recent examples

https://newsnetwork.mayoclinic.org

Software Success Story Example 1: **Autonomous Driving**



http://www.zomazi.com

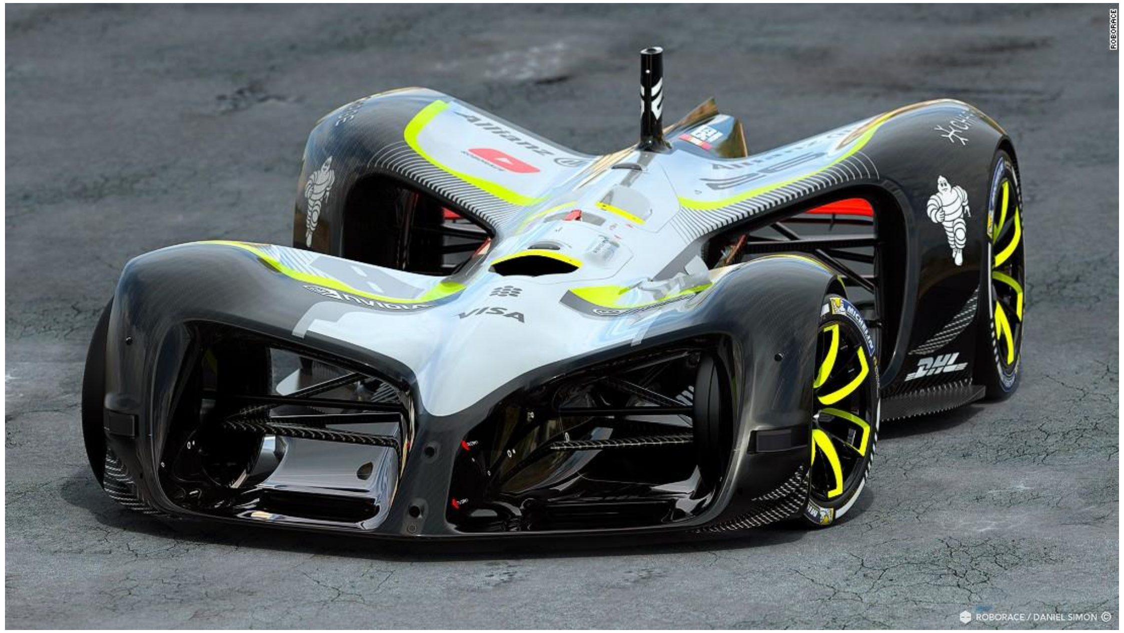
In the near future, road traffic will be autonomous. Even complex situations will be handled without human intervention

http://kozmedia.com

Cars will be driven ***purely by software***, based on information from sensors and from the environment

Software Success Story Example **1**: **Autonomous Driving**

http://i2.cdn.cnn.com



Roborace – Global Championship of Intelligence and Technology

http://roborace.com

The Robocars have top speeds of more than **300 km/h** (190 mph). The series will be held on the same tracks the FIA Formula E Championship uses. It will be the first global championship for driverless (autonomous) cars

https://imgr2.auto-motor-und-sport.de

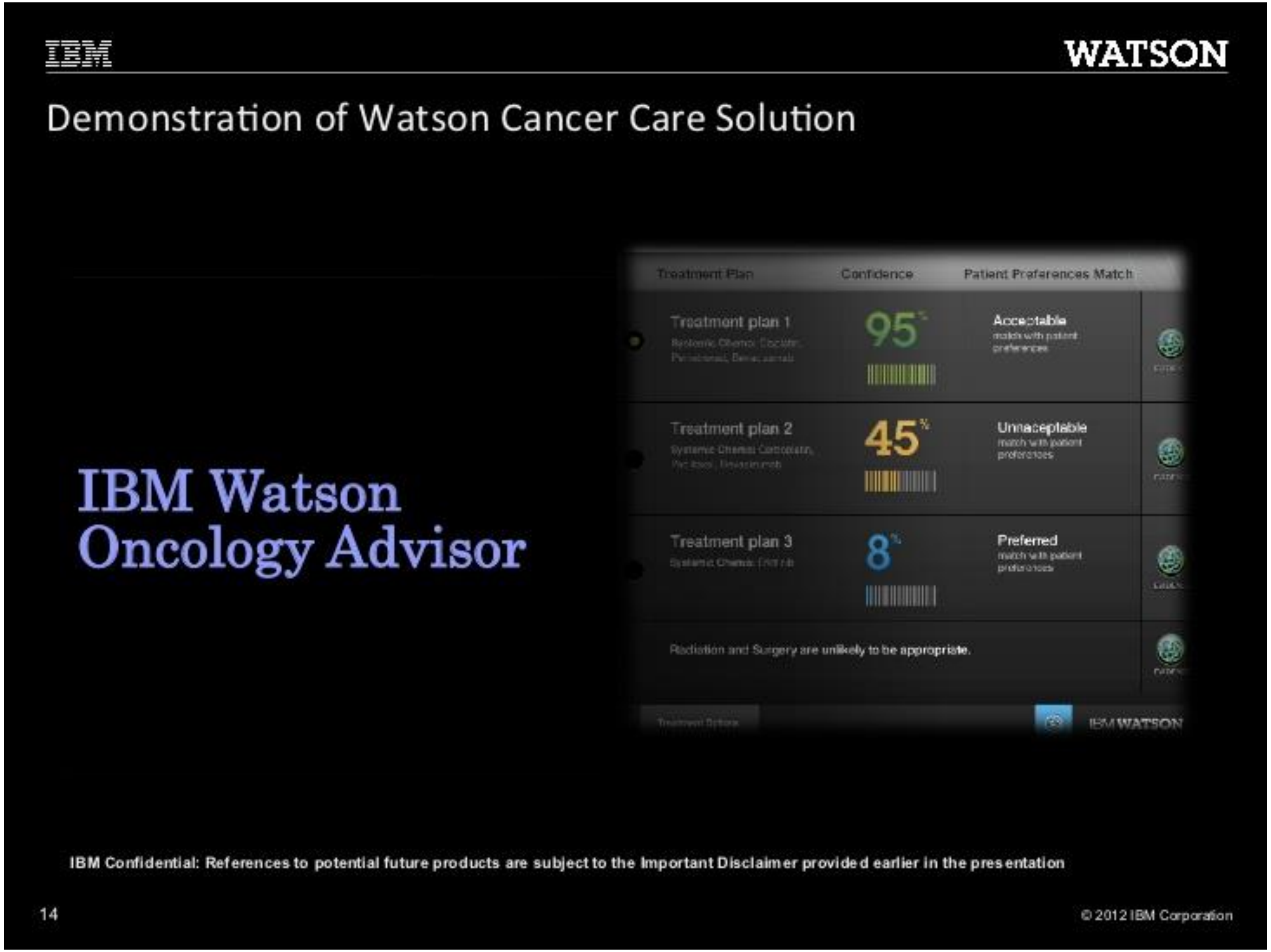
10 teams will have *equal* cars (2 each), but will have to develop their own real-time computing algorithms and artificial intelligence technologies

Software Success Story Example **2**: 2016 European Truck Platooning Challenge

http://www.itsinternational.com

Truck platooning, where trucks travel in convoy very close to each other, provides many benefits. The first truck does the driving while the ones following are connected by a wireless electronic communications system, like the carriages of a train
[https://www.eutruckplatooning.com/default.aspx]

Software Success Story Example **5:** **Medical Oncological Advisor “Watson”**



http://de.slideshare.net

Software expert system for diagnosis and therapy of **cancer**

Knows and «understands» the complete oncological knowledge

Software Success Story Example **5:** Medical Oncological Advisor "Watson"

Did AI just save its first life?

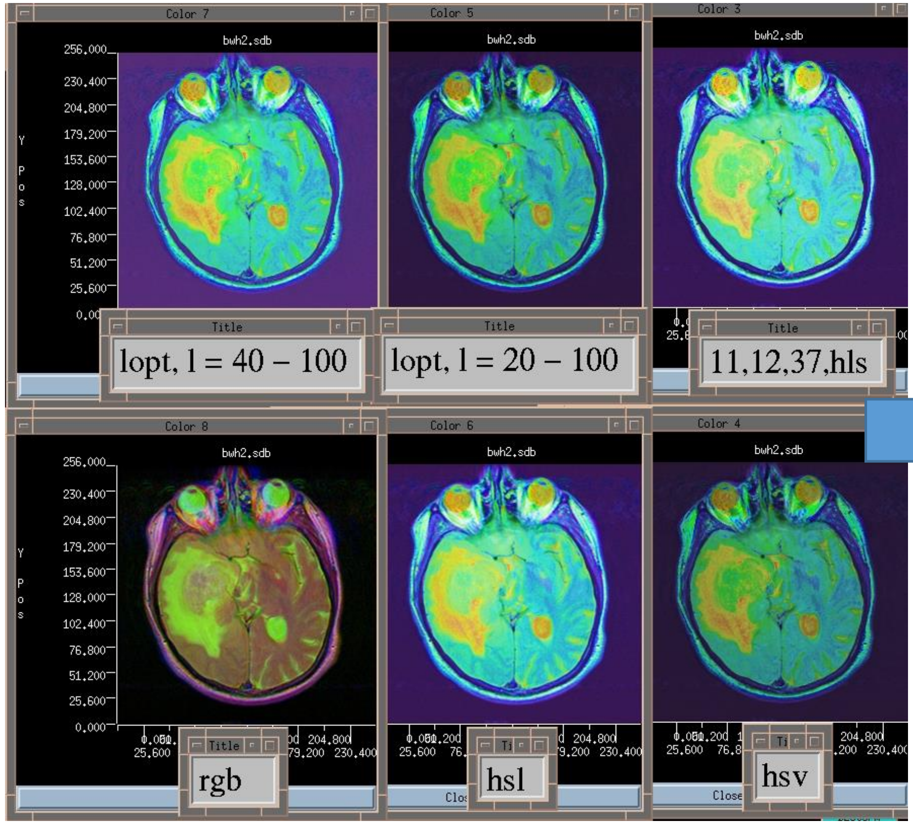
The University of Tokyo is reporting that Watson has correctly diagnosed a **rare form of leukemia** in a 60-year-old Japanese woman. Doctors turned to Watson after the patient failed to respond to drugs they were administering. After 10 minutes of crunching the data, Watson correctly diagnosed her ailment and recommended the appropriate treatment

https://www.top500.org

https://www.ibm.com

Watson Oncology:
Expert system + Doctor interaction = powerful diagnosis & treatment tool

Software Success Story Example **6:** **Early Tumor Detection**



https://www.veterinarypracticenews.com

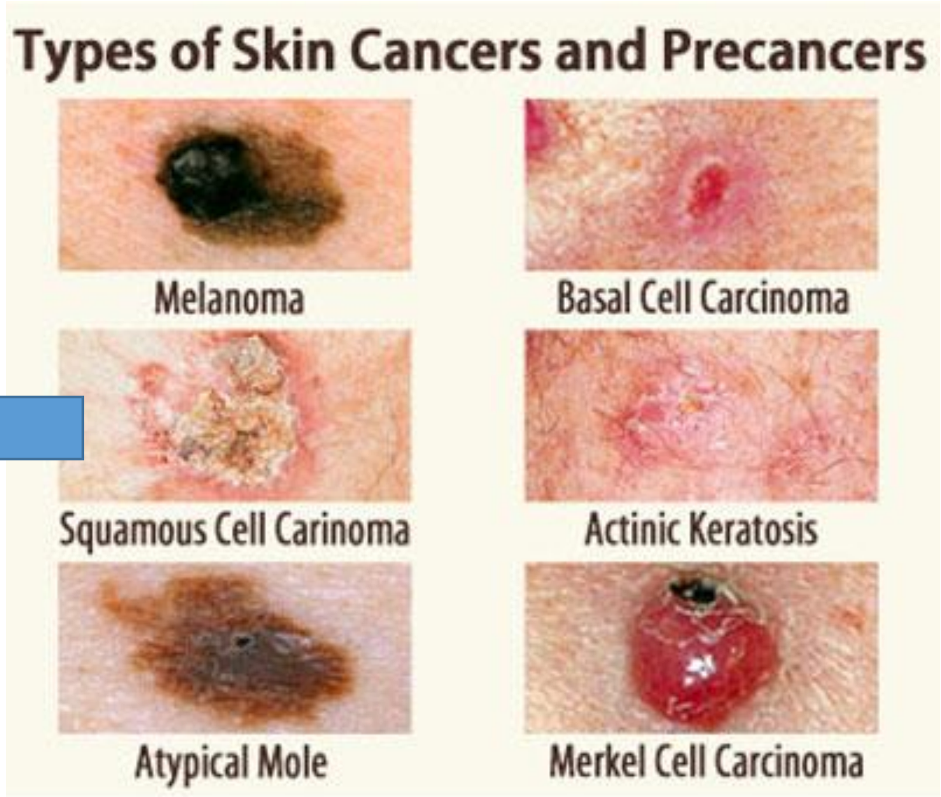
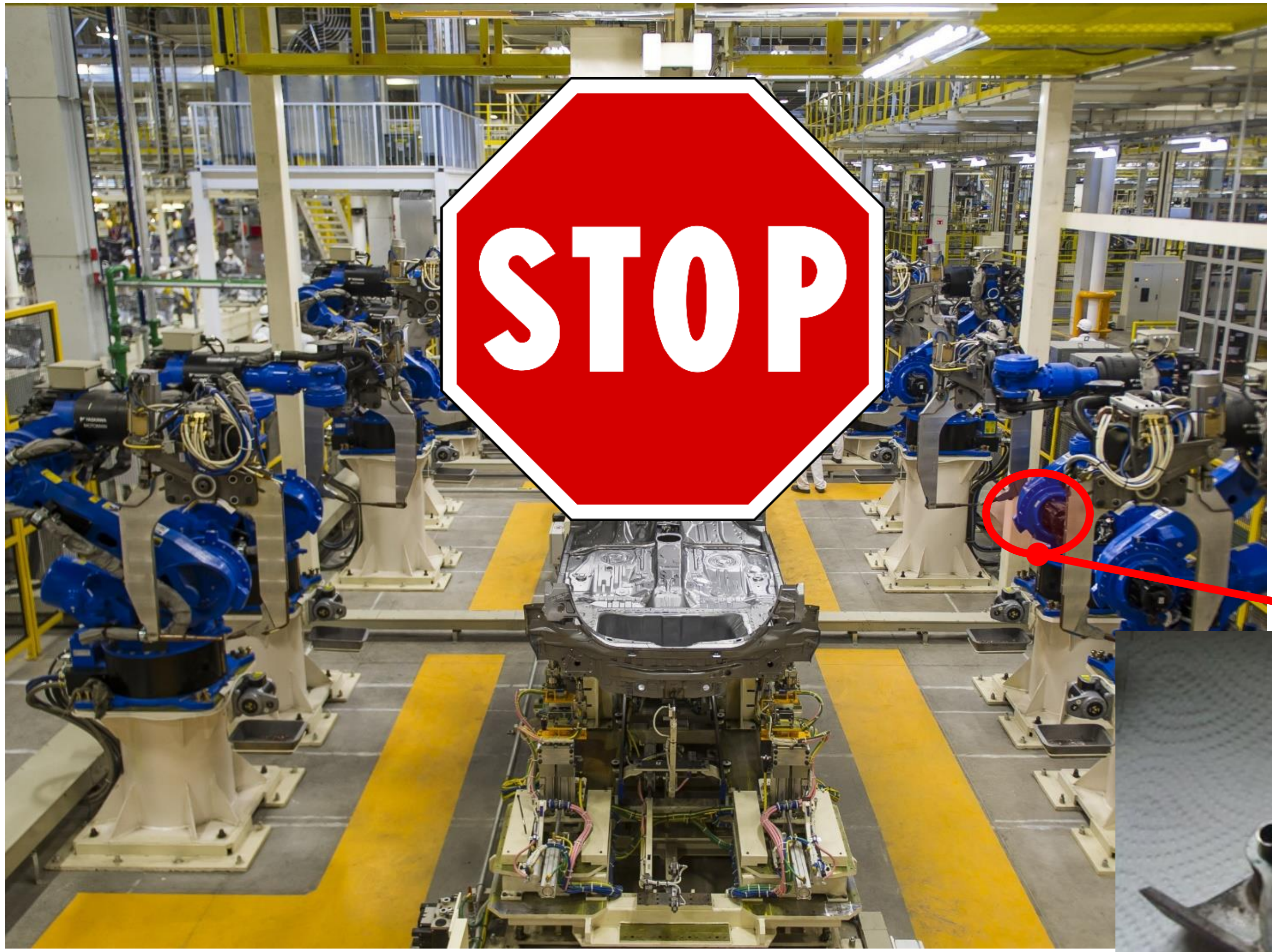


Image processing systems for early detection and precise diagnosis
for various forms of **cancer**
Self-learning, massie improement

Software Success Story Example **7**: **Preventive maintenance**



https://detektor.fm

Today's manufacturing:

- Highly automated
- Complex assembly lines

- 10'000's of devices

- Failure of one device may stop assembly line

How to prevent?

Software Success Story Example **7**: **Preventive maintenance**

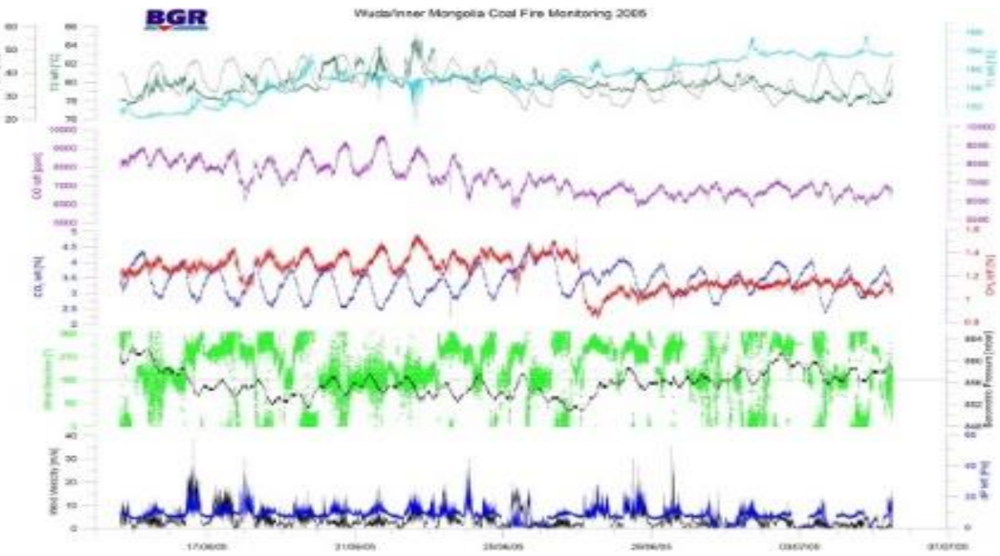
Mechanic & Thermal

Vibrations & Noise

Times & Intervals

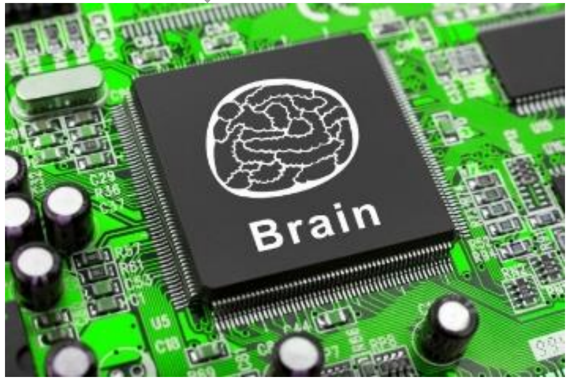
Geometry

Material constants



Millions of values per Minute

https://www.bgr.bund.de



Machine-learning based analysis

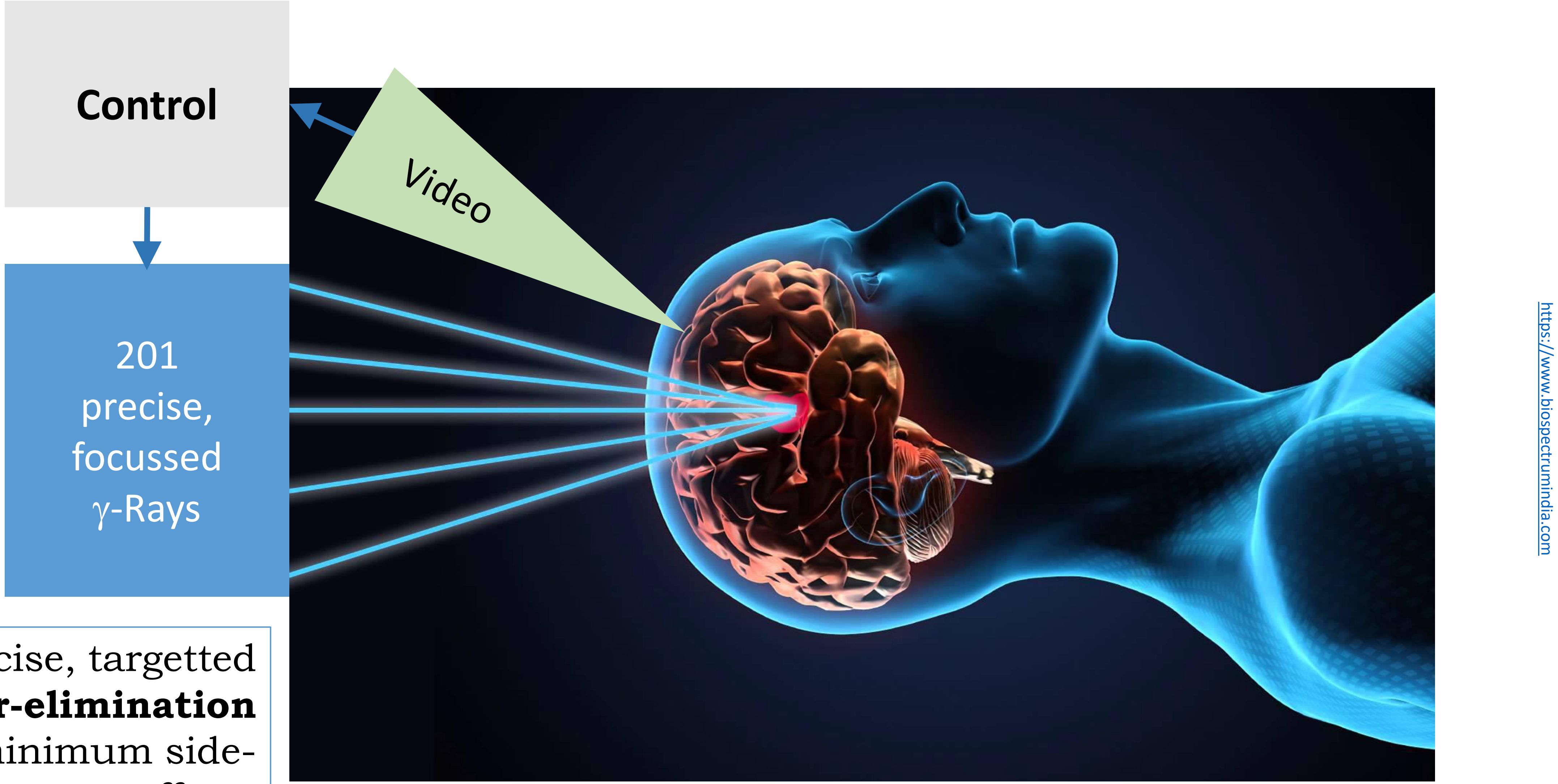
4.6.18
Failure
D394c
[C = 0.84]

Failure Prediction

http://www.indiantradebird.com

http://diysolarpane

Software Success Story Example **8**: **Gamma Knife Surgery**



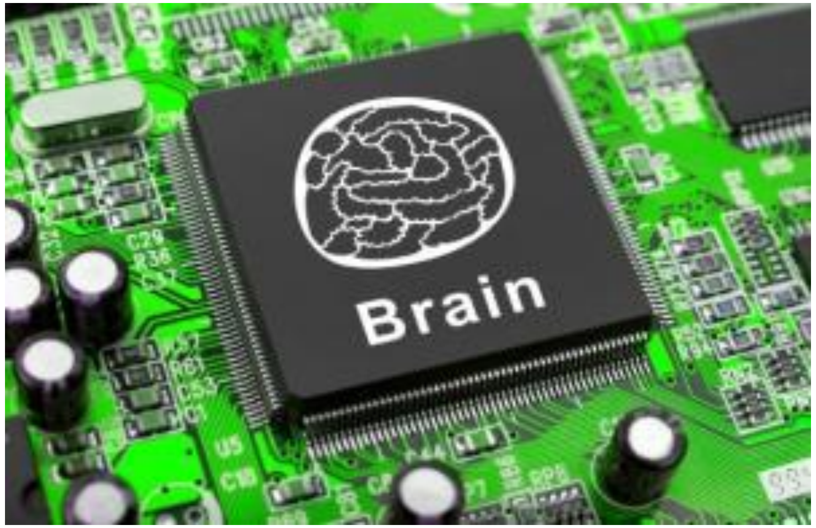
Precise, targetted **tumor-elimination** with minimum side-effects

Software Success Story Example **9**: **Autonomous PREDATOR Drone**

Warbots

http://www.defenseworld.net

Software Success Story Examples: Artificial Intelligence/Machine Learning



https://fst.net.au

The software with the most impact: Artificial Intelligence (AI) and Machine Learning (ML)

https://www.antennevorarlberg.at

Testing ground for AI/ML: Games

Software Success Story Example **10**: **Artificial Intelligence/Machine Learning**

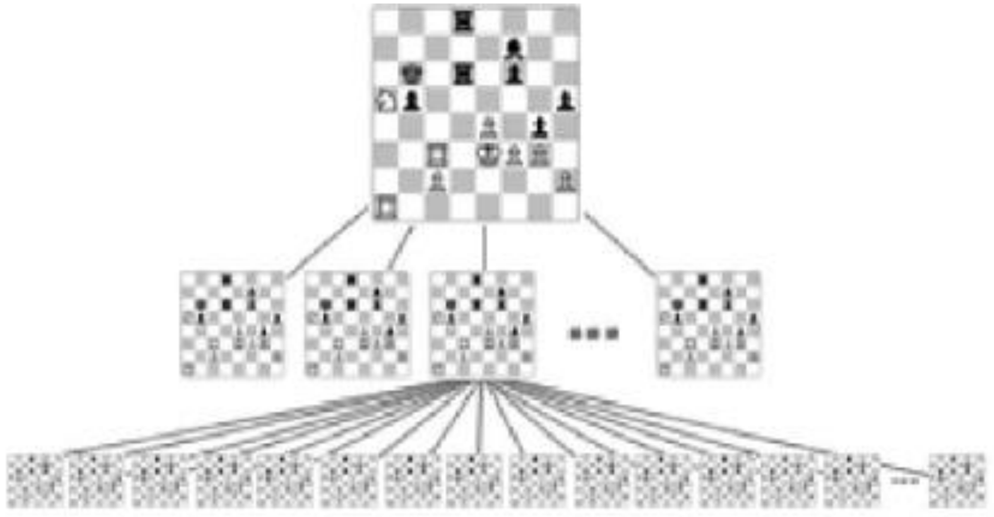
http://static3.businessinsider.com

Note that this is not artificial intelligence, but brutal computing power!

New York, 1997:

Chess World Champion – and arguably the eer-best chess player of all times – **Garry Kasparov** loses the championship against **Deep Blue**.

→ A Computer is chess world champion and has never been conquered again!

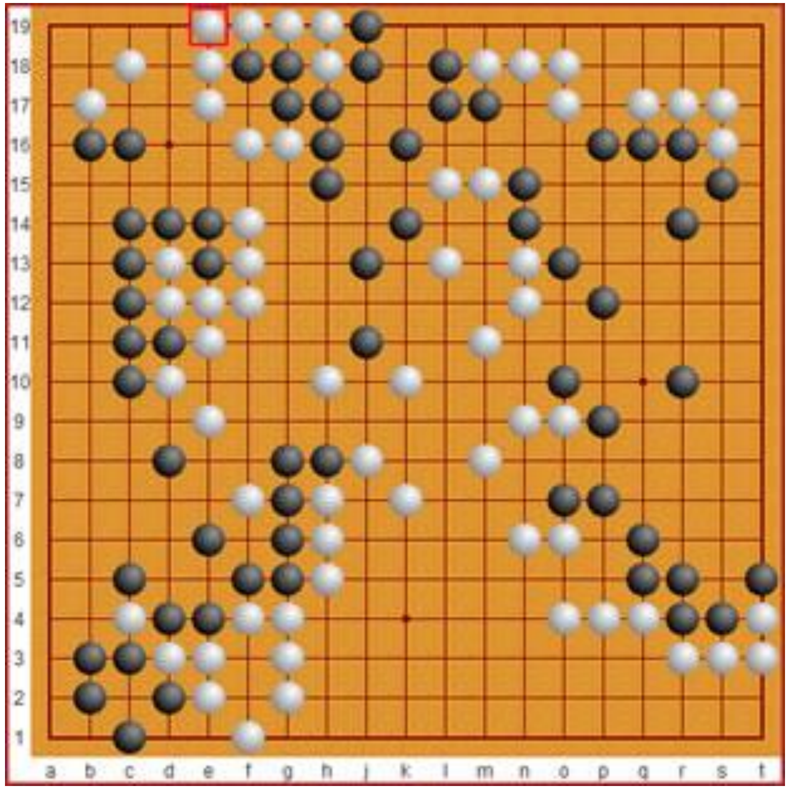


Software Success Story Example **10a**: **GO**

«**GO**» is a strategy board-game which was invented 2`500 years ago in China

https://fr.wikipedia.org

Board: 19 x 19 lines, unlimited number of black and white stones



http://www.brettspielnetz.de

Goal: Occupy as much territory as possible

Software Success Story Example **10a**: **GO**

https://static01.nyt.com

Number of different positions on the GO-board: ~ **4,63 x 10^{170}**

# of atoms in the universe: ~**10^{80}**

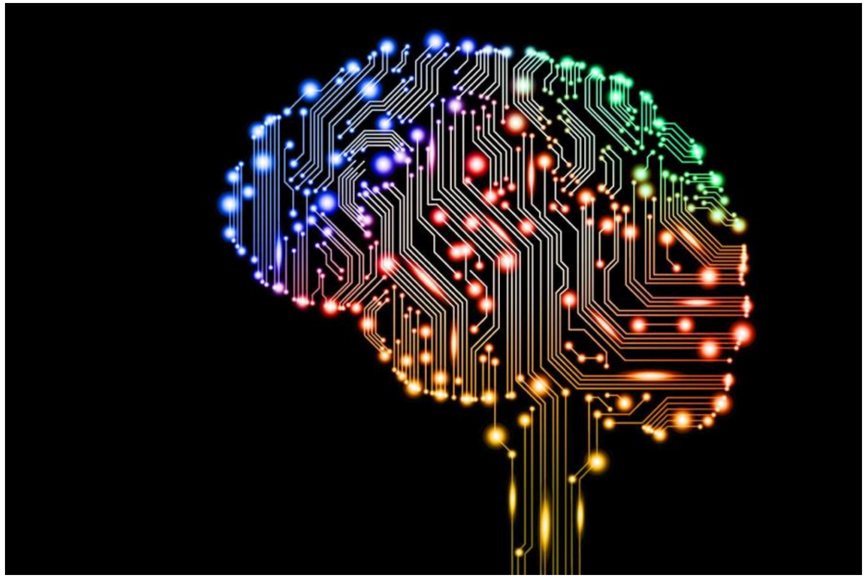
Chess: ~**10^{43}**

Software Success Story Example **10a**: **GO**

http://www.watson.ch

March 2016: The AI-program «AlphaGO» wins a tournament against the GO World champion Lee Sedol 4:1

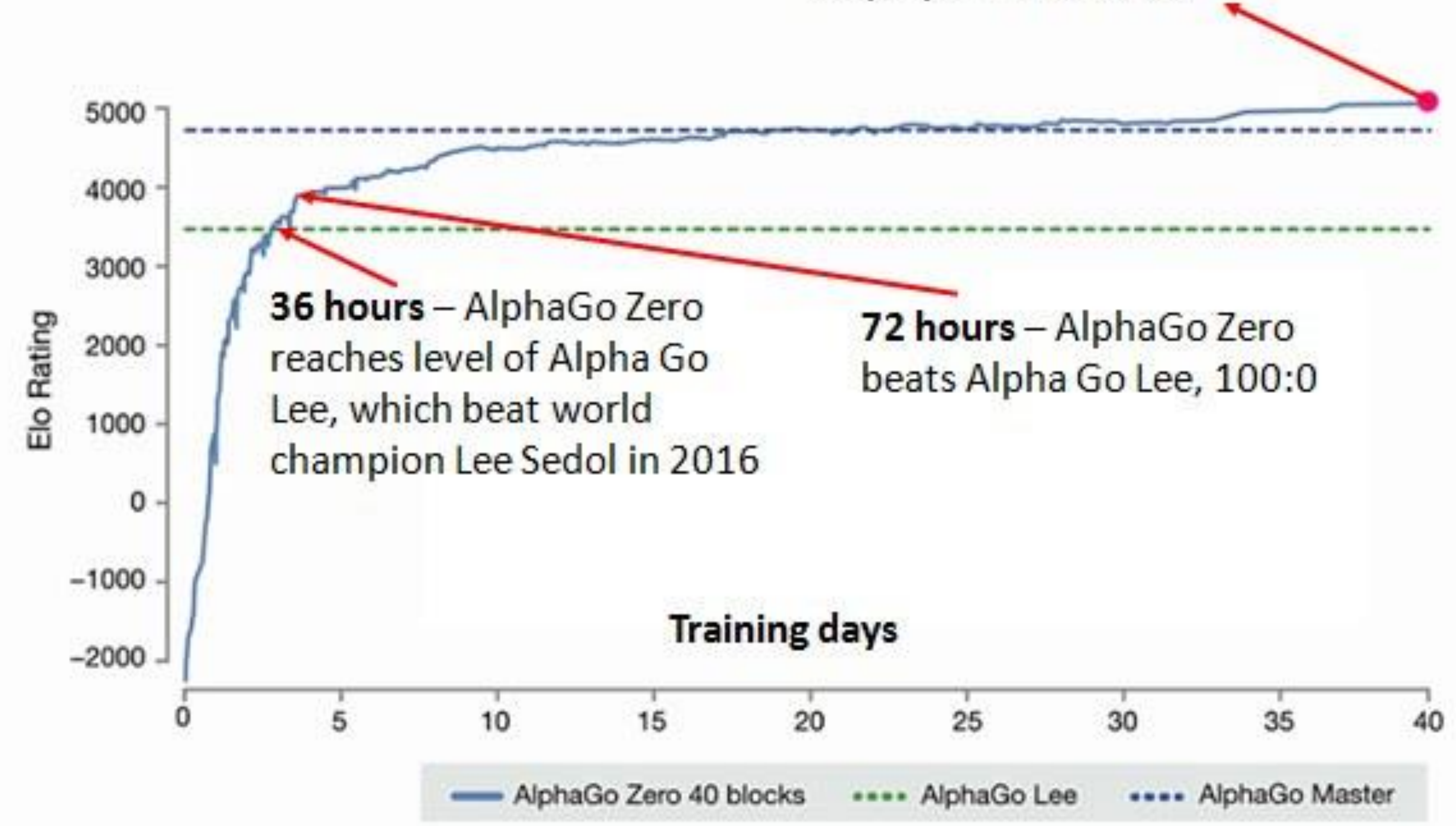
Impressive/Worrying:
«AlphaGO» is **NOT** an algorithm, but a self-learning software [Deep Learning]



http://www.digitaltrends.com

Software Success Story Example **10a**: GO

40 days – AlphaGo Zero surpasses all previous versions, becomes the best Go player in the world

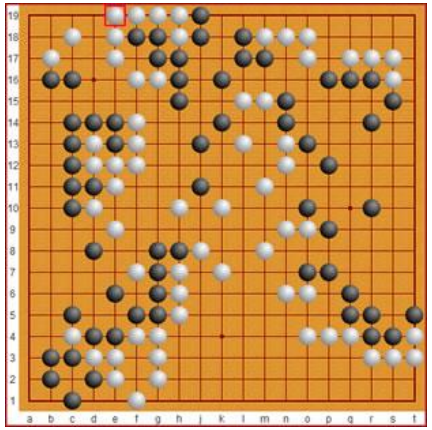


https://blog.yellowant.com

Software Success Story Example **10b**: **Poker**

Complete Information

Incomplete Information



Two-player Poker

Multi-player Poker

Zero-sum Games
Whatever one player wins, the other player loses

Multi-agent Game

Recognized, superhuman AI-milestone

Software Success Story Example **10c**: **Two-Player Poker**

Poker game: «**No Limits Texas Hold em'all**» is a card game 1:1 where strategy & bluffing is important

https://www.pokervip.com/strategy-articles/poker-rules/texas-holdem

# of different hands: ~ $\mathbf{10^{160}}$

# of atoms in the universe: ~$\mathbf{10^{80}}$

Software Success Story Example **10c**: **Two-Player Poker**

Two-Player Poker Tournament January 2017, simulataneously against 4 professionals:

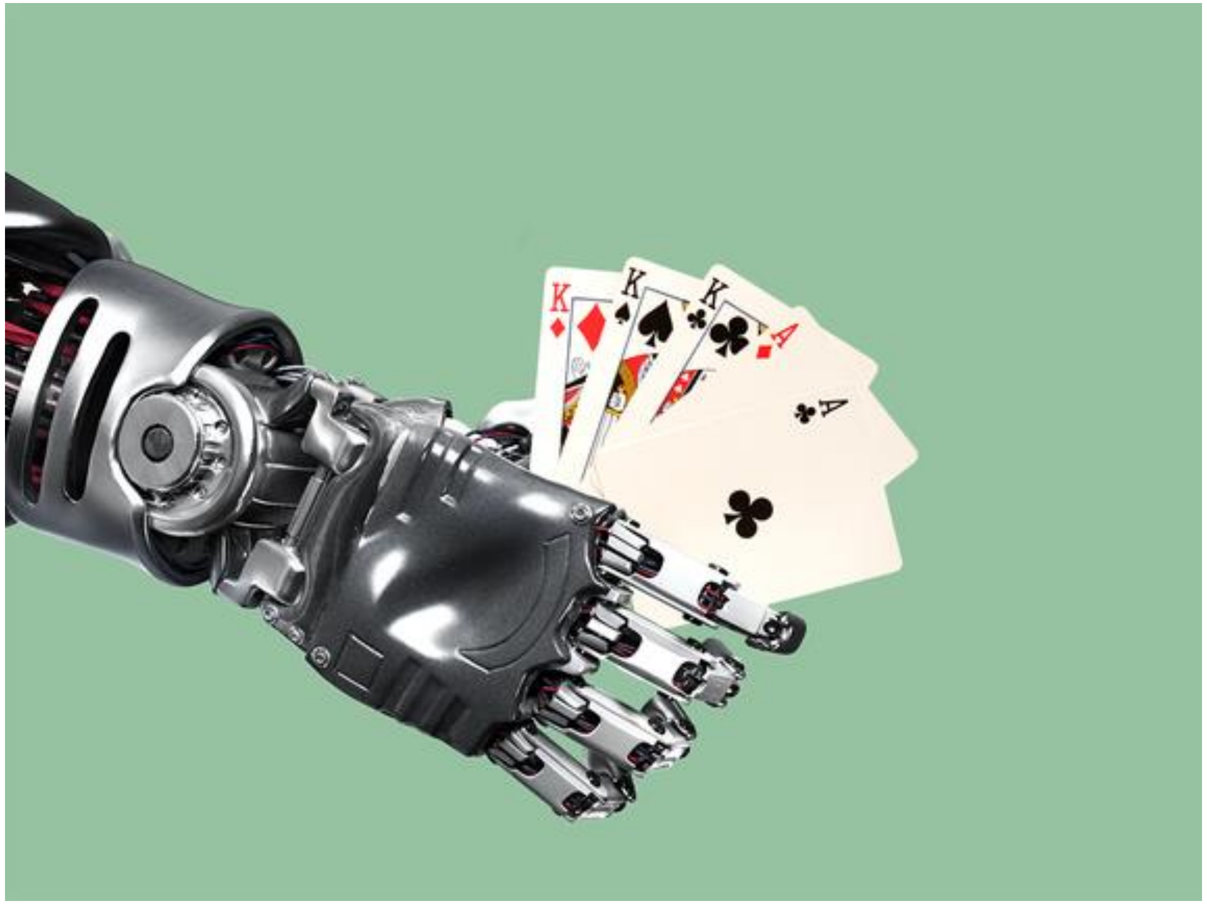
Our representatives of humanity:
Jason Les, Dong Kyu Kim, Daniel McAulay and Jimmy Chou

kept things relatively tight at the outset.

At the end of day 20 and after 120,000 hands, the AI-program **Libratus** claimed victory with daily totals of $206’061 in theoretical chips and an overall pile of $1’766’250

http://www.cmu.edu/news/stories/archives/2017/january/AI-tough-poker-player.html

Software Success Story Example **10d**: **Multi-Player Poker**



https://fivethirtyeight.com

Six-player, no-limit Texas Hold’em all still remains one of the most elusive **challenges for AI-systems**

July 2019:
Pluribus was evaluated by playing against an elite group of players that included several World Series of Poker and World Poker Tour champions

Pluribus played 10,000 hands of poker against five human players selected randomly from the pool. Pluribus’s win rate was estimated to be about 5 big blinds per 100 hands (5 bb/100), which is considered a **very strong victory** over its elite human opponents

Software Success Story Example **10d**: **Multi-Player Poker**

Pluribus learned poker by playing against copies of itself.

Starting from scratch, and playing randomly at first, the program steadily improved its performance.

After eight days, it had devised a “blueprint strategy”, which it uses for the first round of betting. For subsequent rounds, Pluribus looks ahead to hone its strategy. It aims to be unpredictable to wrongfoot its opponents

Noam Brown, Tuomas Sandholm: **Superhuman AI for multiplayer poker.**
Science, 11. July 2019
https://science.sciencemag.org/content/early/2019/07/10/science.aay2400/tab-pdf

⇒ Opens up new **AI-applications** with massively incomplete information and a high degree of uncertainty

Software Success Story Example **10d**: **Rubik's Cube**

Starting point: Random colour combination of the faces

End point: All faces have one colour

$$8! \times 3^7 \times (12!/2) \times 2^{11} = 43,252,003,274,489,856,000$$

1

Number of combinations

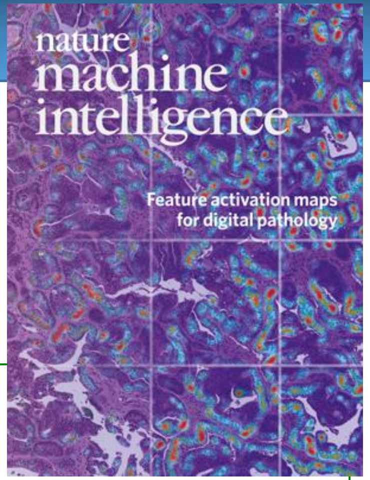
https://gadgetlot.com/rubiks-cube-3-by-3-puzzle/

Determinstic Algorithms:

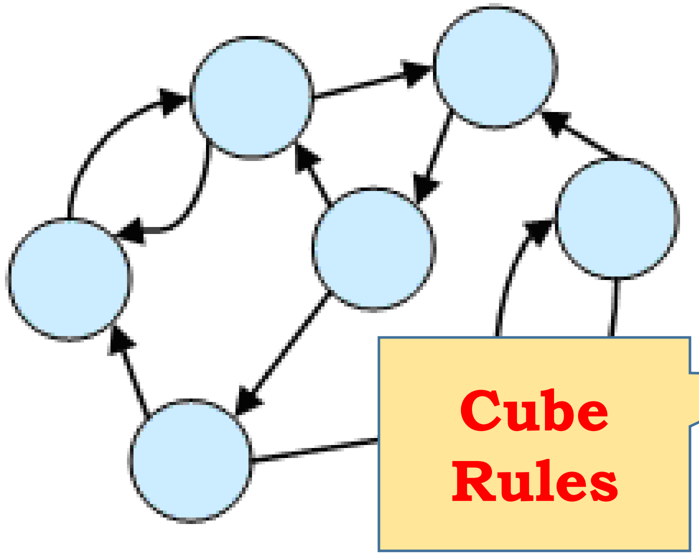
2008: Tomas Rokicki proved the minimum of moves to be **20**

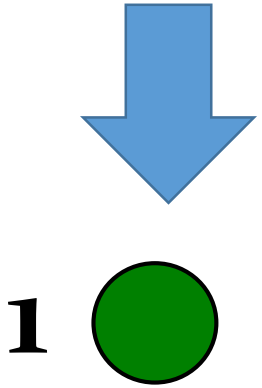
Software Success Story Example **10d**: **Rubik's Cube**

$$8! \times 3^7 \times (12!/2) \times 2^{11} = 43,252,003,274,489,856,000$$



https://boardgamegeek.com





July 2019: NATURE MACHINE INTELLIGENCE

We solve the Rubik's cube with **DeepCubeA**, a <u>deep reinforcement learning</u> approach that learns how to solve increasingly difficult states in reverse from the goal state ***without any specific domain knowledge***.

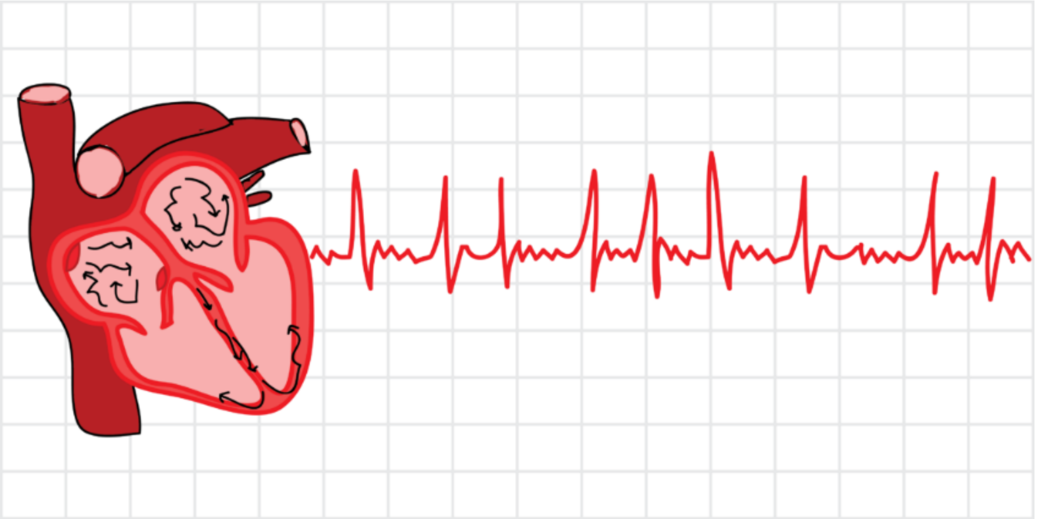
DeepCubeA solves 100% of all test configurations, finding a shortest path to the goal state 60.3% of the time

DeepCubeA is able to solve *planning problems* with large state spaces and few goal states by learning a cost-to-go function, parameterized by a DNN, which is then used as a heuristic function for weighted A* search

https://www.nature.com/articles/s42256-019-0070-z

Software Success Story Example **11**: **Atrial Fibrillation Detection**

Atrial fibrillation (Vorhofflimmern) is frequently asymptomatic and thus underdetected but is associated with stroke, heart failure, and death



www.scopeblog.stanfornd.edu

Existing screening methods require prolonged monitoring and are limited by cost and low yield

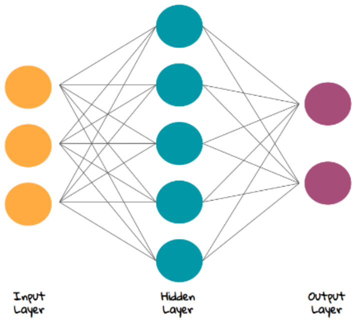
https://www.thelancet.com/journals/lancet/article/PIIS0140-6736(19)31721-0/fulltext

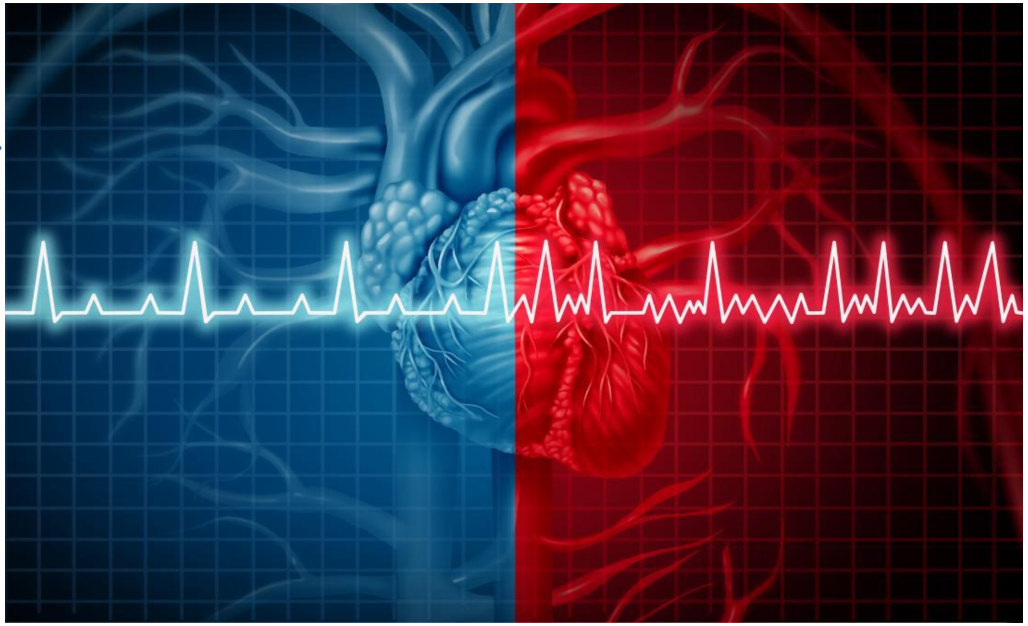
Software Success Story Example **11**: **Atrial Fibrillation Detection**

We developed an artificial intelligence (AI)-enabled electrocardiograph (ECG) using a convolutional neural network to detect the electrocardiographic signature of atrial fibrillation present during normal sinus rhythm

https://www.thelancet.com/journals/lancet/article/PIIS0140-6736(19)31721-0/fulltext

Training data:
We included 180 922 patients with 649 931 normal sinus rhythm





www.medicalnewstoday.com

Algorithm result:
overall accuracy of **79,4%**

Software Success Stories

http://mistakebook.azurewebsites.net

... innumerable and new ones daily!

... but

1.1

«Software Everywhere»

- Success Stories
- Failure Stories

… Some recent examples

https://newsnetwork.mayoclinic.org

Software Catastrophe Example **1: Crash Airbus A400M (9. Mai 2015)**

http://www.reuters.com

A400M: Military Transport Plane

Capacity: 37’000 kg

Range: > 3’000 km

Failure of the thrust control of 3 engines shortly after the start
$\Rightarrow$ **Crash**

http://www.ouest-france.fr

Software Catastrophe Example **1: Crash Airbus A400M (9. Mai 2015)**

http://defence-blog.com

Ground crew software update

0

Engine Control Data

http://www.triumphgroup.com

Start

Control Program

Check **completeness** and **integrity** of required data

Software Catastrophe Example **2**: **US$ 951 Million cyber-theft**

In February 2016, instructions to **steal US$ 951 million** from the central bank of Bangladesh, were issued via the SWIFT network

Five transactions issued by hackers, worth $101 million, succeeded

The Federal Reserve Bank of NY blocked the remaining thirty transactions, amounting to $850 million

Software Catastrophe Example **3**: **Unwanted acceleration of Toyota cars**

http://businessethicscases.blogspot.ch

The unwanted acceleration of Toyota and Lexus cars caused

89 traffic deaths and **52 injured** from 2000 to 2010

Software Catastrophe Example **3**: **Unwanted acceleration of Toyota cars**

http://www.autoevolution.com

Toyota claimed in the beginning that the ***doormat*** was the source of the acceleration

Independent research demonstrated a ***software-problem*** in the throttle control

19. March 2014: Toyota pays a US-fine of 1.2 Billion US$

Software Catastrophe Example **4**: **Automated Trading Big Loss**

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)

Software Catastrophe Example **4**: **Automated Trading Big Loss**

http://www.nj.com

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 Millionen US$**

https://www.mytechlogy.com

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

Software Catastrophe Example **5**: **Blockchain Code Exploit**

http://www.extremetech.com

A **blockchain** is a cryptographic, anonymous public ledger of all cryptocurrency transactions that have ever been executed in a community.

The blockchain-technology is the base for nearly all **FinTech** ventures.

http://www.bitcoinisle.com

Anyone who invested Ether into the ***DAO fund*** received a particular number of DAO tokens, which enabled them to vote on the projects that the DAO will fund. By the end of May, the DAO had raised more than **US$150 million** worth of Ether from investors.

http://fortune.com

http://www.coindesk.com

Software Catastrophe Example 6: Cryptocurrency Exchange Hacks

https://cryptoeddy.com

A brief History of Crypto Exchanges Hacks
Total loss to date (Jul 11 – Sep 18):
$1,542,620,000.-
Source: https://discover.ledger.com/hackstimeline/

+ Wallet hacking
+ Mining hacking

Software Catastrophe Example 7: US Clinton e-Mail Hack

https://www.theatlantic.com

In March 2016, the personal Gmail account of John Podesta, the chairman of Hillary Clinton's 2016 U.S. presidential campaign, was compromised in a data breach, and a collection of his **e-mails**, many of which were work-related, were stolen https://en.wikipedia.org/wiki/Podesta_emails

The e-mails were subsequently published by WikiLeaks.

https://www.theatlantic.com:

"Conservatives will see corruption and liberals will see corporatism and expedience, but the exchanges simply expose the candidate who's been there all along"

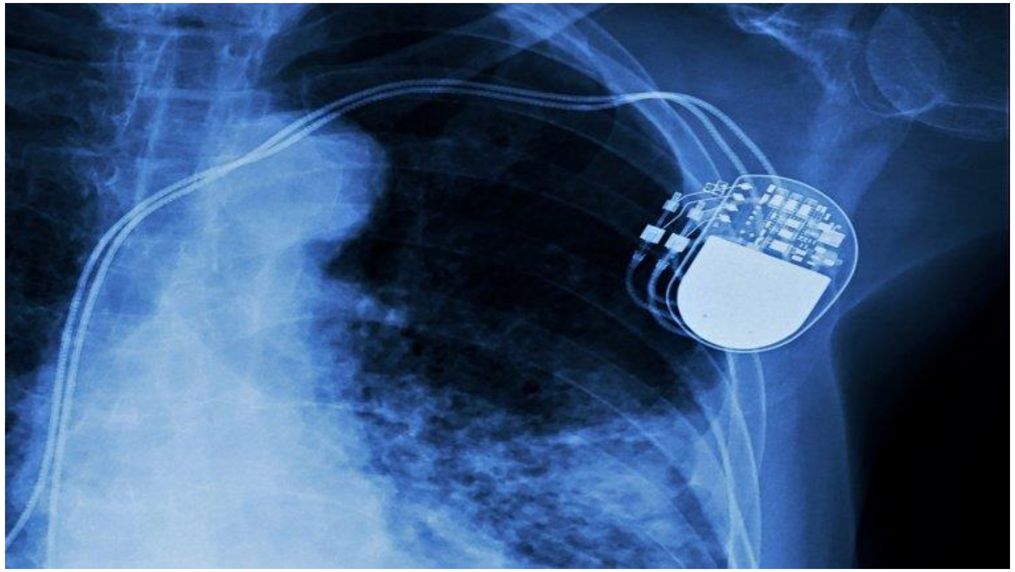
The leaks certainly damaged Hilary Clinton's campaign and possibly decided the outcome

Software Catastrophe Example **8**: **Heart Pacemaker Vulnerability**

https://www.cisomag.com/wp-content/uploads/2017/08/Pacemaker.jpg

August 30, 2017:

An estimated 465,000 people in the US are getting notices that they should ***update the firmware*** that runs their life-sustaining pacemakers or risk falling victim to potentially ***fatal hacks***



https://www.hackread.com/pacemaker-ecosystem-security-flaws-leave-vulnerable-hackers/

https://arstechnica.com/information-technology/2017/08/465k-patients-need-a-firmware-update-to-prevent-serious-pacemaker-hacks/

Software Catastrophe Example **9**: EQUIFAX Hacking

http://cdn.mos.cms.futurecdn.net

7. September 2017:

Data of 143 million Americans exposed in hack of credit reporting agency Equifax

https://www.washingtonpost.com

Hackers gained access to *sensitive personal data* — Social Security numbers, birth dates, home addresses, credit histories — for up to 143 million Americans, a major cybersecurity breach at a firm that serves as one of the three major clearinghouses for Americans' ***credit histories***

https://www.hackread.com

Software Catastrophe Example **10**: CAPITOL ONE Hacking

A hacker gained access to 100 million Capital One credit card applications and accounts

By Rob McLean, CNN Business
Updated 2117 GMT (0517 HKT) July 30, 2019

https://edition.cnn.com

Paige Thompson is accused of breaking into a Capital One server and gaining access to 140,000 Social Security numbers, 1 million Canadian Social Insurance numbers and 80,000 bank account numbers, in addition to an undisclosed number of people's names, addresses, credit scores, credit limits, balances, and other information, according to the bank and the US Department of Justice

Software Catastrophe Example **11**: Boeing 737 MAX Accidents

Both planes crashed **nose-down**

What happened?

Lion Air Flight 610: On 29 October 2018, the Boeing 737 MAX 8 crashed into the Java Sea 12 minutes after takeoff, killing all 189 passengers and crew

https://www.abc.net.au

Ethiopian Airlines Flight 302: Six minutes after takeoff, the plane crashed near the town of Bishoftu, Ethiopia, killing all 157 people aboard.

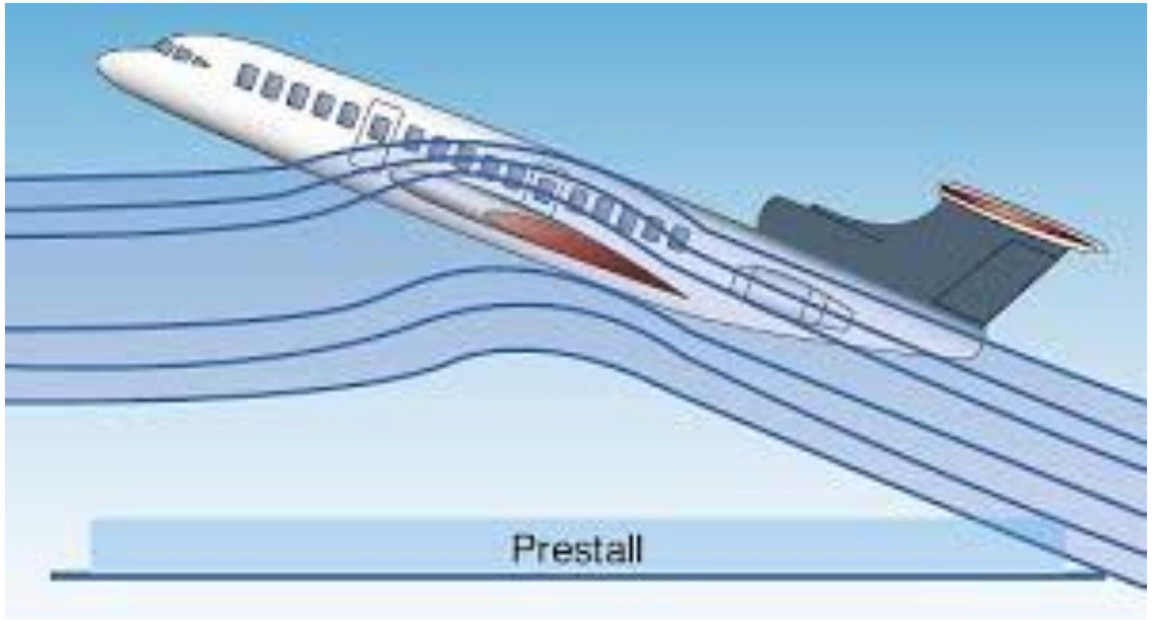
https://www.aerotelegraph.com

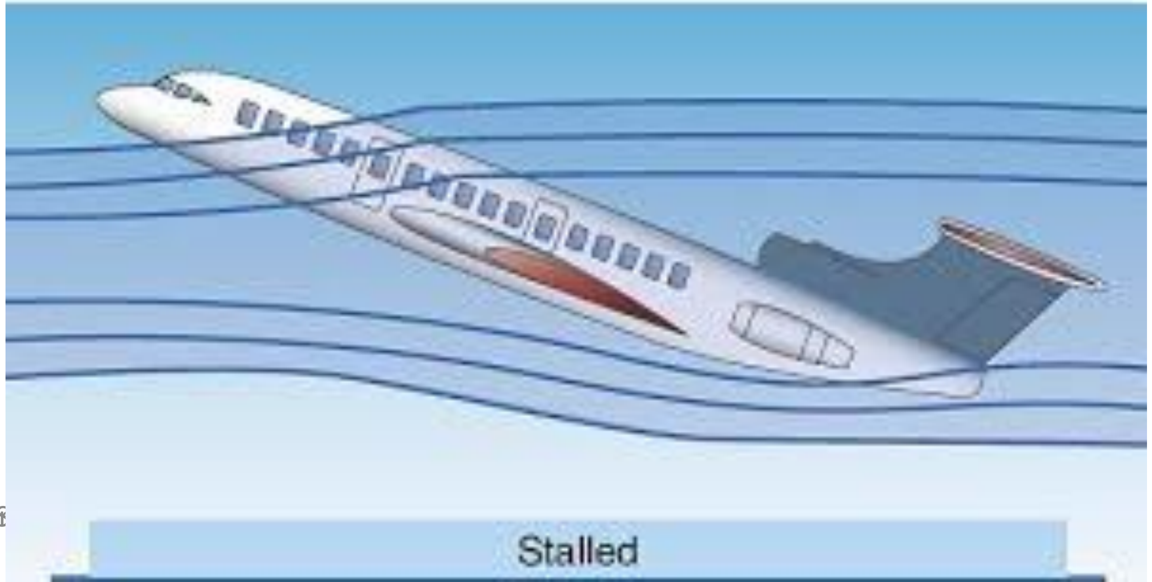
Software Catastrophe Example **11**: Boeing 737 MAX Accidents

The 737 MAX was equipped with new, more fuel-efficient engines

https://leehamnews.com

Airflow
⇓
Lift





https://en.wikipedia.org

The larger engines augmented the risk of **stalling**

Lift
Loss
⇓
Stalling

Software Catastrophe Example **11**: Boeing 737 MAX Accidents

https://www.youtube.com

Dangerous nose-up angle

→ Risk of stalling (= loss of uplift)

Software-Fix:
MCAS takes readings from sensors to determine how much the plane's nose is pointing up or down. If the software detects the nose is pointing up at a dangerous angle it automatically pushes the nose to **stop the plane stalling**

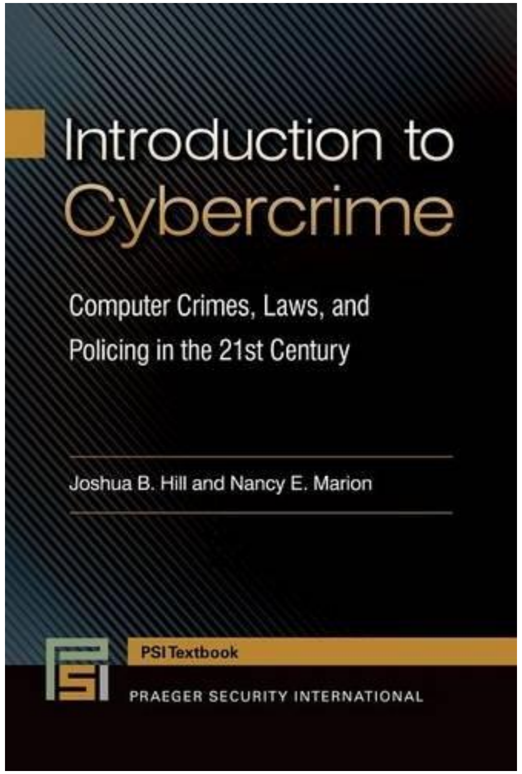
https://www.theguardian.com

https://news.delta.com

… However:

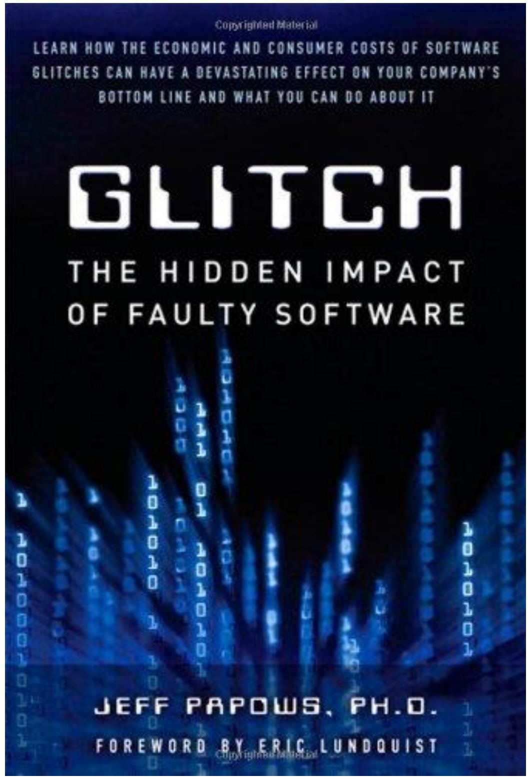
- The pilots were **not** informed about this (new) functionality
- The MCAS (= Software) decisions/actions could **not** be overridden by the pilots

Textbook



Nancy Marion:
Introduction to Cybercrime – *Computer Crimes, Laws, and Policing in the 21st Century*
Praeger Security International,2016. ISBN 978-1-4408-3533-9

Textbook



Jeffrey Papows:
Glitch – *The Hidden Impact of Faulty Software*
Prentice Hall Inc., USA, 2010. ISBN 978-0-132-16063-6

«Software Everywhere»

... and what is the message?

«Software Everywhere» … and what is the message?

Success Stories

Software generates **Business Value**

⇒ Products, Services, Quality of Life, …

http://smallbusinessbc.ca

Failure Stories

Software creates **Risks**

⇒ Accidents, failures, malfunctions, …

http://carpng.com

https://www.decisivedge.com

Reasons:

https://www.afcea.org

Software Fault

Software Vulnerability

Application-Software

- Bug
- Malfunction
- Fault/Error/Failure
- Design/Implementation Flaw
- ...

Trusted Engineering

Application & Systems-Software

- Malware entry-point
- Unauthorized access way
- Insider crime
- ...

Message

Responsibility of the Software Community:

http://img.quotery.com

To build and operate **dependable/trustworthy** software

https://de.freepik.com

... «The software does what it should do, and does not what it should not do»

Requirements Specifications

Dependability/Trustworthiness is a consequence of good quality properties

Functional Properties

Non-Functional Properties

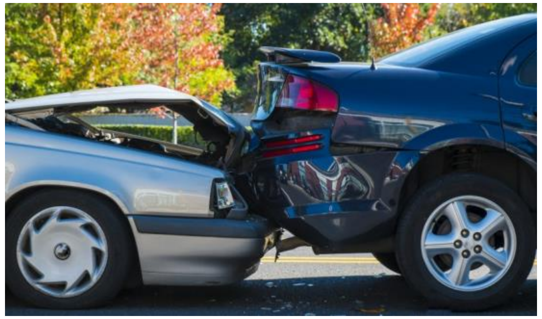
Quality Properties

https://www.usabilityblog.de

Business-/Applications Functionality

Security

Safety



Availability

... etc.

Objective:

To build and operate **dependable/trustworthy** software

… *«The software does what it should do, and does not what it should not do»*

To generate **business value** for its owner (and the community)

… *«The software industry is today one of the largest industries in the world»*

To maintain a high **changeability** of the software:

… *«The software must efficiently be adaptable to new requirements»*

FPSS

Our objective is:

to build, evolve, and maintain
long-lived, mission-critical IT-systems
with a strong dependability,
an easy changeability
and a high business value.

http://www.gettyimages.ch

Systems-Engineering

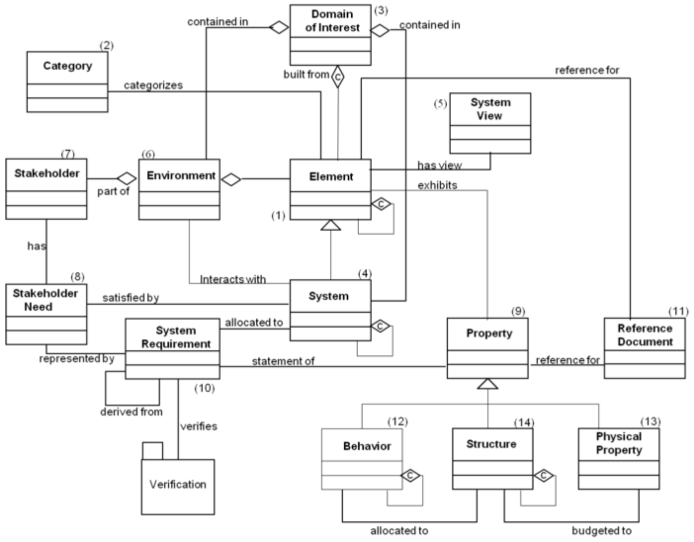
Software-Engineering

... and some definitions

Future-Proof Software-Systems [Part 1]

Existing Software-System

https://www.sebokwiki.org



New Requirements

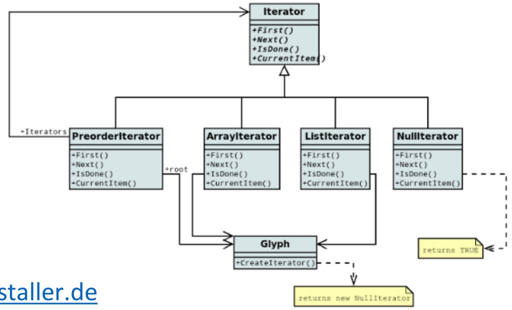
- ✓ Functional
- ✓ Non-functional (Properties)
- ✓ Legal
- ✓ Operational
- ✓ …

https://kms.com.au

1 Managed Evolution Strategy

Development, Integration & Deployment Processes

2 Architecture Principles



http://dia-installer.de

Extended Software-System

```
...code omitted...
    Text  textRef = TextBuilder.create()
      .layoutY(100)
      .textOrigin(VPos.TOP)
      .textAlignment(TextAlignment.JUSTIFY)
      .wrappingWidth(400)
      .text(message)
      .fill(Color.rgb(187, 195, 107))
      .font(Font.font("SansSerif", FontWeight.BOLD, 24))
      .build();

    TranslateTransition  transTransition = TranslateTransitionBuilder.create()
      .duration(new Duration(75000))
      .node(textRef)
      .toY(-820)
      .interpolator(Interpolator.LINEAR)
      .cycleCount(Timeline.INDEFINITE)
      .build();

    Scene scene  = SceneBuilder.create()
      .width(516)
      .height(387)
      .root(
        GroupBuilder.create()
          .children(
            ImageViewBuilder.create()
              .image(new Image("http://projavafx.com/images/earthrise.jpg"))
              .build(),
            ScrollPaneBuilder.create()
              .layoutX(50)
              .layoutY(180)
              .prefWidth(440)
              .prefHeight(85)
              .hbarPolicy(Scr        NEVER)
              .vbarPolicy(Scr        NEVER)
              .pannable(true)
              .content(textRe
              .style("-fx-ba         transparent;")
              .build()
          )
          .build()
      )
      .build();
...code omitted...
```

http://what-when-how.com https://www.techwell.com

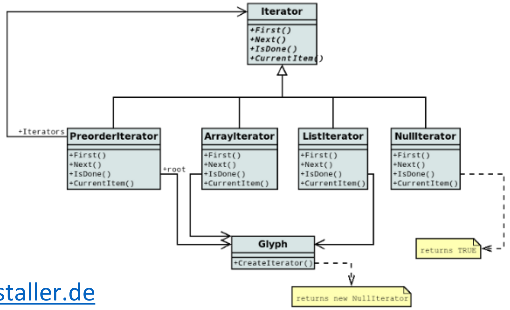
Future-Proof Software-Systems [Part 1]

https://kms.com.au

Managed Evolution Strategy

Development, Integration & Deployment Processes

Architecture Principles



http://dia-installer.de

Enables you to work as a successful **software architect** in industry

Software engineering is the application of engineering to the design, development, implementation, testing and maintenance of software using systematic methods

DEFINITIONS

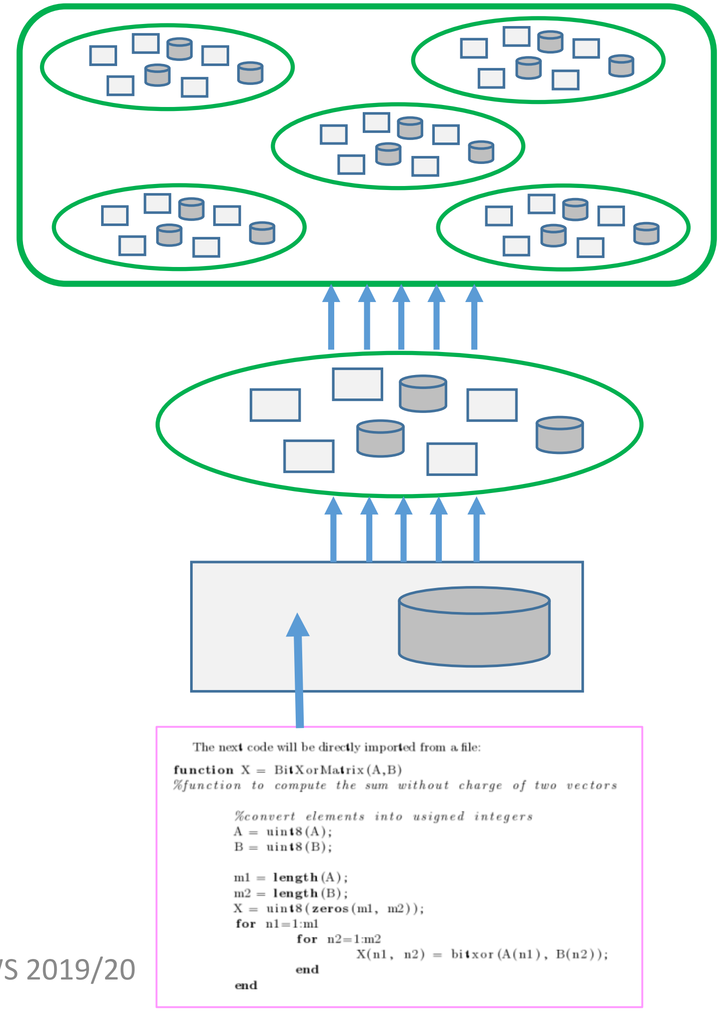
Software Hierarchy:

Application Landscape

Application

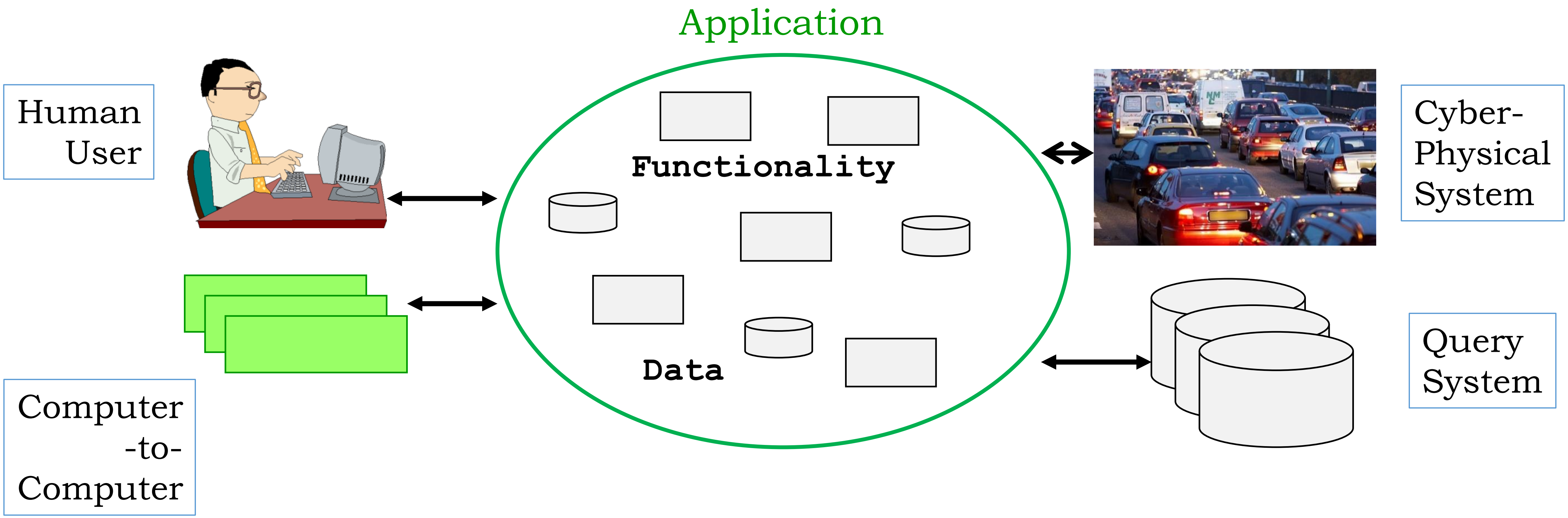
Component

Program, Module



DEFINITIONS

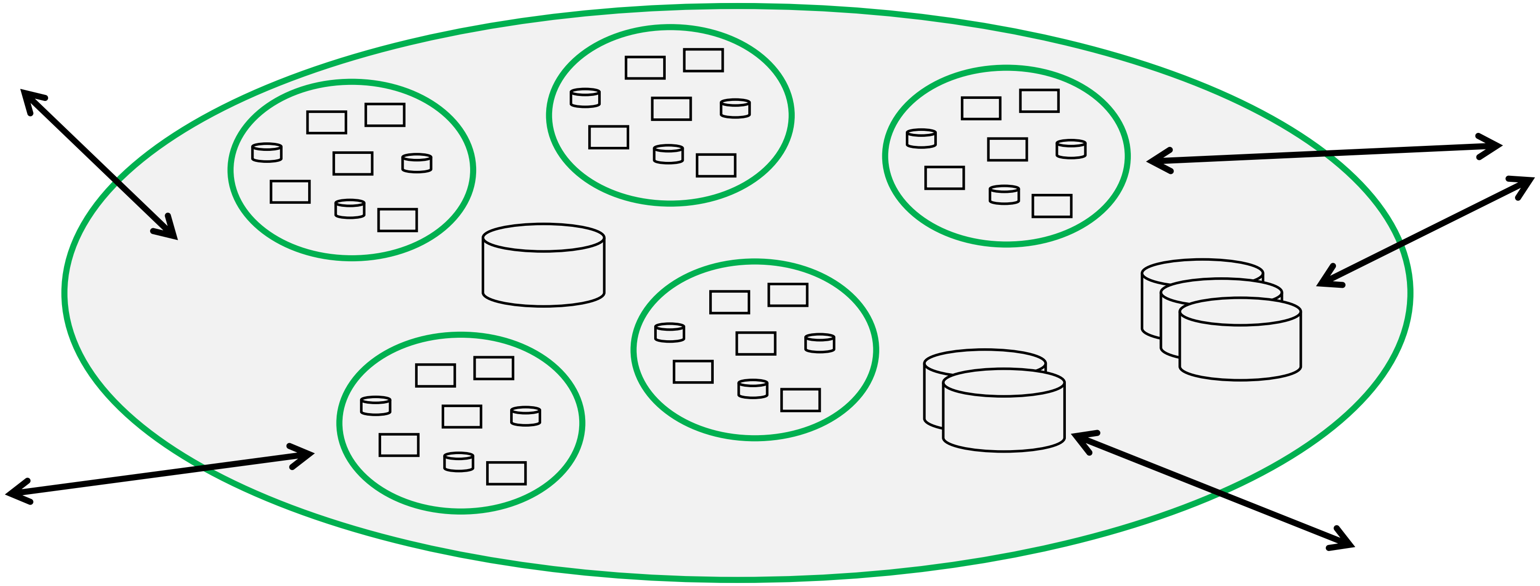
Application (software) =
Software designed to fulfill <u>specific needs</u> of a user: for example, software for navigation, payroll, or process control (IEEE Std 610.12-1990)



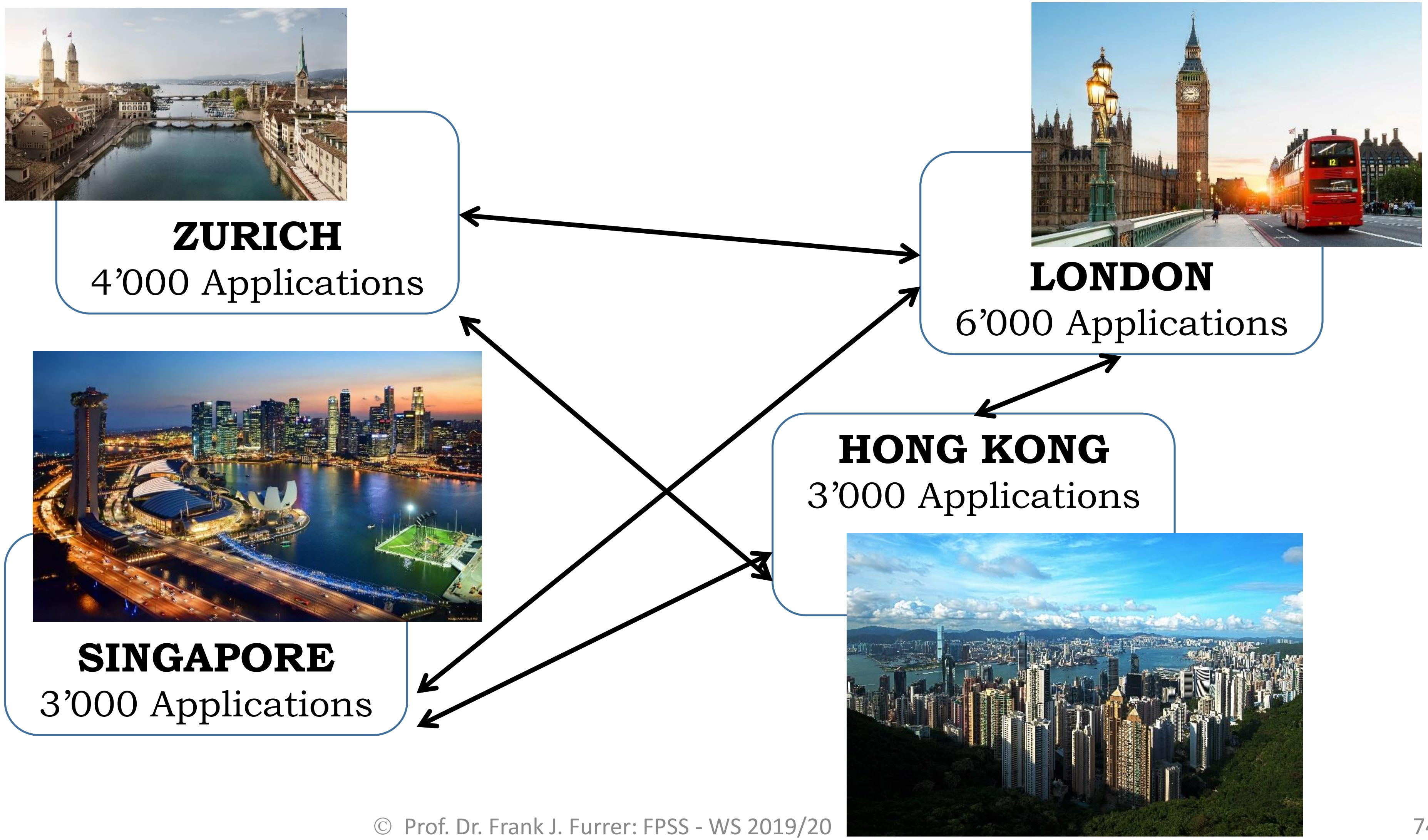
DEFINITIONS

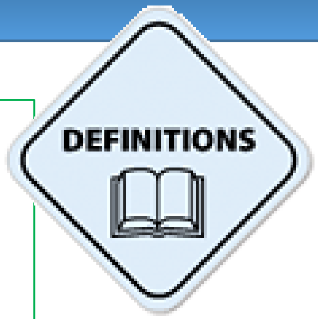
Application Landscape =
Set of interacting applications and data cooperating to achieve a common objective: for example operate a bank, drive a car, or control a manufacturing process

Application Landscape



Example: CREDIT SUISSE **distributed** Application Landscape

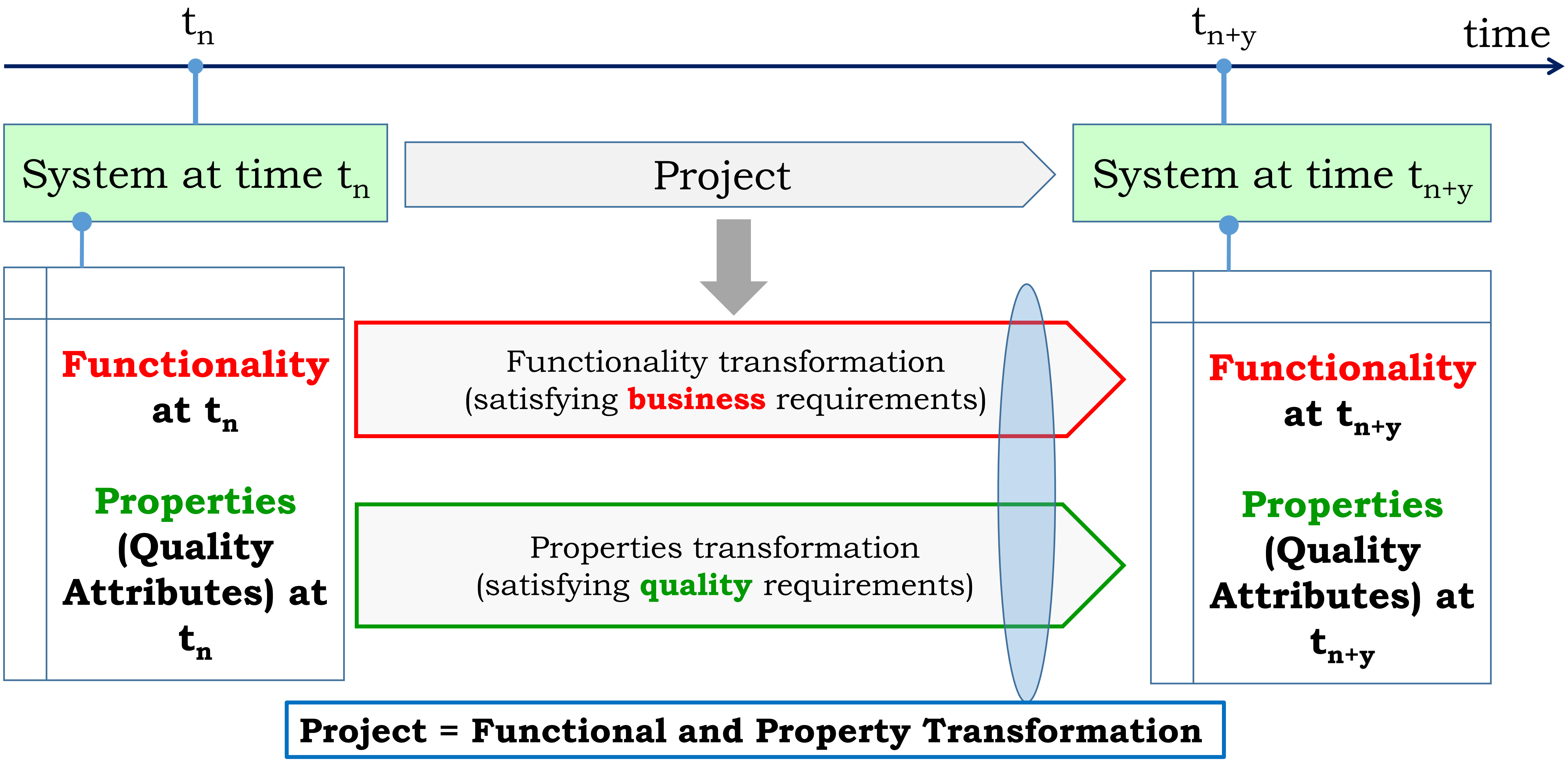




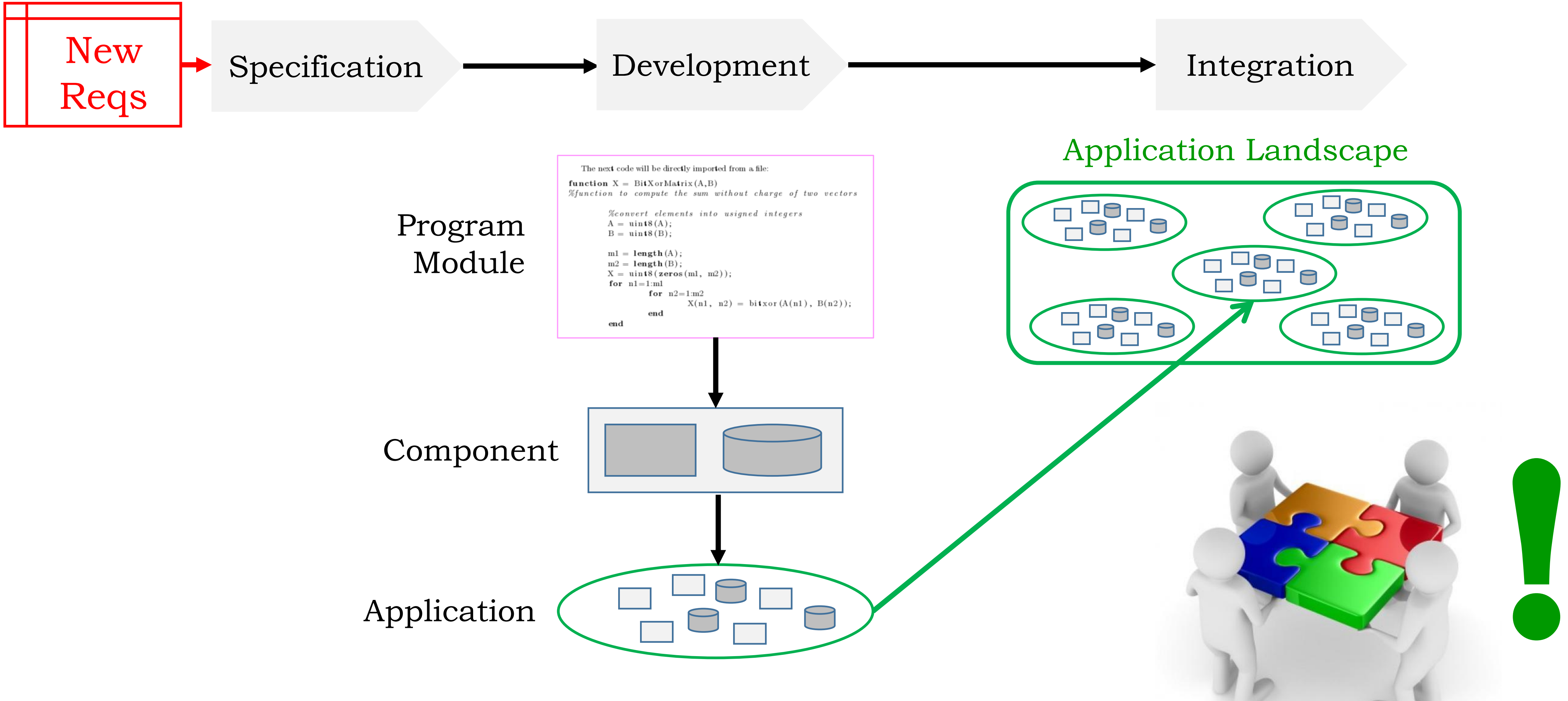
Project =
Planned set of interrelated tasks to be executed over a fixed period and within certain cost and other limitations

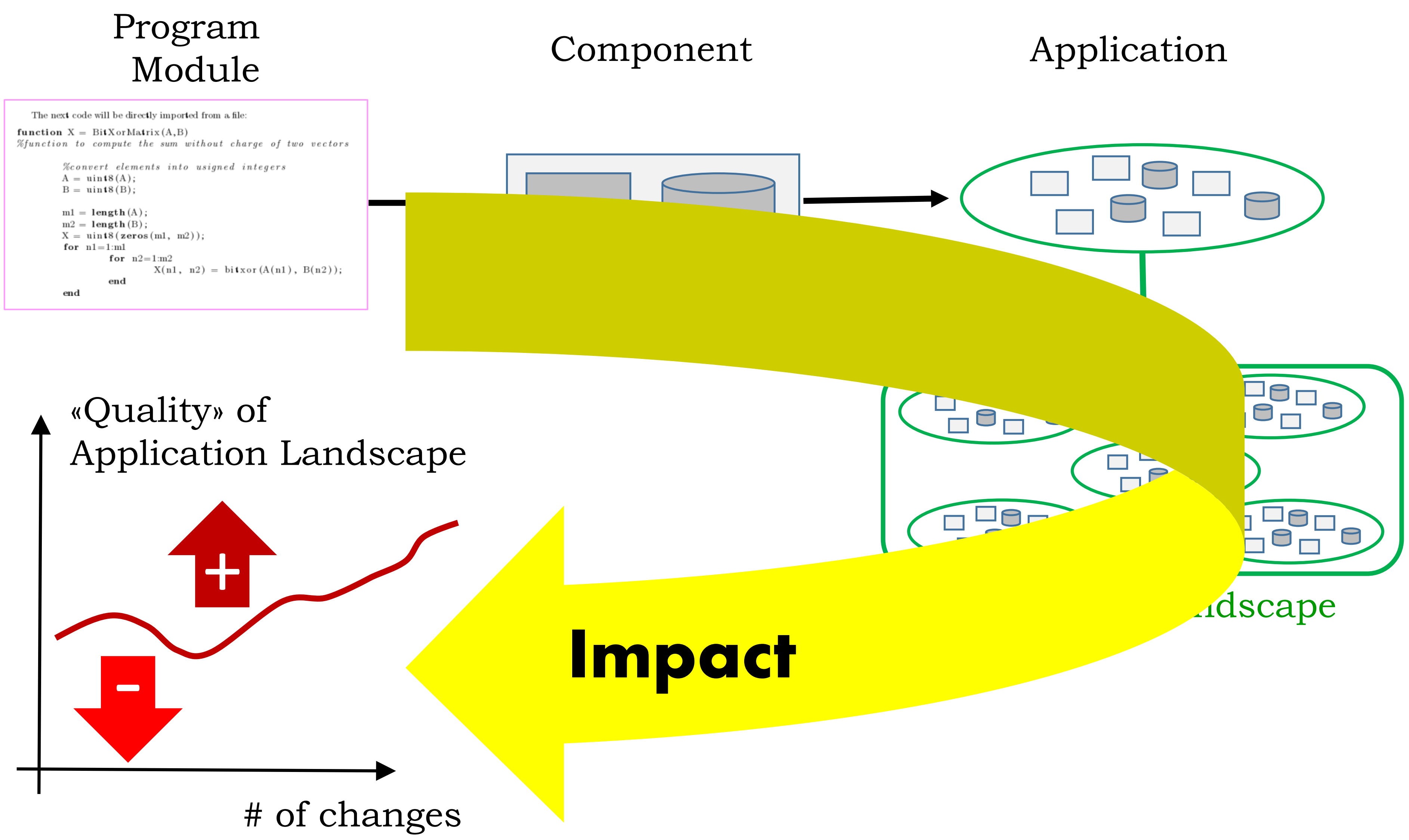
http://www.businessdictionary.com/definition/project.html

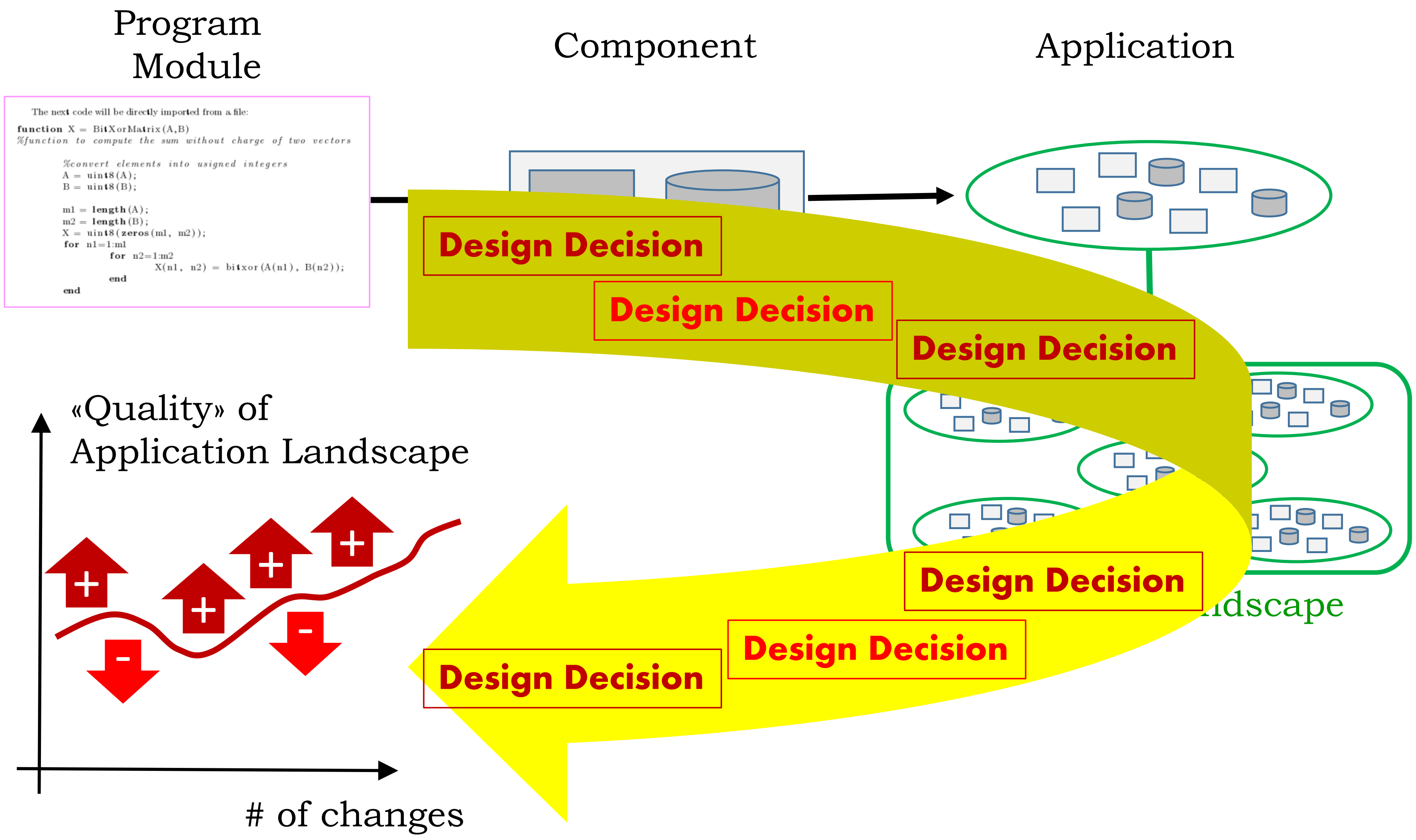
http://wolcottwhisper.com


t_n
t_{n+y}
time
System at time t_n
Project
System at time t_{n+y}
Functionality at t_n
Properties (Quality Attributes) at t_n
Functionality transformation (satisfying business requirements)
Properties transformation (satisfying quality requirements)
Functionality at t_{n+y}
Properties (Quality Attributes) at t_{n+y}
Project = Functional and Property Transformation

Software Development & Integration

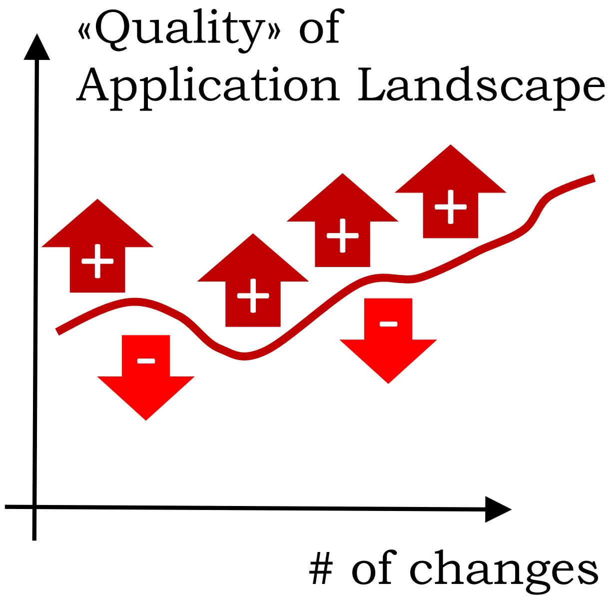



Program Module
Component
Application
The next code will be directly imported from a file:
function X = BitXorMatrix(A,B)
%function to compute the sum without charge of two vectors
%convert elements into usigned integers
A = uint8(A);
B = uint8(B);
m1 = length(A);
m2 = length(B);
X = uint8(zeros(m1, m2));
for n1=1:m1
for n2=1:m2
X(n1, n2) = bitxor(A(n1), B(n2));
end
end
«Quality» of Application Landscape
+
-
# of changes
Impact
ndscape



The «quality» of the application landscape is the sum of all design & implementation decisions

http://searchengineland.com



… defined later

The «quality» of the application landscape is a consequence of:

- Architecture choices
- Design decisions
- Implementation options

The «quality» of the application landscape is the sum of all design & implementation decisions

Architect

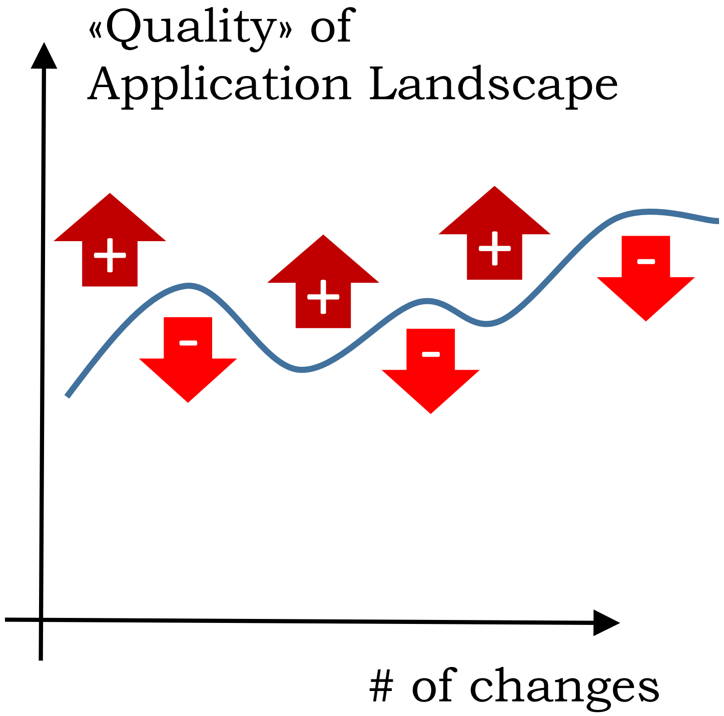
http://exploringthemind.com

Good design decision

- Proven principles
- Good people
- Quality process
- Sufficient resources

Bad design decision

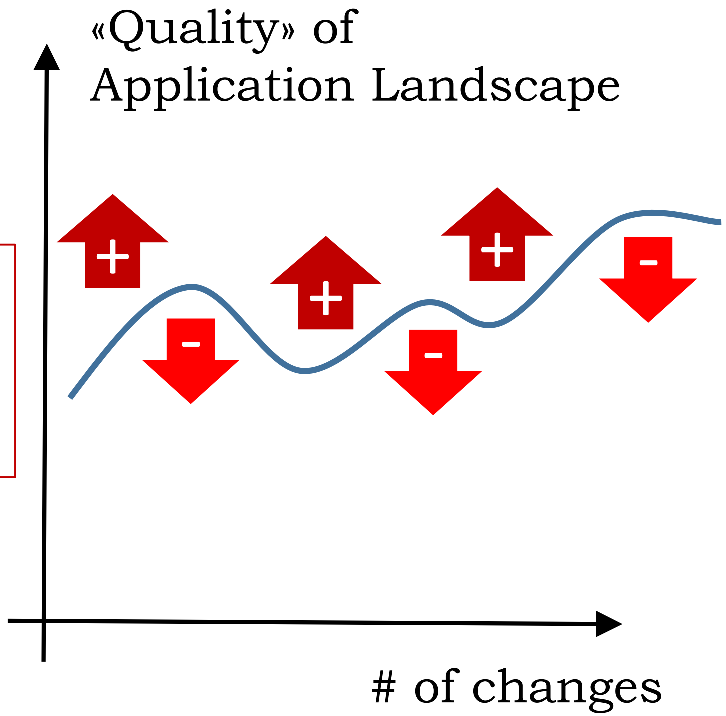
- Missing overall architecture
- Lack of principles enforcement
- Careless people
- Technical debt accumulation
- Time & resource shortage



Good design decision

https://www.entrepreneur.com

- Principles
- People
- Processes

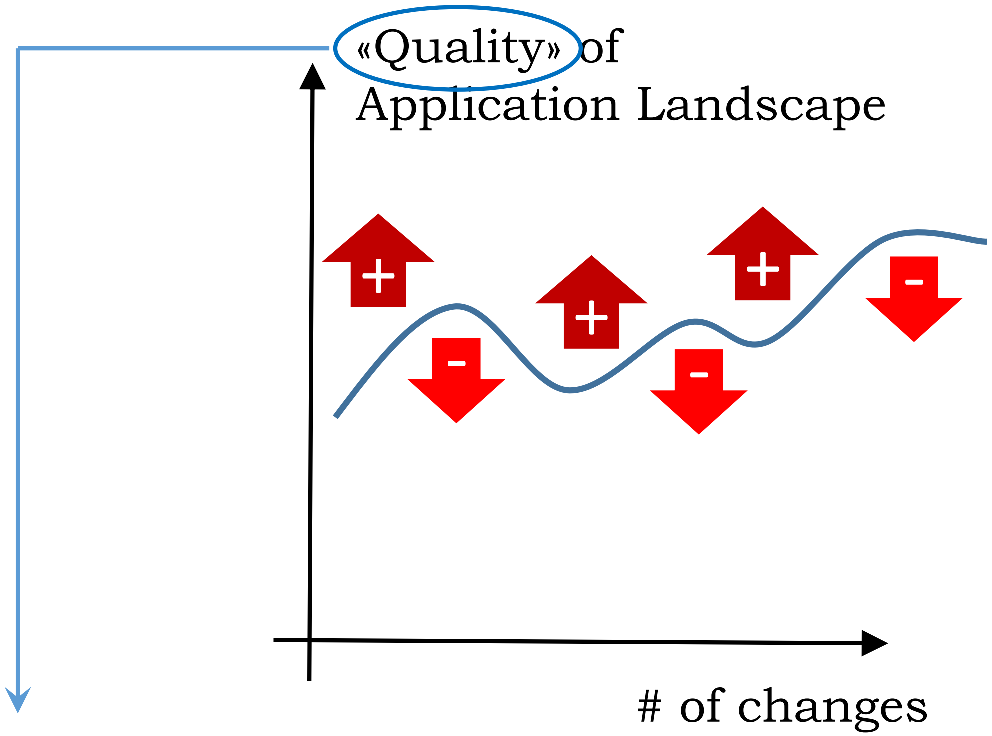


Bad design decision

https://www.emaze.com

Architect

Even the best designed system will be killed by a sequence of bad design or implementation decisions http://media.istockphoto.com



Functional:

- free of defects
- match specifications

Non-Functional:

- … «-illities»
- Security, agility, safety, …

- … «-illities»

Non-functional properties [= Quality Attributes]

- Safety
- Security
- Availability
- Integrity
- Performance
- Maintainability
- Recoverability
- Resource consumption (power, memory, …)
- Diagnosability
- …

Which quality attributes are most important?

⇒ Depends on the application!

- … «-illities»

Application → **Quality Attributes Scorecard**

«Fit for Purpose»

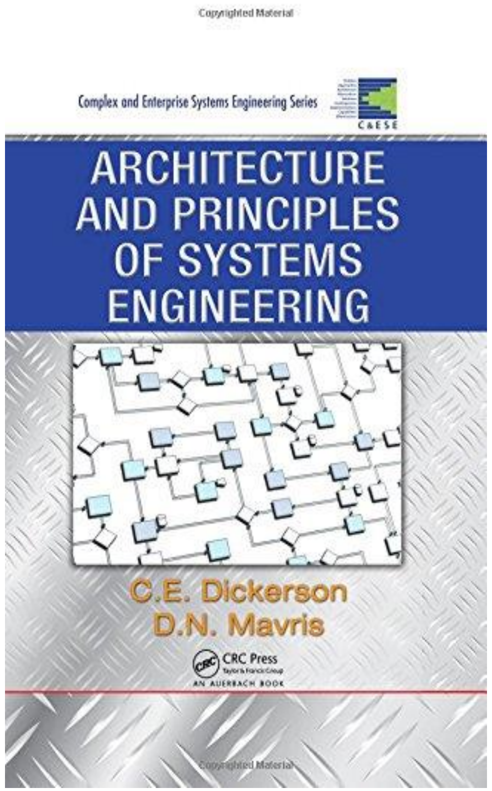
Quality Attributes:

- Resources
- Security
- Safety
- Availability
- Performance
- Integrity
- Maintainability
- Standards conformance
- …

Dependability

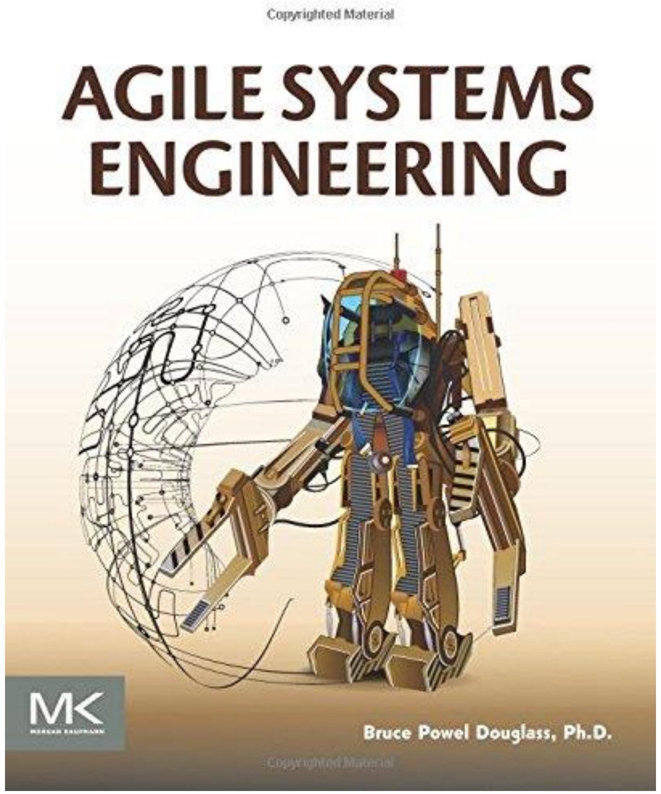
Dependability objective:
No harm to life and property (internal and external)

Textbook



C.E. Dickerson, D.N. Mavris:
Architecture and Principles of Systems Engineering
CRC Press (Taylor & Francis), Boca Raton, USA, 2010. ISBN 978-1-4200-7253-2

Textbook



Bruce Powel Douglass:
Agile Systems Engineering
Morgan Kaufmann Publishers (Elsevier), Waltham, MA, USA, 2016. ISBN 978-0-12-802120-0

Change

Uncertainty

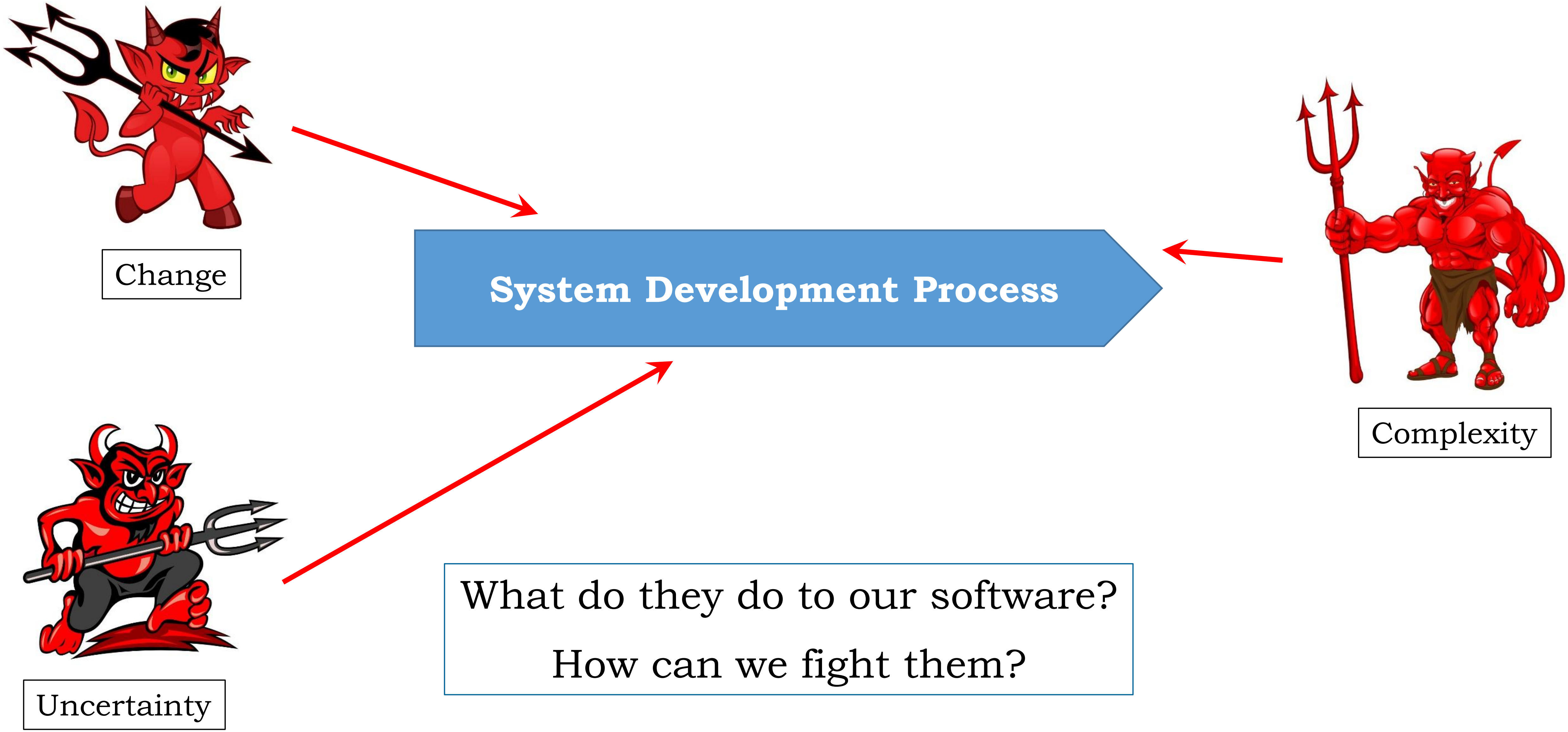
The

3 Devils

of

Systems-Engineering

Complexity

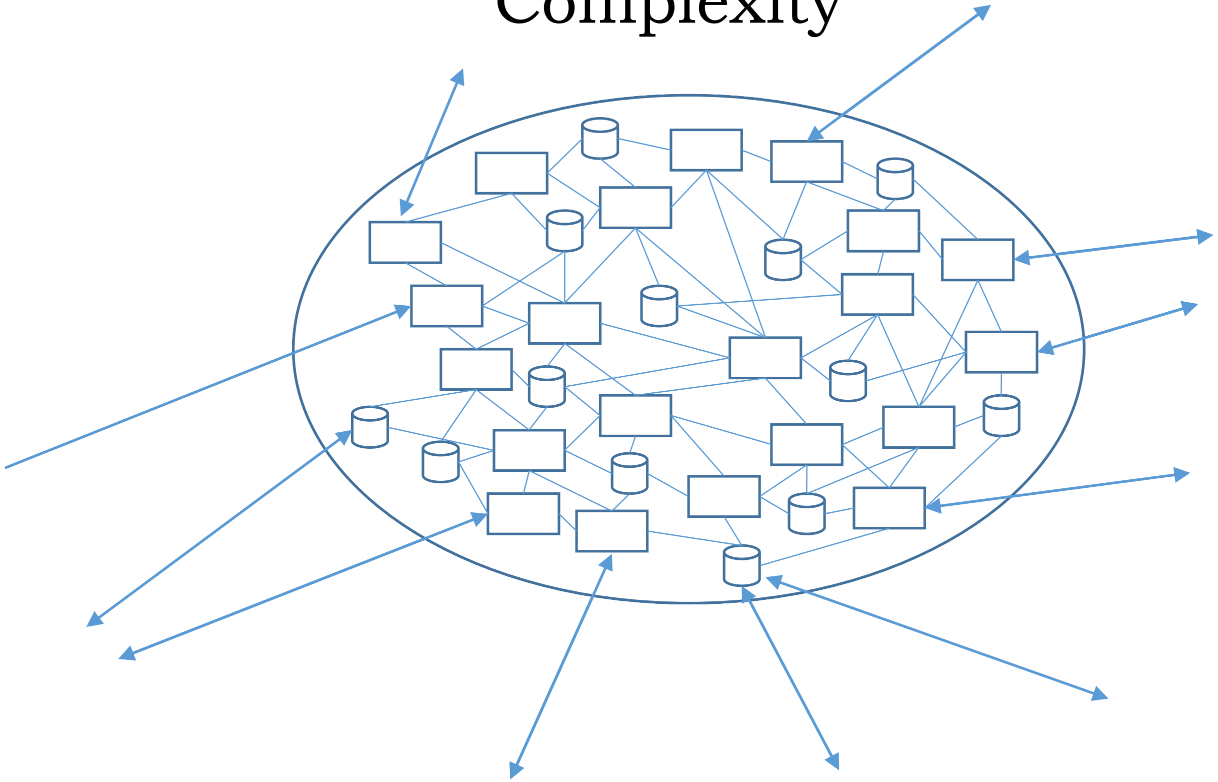


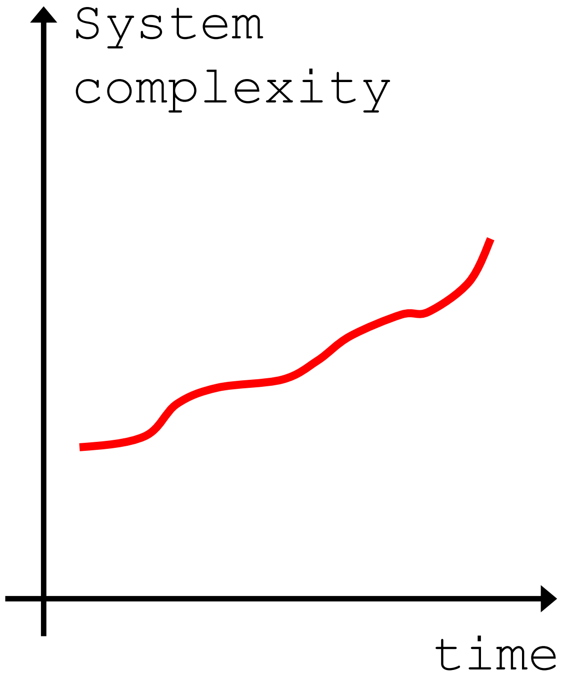
What do they do to our software?

How can we fight them?

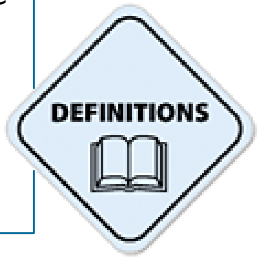
Complexity

Complexity





"**Complexity** is that property of an IT-system which makes it difficult to formulate its overall behaviour, even when given complete information about its parts and their relationships"



Complexity: Negative Impact on Software-Systems

Complexity

High difficulty to understand and document

Loss of conceptual integrity

Duplication of models, functionality and data

Inconsistent architecture

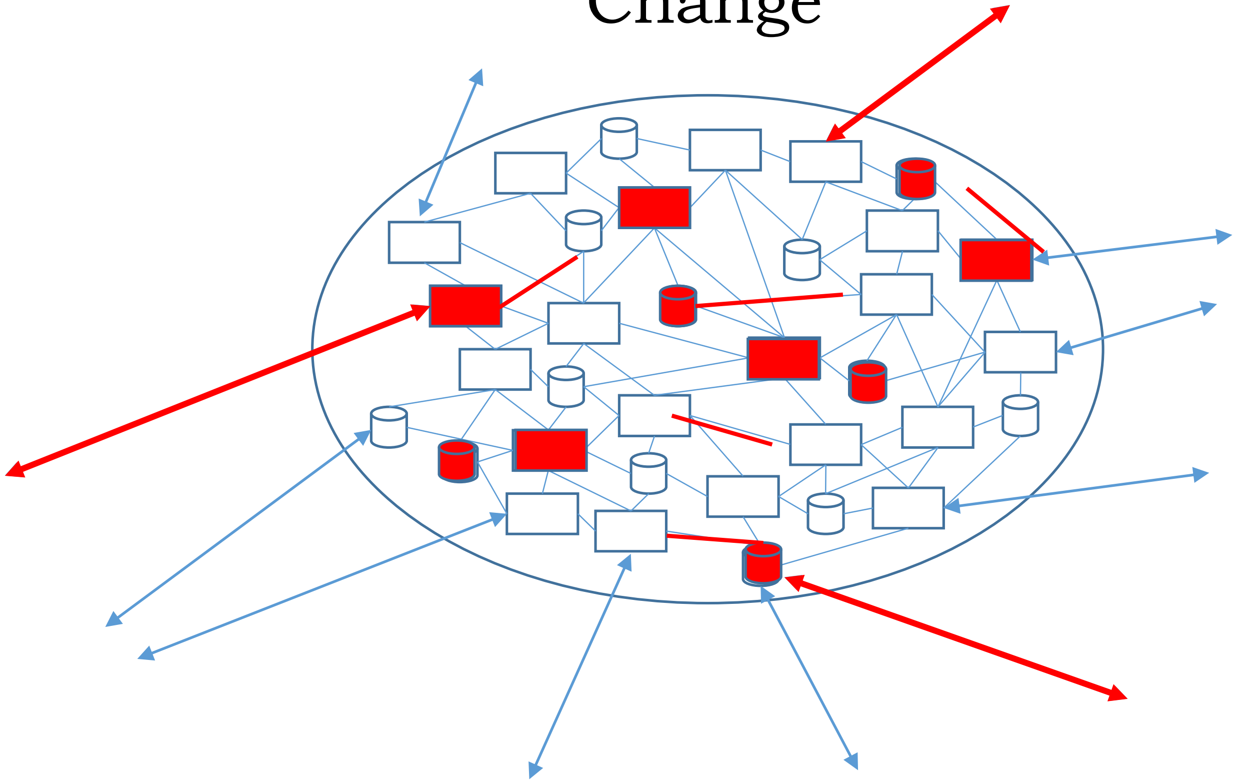
«Far»-effects: Changing one part may having unexpected effects in another part

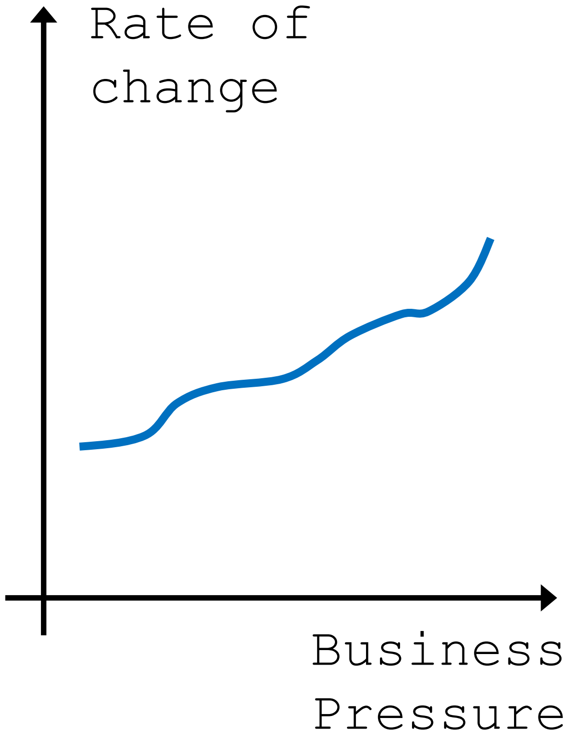
Emergence: The system develops unexpected poperties or behaviour

Complexity must be managed through the whole systems engineering process

Change

Change





"Continuous – sometimes disruptive – **change** forces relentless adaptation of the system to new requirements, to changes in the environment and to technological progress"

DEFINITIONS

Change: Negative Impact on Software-Systems

Change

High intricacy to coordinate and balance

Uncoordinated projects

Redundancy in req's, spec's and implementation artefacts

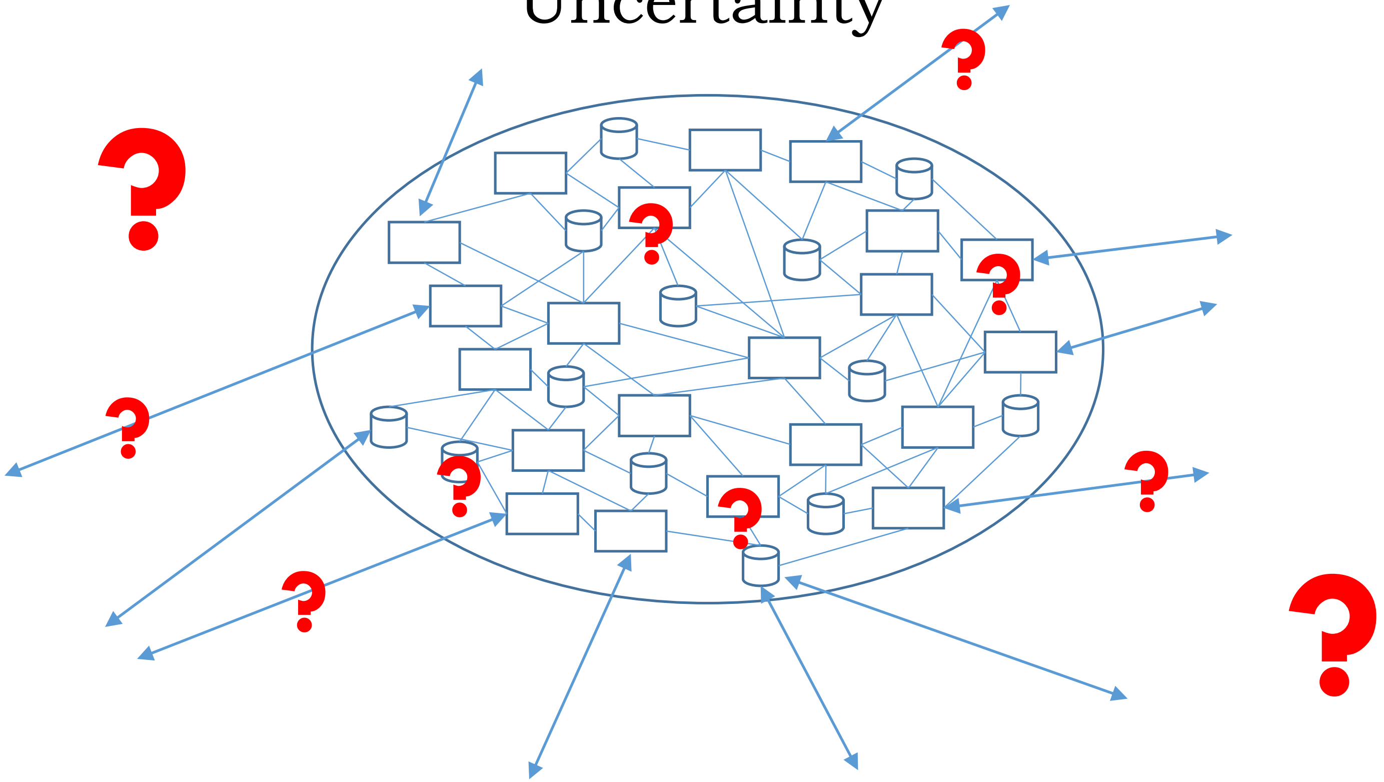
Architecture erosion

Accumulation of technical debt

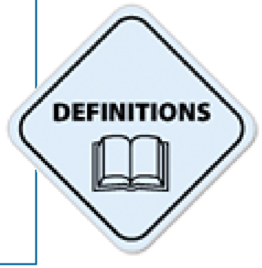
Conflicting req's, spec's and implementations

Change must be organized and coordinated

Uncertainty



"**Uncertainty** – both during development and during operation – forces weakly founded decisions with possibly far-reaching consequences"



Uncertainty: Negative Impact on Software-Systems

Uncertainty

Unknown or unforeseen impacts or effects

↓

Unfounded or inadequate decisions

Badly adjusted implementations

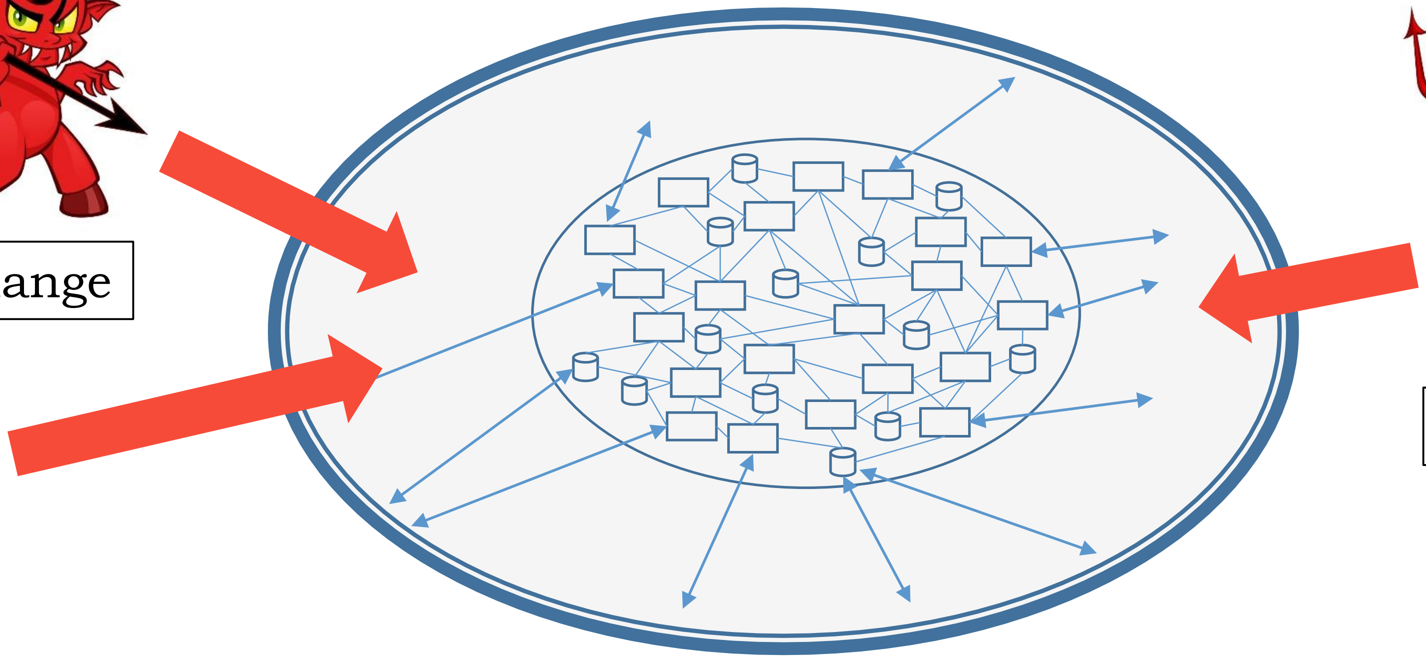
Unanticipated risks or hazards

Unprepared disasters and catastrophes

Sudden changes in markets, operating environment or user behaviour

Uncertainty must be assessed, risk-mitigated, and tracked

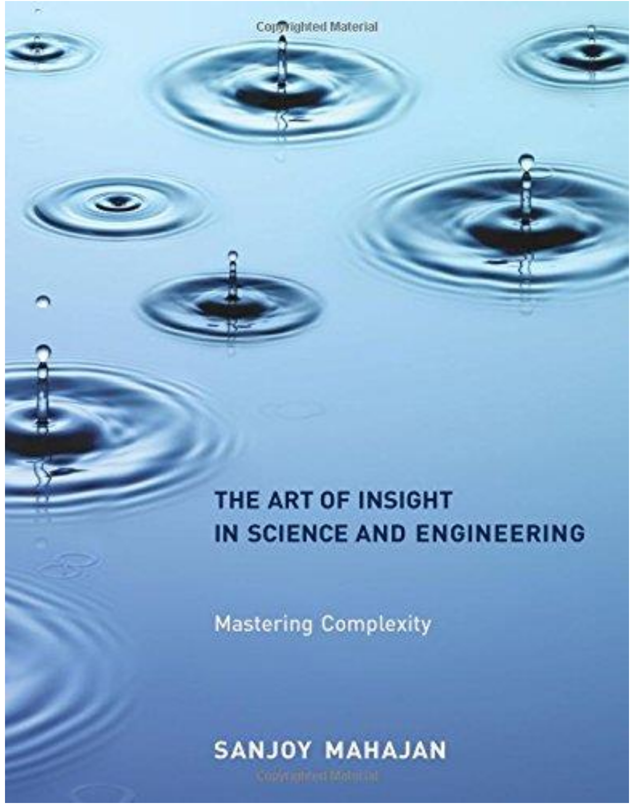
How can we successfully fight them?



… you cannot fight complexity, change and uncertainty
⇒ **You can only manage it !**

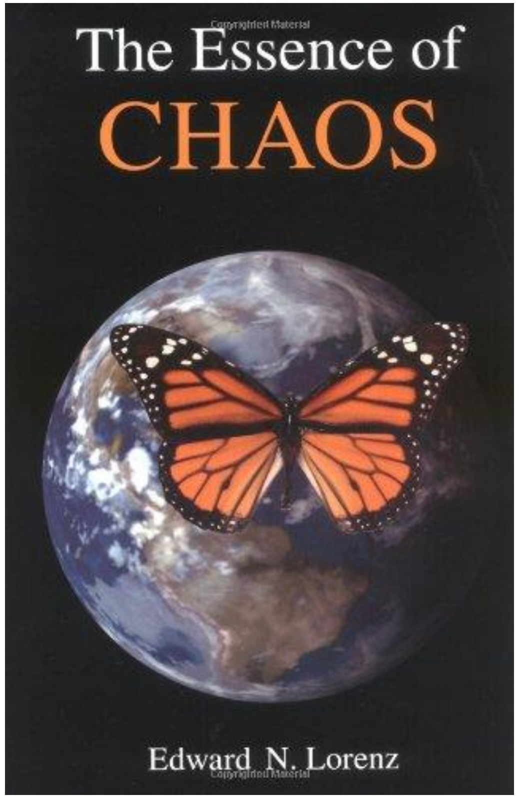
… by using principles, methods, strategies, and processes for **future-proof software-systems**

Textbook



Sanjoy Mahajan:
The Art of Insight in Science and Engineering – *Mastering Complexity*
The MIT Press, Cambridge, USA, 2014. ISBN 978-0-262-52654-8

Textbook



Edward N. Lorenz:
The Essence of Chaos
Jessie & John Danz Lectures (Reprint), 1996.
ISBN 978-0-2959-7514-6

Future-Proof Software-Systems

http://photos.gograph.com

To build and operate **dependable** software

… *«The software does what it should do, and does not what it should not do»*

To generate **business value** for its owner (and the community)

… *«The software industry is today one of the largest industries in the world»*

To maintain a high **changeability** of the software:

… *«The software must efficiently be adaptable to new requirements»*

… this is the fundamental objective of future-proof software-systems engineering

Future-Proof Software-Systems engineering
– and generally modern software development –
is strongly based on **semi-formal** and **formal methods**

Formal methods used in developing computer systems are *mathematically based techniques* for describing system behaviour and system properties. Such formal methods provide frameworks within which people can specify, develop, and verify systems in a *systematic*, rather than ad hoc, manner

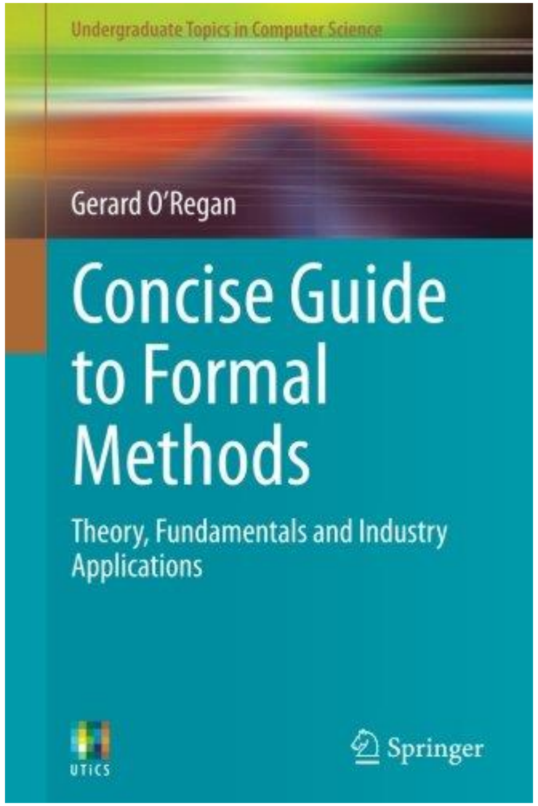
Encyclopedia of Software Engineering, 2nd edition, 2002

www.springer.com

Using semi-formal and formal methods means (during the architecting phase):

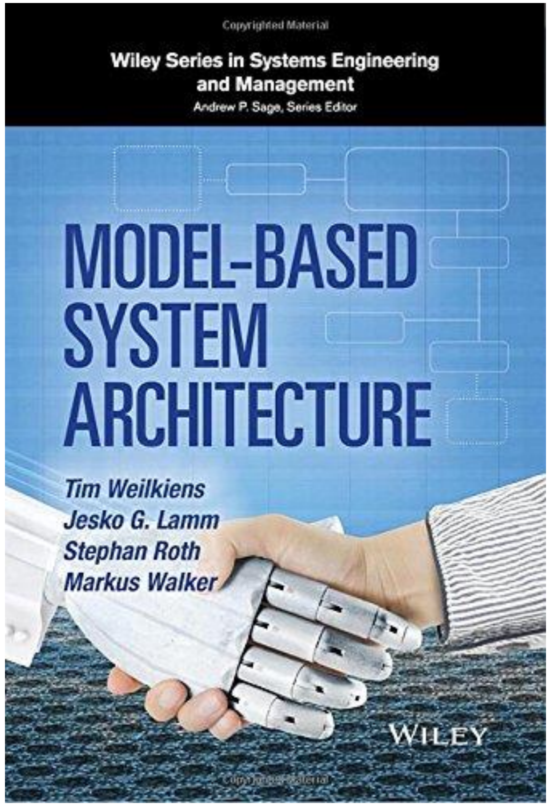
- Precise definitions
- Adequate models
- Strong, enforcable principles
- Proven patterns
- Reliable industry standards
- Time-tested reference architectures
- Established frameworks
- Architecture Description Languages

Textbook

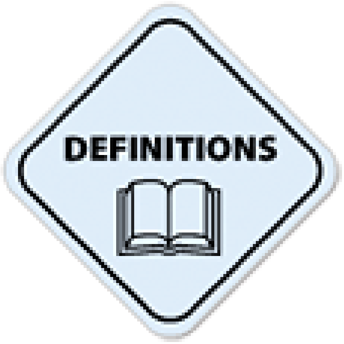


Gerard O'Regan:
Concise Guide to Formal Methods – *Theory, Fundamentals and Industry Applications*
Springer Verlag, 2017. ISBN 978-3-319-64020-4

Textbook



Tim Weilkiens, Jesko G. Lamm, Stephan Roth, Markus Walker:
Model-Based System Architecture
John Wiley & Sons, Inc., USA, 2016. ISBN 978-1-118-89364-7



http://www.iconshut.com

A *future-proof software-system* is a structure
that enables the management
of complexity, change and uncertainty
with the least effort,
with acceptable risk,
and with specified quality properties

Activity: Steering the development & evolution
→ **Strategy**

Parts of the system and their relationsships
→ **Architecture**

A *future-proof software system* is a structure
that enables the management
of complexity, change and uncertainty
with the least effort,
with acceptable risk,
and with specified quality properties

Acceptable probability for undesired effects and consequences
→ **Dependability**

Assuring the desired non-functional properties
→ **„Fit for Purpose“**

Best value for the parameters ‘money‘ and ‘time-to-market‘
→ **Changeability**

Managed Evolution Strategy

Architecture Principles

A *future-proof software-system* is a structure
that enables the management
of complexity, change and uncertainty
with the least effort, with acceptable risk, and with specified quality properties

Dependability

Changeability

Domain-specific
Quality Properties

Our objective is:

to build, evolve, and maintain

long-lived, mission-critical IT-systems

with a strong dependability,

an easy changeability

and a high business value.

http://www.gettyimages.ch

FPSS Properties

Primary Characteristics:
- Business Value
- Changeability
- Dependability

Secondary Characteristics (Domain-specific)**:**
- Non-functional properties:
  - Performance, Real-time, …
  - Hardware Resource Consumption
  - Adherence to industry-standards
  - etc.

What are the characteristics of Future-Proof Software-Systems?

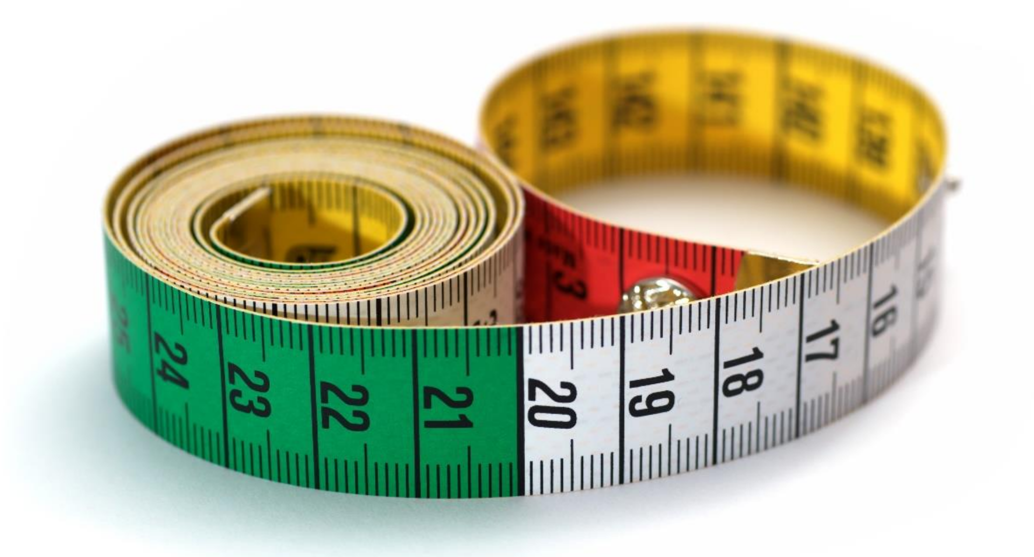
Primary Characteristics:

- Business Value
- Changeability
- Dependability

Definition
Metric
Example
Importance

If it can't be expressed in figures, it is not science; it is opinion

Robert Heinlein (1973)



Business Value

Business Value: Definition

http://blog.enfocussolutions.com

Business Value (of a software development) =

The opportunity to gain an *advantage* for the business

- Financial advantage (earnings), but also:
- Cost avoidance
- Competitive advantage (innovative functionality),
- Compliance to laws and regulations,
- Process improvements
- etc.

Business Value: Metric

Metric: **NPV** (Net Present Value)

$$\mathbf{NPV} = \sum_{n} \frac{\text{Benefit}_{\text{year-}\mathbf{n}}}{(1 + i)^{\mathbf{n}}} - I$$

NPV = Net Present Value(€)
I = Investment(€)
i = Yearly interest rate (%)
n = year (n=0: Project start)

NPV is the most common formula for calculating business value. It comes from business-economics.

Business Value: Example

Business Value = Net Present Value (NPV)


http://www.eco-way.ch/?p=10846

Investment:
- 860‘000.- €

√ NPV = +165‘000 €

Earnings:	Year1	Year2	Year3	Year5	Year6
	240‘000 €	270‘000 €	230‘000 €	280‘000 €	300‘000 €
8 %/year:	$(1+0.08)^{-1}$	$(1+0.08)^{-2}$	$(1+0.08)^{-3}$	$(1+0.08)^{-4}$	$(1+0.08)^{-5}$
	1.08	**1.17**	**1.26**	**1.36**	**1.47**

Discount Factor

+ 222‘000 €
+ 230‘000 €
+ 182‘000 €
+ 205‘000 €
+ 186‘000 €
+ 1‘025‘000 €

Changeability

A *future-proof software-system* is a structure

that enables the management

of complexity, change and uncertainty

with the least effort, with acceptable risk, and with specified quality properties

Changeability

Changeability: Definition

Changeability =

The *capability* to develop and introduce new features with:

- short time-to-market
- reasonable development cost

Important note: This capability is a property of an *organization*, but is heavily based on a good, evolvable *structure* of the system

Architecture

Changeability: Metric

Size

Functionality with:

TtM

- short time-to-market

DevC

- reasonable development cost

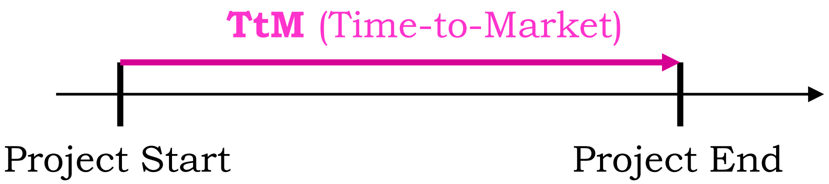
Metric Idea: **Changeability ~ Size^2/(TtM*DevC)**

Changeability: Metric

Amount of functionality: Functional Size

Size Unit: #UCP or #FP

Functionality produced



Unit: days (d)

Project Duration

Unit: k€

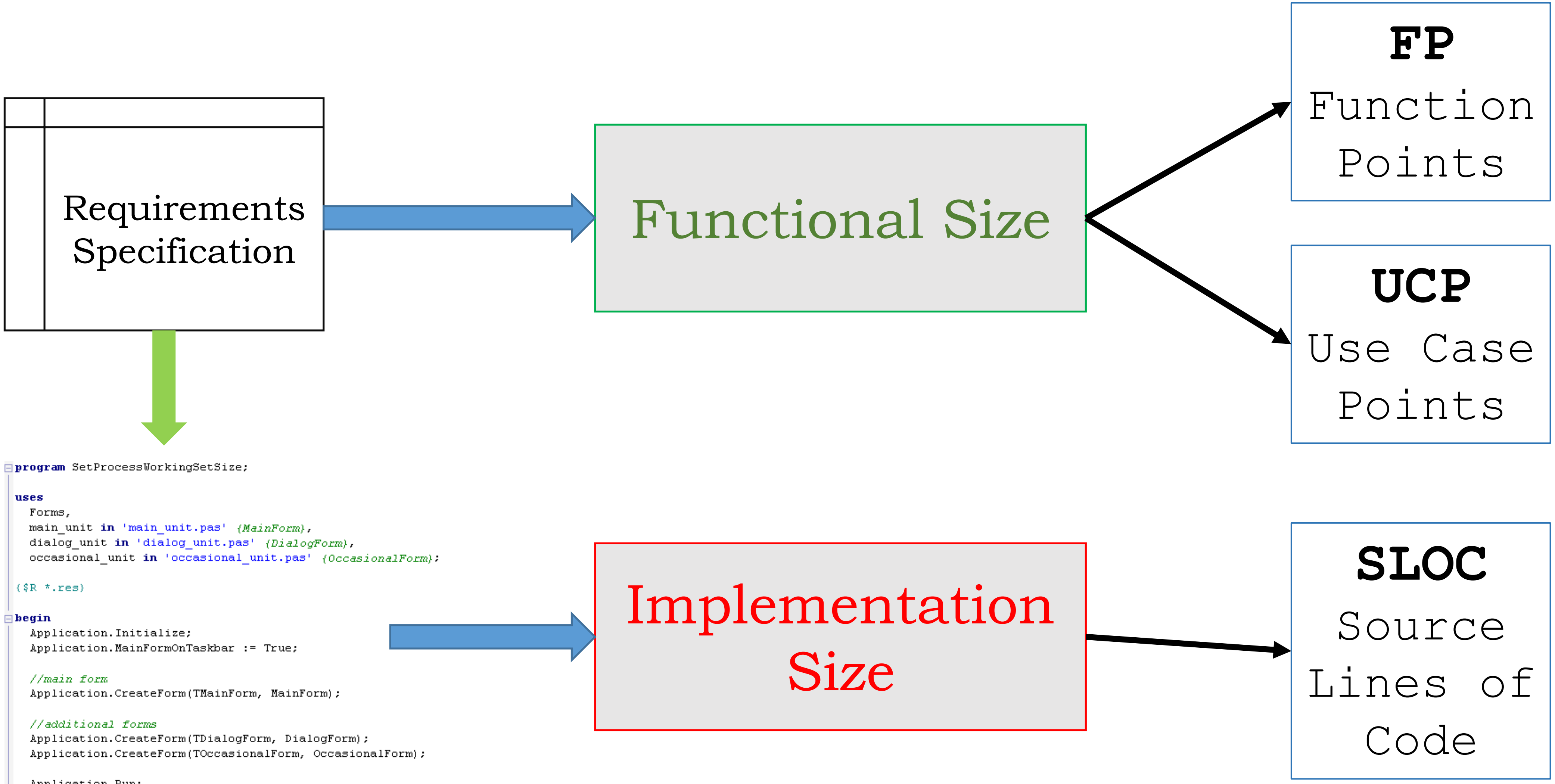
Project Cost

$$\textbf{Changeability} = \frac{\Sigma TtM_i * \Sigma DevC_i}{(\Sigma Size_i)^2}$$

Unit: (days*k€)/#UCP²

Metric for Changeability

Clarification: **Software Size**

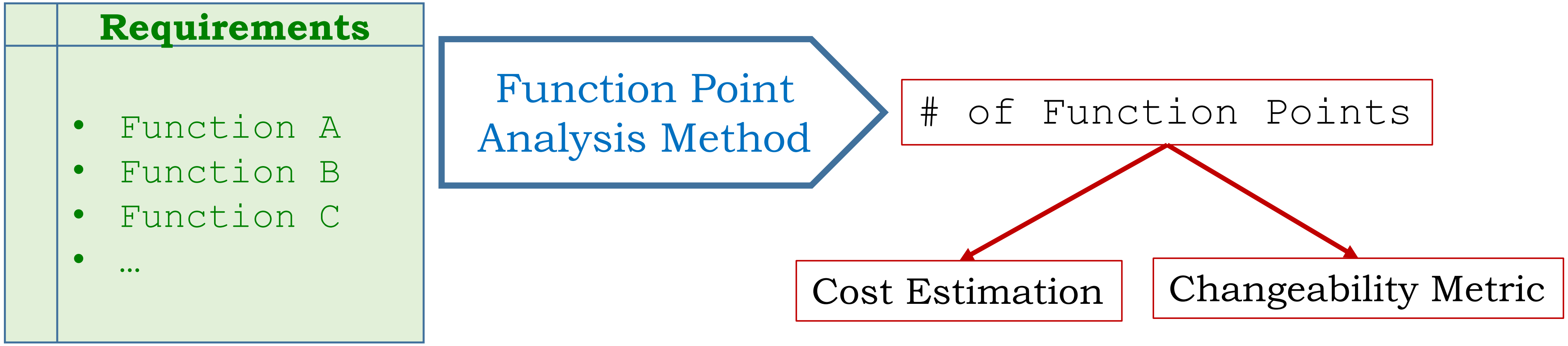



```
program SetProcessWorkingSetSize;

uses
  Forms,
  main_unit in 'main_unit.pas' {MainForm},
  dialog_unit in 'dialog_unit.pas' {DialogForm},
  occasional_unit in 'occasional_unit.pas' {OccasionalForm};

{$R *.res}

begin
  Application.Initialize;
  Application.MainFormOnTaskbar := True;

  //main form
  Application.CreateForm(TMainForm, MainForm);

  //additional forms
  Application.CreateForm(TDialogForm, DialogForm);
  Application.CreateForm(TOccasionalForm, OccasionalForm);

  Application.Run;
end.
```

Clarification: **Function Points** (FP)

FP Definition:

A <u>function point</u> is a unit of measurement to express the amount of **business functionality** an information system provides to its users (https://en.wikipedia.org/wiki/Function_point)



- David Garmus, David Herron: **Function Point Analysis – *Measurement Practices for Successful Software Projects***. Addison-Wesley, Boston, USA, 2001. ISBN 978-0-201-69944-3
- IFPUG: International Function Point Users Group (http://www.ifpug.org)

Clarification: **Use Case Points** (UCP)

DEFINITIONS

UCP Definition:

Use Case Points (UCP) is an estimation method that provides the ability to estimate an **application's size** and effort from its use cases (http://www.codeproject.com/Articles/9913/Project-Estimation-with-Use-Case-Points)

Use Cases

UCP = TCF * ECF * UUCP * PF

1. Technical Complexity Factor (TCF).
2. Environment Complexity Factor (ECF).
3. Unadjusted Use Case Points (UUCP).
4. Productivity Factor (PF).

Roy Clem: ***Project Estimation with Use Case Points***. Code Project, 22 March 2005
http://www.codeproject.com/Articles/9913/Project-Estimation-with-Use-Case-Points

Changeability: Example

Project data

Project	Size (#UCP)	TtM_i (days)	$DevC_i$ (k€)	End Date
P_1	1'200	900	5'600	Jan 2012
P_2	650	645	2'566	Jan 2012
P_3	4'400	5'280	27'270	March 2012
P_4	980	620	5'400	April 2012
P_5	11'250	6'600	75'600	April 2012
P_6	2'300	1'900	13'900	June 2012
P_7	800	390	6'200	August 2012
P_8	1'850	1'250	13'200	August 2012
etc.	…	…	…	…

measurement period

CREDIT SUISSE values:

~ 4.2 k€/UCP

~ 0.8 days/UCP

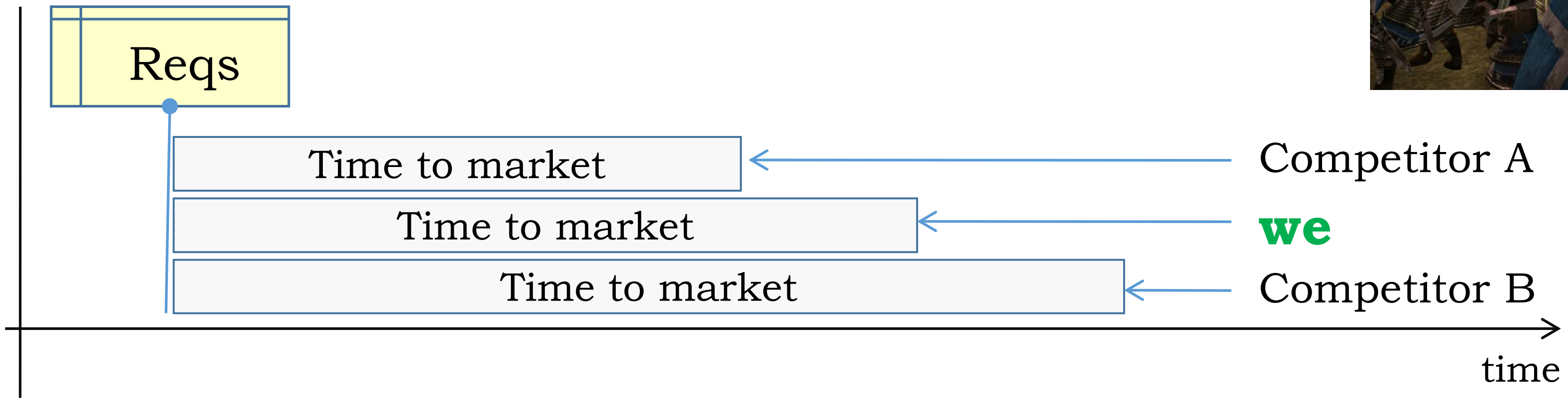
[Murer/Bonati/Furrer ISBN 978-3-642-01632-5]

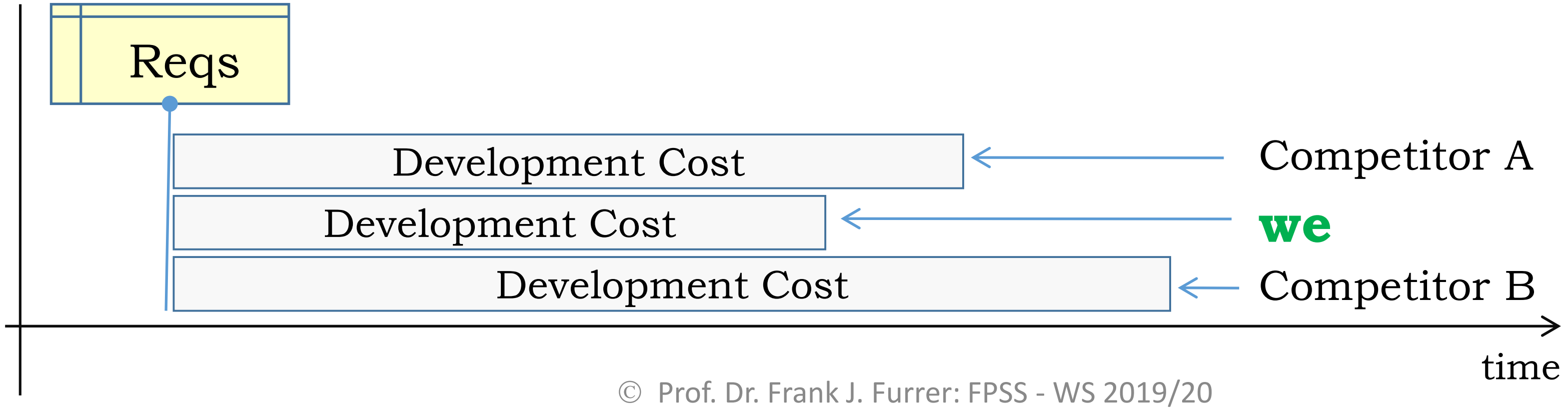
$$\text{Changeability} = \frac{\sum TtM_i * \sum DevC_i}{(\sum Size_i)^2}$$

Unit: (days*k€)/#UCP2

Changeability: Importance

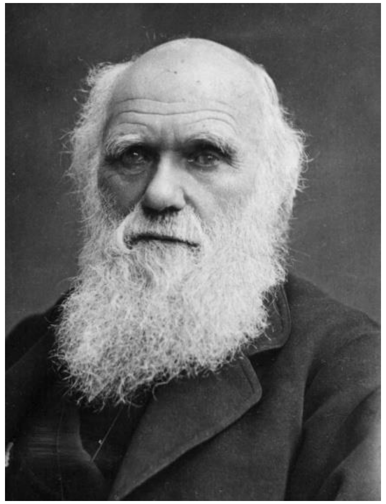
Why is *changeability* so important?





Changeability: Importance

Why is *changeability* so important?



"It is not the strongest of the species that survives,
nor the most intelligent that survives.
It is the one that is the most adaptable to change."

Charles Darwin: The Origin of Species (1859)

Today: «most adaptable to change» applies to software-systems and the companies which live from them

Changeability: Importance

Why is *changeability* so important?

✓ Changeability impacts **every** project

- Low changeability: (all) projects are late and expensive
  = **high** resistance to change ← bad!
- High changeability: (all) projects are in time and cost-efficient
  = **low** resistance to change ← good!

✓ High changeability allows to use the company resources more efficient

✓ Changeability is an important competitive market factor

A *future-proof software-system* is a structure

that enables the management

of complexity, change and uncertainty

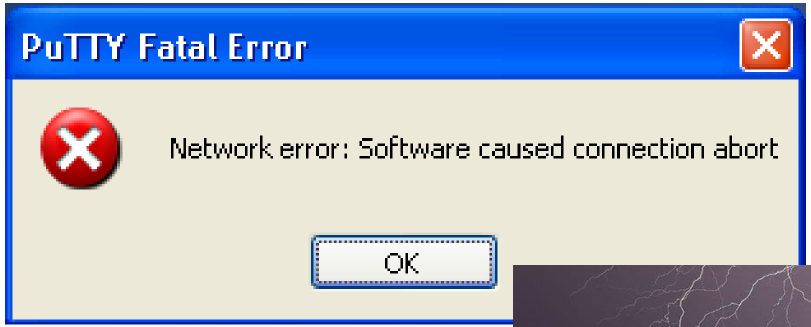
with the least effort, with acceptable risk, and with specified quality properties

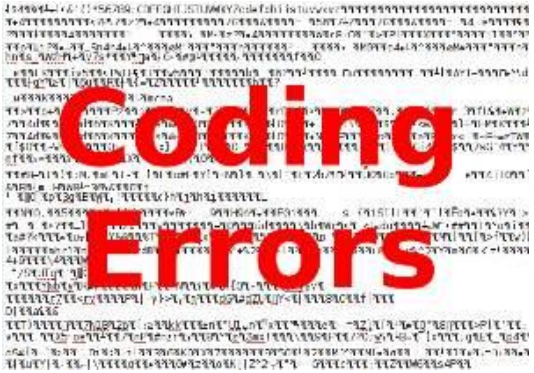
Dependability

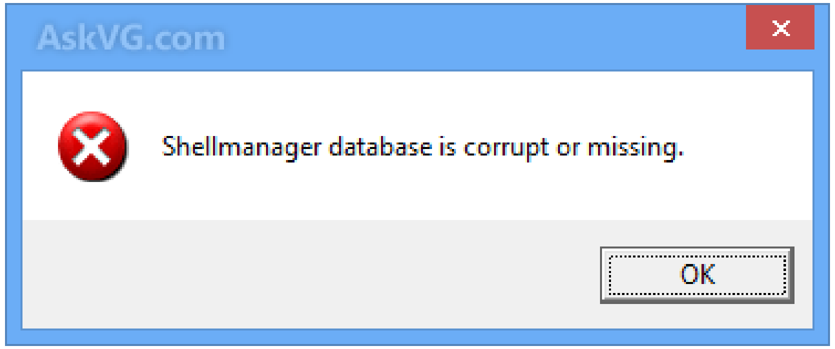
Dependability

Why is *dependability* very important for future-proof software?

⇒ The world has become a dangerous place for software





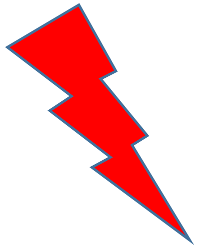


Dependability Basics

https://www.beritateknologi.com

Disruption

Cause



Incident

System & Environment

Impact

Instrument Landing System Failure

https://upload.wikimedia.org

Consequences

Effect

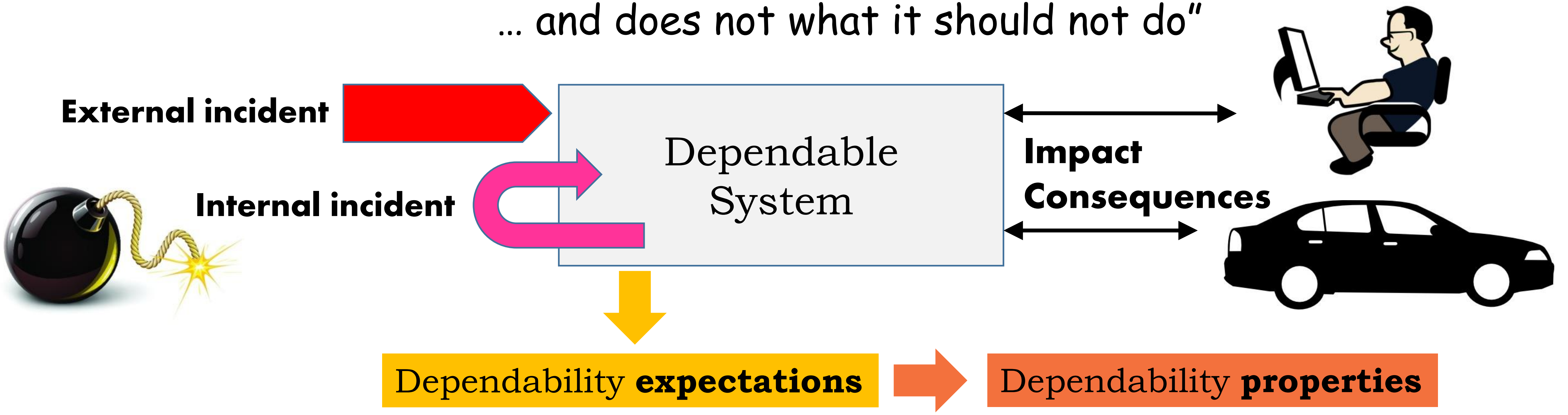
http://www.403wg.afrc.af.mil

DEFINITIONS

Dependable System

"Dependability" refers to the user's ability to depend on a system in its intended environment, with its intended use, as well as when these assumptions are violated or external events cause disruptions.

"The software does what it should do,
… and does not what it should not do"



Dependability **expectations**

Examples

e-banking system:

- *security* (= defense against hackers)
- *integrity* (= don't digitally lose my money)
- *confidentiality* (= "it's my business")
- *availability* (= 24 h/7 days).

Dependability expectations
=
Application domain

Car:

- *safety* (= no accidents)
- *security* (= no hostile influence)
- *reliability* (= no engine failures on the motorway)
- *conformance* to all laws and regulations

https://www.tylerandmimiford.com

Dependability

http://www.swintoncounseling.com

Application Domain

Resilience

General Principles
⇒ valid for **all** dependable systems

User expectations

Safe to use
Efficient
Frienly
…

Community expectations

Safe and secure
Clean operation
Energy-efficient
…

Legal & compliance expectations

Comply with laws
Adhere to regulations
Auditable
Certified
…

Dependability **expectations** → Dependability **properties**

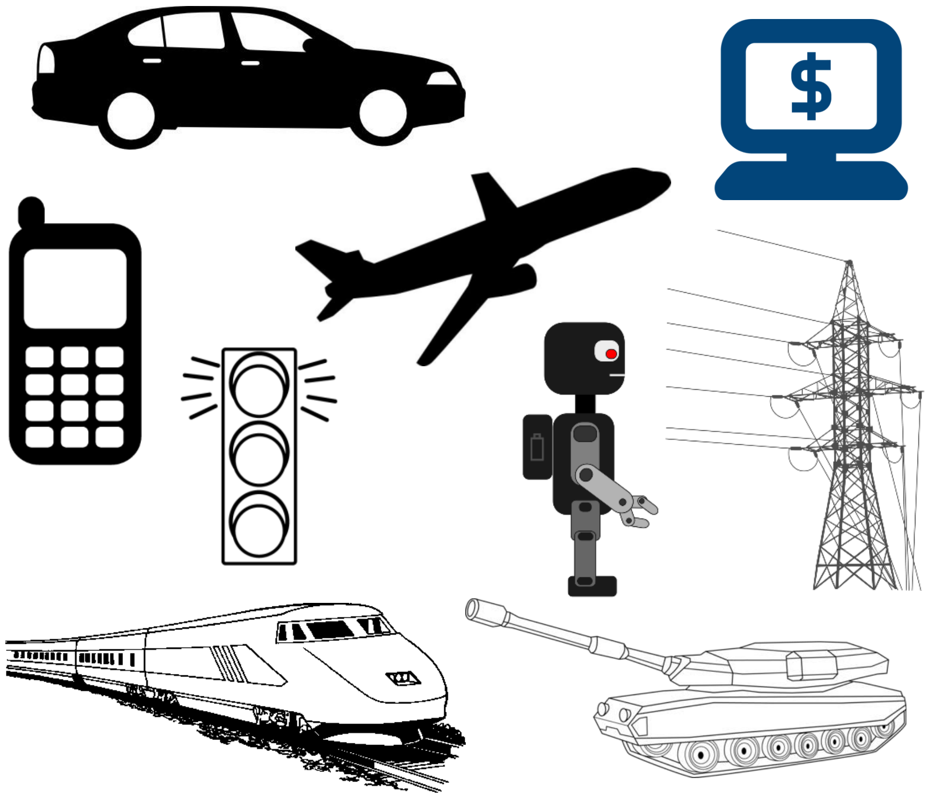
Resilience

General property

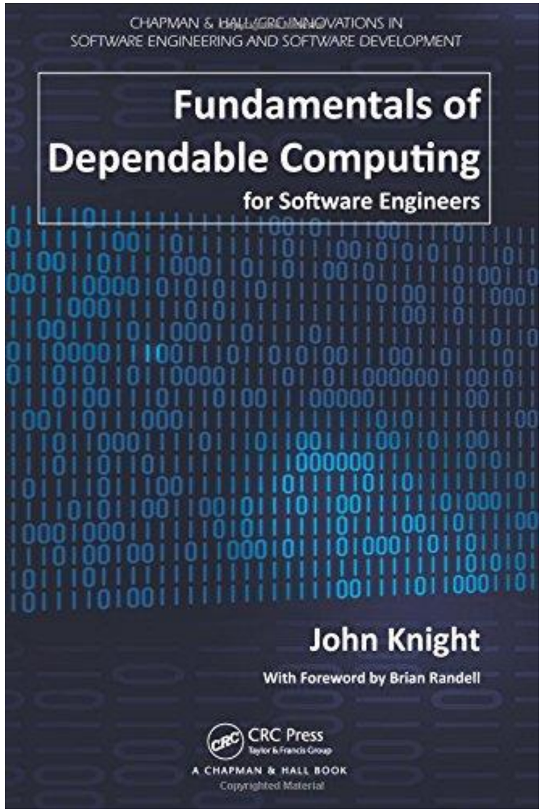
Specific properties

- **Safety**
- **Security**
- **Integrity**
- **Confidentiality**
- **Real-time capability**
- **…**

Application Domain

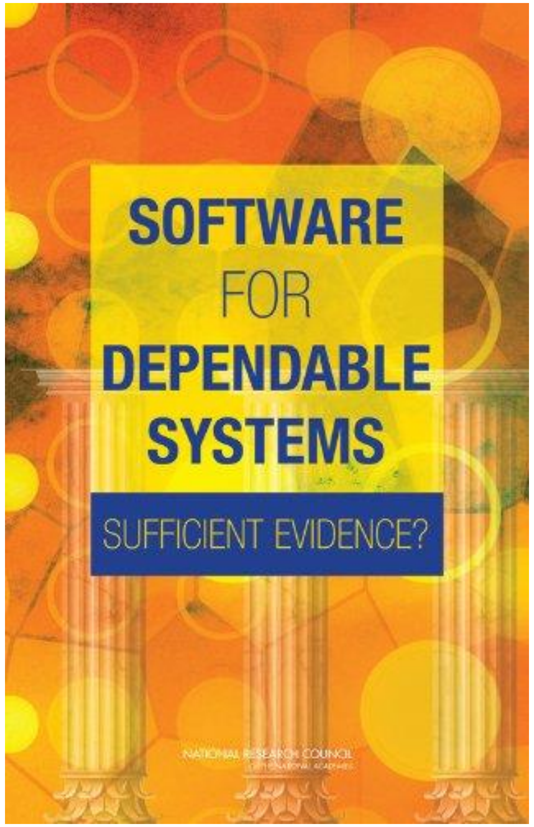


Textbook

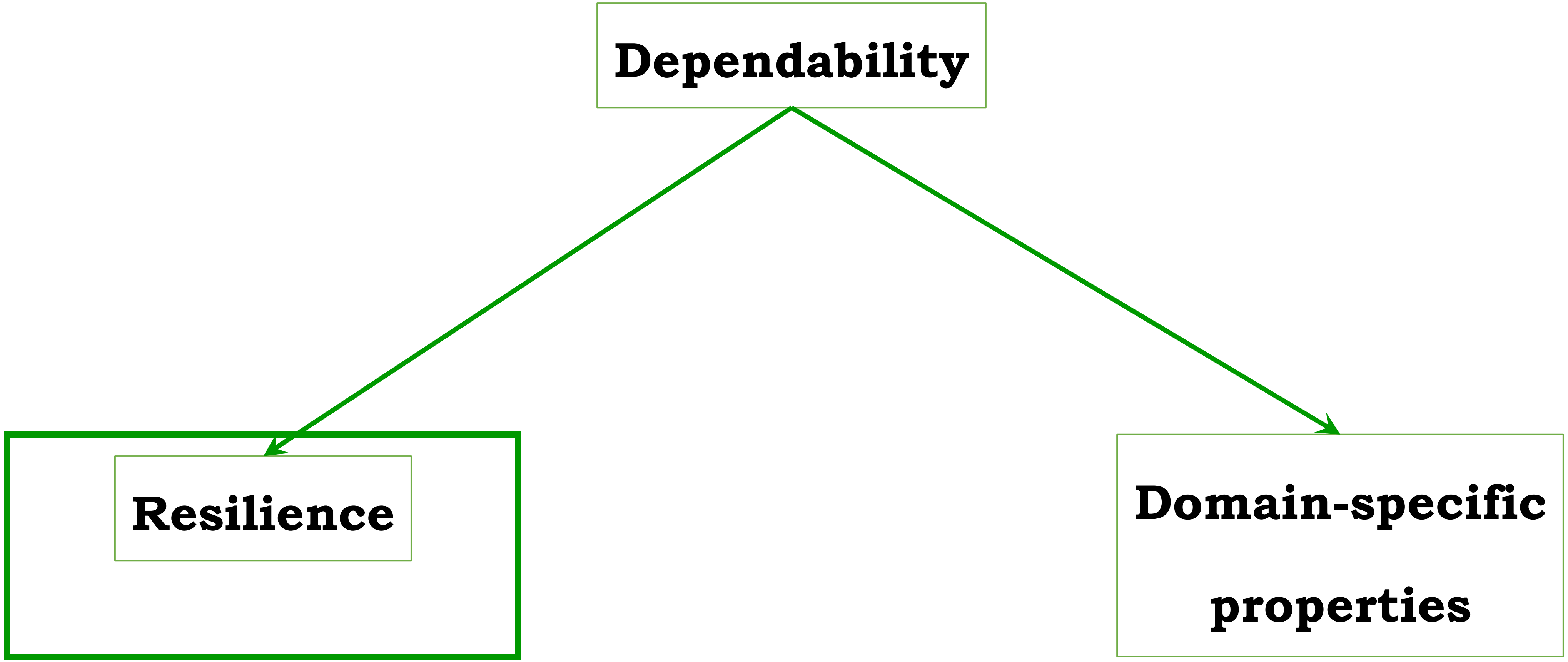


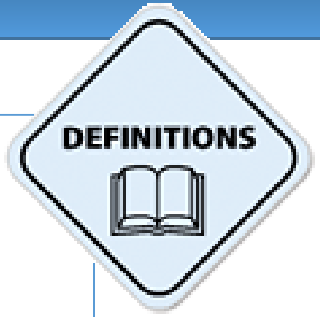
John Knight:
Fundamentals of Dependable Computing for Software Engineers
CRC Press (Taylor & Francis), USA, 2012
ISBN 978-1-4398-6255-1

Textbook



Daniel Jackson, Martyn Thomas, and Lynette I. Millett (Editors: **Software for Dependable Systems: Sufficient Evidence?**
U.S. National Academy Press, 2007. ISBN 978-0-309-10394-7 [https://www.nap.edu/download/11923]


Dependability
Resilience
Domain-specific properties



Resilience:

General Principles

⇒ valid for **all** dependable systems

Resilience is the *capability* of a system

- to *absorb* the **incident**,
- to *recover* to an acceptable level of performance,
- to *sustain* that level for an acceptable period of time

http://www.incose.org/practice/techactivities/wg/rswg/

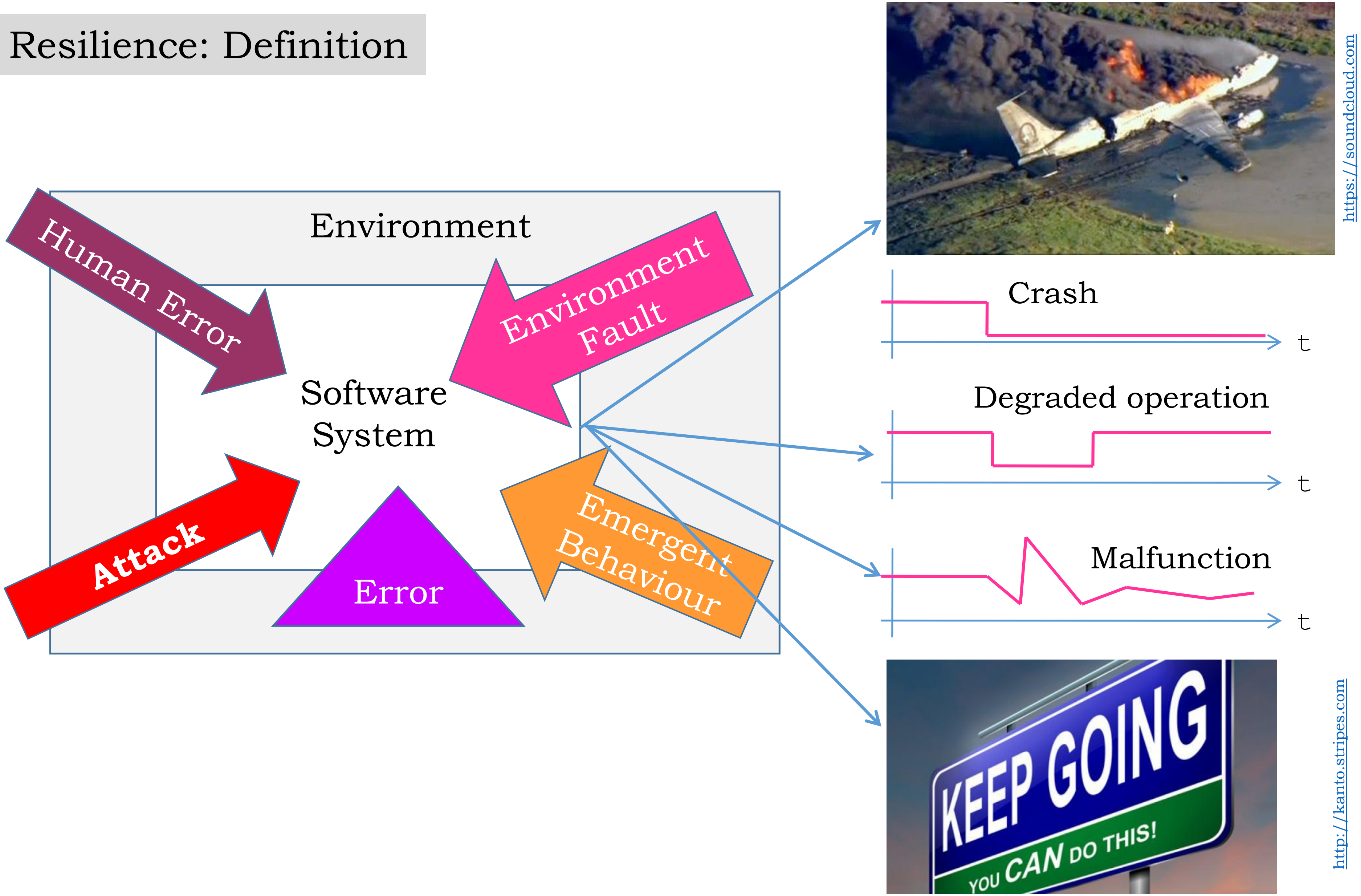
Engineering Profession: Resilience Engineer

Resilience Engineering Tasks:

- *Before* – Allows anticipation and corrective action to be considered
- *During* – How the system survives the impact of the disruption
- *After* – How the system recovers from the disruption

https://www.npmjs.com

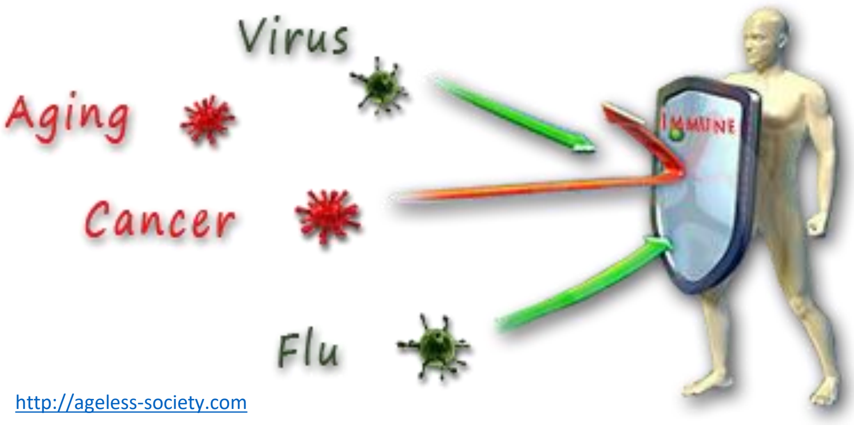
Resilience: Definition



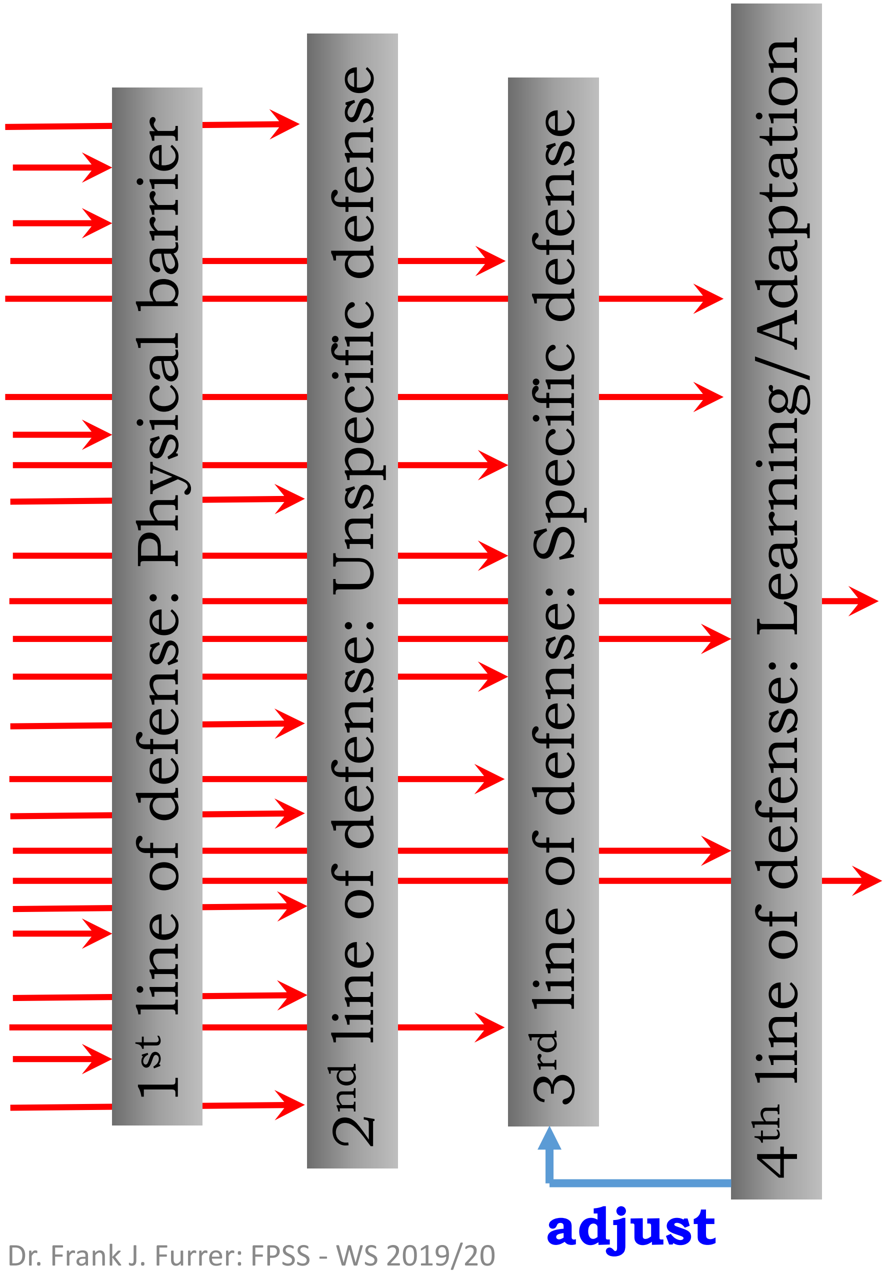
Worst Case

Best Case
(= Engineering objective)

Resilience Example: Human Immune System

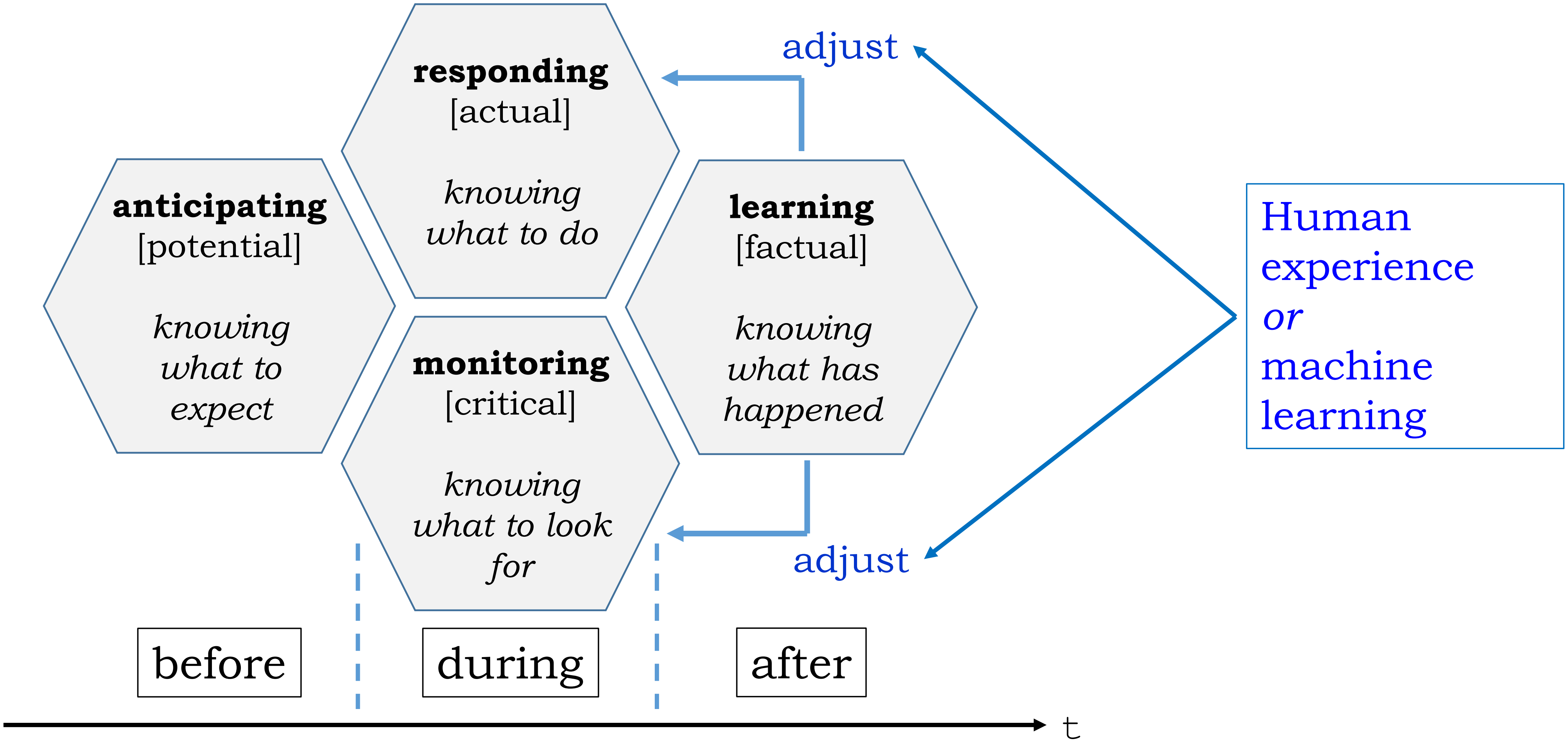


http://ageless-society.com



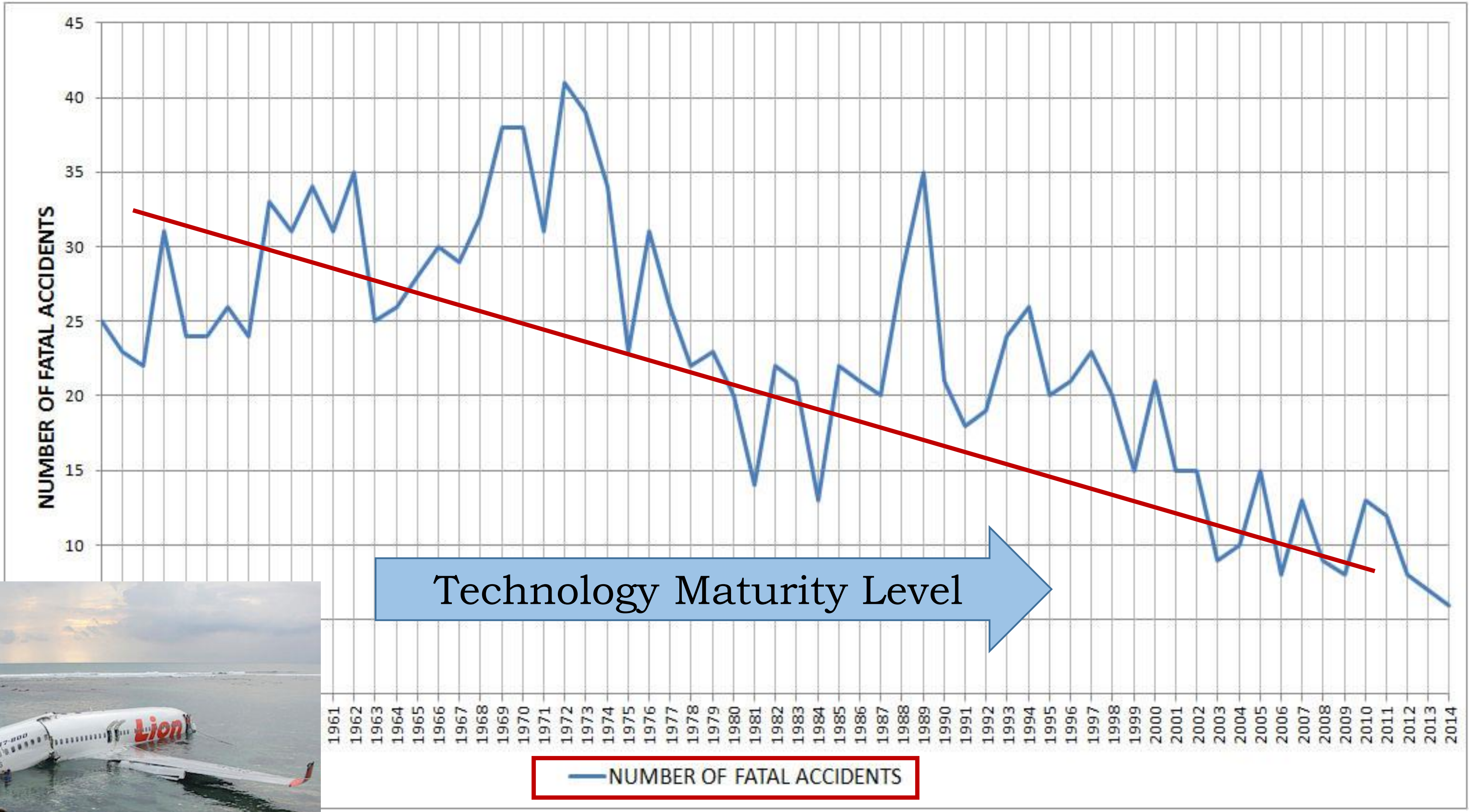
http://images.girlslife.com

The four cornerstones of resilience



Hollnagel, 2011, ISBN 978-1-4724-2074-9

Airplane Accident Learning Curve



Source: http://www.planecrashinfo.com/cause.htm [16.1.2017]

Dependability Metric

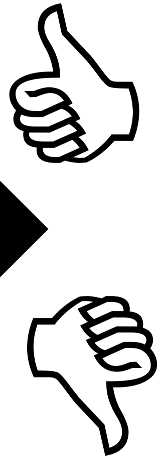
Resilience Metric

Metrics for *domain-specific* quality properties

- **Safety**
- **Security**
- **Integrity**
- **Confidentiality**
- **Real-time capability**
- **…**

Empirical Dependability Metric

Incident → System & Environment → **Consequences**



Damage **Potential** of the Incident:

- catastrophic
- critical
- severe
- marginal
- negligible

Response of the system

Resulting **Consequences**:

- catastrophic
- critical
- severe
- marginal
- negligible

Incident

Response of the system

Consequenes

Damage **Potential**:
- catastrophic: 5
- critical: 4
- severe: 3
- marginal: 2
- negligible: 1

Weight:
- Predicted *(= before the incident)*
- Actual *(= after the incident)*

Actual Damage:
- catastrophic: **5**
- critical: **4**
- severe: **3**
- marginal: **2**
- negligible: **1**

Assessment/Simulation **before** the incident

Actual damage **after** the incident

Resilience against 1 incident: φ_1 = [Potential – Damage]

Example:

Example 1: Damage potential = **2** (marginal)
Actual damage = **3** (severe)

Resilience = [Potential – Damage]: **2** – **3** = **-1**

Amplification

Example 2: Damage potential = **4** (critical)
Actual damage = 2 (marginal)]

Resilience = [Potential – Damage]: **4** – **2** = **+2**

Resilience

Empirical resilience metric:

System resilience over a time period τ:

$$\rho_\tau = \frac{1}{n} \sum [\text{Potential}_i - \text{Impact}_i];\ i = 1 \ldots n$$

n incidents in a time period **τ**

Scientific resilience metric: https://blogs.gwu.edu/seed/files/2012/07/Reliability-Engineering-and-System-Safety-2014-Francis-1y5jkh9.pdf

Example: Banking System Incidents

Date	Incident	Damage Potential	Damage	Impact		Remarks
4.1.13	DB2 Database Crash	4	Operational blackout for 3 hrs (Recovery time)	2	**4 – 2 = 2**	Save & recovery procedures worked well
6.1.13	Semnager Virus Infection	3	Small number of customers affected	2	**3 – 2 = 1**	Payment check procedures worked well
21.2.13	Crash of authentication servers	3	Employees could not access the IT system for 1 hour	1	**3 – 1= 2**	Backup/recovery mechanisms worked well
4.5.13	Fibre trunk cable damaged (by construction work)	4	No external communications for 5 hours	3	**4 – 3 = 1**	Emergency repair in time
9.12.13	Illegal financial transaction executed (fault in sanction filter)	3	Legal & compliance consequences	3	**3 – 3 = 0**	Sanction filter update process improved

System resilience over 5 incidents: $\rho_5 = \frac{1}{n} \sum \rho_i$ **= 1.20**

Why is *resilience* so important?

Software has an enormous impact on people and society:

- Functionality in all areas of life and work
- Tremendous business opportunities & risks
- etc.

Software failures may have grave consequences:

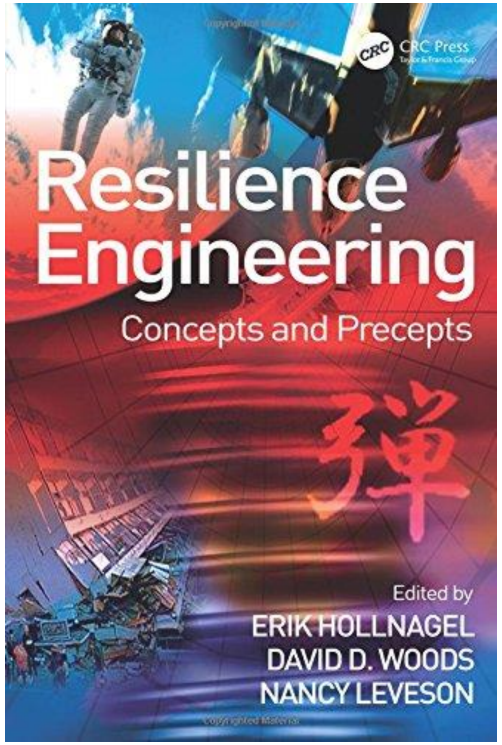
- Accidents in safety-critical systems (death, injury)
- Financial or reputation loss
- Legal & regulatory consequences
- Product liability cases
- etc.

Resilience must be planned and built-in

- **Not** added as an afterthought!

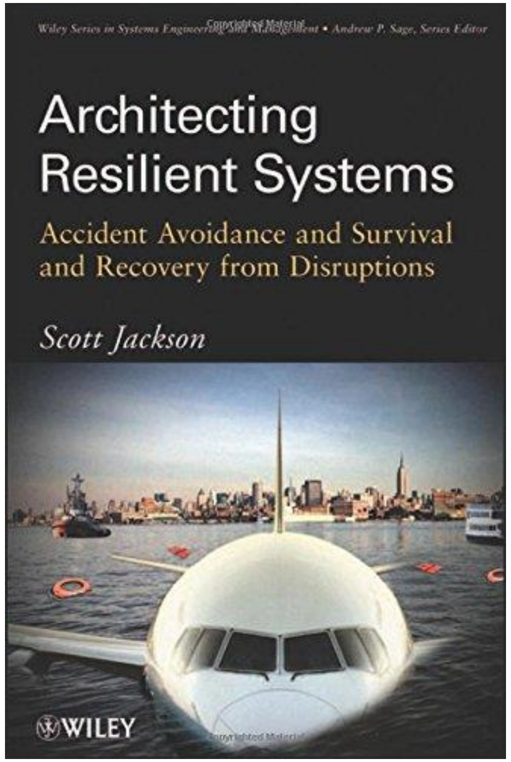
https://www.intivix.com

Textbook

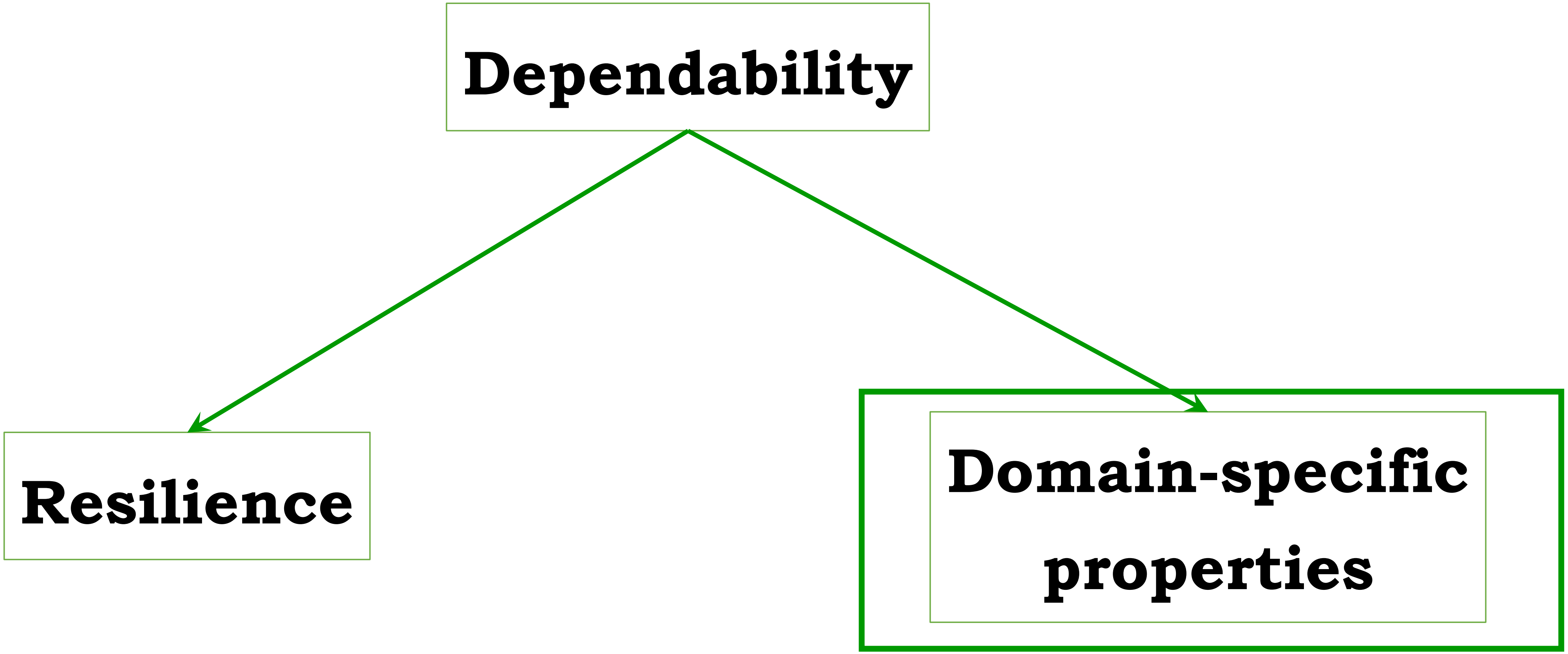


Erik Hollnagel, David D. Woods, Nancy Leveson (Editors):
Resilience Engineering – *Concepts and Precepts*
Ashgate Publishing Ltd., Aldershot, UK, 2006.
ISBN 978-0-7546-4904-5

Textbook



Scott Jackson:
Architecting Resilient Systems – *Accident Avoidance and Survival and Recovery from Disruptions*
John Wiley & Sons, Inc., New Jersey, USA, 2010. ISBN 978-0-470-40503-1


Dependability
Resilience
Domain-specific properties

Dependability **expectations** → Dependability **properties**

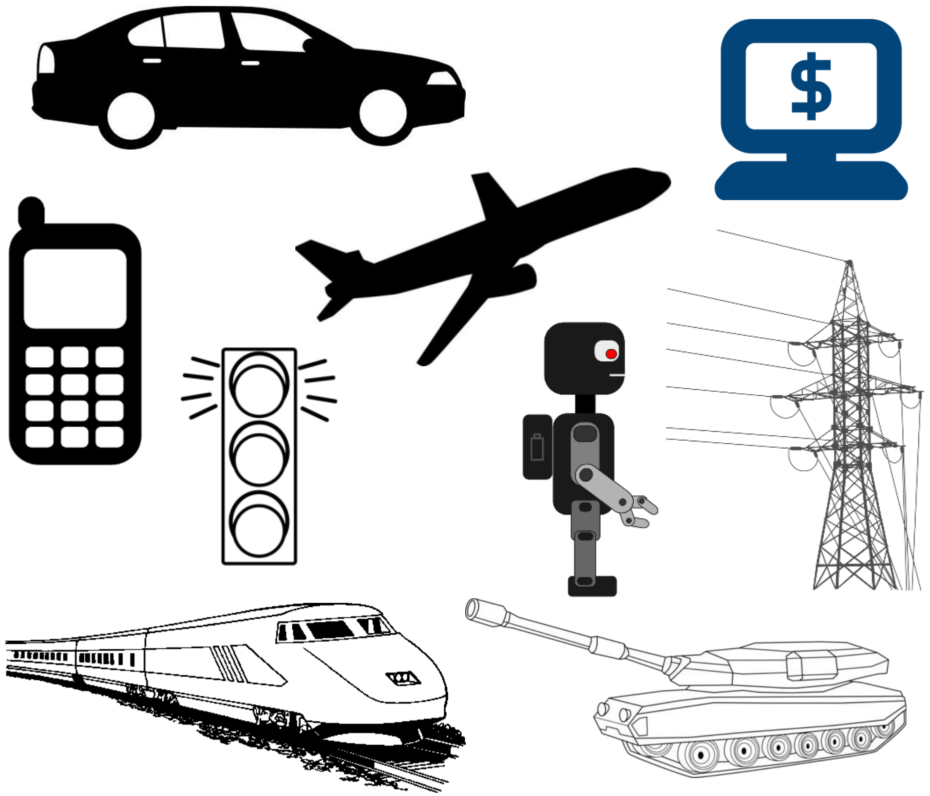
Resilience

General property

Specific properties

- **Safety**
- **Security**
- **Integrity**
- **Confidentiality**
- **Real-time capability**
- **...**

Application Domain



Safety

DEFINITIONS

Safety is the sustainable state in which the risk of harm to people, organizations or property is maintained below an *acceptable level*

Accident

Attack

Failure

Malfunction

Metrics

Company Management

http://clipart-library.com

- **Safety**
- **Security**
- **Integrity**
- **Confidentiality**
- **Real-time capability**
- **...**

Metrics for *domain-specific* quality properties

Company-specific set of **safety** metrics

Certification Authorities

Joint Aviation Authorities · Europe · JAA

Military Aviation Authority MAA

Federal Aviation Administration

EASA European Aviation Safety Agency

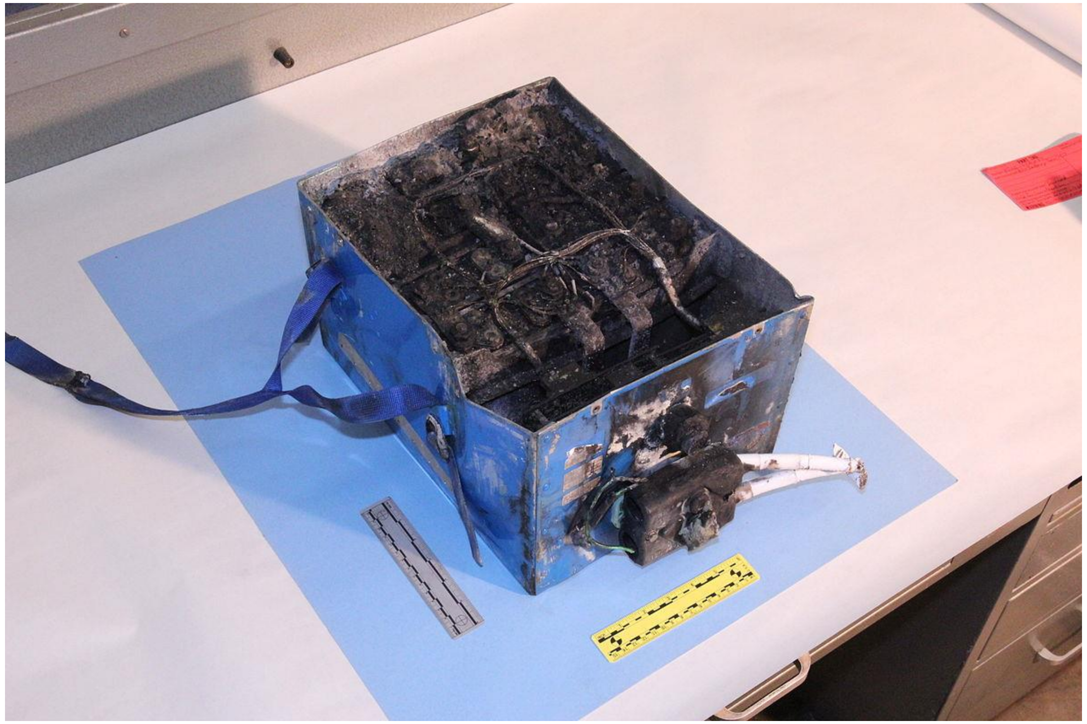
http://bigcashblog.com

Domain Experts

Example: Aircraft Safety Incident

In the Boeing 787 Dreamliner's first year of service (2014), at least four aircraft suffered from electrical system problems stemming from its **lithium-ion batteries**

http://www.boeing.com



https://upload.wikimedia.org

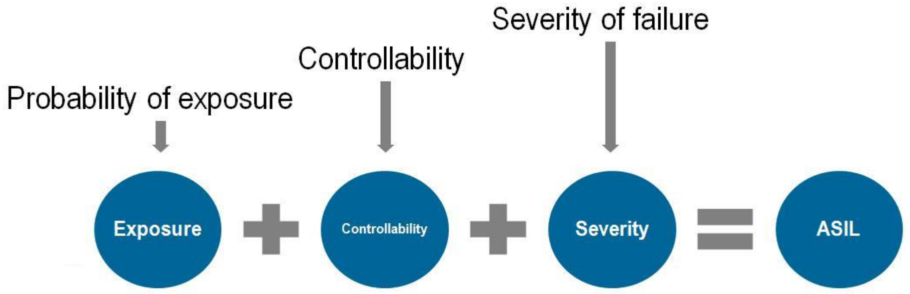
The Lithium batteries caused:

- An **electrical fire** aboard an All Nippon Airways 787
- A similar **fire** found by maintenance workers on a landed Japan Airlines 787 at Boston's Logan International Airport

Safety Metric Example: Automotive Safety Integrity Level (ASIL)

Automotive Safety Integrity Level (**ASIL**) is a risk classification scheme defined by the ISO 26262 (Functional Safety for Road Vehicles). ASIL is established by performing a ***risk analysis*** of a potential hazard by looking at the Severity, Exposure and Controllability of the **vehicle operating scenario**

http://www.ni.com

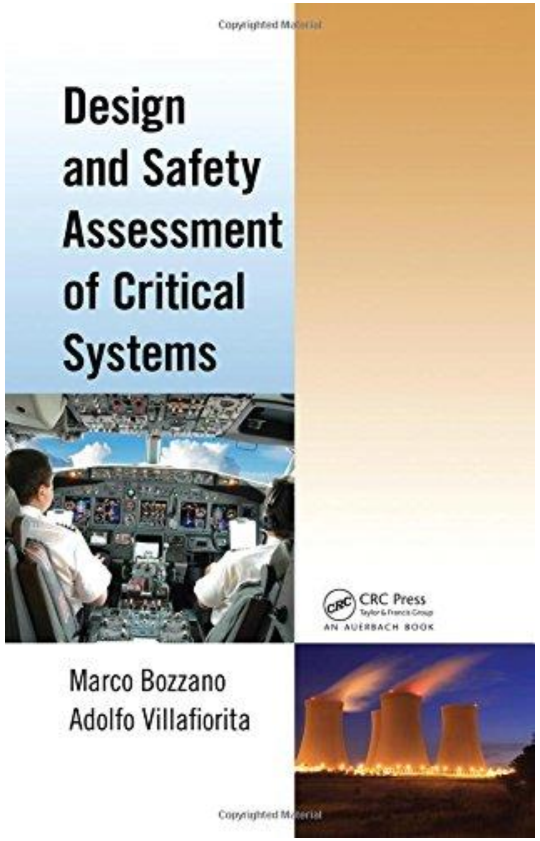


http://www.dailyecho.co.uk

ASIL	Impact of Failure	Controllability	Exposure	In-Car Examples
A	Slight injury	Normally controllable	High probability	• Lag in display from rear-view camera
B	Severe injury	Normally controllable	High probability	• Failure of collision avoidance tone
C	Fatal/Survival uncertain	Difficult to control	Medium Probability	• Anti-Lock Braking system wheel lock-up • Out-of-control automatic transmission
D	Fatal / survival uncertain	Difficult to control	High Probability	• Steering-control lock-up • Airbag deployment while driving

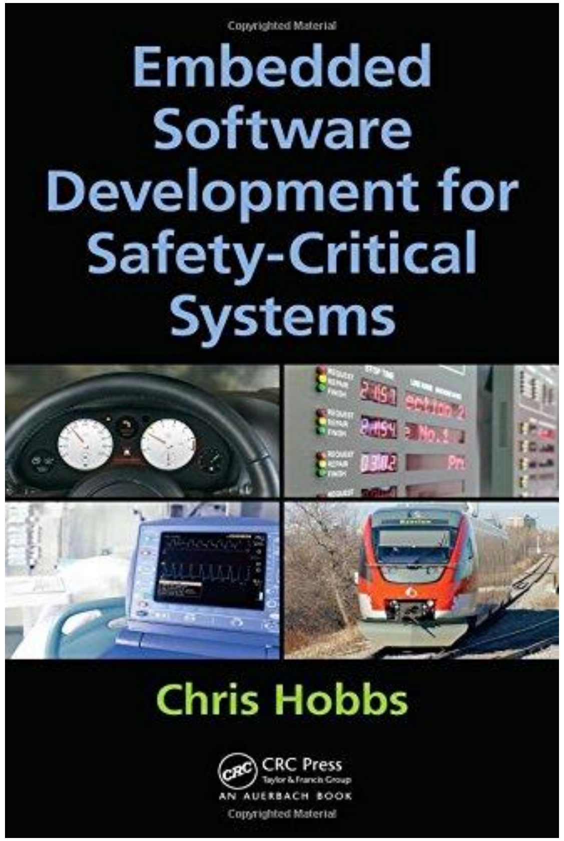
http://cdn.electronics-eetimes.com

Textbook



Marco Bozzano, Adolfo Villafiorita:
Design and Safety Assessment of Critical Systems
CRC Press (Taylor & Francis Ltd., USA), 2010.
ISBN 978-1-439-80331-8

Textbook

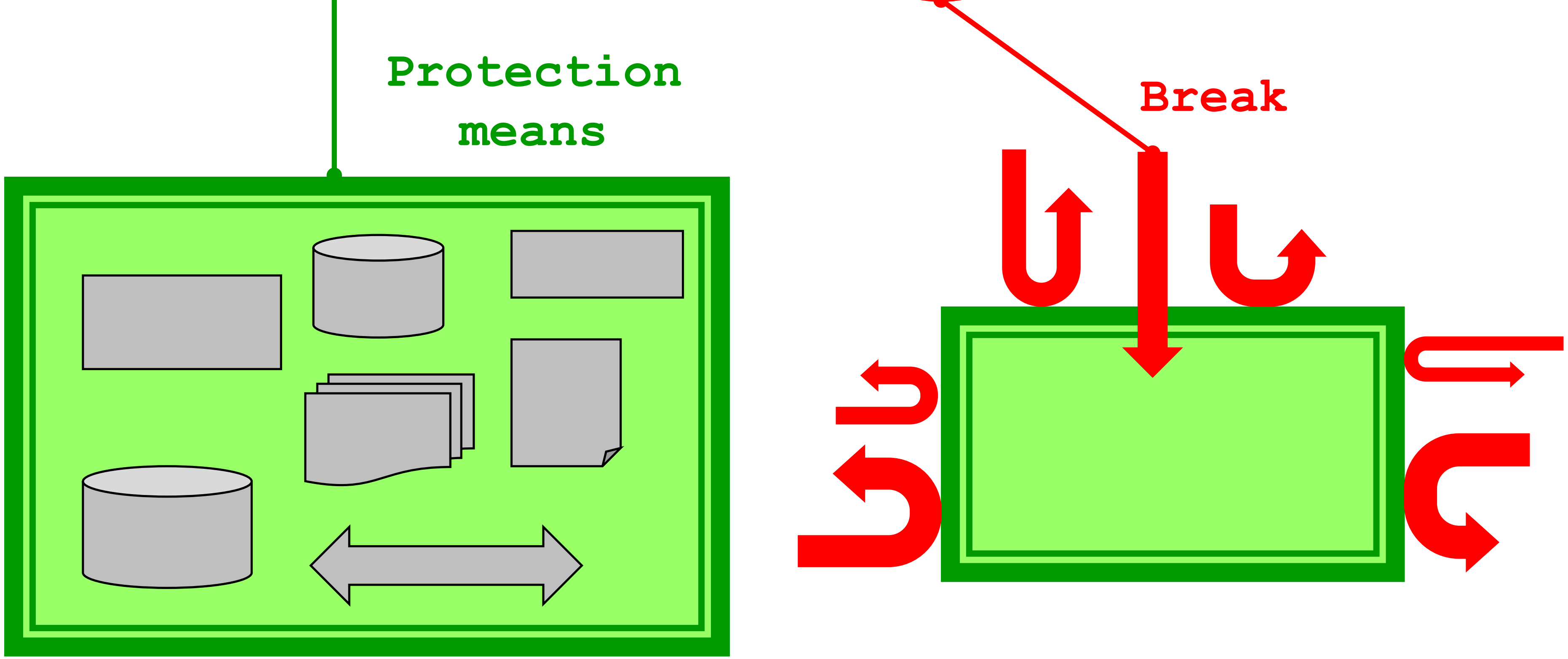


Chris Hobbs:
Embedded Software Development for Safety-Critical Systems
CRC Press (Taylor & Francis Inc.), USA, 2015.
ISBN 978-1-498-72670-2

Security

DEFINITIONS

Protection of an information system's assets with a known and acceptable **risk** of a security break



Today: IT-Security ⇒ Technology battle

Metrics

Company Management

http://clipart-library.com

- **Safety**
- **Security**
- **Integrity**
- **Confidentiality**
- **Real-time capability**
- **…**

Metrics for *domain-specific* quality properties

Company-specific set of **security** metrics

http://bigcashblog.com

Domain Experts

Security Standards

Information Security Management System
ISO 27001
Certified

NIST
National Institute of Standards and Technology
U.S. Department of Commerce
The Cybersecurity Framework
Cybersecurity Means Business

PCI Security Standards Council ®

Example: Security break

https://s3-eu-west-1.amazonaws.com

05.05.2017:

Hackers have stolen the personal data and financial details of tens of thousands of UK Debenhams customers, the company has admitted.

In a cyber attack against a third party firm that runs the retailer's online florist, Debenhams Flowers, hackers managed to take the *names, addresses and financial information of 26,000 customers*

http://www.telegraph.co.uk/technology/2017/05/05/debenhams-flowers-hack-credit-card-details-26000-people-stolen/

https://blog.bankbazaar.com

Security Metric Example (1/2): Open Vulnerabilities in Server-Farm

Vendor Communications

Microsoft
ORACLE
CITRIX
rti
KASPERSKY lab
etc.

Server

Operating System
Middleware
Communications-SW
Load-Balancer
Utilities
Monitoring
…

Vulnerability List

Institutional Communications

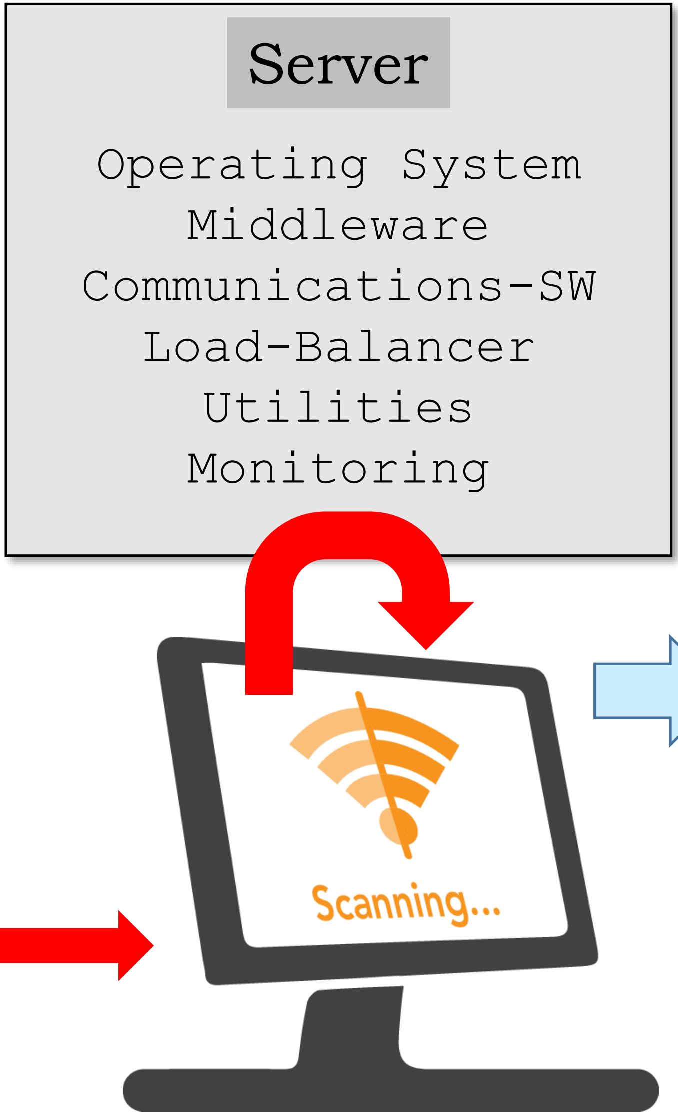
NIST
National Institute of Standards and Technology
Technology Administration, U.S. Department of Commerce

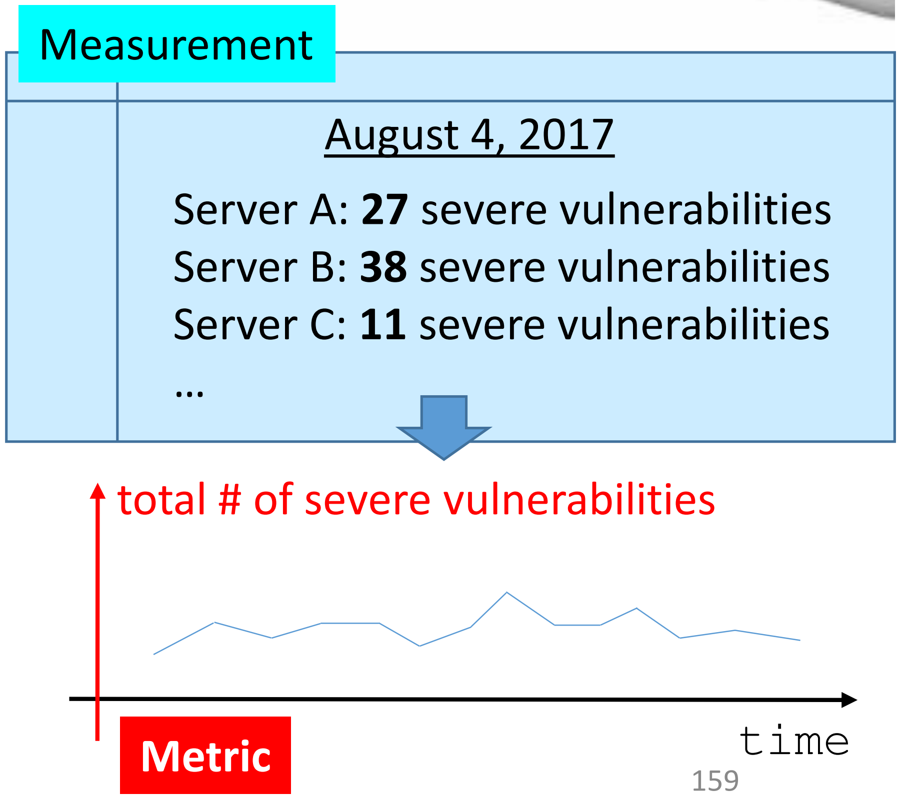
https://nvd.nist.gov/vuln/full-listing

US-CERT
UNITED STATES COMPUTER EMERGENCY READINESS TEAM

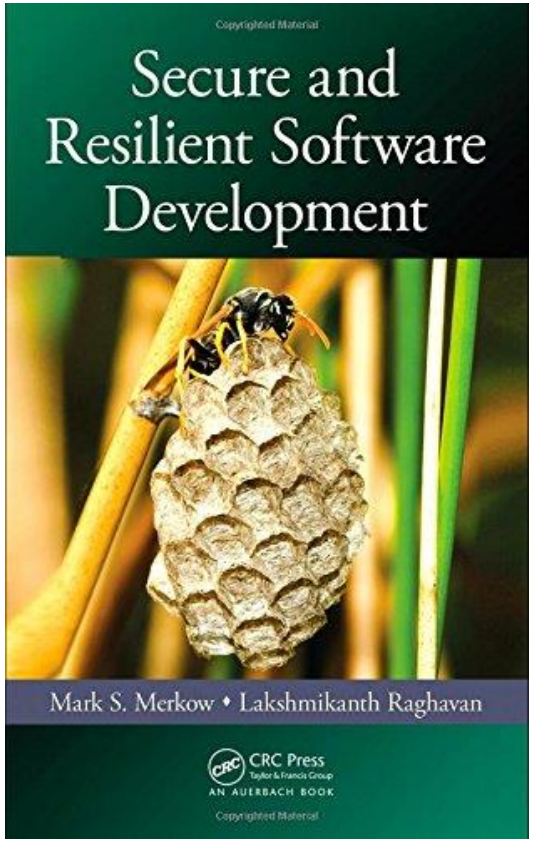
https://www.us-cert.gov/ncas/alerts/TA15-119A

Security Metric Example (2/2): Open Vulnerabilities in Server-Farm



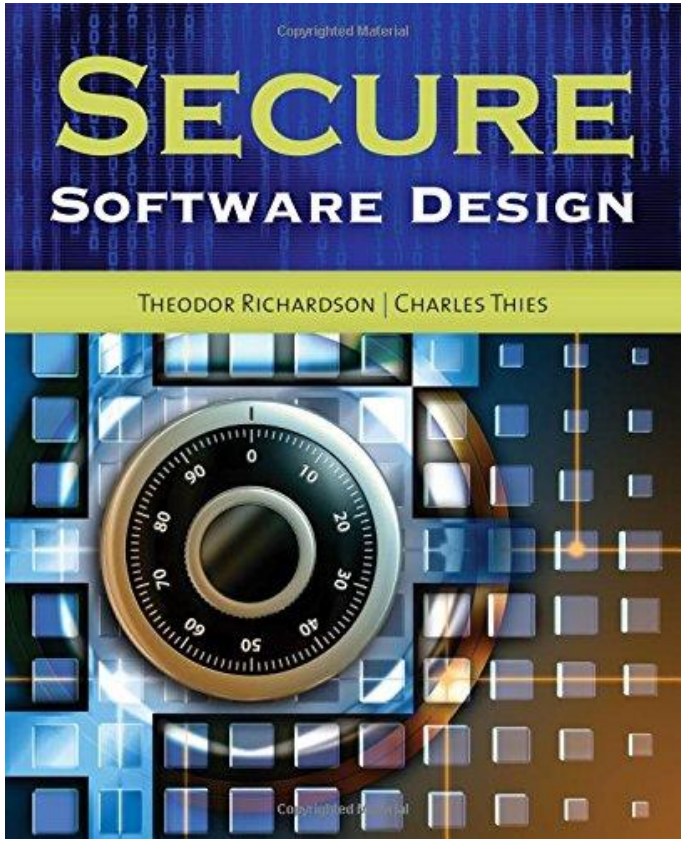


Textbook



Mark S. Merkow:
Secure and Resilient Software Development
Auerbach Publishers Inc., USA, 2010. ISBN 978-1-439-82696-6

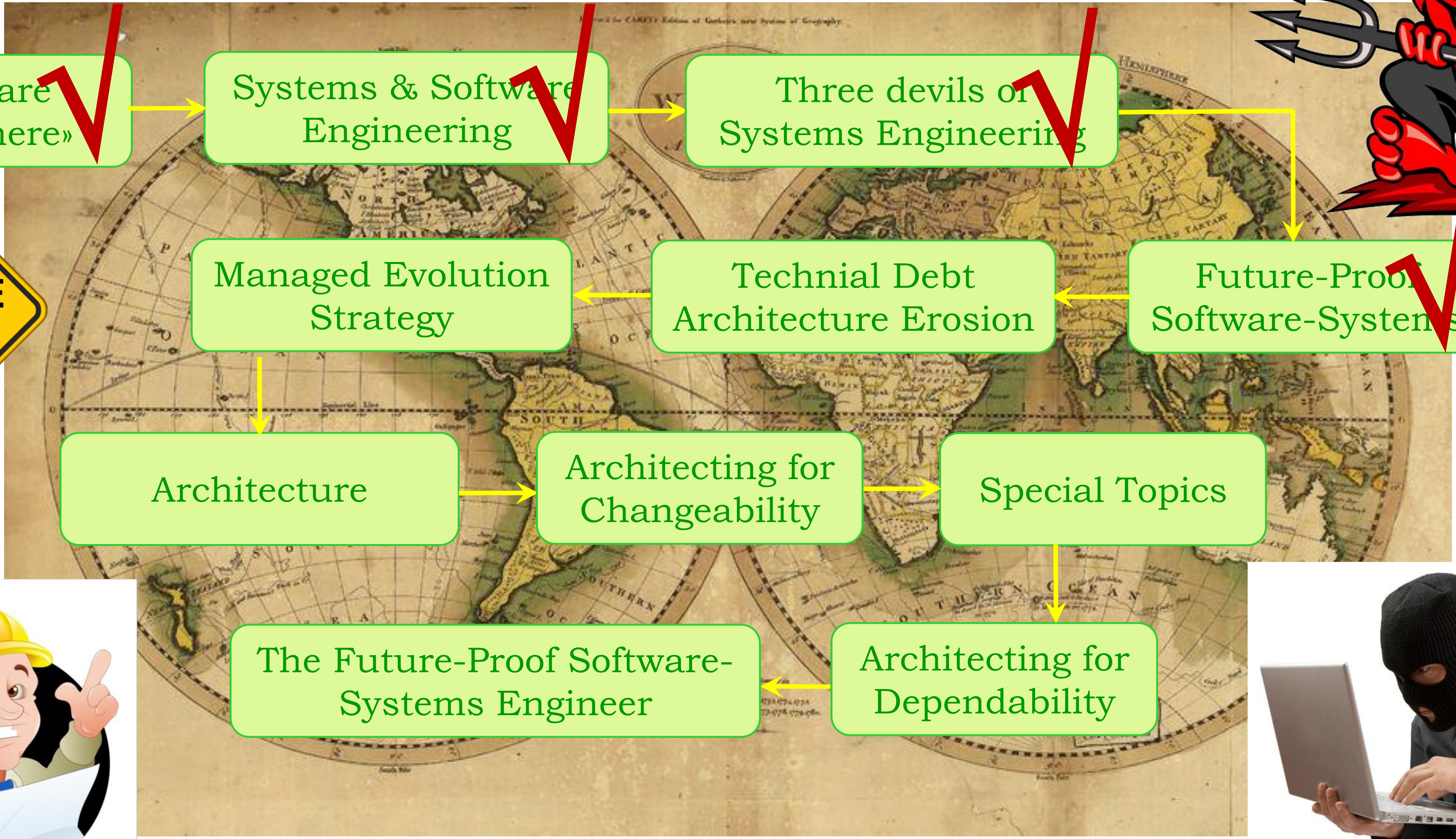
Textbook



Theodor Richardson, Charles N. Thies:
Secure Software Design
Jones & Bartlett Publisher, Inc., 2012. ISBN 978-1-4496-2632-7

Future-Proof Software-Systems [Part 1]

Our journey:



- «Software everywhere» √ → Systems & Software Engineering √ → Three devils of Systems Engineering √ → Future-Proof Software-Systems √
- → Technial Debt Architecture Erosion → Managed Evolution Strategy
- → Architecture → Architecting for Changeability → Special Topics
- → Architecting for Dependability → The Future-Proof Software-Systems Engineer

Part 1: Introduction