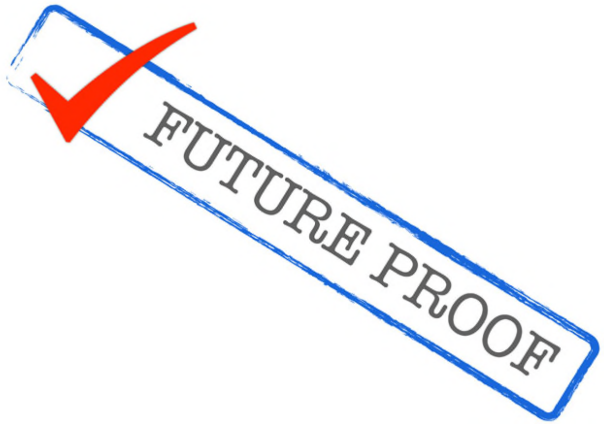


https://static1.squarespace.com

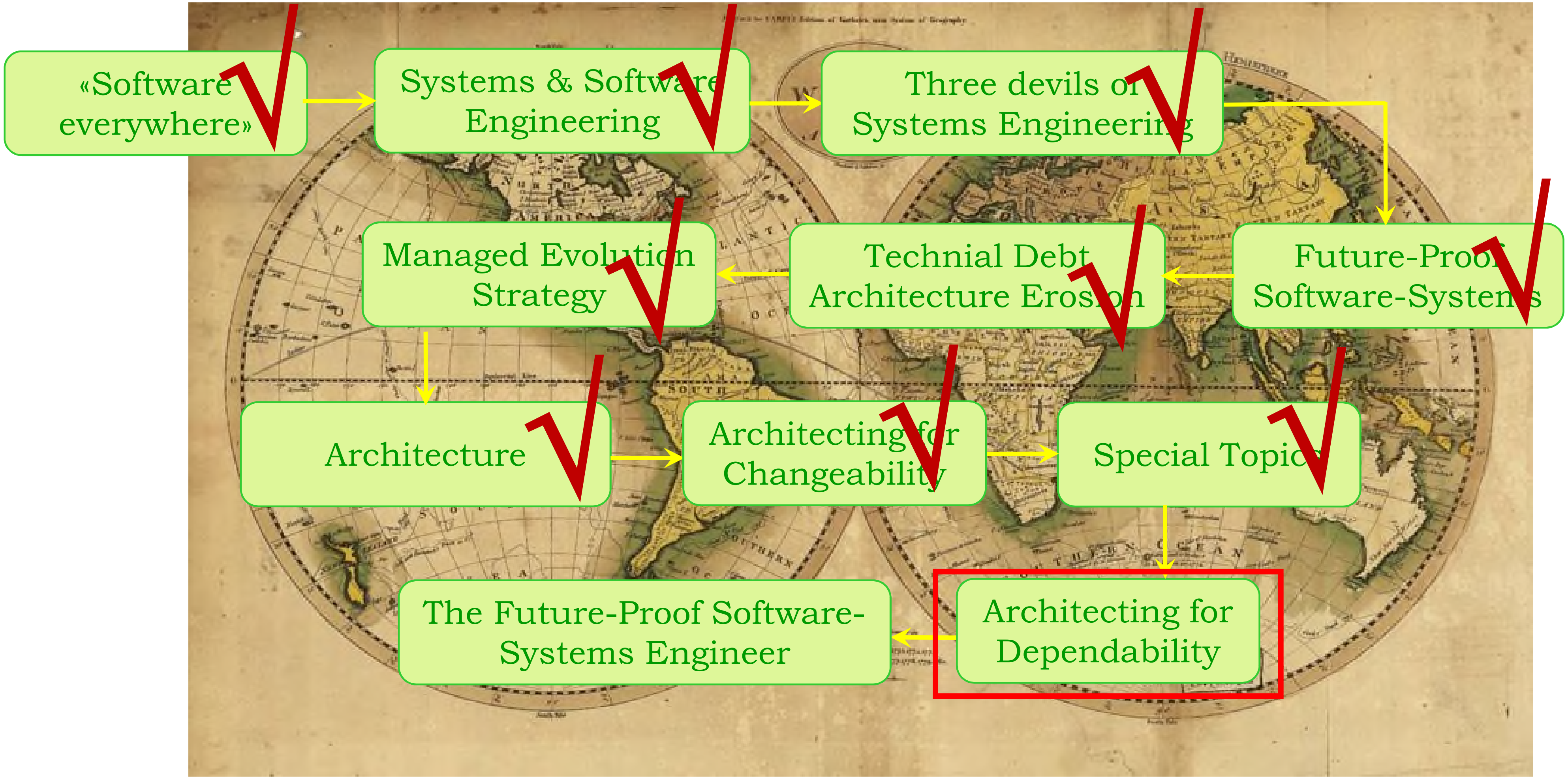
Future-Proof Software-Systems (FPSS)

Part 4B: Architecting for Dependability

Lecture WS 2017/18: Prof. Dr. Frank J. Furrer

Version 1.0

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

Remember

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

Content

Part 4B:

- ***Specific*** Dependability Architecture Principles (Examples)

- **Introduction**
- Safety
- Security
- Confidentiality
- Integrity
- Availability
- Real-Time Capability
- **Dependability Engineering and Methodology**

Introduction

Software Challenges

Software Business Value

Generates customer satisfaction and enables revenues

Software Changeability

enables success in the competitive business arena

Part 4

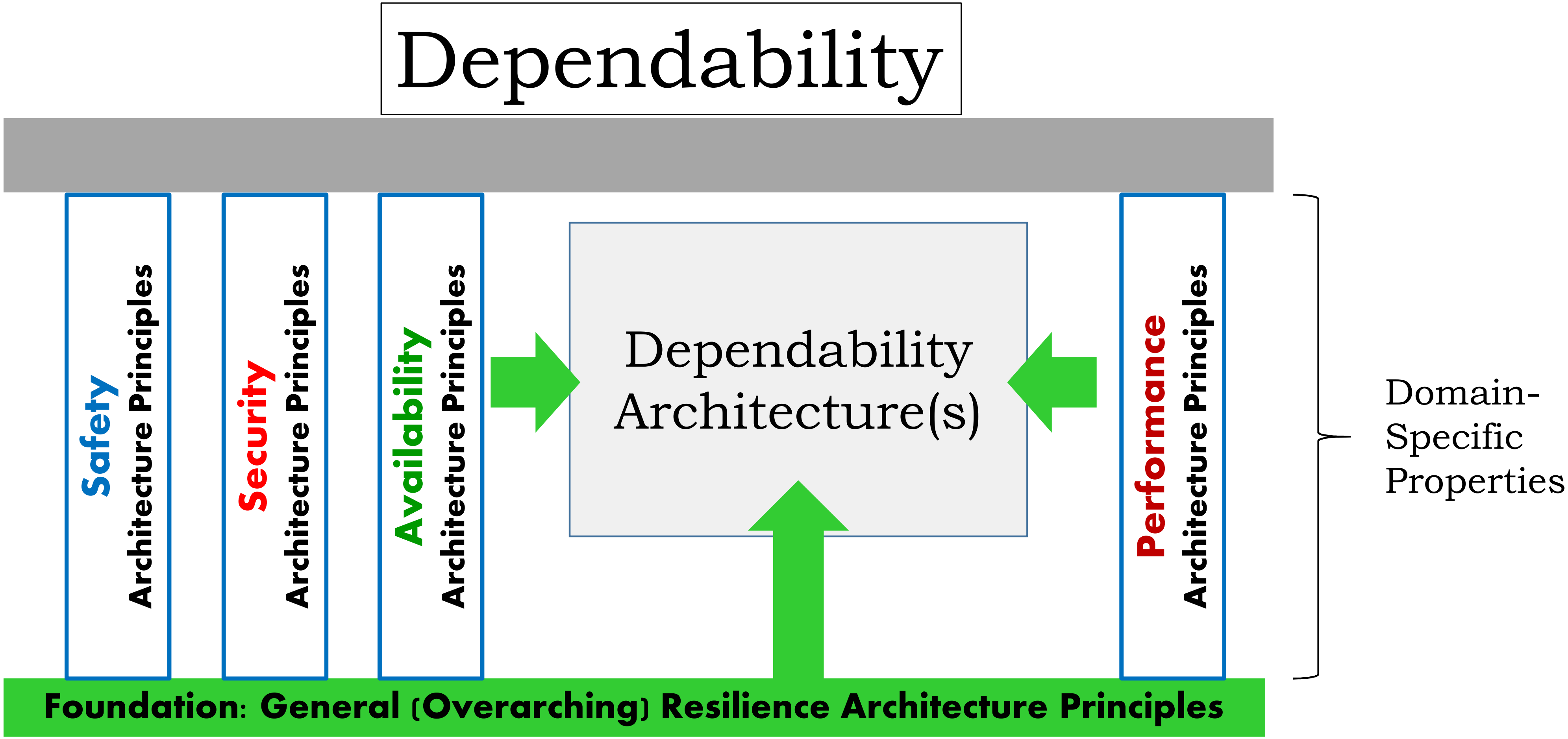
Software Dependability

assures survival in the hostile world

http://www.smallfuel.com

http://www.clipartsheep.com

http://www.clipartsheep.com


Dependability
Safety Architecture Principles
Security Architecture Principles
Availability Architecture Principles
Dependability Architecture(s)
Performance Architecture Principles
Domain-Specific Properties
Foundation: General (Overarching) Resilience Architecture Principles

Which are the elements of dependability ?

The **quality properties** of the system:

- Safety
- Security
- Integrity
- Confidentiality
- Real-time behaviour
- Availability
- …

Software Quality Properties	
accessibility	maintainability
accountability	manageability
accuracy	mobility
adaptability	modifiability
administrability	modularity
affordability	operability
agility	orthogonality
auditability	portability
autonomy	precision
availability	predictability
business value	process capabilities
capacity	producibility
changeability	provability
compatibility	reactivity
composability	real-time capability
confidentiality	reconfigurability
configurability	recoverability
correctness	relevance
credibility	reliability
customizability	repeatability
diagnosability	reproducibility
debugability	resilience
defendability	resistance to change
degradability	responsiveness
determinability	reusability
demonstrability	robustness
dependability	safety
deployability	scalability
discoverability	seamlessness
distributability	self-sustainability
durability	serviceability
effectiveness	securability
efficiency	simplicity
energy efficiency	stability
evolvability	standards compliance
extensibility	survivability
fail-safety	sustainability
failure transparency	tailorability
fault-tolerance	testability
fidelity	timeliness
flexibility	traceability
inspectability	transparency
installability	trustworthiness
integrity	ubiquity
interchangeability	understandability
interoperability	upgradability
learnability	vulnerability
	usability

Dependability: Protection of the System's Users & Environment

Attacks

Failures

Incidents

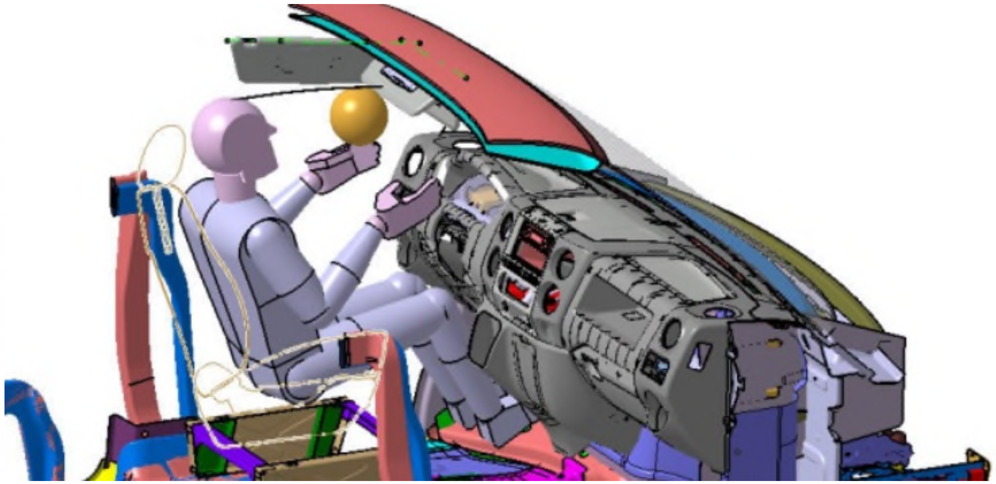
Disasters

Faults

https://www.scienceabc.com

Dependability Engineering

= Respected, viable, interdisciplinary engineering discipline



https://brace-automotive.com

With:

- Extensive *methodologies*
- Detailled *standards* + regulations
- Architecture *principles*, patterns and frameworks
- Strict *certification* procedures

Dependability Engineering

= Respected, viable, ***interdisciplinary*** engineering discipline

http://www.testing-expo.com

Safety-Engineering

http://mgconsulthr.com

Availability-Engineering

System Dependability Engineer

https://resources.workable.com

Security-Engineering

etc.

https://www.rubicon.eu

Certification-Engineering

Example: Security Engineering

https://resources.workable.com

Extensive *methodologies*

The OCTAVE Method

- For more information, you can download the Octave[SM] method implementation guide from www.cert.org/octave/omig.html

ISO 27001:2013
INFORMATION SECURITY MANAGEMENT SYSTEM

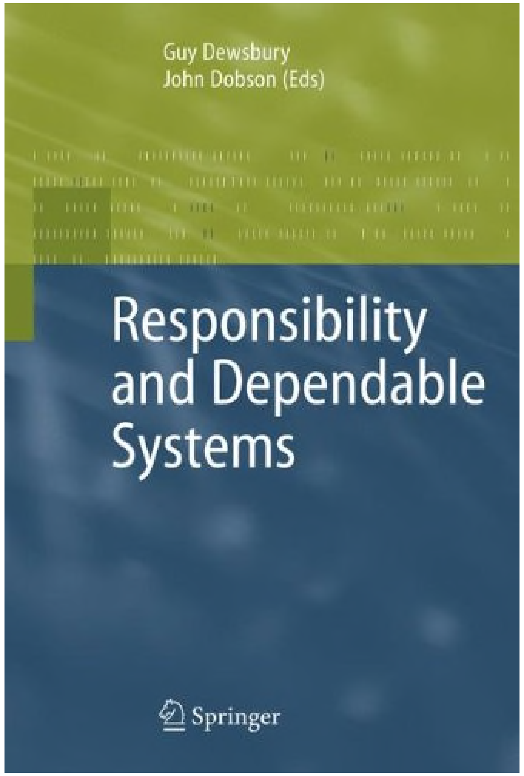
Detailled standards + regulations

Architecture principles, patterns and frameworks

CERTIFIED
ISO 27001
INFORMATION SECURITY

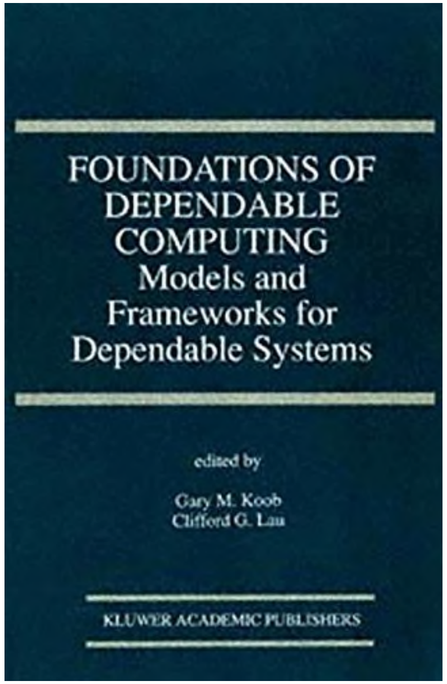
Strict certification procedures

Textbook



Guy Dewsbury, John Dobson (Editors):
Responsibility and Dependable Systems
Springer Publishing, London, UK, 2007. ISBN 978-1-849-96631-3

Textbook



Gary M. Koob, Clifford G. Lau:
Foundations of Dependable Computing:
Models and Frameworks for Dependable Systems
Springer-Verlag 2013 (Reprint of the original 1st edition 1994). ISBN 978-1-475-78315-5

Dependability Scorecard

Which **quality poperties** are important in my system ?

Software Quality Properties	
accessibility	maintainability
accountability	manageability
accuracy	mobility
adaptability	modifiability
administrability	modularity
affordability	operability
agility	orthogonality
auditability	portability
autonomy	precision
availability	predictability
business value	process capabilities
capacity	producibility
changeability	provability
compatibility	reactivity
composability	real-time capability
confidentiality	reconfigurability
configurability	recoverability
correctness	relevance
credibility	reliability
customizability	repeatability
diagnosability	reproducibility
debugability	resilience
defendability	resistance to change
degradability	responsiveness
determinability	reusability
demonstrability	robustness
dependability	safety
deployability	scalability
discoverability	seamlessness
distributability	self-sustainability
durability	serviceability
effectiveness	securability
efficiency	simplicity
energy efficiency	stability
evolvability	standards compliance
extensibility	survivability
fail-safety	sustainability
failure transparency	tailorability
fault-tolerance	testability
fidelity	timeliness
flexibility	traceability
inspectability	transparency
installability	trustworthiness
integrity	ubiquity
interchangeability	understandability
interoperability	upgradability
learnability	vulnerability
	usability

Application Domain

Dependability **expectations**

Dependability **properties**

Application Domain

Car:

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

Dependability **expectations**

Dependability **properties**

#	System Quality Property	Weight 0: irrelevant 10: highest importance
	Primary Characteristics	
1	Business Value	**10**
2	Changeability	**10**
Dependability:		
3	Safety	9
4	Fault-Tolerance	9
5	Compliance to laws & regulations	9
6	Integrity (Sensor Data)	9
7	Availability	8
8	Security	7
9	Diagnosability	6
	Secondary Characteristics	
10	Resources (Memory, CPU, …)	8
11	Compliance to industry-standards	7
12	Usability (User Interfaces)	9
etc		

Dependability **properties**

#	System Quality Property	Weight 0: irrelevant 10: highest importance
Primary Characteristics		
1	Business Value	**10**
2	Changeability	**10**
Dependability:		
3	Safety	9
4	Fault-Tolerance	9
5	Compliance to laws & regulations	9
6	Integrity (Sensor Data)	9
7	Availability	8
8	Security	7
9	Diagnosability	6
Secondary Characteristics		
10	Resources (Memory, CPU, ...)	8
11	Compliance to industry-standards	7
12	Usability (User Interfaces)	9
etc		

Dependability Specifications

Dependability Metrics

http://www.lockdeck.com

https://media.licdn.com

Application Domain

Car:

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

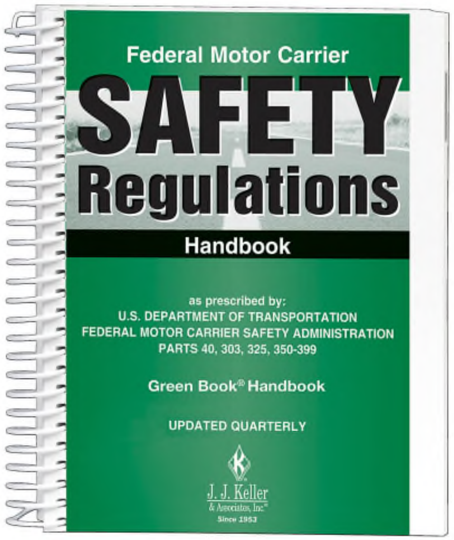
Dependability **expectations**

Dependability **properties**

#	System Quality Property	Weight 0: irrelevant 10: highest importance
	Primary Characteristics	
1	Business Value	**10**
2	Changeability	**10**
Dependability:		
3	Safety	9
4	Fault-Tolerance	9
5	Compliance to laws & regulations	9
6	Integrity (Sensor Data)	9
7	Availability	8
8	Security	7
9	Diagnosability	6
	Secondary Characteristics	
10	Resources (Memory, CPU, …)	8
11	Compliance to industry-standards	7
12	Usability (User Interfaces)	9
etc		

Dependability Scorecard

Application Domain



Dependability **expectations**

Dependability **properties**

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

#	System Quality Property	Weight 0: irrelevant 10: highest importance
	Primary Characteristics	
1	Business Value	**10**
2	Changeability	**10**
Dependability:		
3	Safety	9
4	Fault-Tolerance	9
5	Compliance to laws & regulations	9
6	Integrity (Sensor Data)	9
7	Availability	8
8	Security	7
9	Diagnosability	6
	Secondary Characteristics	
10	Resources (Memory, CPU, ...)	8
11	Compliance to industry-standards	7
12	Usability (User Interfaces)	9
etc		

#	System Quality Property	Weight 0: irrelevant 10: highest importance
	Primary Characteristics	
1	Business Value	**10**
2	Changeability	**10**
Dependability:		
3	Safety	9
4	Fault-Tolerance	9
5	Compliance to laws & regulations	9
6	Integrity (Sensor Data)	9
7	Availability	8
8	Security	7
9	Diagnosability	6
	Secondary Characteristics	
10	Resources (Memory, CPU, …)	8
11	Compliance to industry-standards	7
12	Usability (User Interfaces)	9
etc		

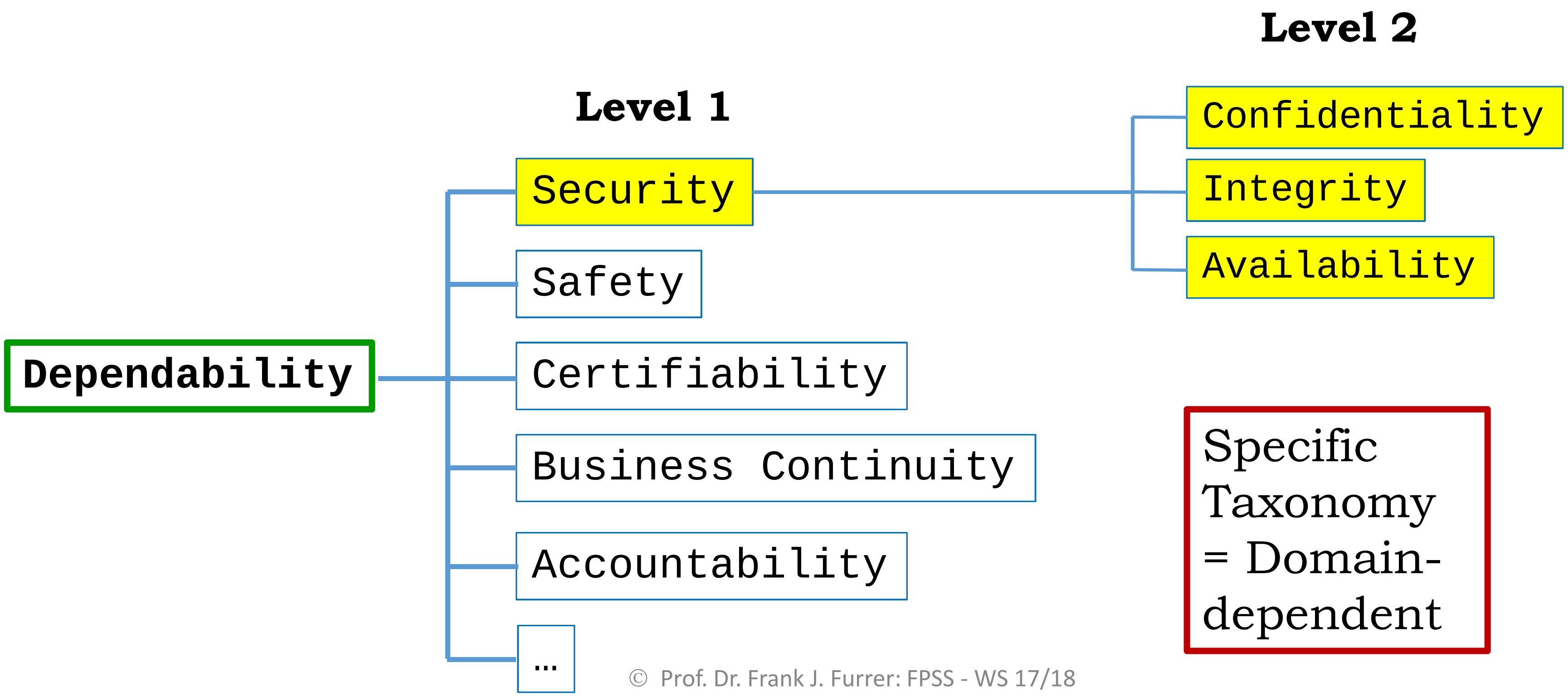
The **dependability scorecard** defines the *dependability properties* of the application domain

Dependability Taxonomy

Dependability *Scorecard*: **Business** View

Dependability *Taxonomy*: **Engineering** View

refinement



Specific Taxonomy = Domain-dependent

Dependability Specifications

#	System Quality Property	Weight 0: irrelevant 10: highest importance
Primary Characteristics		
1	Business Value	**10**
2	Changeability	**10**
Dependability:		
3	Safety	9
4	Fault-Tolerance	9
5	Compliance to laws & regulations	9
6	Integrity (Sensor Data)	9
7	Availability	8
8	Security	7
9	Diagnosability	6
Secondary Characteristics		
10	Resources (Memory, CPU, …)	8
11	Compliance to industry-standards	7
12	Usability (User Interfaces)	9
etc		

Level 1

Level 2

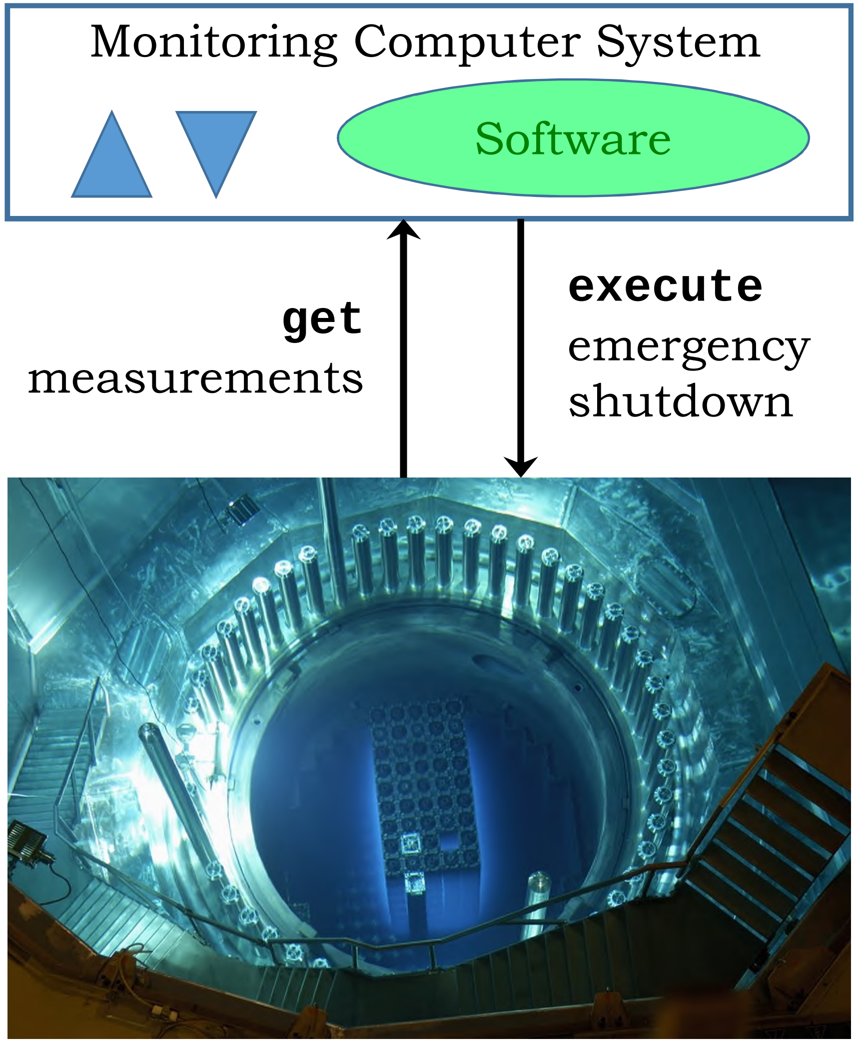
Dependability
- Security
  - Confidentiality
  - Integrity
  - Availability
- Safety
- Certifiability
- Business Continuity
- Accountability
- …

Dependability Specifications

Dependability Metrics

https://meda.licdn.com

Example: Nucelar Reactor Automatic Shutdown [simplified] (1/2)



https://www.xceed-eng.com

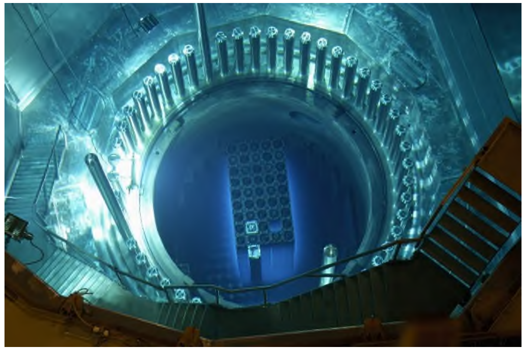
Functional Requirements

- Gather all necessary *measurement values* and indicators
- Continuously *supervise* their evolution
- Detect any *abnormal* condition
- Decide on *shutdown*
- *Execute* emergency shutdown procedure

Dependability Requirements

- *Reliability*
- *Availability*
- *Safety*
- *Auditability*
- *Conformance to regulations*
- *Certification*

Example: Nucelar Reactor Automatic Shutdown [simplified] (2/2)



Dependability Specifications
[Examples only]

- **Safety**: «No false positives / No false negatives»
- **Reliability**: Fully formally verified software
- **Availability**: Downtime max. 1 min / Max. 10x/year
- **Auditability**: All data, actions and decisions must be fully logged. Logs must be unforgeable (#Certs or blockchain)
- **Conformance to regulations**: All relevant regulations must be fully implemented
  U.S. Nuclear Regulatory Commission (https://www.nrc.gov/)
- **Certification**: The system must be fully certified

https://i.pinimg.com

Dependability Specifications

Higher degree of formalization & precision

Functional Specifications

Requirements: Textual, Models

Specifications: (semi-) formal models

Implementation: validated, verified

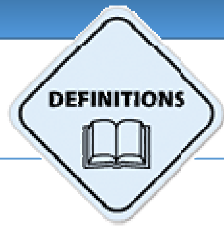
Dependability Specifications

Requirements: Textual, Models

Specifications: (semi-) formal models

Metrics: formal

Implementation: validated, verified & (often) certified



Dependability Specification

The dependability specification of a system defines all *non-functional specifications* (safety, security, …)

which are necessary to **reduce the risk** of operation of the system

to a known, **acceptable level**

https://top10binarydemo.com

Dependability Metrics

DEFINITIONS

Dependability Metric

A dependability metric for a *non-functional property* of a system

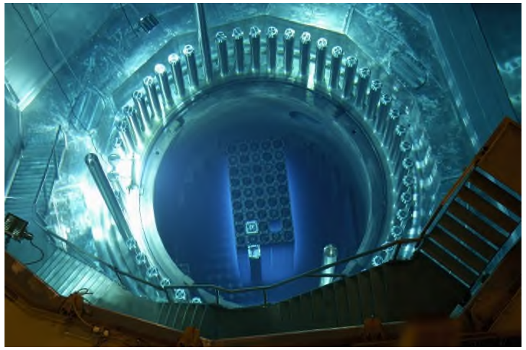
(safety, security, …)

is a **quantitative measure**

of the **effectiveness** of the respective risk-mitigation provisions

https://www.bionicturtle.com

Example: Nucelar Reactor Automatic Shutdown [simplified] (2/2)



Dependability Requirements
[Examples only]

Metrics needed

- **Safety**: «No false positives / No false negatives»
- **Reliability**: Fully formally verified software
- **Availability**: Downtime max. 1 min / Max. 10x/year
- **Auditability**: All data, actions and decisions must be fully logged. Logs must be unforgeable (#Certs or blockchain)
- **Conformance to regulations**: All relevant regulations must be fully implemented
  U.S. Nuclear Regulatory Commission (https://www.nrc.gov/)
- **Certification**: The system must be fully certified

https://i.pinimg.com

Example: Nucelar Reactor Automatic Shutdown [simplified] (1/2)

- **Safety**: «No false positives / No false negatives»
  - **Express in measurable probabilities**
- **Reliability**: Fully formally verified software
  - **Which parts, which method**
- **Availability**: Downtime max. 1 min / Max. 10x/year
  - **Express in measurable up-/downtime**
- **Auditability**: All data, actions and decisions must be fully logged. Logs must be unforgeable
  - **List of log item /#Certs or blockchain**
- **Conformance to regulations**: All relevant regulations must be fully implemented
  - **List regulations / Which parts**
- **Certification**: The system must be fully certified
  - **List certification bodies / Which parts**

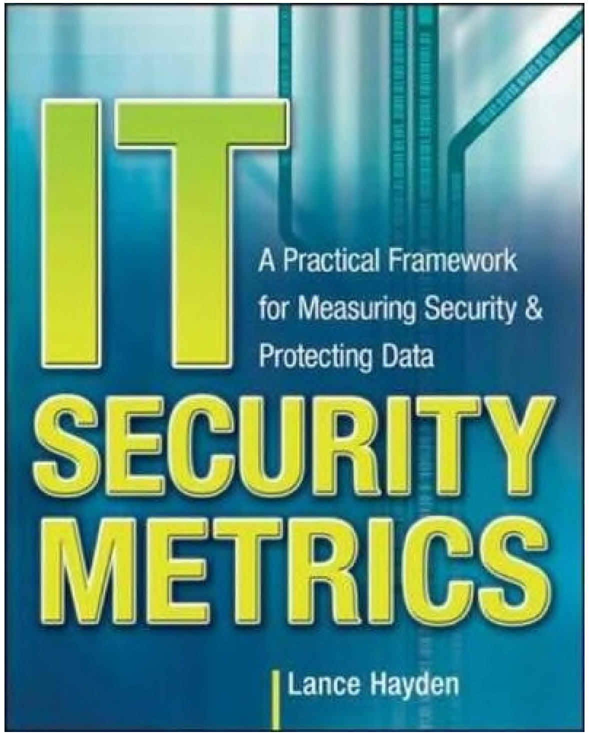
Note:
Defining, implementing and use of metrics is *difficult* and *expensive*

Be careful when using metrics

... and be sure that they are useful!

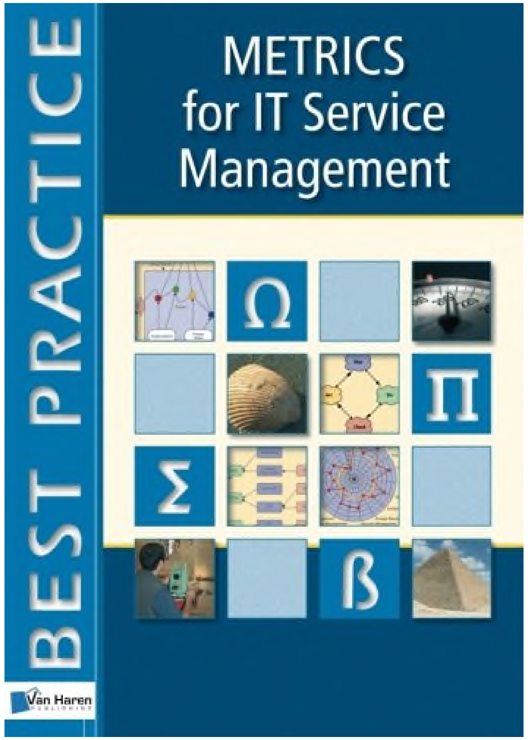
For many application domains proven metrics are available

Textbook



Lance Hayden:
IT Security Metrics: *A Practical Framework for Measuring Security and Protecting Data*
McGraw-Hill Education Ltd., 2010. ISBN 978-0-071-71340-5

Textbook



Peter Brooks:
Metrics for IT Service Management
Van Haren Publishing, 2006. ISBN 978-9-0772-1269-1

What can we learn in this lecture?

https://www.npmjs.com

We cannot become full **Resilience Engineers** in this lecture

… but we can learn:

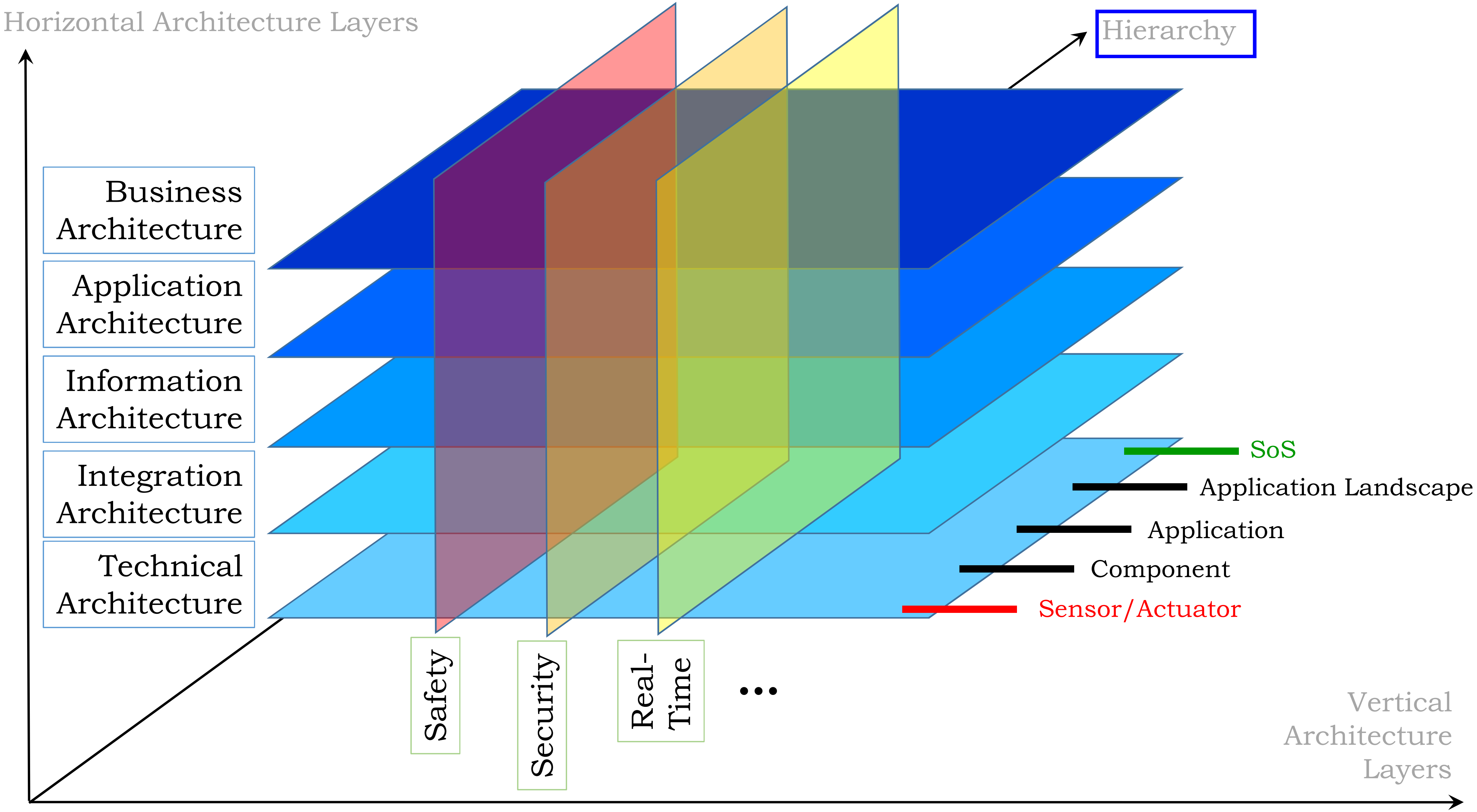
- The *methodology*
- Some important *principles & patterns*

What can we learn in this lecture?

Choice for the Lecture

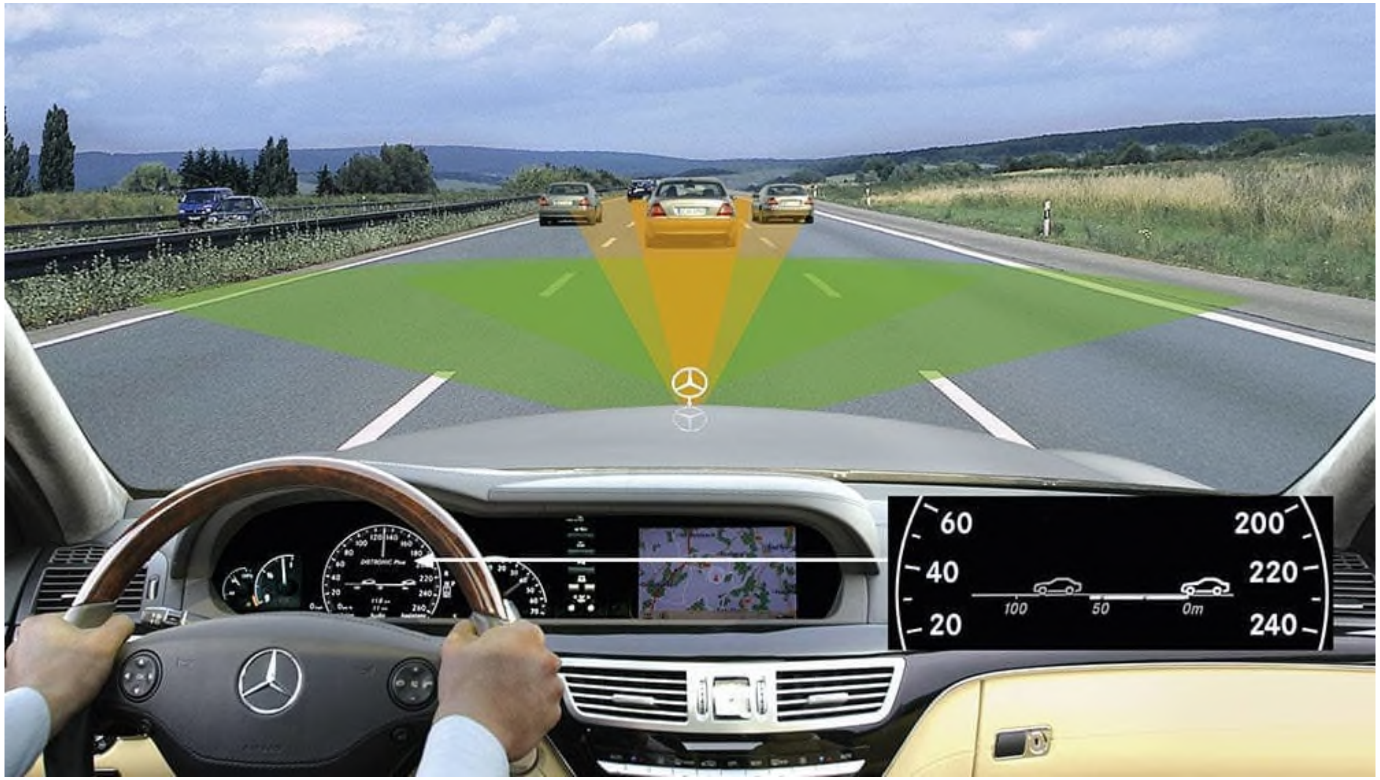
Dependability Property Topics:

- Safety
- Security
- Confidentiality
- Integrity
- Availability
- Real-Time Capability



Horizontal Architecture Layers
Hierarchy
Business Architecture
Application Architecture
Information Architecture
Integration Architecture
Technical Architecture
SoS
Application Landscape
Application
Component
Sensor/Actuator
Safety
Security
Real-Time
...
Vertical Architecture Layers

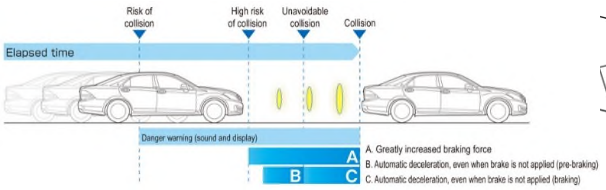
Example: Safety – Collision Avoidance System (1/3)

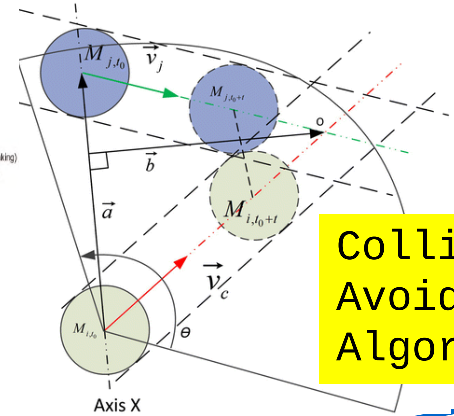


- Radar captures objects in front
- Radar measures relative speeds
- Speedometer provides absolute speed
- Algorithm calculates safe distance
- Safety program brakes car if distance too small (inkl. Δv)

Example: Safety – Collision Avoidance System (2/3)

http://www2.toyota.co.jp

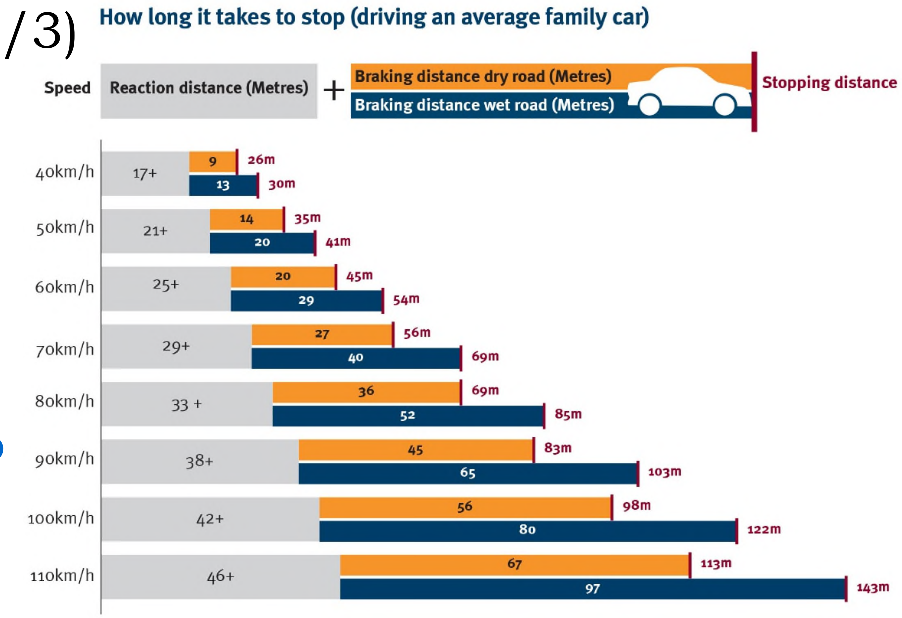




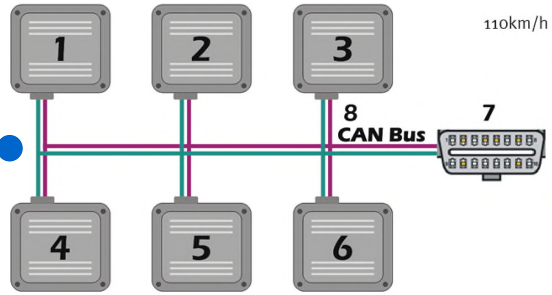
Collision Avoidance Rules

- Business Architecture
- Applications Architecture
- Information Architecture
- Integration Architecture
- Technical Architecture

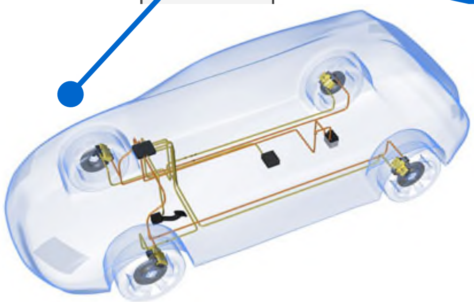
Collision Avoidance Algorithm



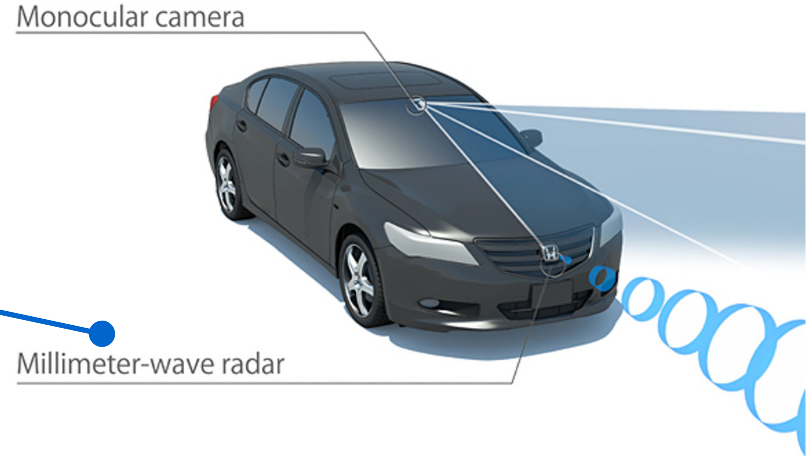
https://www.qld.gov.au



https://media.licdn.com



brake-by-wire



http://www.extremetech.com

Example: Safety – Collision Avoidance System (3/3)

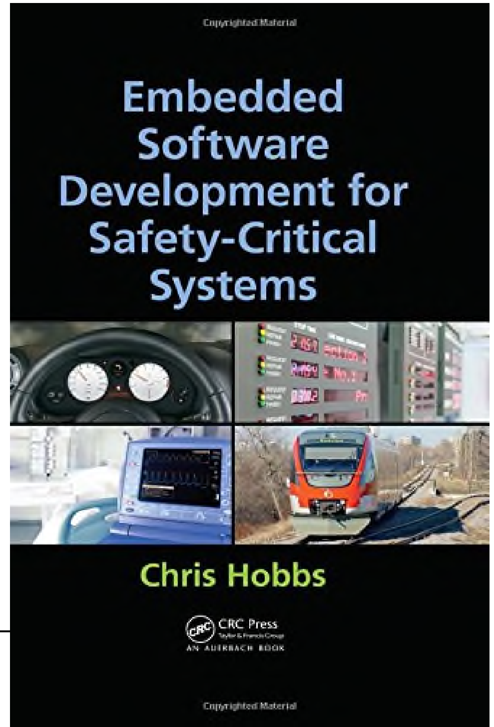
ISO 26262 is an *international standard* for functional safety of electrical and/or electronic systems in **production automobiles** defined by the International Organization for Standardization (ISO) in 2011

https://www.qa-systems.com

ISO 26262
Road Vehicles - Fuctional Safety

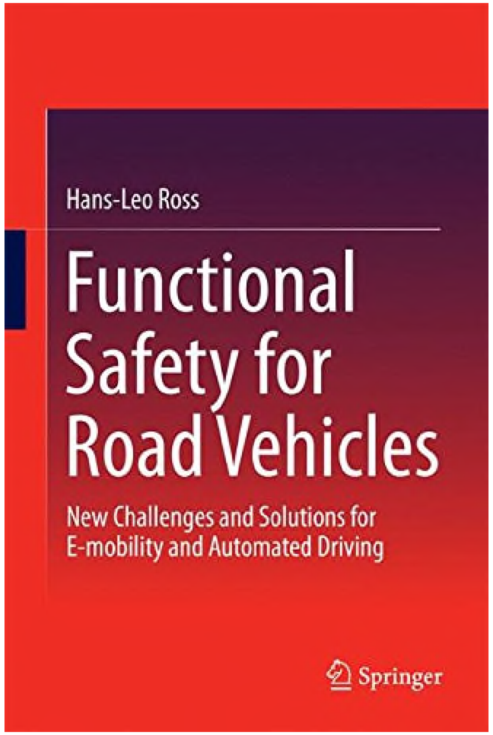
- Business Architecture
- Applications Architecture
- Information Architecture
- Integration Architecture
- Technical Architecture

Specific:
- Safety architecture
- Safe system development process
- Validation
- Verfication
- Certification



ISBN 978-1-4987-267022

Textbook

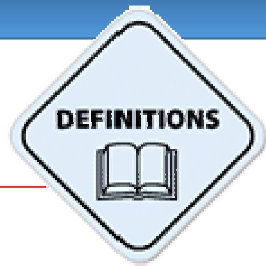


Hans-Leo Ross:
Functional Safety for Road Vehicles: *New Challenges and Solutions for E-mobility and Automated Driving*
Springer-Verlag, 2016. ISBN 978-3-319-33360-1

Textbook

https://www.iso.org/home.html

Dependability Property: Safety



Safety is the state of being **protected**

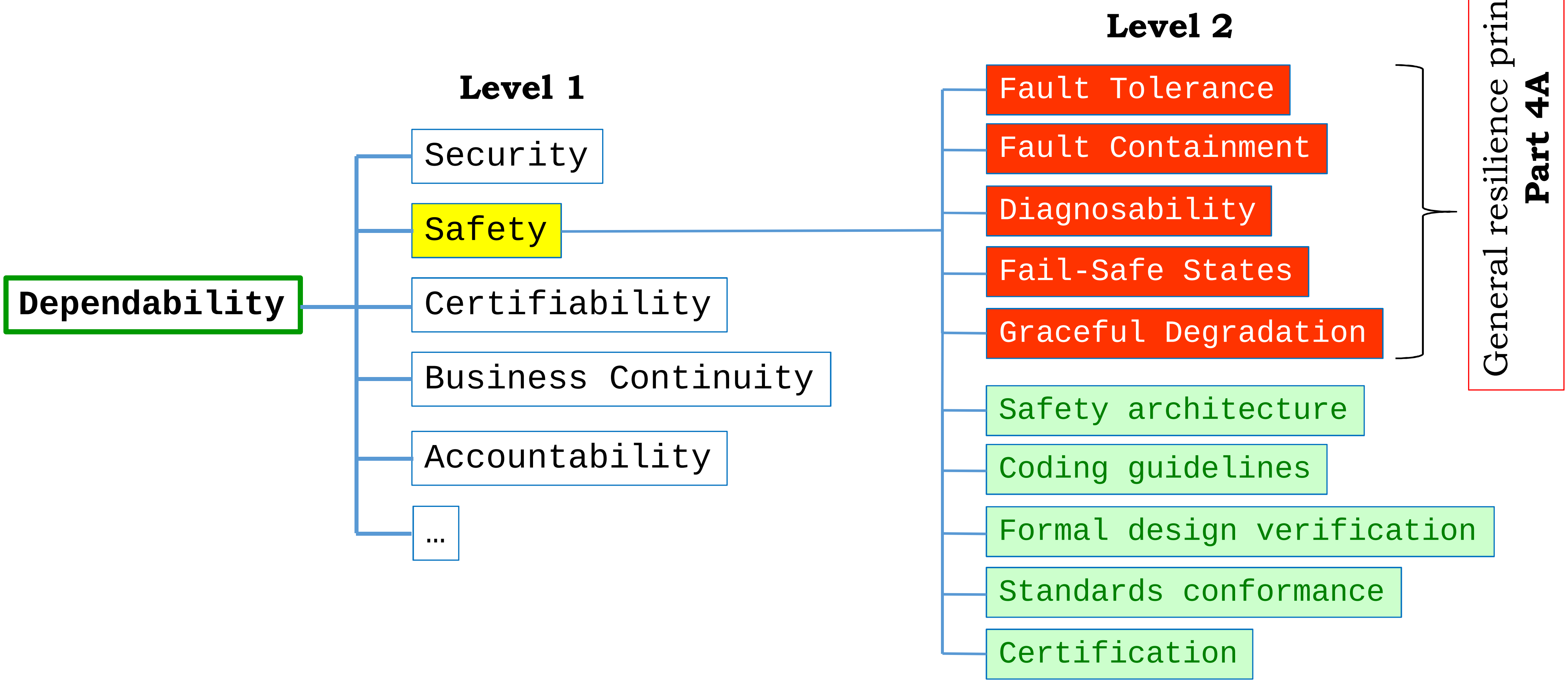
against *failure*, *damage*, *error*, *accidents*, *harm*, or any other event

that could be considered non-desirable

in order to achieve an **acceptable level of risk**

http://www.smarttalent.net

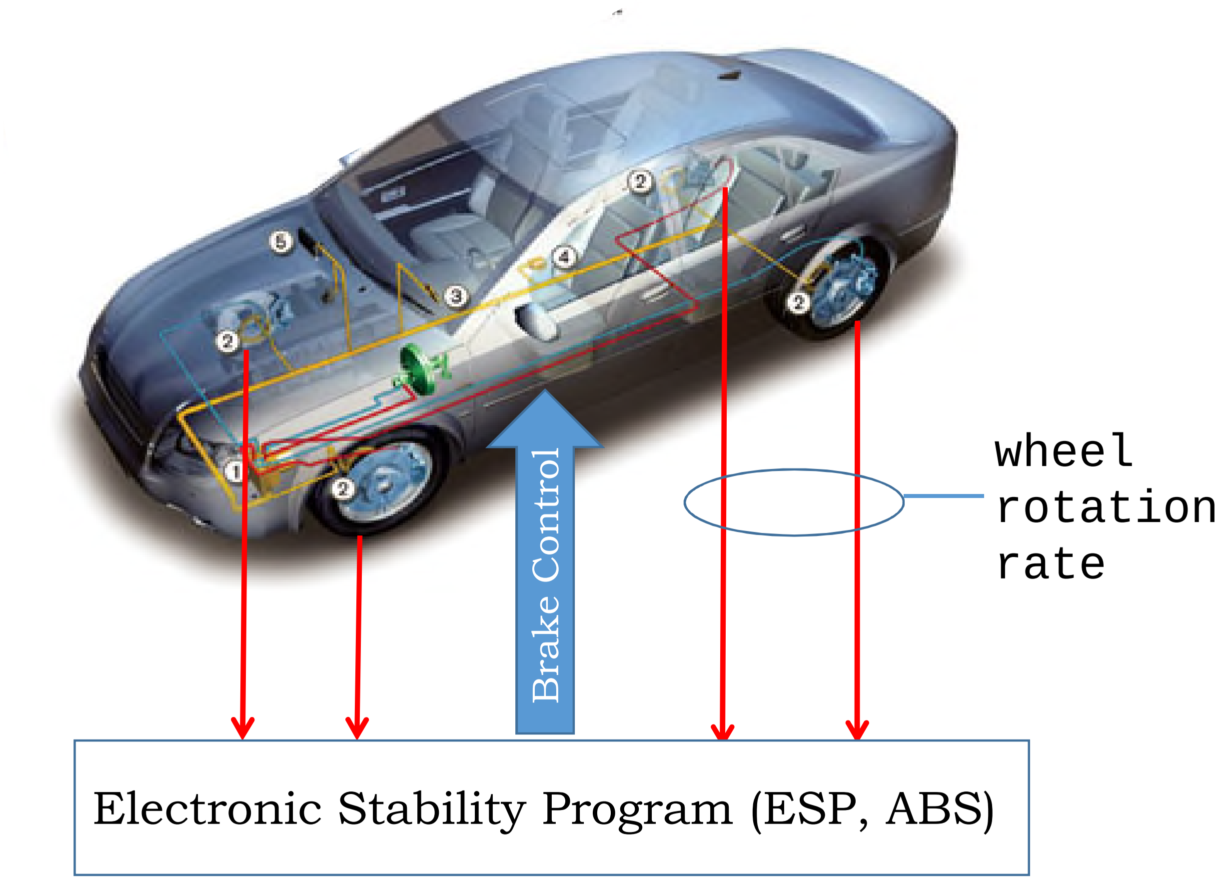
Dependability Taxonomy: Safety



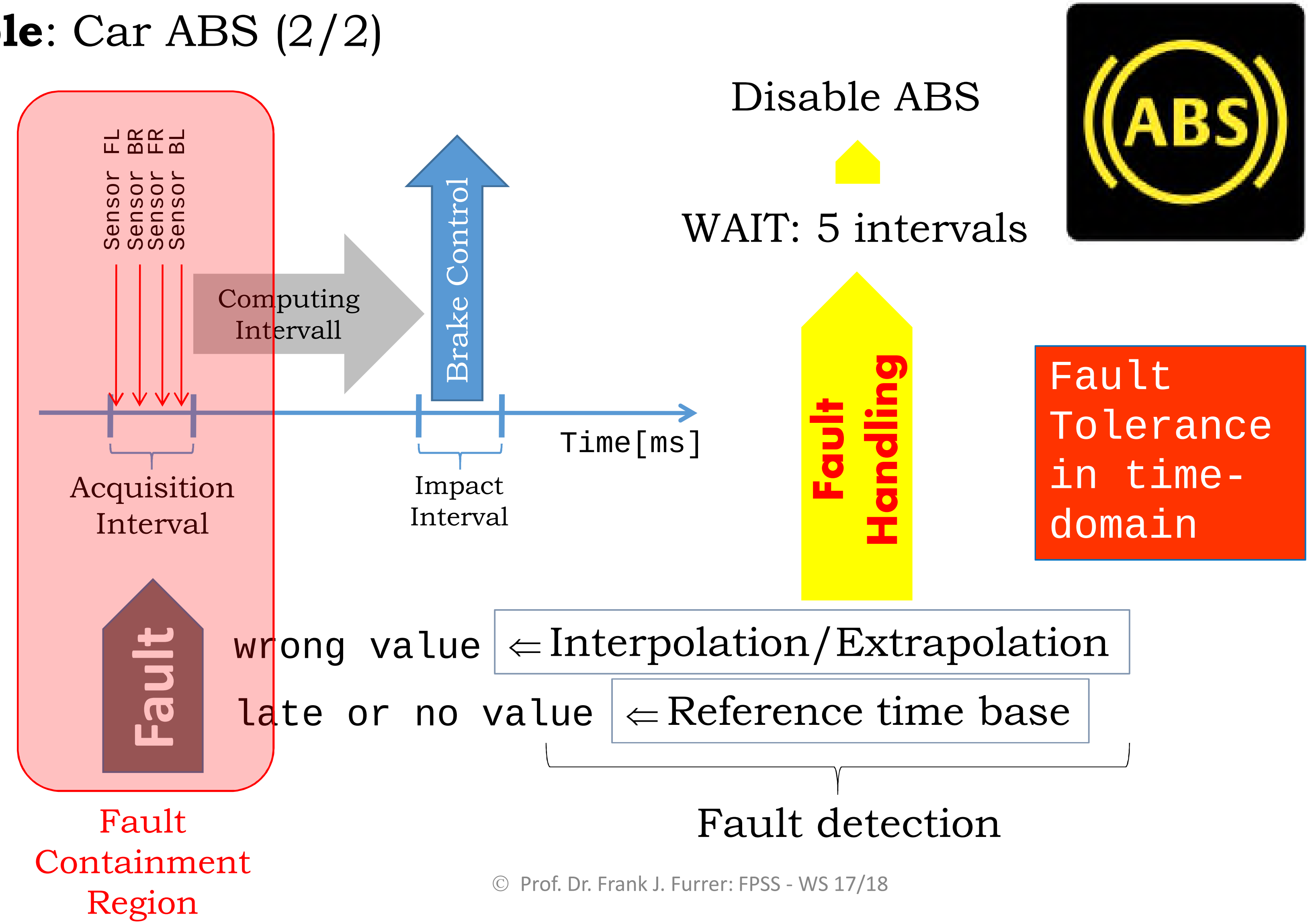
Fault Tolerance

Example: Car ABS (1/2)

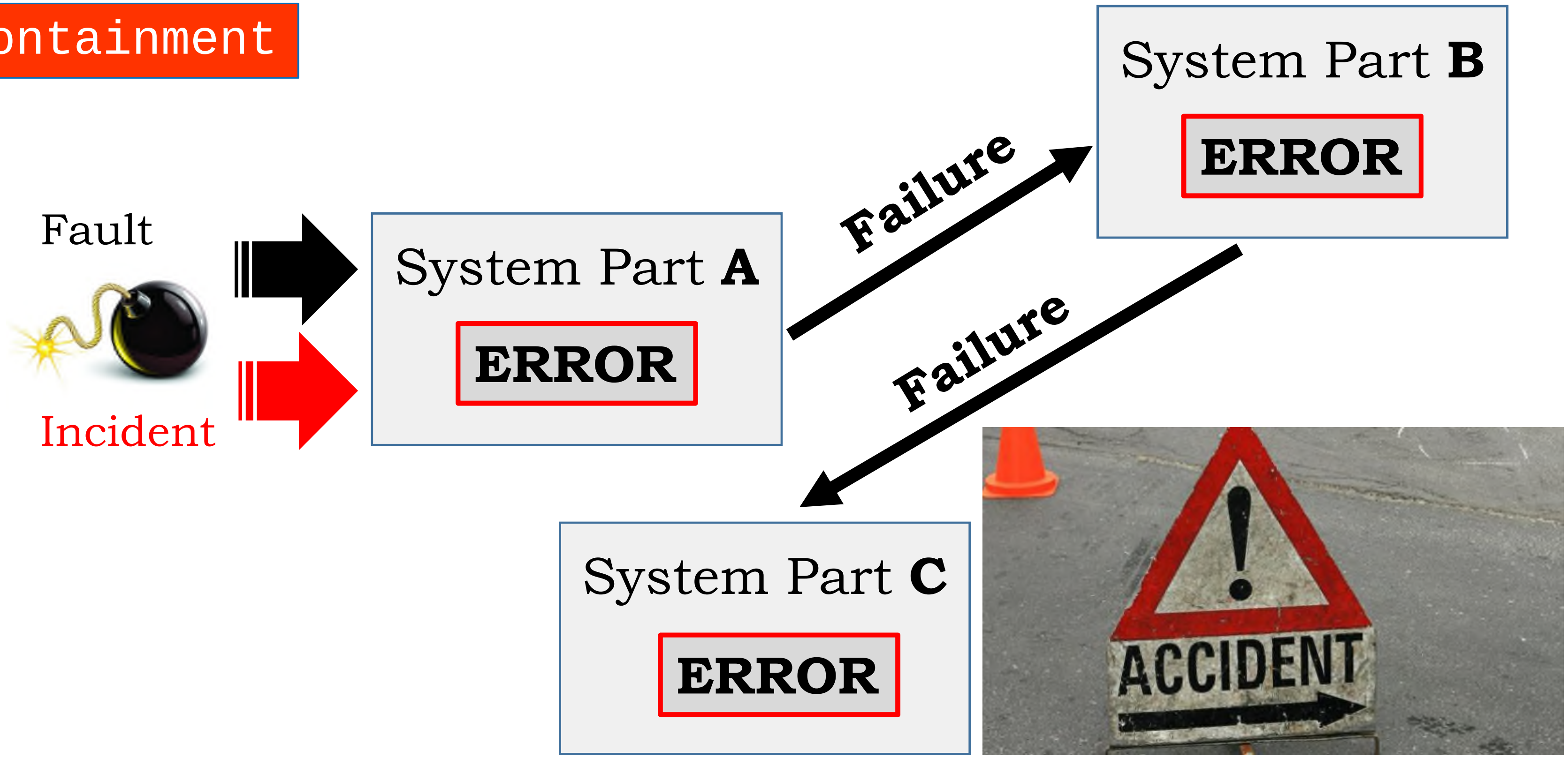
http://www.tomorrowstechnician.com



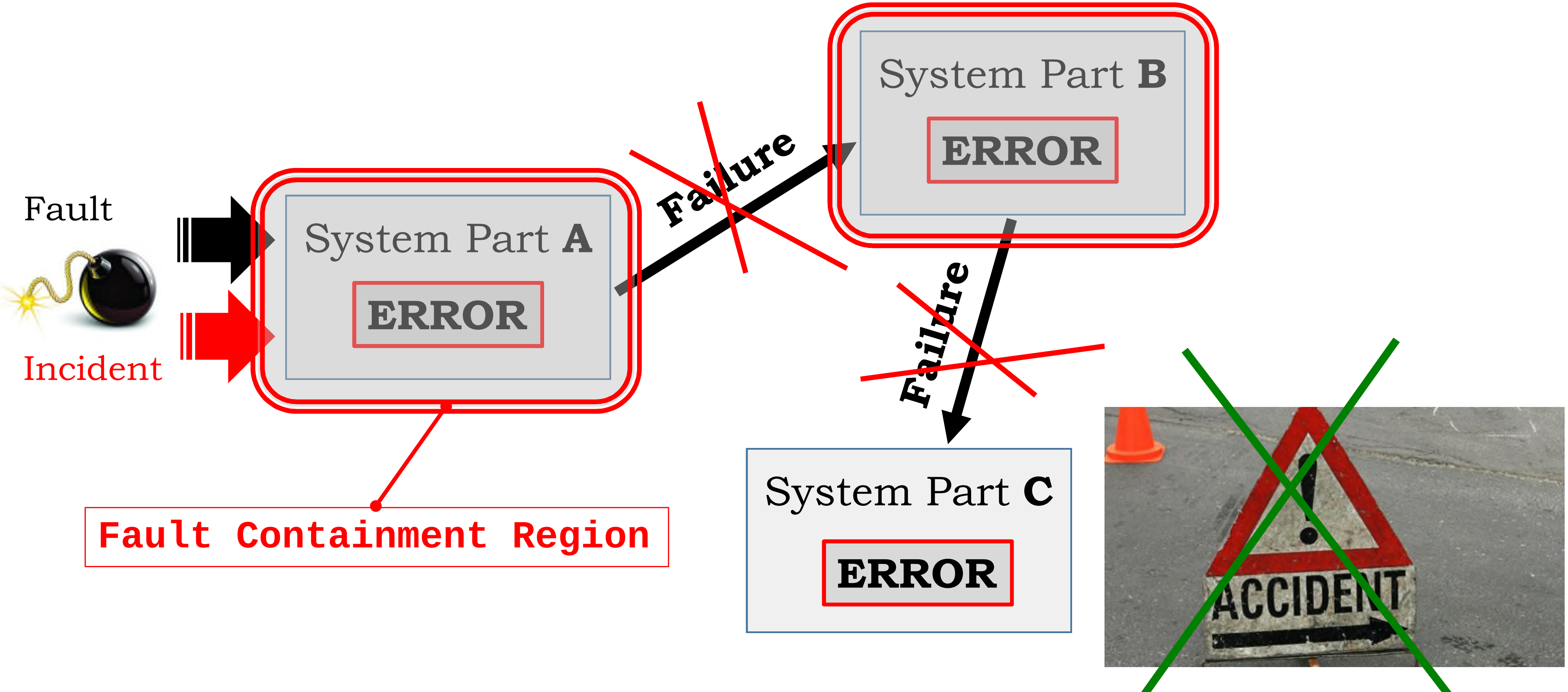
Example: Car ABS (2/2)



Fault Containment



The consequences of a *fault* – the ensuing *error* – can **propagate** either by an erroneous message or by an erroneous output action of the faulty part



Build **error** propagation boundaries around each system part

Diagnosability

DEFINITIONS

Diagnosability is the property of a partially observable system with a given set of possible failures, that these failures can be ***detected*** with certainty with a finite observation

https://www.phmsociety.org

Post-mortem

Preventive

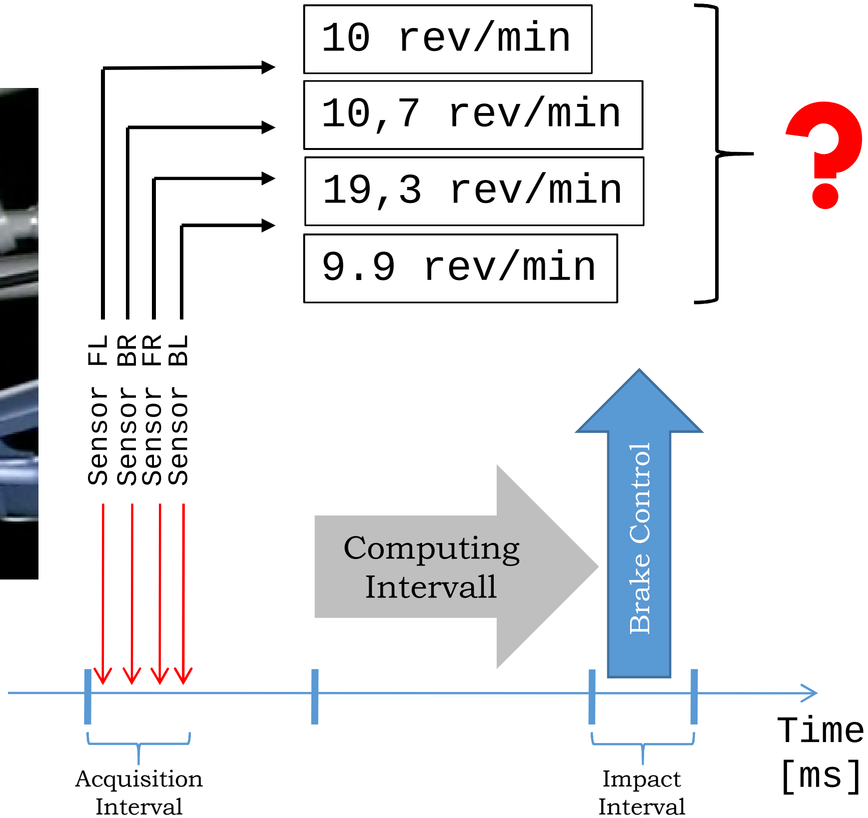
https://www.dexcom.com

https://www.ashdown-ingram.com.au

Example: *Inconsistent* wheel rotation information

http://www.cdxetextbook.com/brakes

Wheel rotation speed sensor



Diagnosability for Failure Prevention

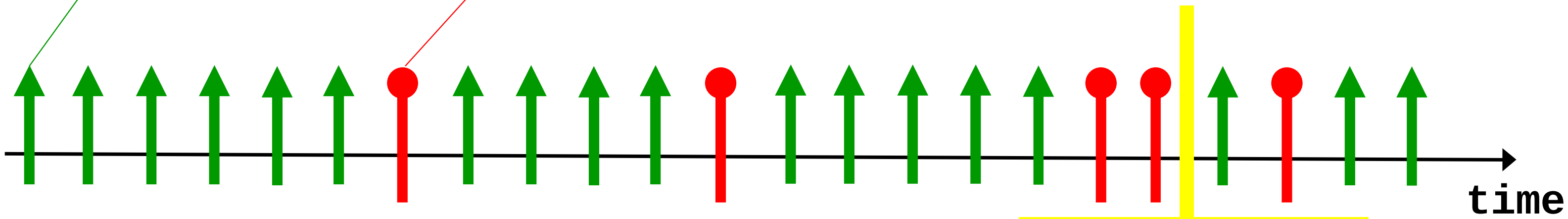
Example: Emerging failure detection

Wheel rotation speed sensor

http://www.cdxetextbook.com/brakes

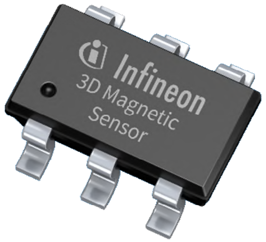
Redundancy

Correct value | Faulty value



The coming sensor failure is ***anticipated***
- Corrective action possible

Diagnosability for Failure Prevention



Monitoring

System element

ECU

- Intermittent failure detection
- Abnormality checking
- Deviations
- …

Diagnosability for Failure Prevention

- Intermittent failure detection
- Abnormality checking
- Deviations
- …

Warning

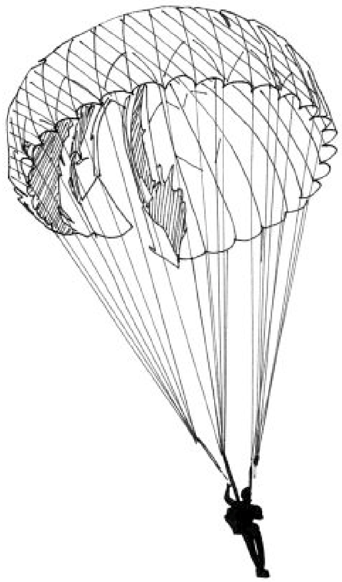
Degraded operation

Maintenance intervention

Fail-Safe States

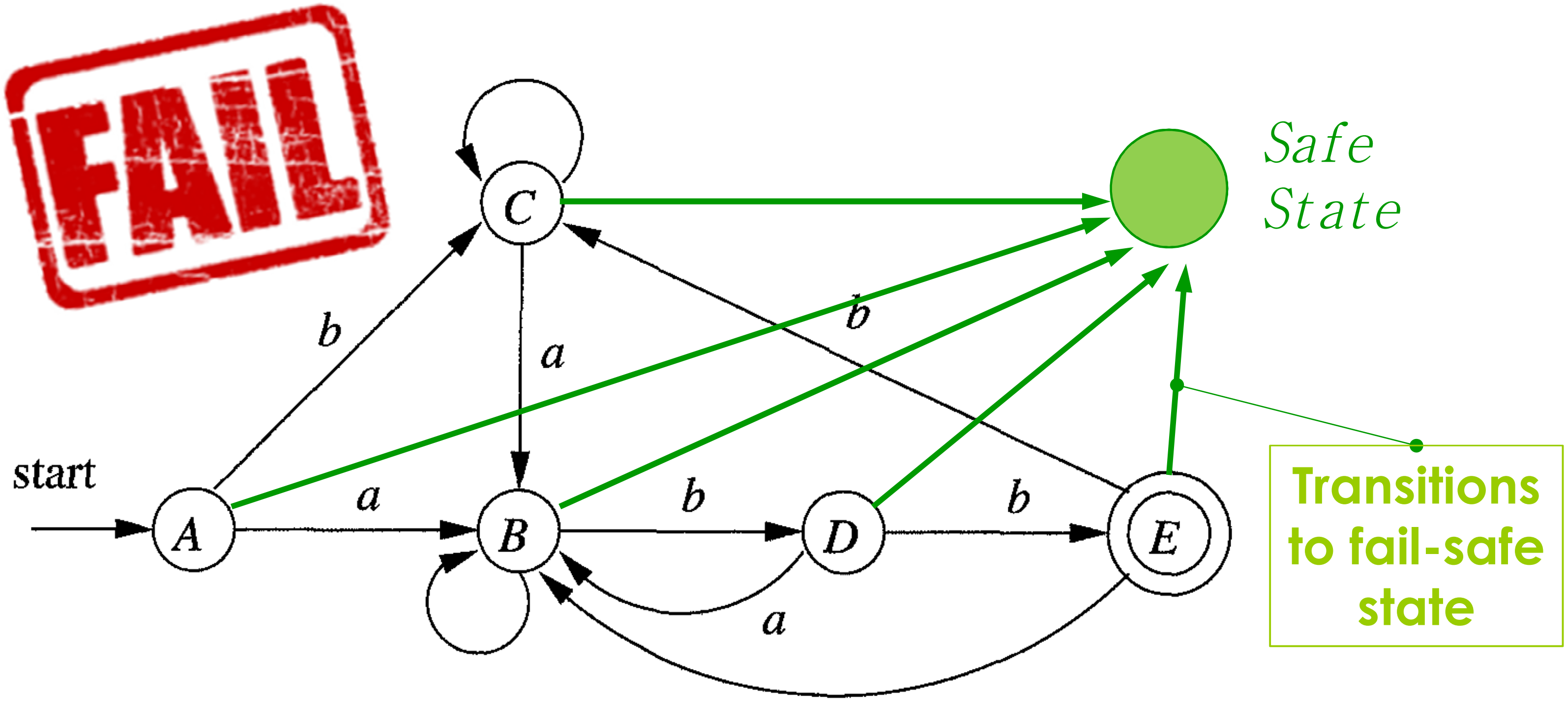
Fail-safe means that a system will not endanger lives or property when it fails.

It will go into a *fail-safe state* and stop working.



https://s-media-cache-ak0.pinimg.com

Fail-Safe States



Safe State

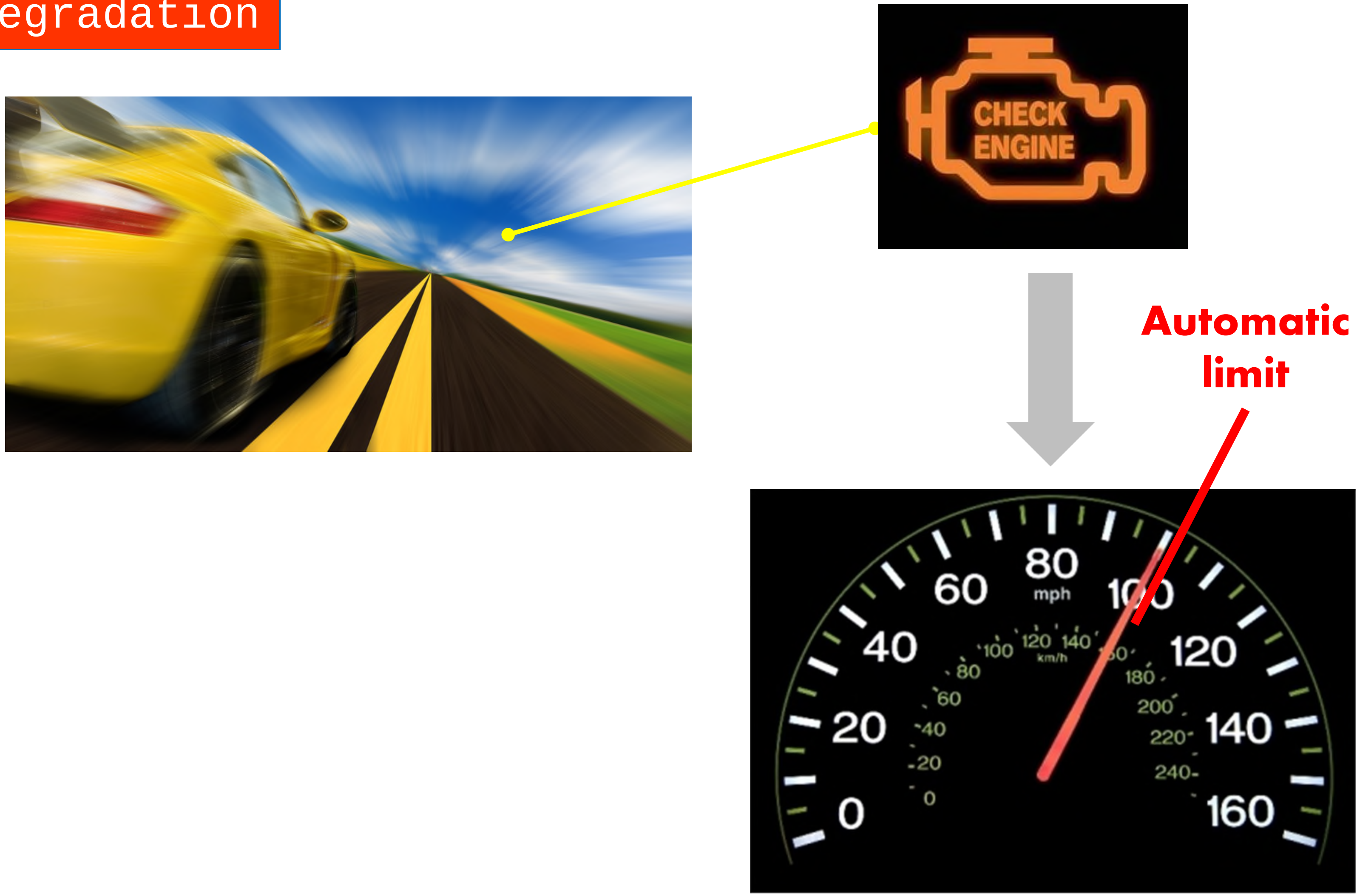
Which is a safe state?

How can we find a safe state?

Difficult Engineering Task

Graceful Degradation

http://www.suggestsoft.com



`Safety architecture`

`Coding guidelines`

`Formal design verification`

`Standards conformance`

`Certification`

`…`

https://www.docmanco.co.nz

https://www.npmjs.com

Safety Engineer:

- Interesting, promising engineering profession
- More and more specialized
- Bright future

Example: Design Verification

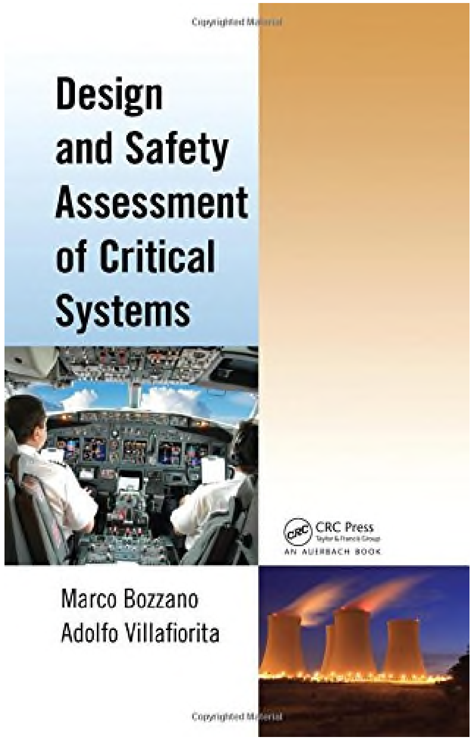
ASIL:
Automotive Safety Integrity Level

Methods		A	B	C	D
		ASIL			
1a	Informal verification by walkthrough	++	+	o	o
1b	Informal verification by inspection	+	++	++	++
1c	Semi-formal verification[a]	+	+	++	++
1d	Formal verification	o	+	+	+

[a] Method 1c can be supported by executable models.

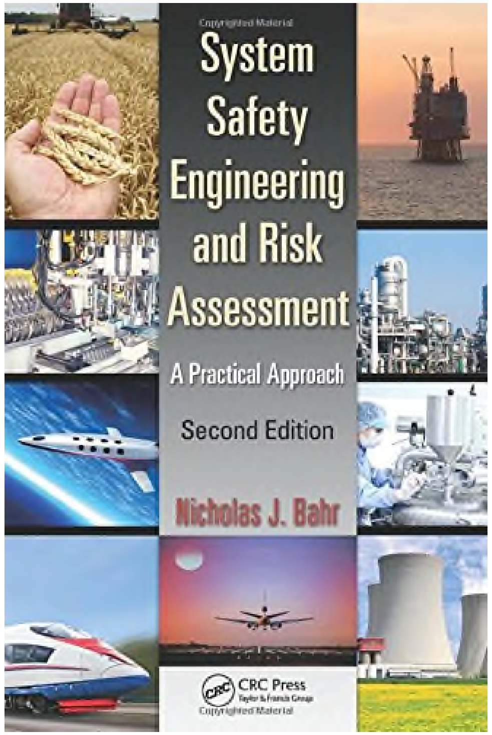
ISO 26262

Textbook



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

Textbook



Nicholas J. Bahr:
System Safety Engineering And Risk Assessment: *A Practical Approach*
Taylor & Francis Inc., USA, 2[nd] revised edition, 2014). ISBN 978-1-466-55160-2

Dependability Property: Security

DEFINITIONS

Security

Information **Security** protects the **confidentiality**, **integrity** and **availability** (CIA) of computer system **data** and **functionality** from *unauthorized and malicious accesses*

https://cdn-01.media-brady.com

Example: Credit Card electronic theft

21/10/2016 10:39
[http://www.**huffingtonpost**.in/2016/10/20](http://www.huffingtonpost.in/2016/10/20)

Massive ATM Card Hack Hits Indian Banks
3.2 Million Debit Cards Affected

A massive debit card hack has reportedly hit major Indian banks such as HDFC Bank, ICICI Bank, Yes Bank, Axis Bank and SBI, compromising as many as 3.2 million debit cards in October 2016

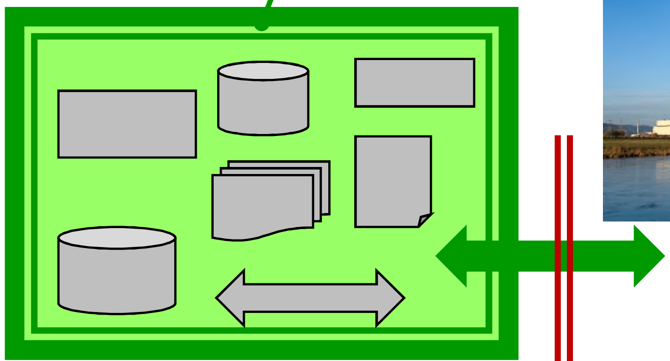
Protection Asset: **Data**
[Credit Card Data]

http://verastic.com

DEFINITIONS

Security =

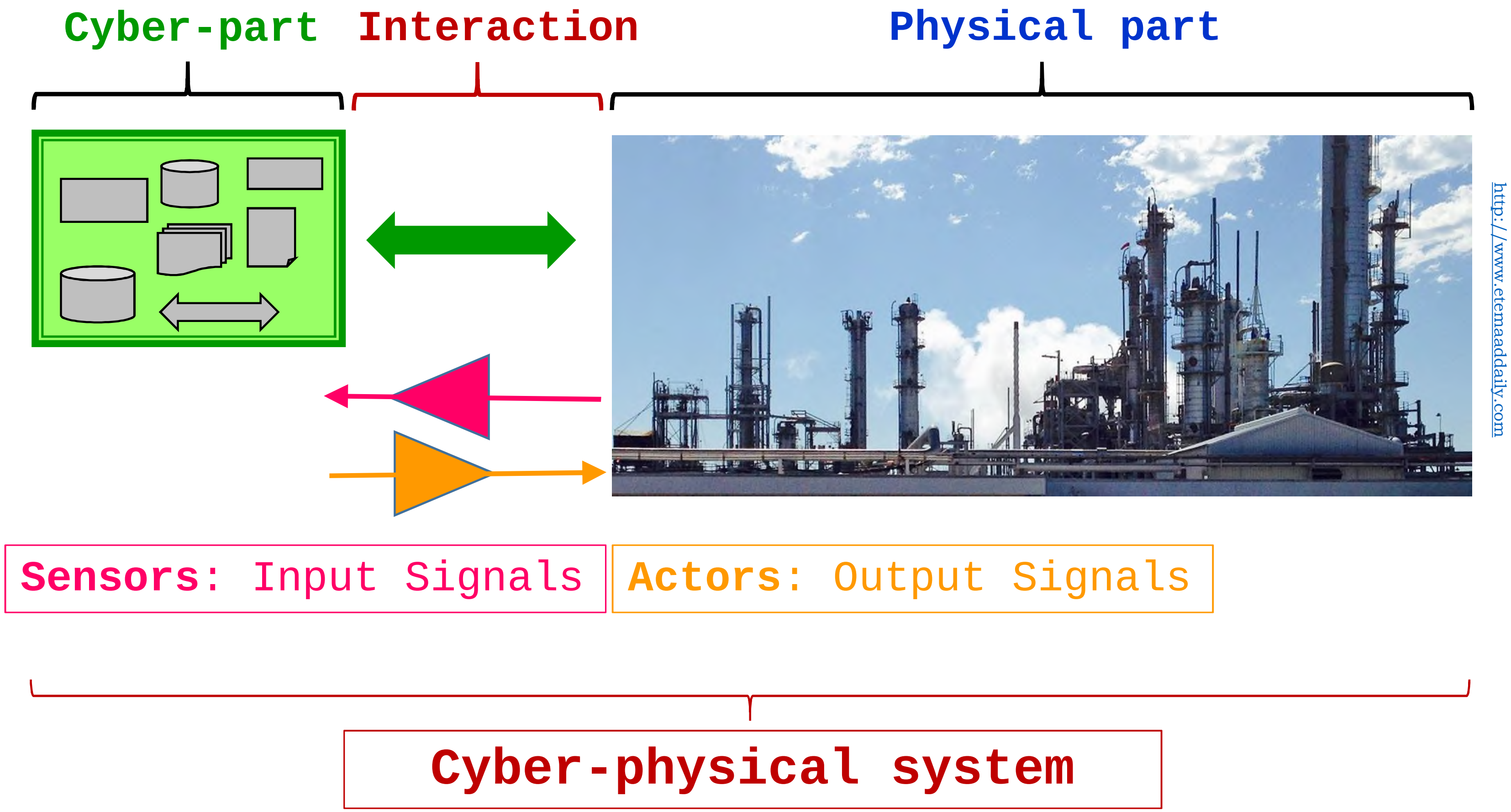
Protection of an information system's assets with a known and acceptable **risk** of a security break



Protection means

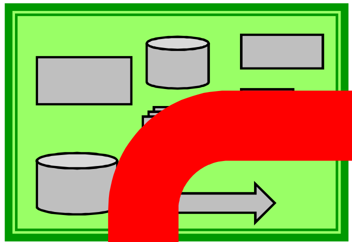
Cyber-physical system

Cyber-physical systems attacks ⇒ **physical** damage



Cyber-physical systems attacks ⇒ **physical** damage

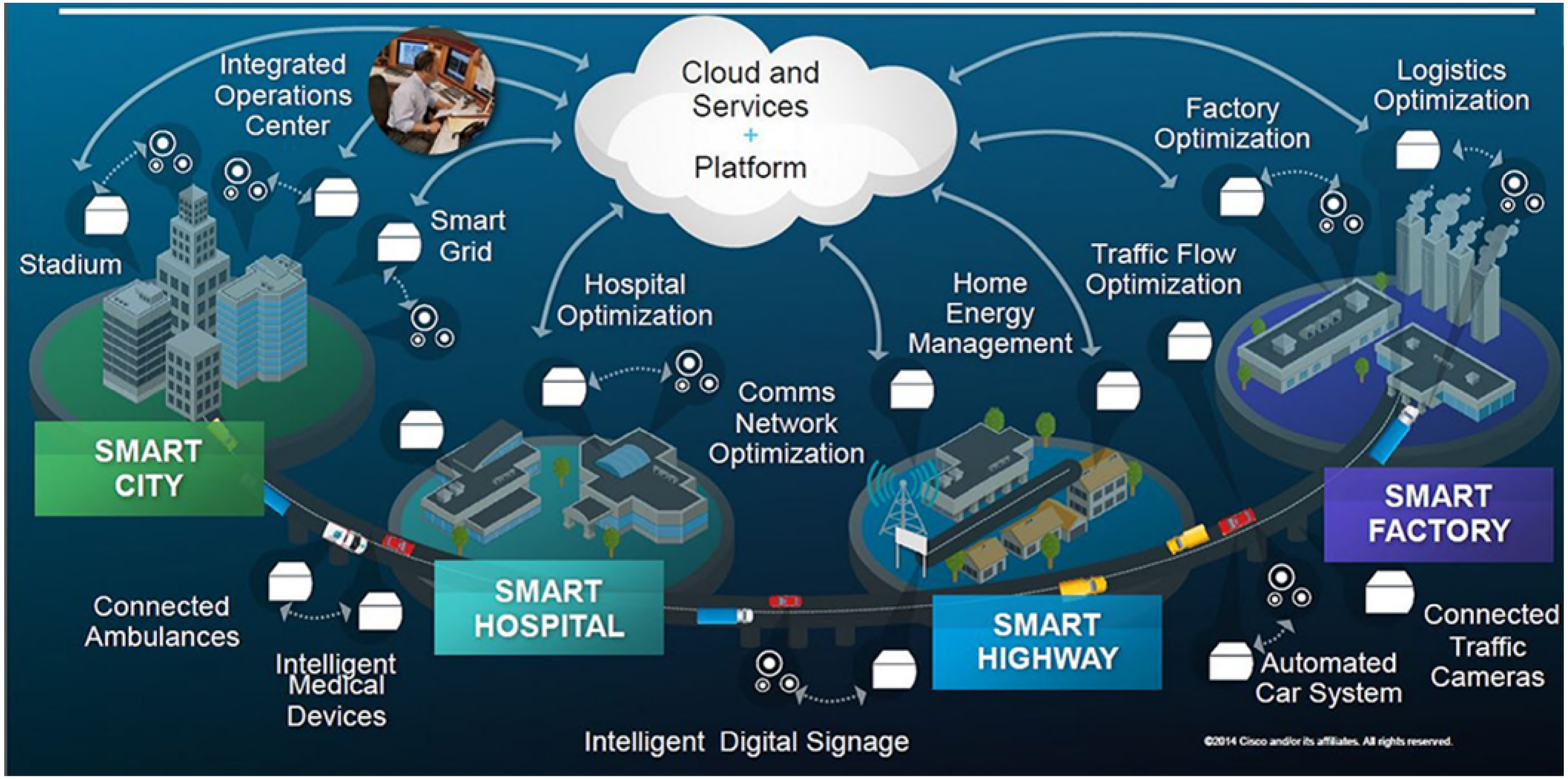
Cyber-part Interaction Physical part



http://www.etemaaddaily.com

Internet of Things (IoT)

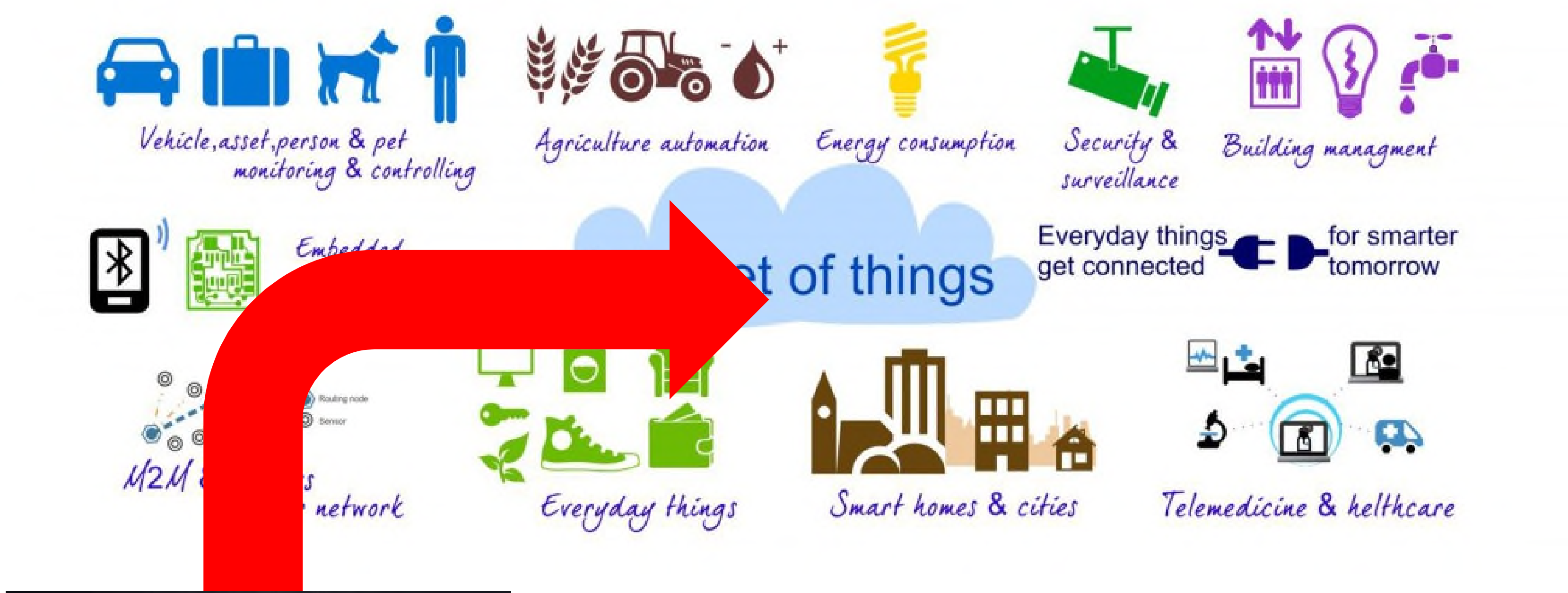
https://cdn.datafloq.com



The Internet of Things (**IoT**) is a computing concept where everyday physical objects are connected to the Internet and are able to identify themselves to other devices and to exchange data and control

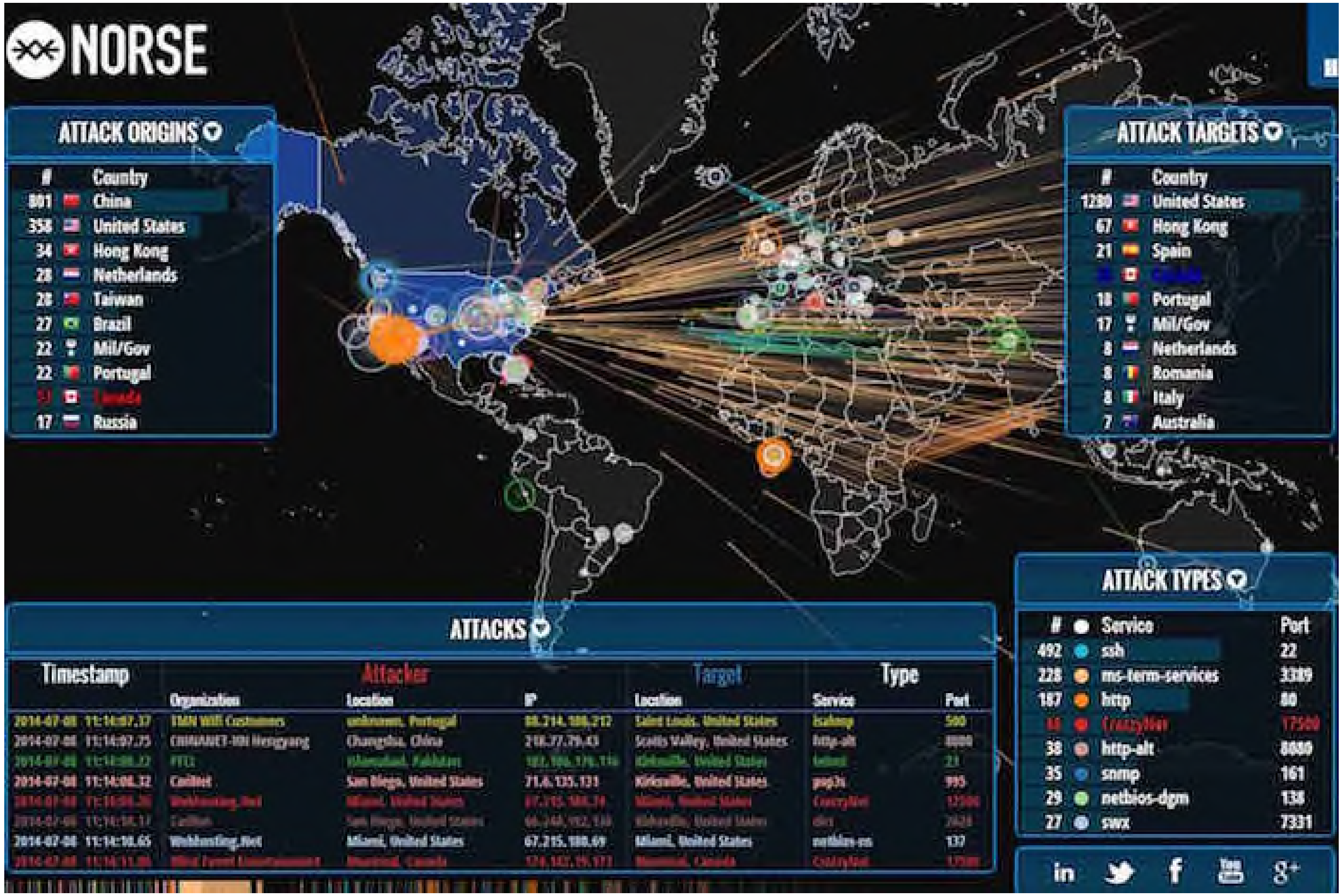
IoT systems attacks ⇒ **physical** damage

https://media.licdn.com



Cyber-physical system attack

Cyber-physical systems attacks ⇒ **CYBER WARFARE**



http://paratusnews.com/wp-content/uploads/2016/10/norse-cyberwar-map-copy.jpg

https://media.licdn.com

Who is ***responsible*** for a successful cyber-physical attack?

The Target !

The Attacker?

http://www.cbronline.com

The Target?

https://media.licdn.com

Who is ***responsible*** for a successful cyber-physical attack?

The Target !

Successful attack = ***vulnerability*** in the attacked system

No possibility to stop an attack

Vulnerability = lack of care and diligence in our system

So – get ready to ***secure*** your system!

http://www.wasistwas.de

... by building secure software

Dependability Taxonomy: Security

Level 1

- **Dependability**
  - Security
  - Safety
  - Certifiability
  - Business Continuity
  - Accountability
  - …

Level 2

- Security
  - Confidentiality
  - Integrity
  - Availability
  - Business Continuity
  - Physical Security
  - Isolation

https://png.pngtree.com

Specific Taxonomy = **Domain-dependent**

Information Security: The CIA-Triad

https://png.pngtree.com

https://vstracking.com

C: Confidentiality

Allows only *authorized users* to access sensitive information and functionality.

I: Integrity

The information and functionality in the system is *correct* and *consistent* at any time (as specified by the rightful owner).

A: Availability

Percentage of time a computer system's information and functionality is *ready* for the intended use.

DEFINITIONS

Information ***security*** protects the confidentiality, integrity and availability of computer system data and functionality from unauthorized and malicious accesses

Protection Assets

- …
- Customer Data
- …

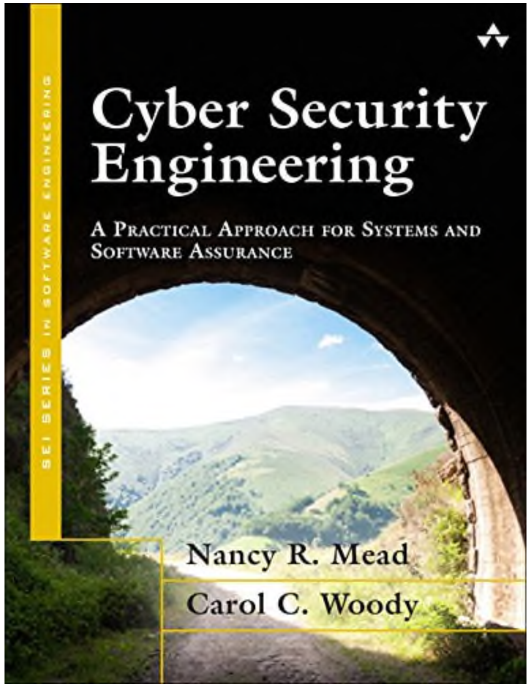
Threats & Dangers

- …
- Unauthorized Access
- …

Protection Means

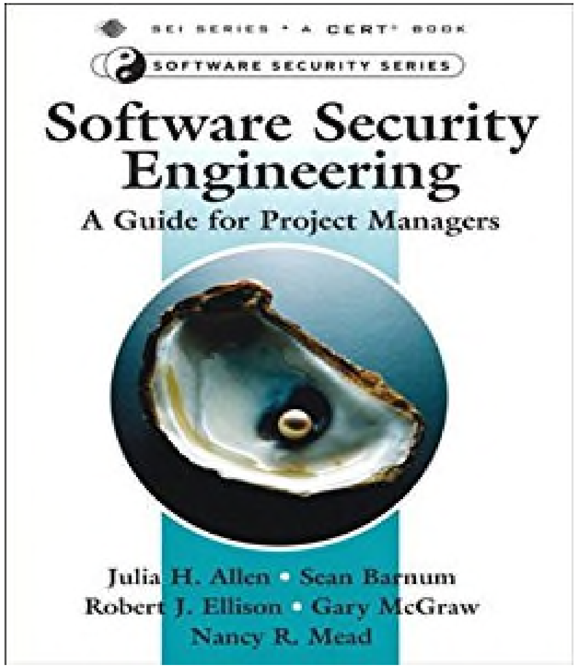
- Authentication
- Authorization
- Access Control
- …

Textbook



Nancy R. Mead, Carol Woody:
Cyber Security Engineering: *A Practical Approach for Systems and Software Assurance*
Addison Wesley, USA, 2016. ISBN 978-0-134-18980-2

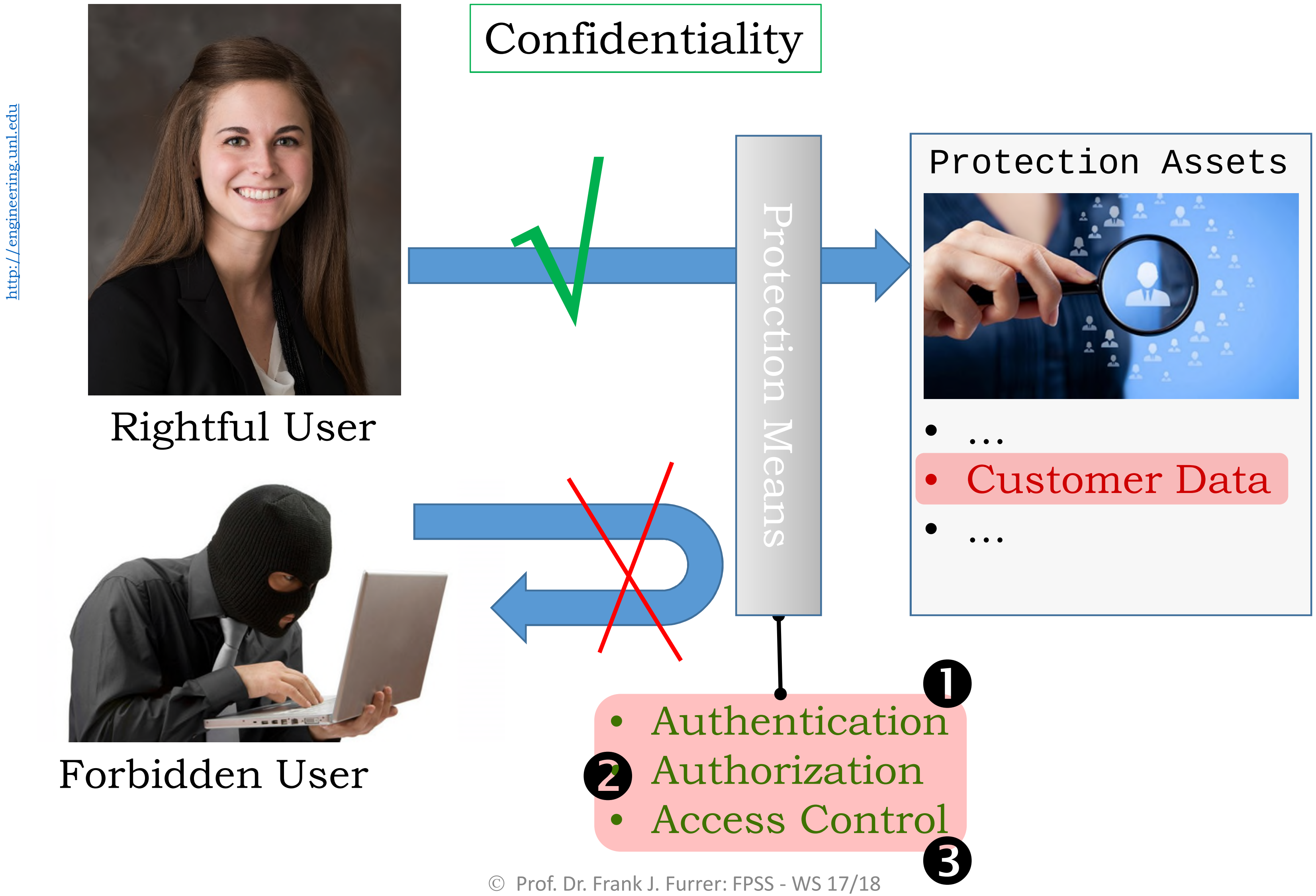
Textbook



Julia H. Allen, Sean Barnum, Robert J. Ellison, Gary McGraw, Nancy R. Mead:
Software Security Engineering: *A Guide for Project Managers*
AddisonWesley Professional, USA, 2008. ISBN 978-0-321-50917-8

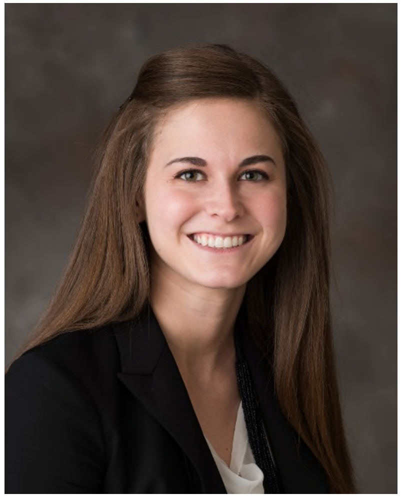
https://vstracking.com

Dependability Property: Confidentiality


Confidentiality
http://engineering.unl.edu
Rightful User
Forbidden User
Protection Means
Protection Assets
• ...
• Customer Data
• ...
• Authentication
• Authorization
• Access Control
❶
❷
❸

Authentication → Establishing, maintaining and using an ***electronic identity***

http://engineering.unl.edu



?

Electronic channel

http://www.rand.org

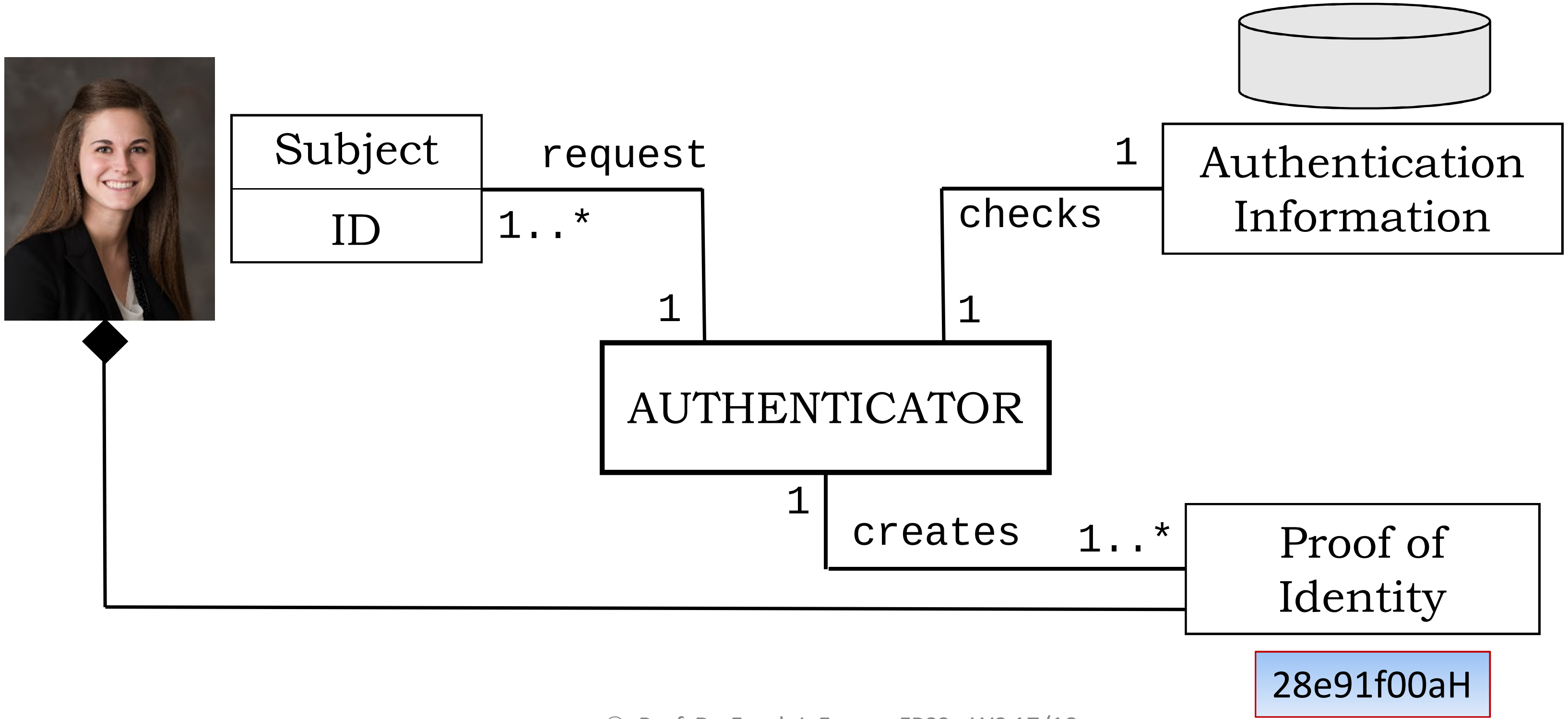
Real world

Computer representation

Authentication

Authenticator Pattern
[Schuhmacher, ISBN 978-0-470-85884-4]

Step 1: Creation

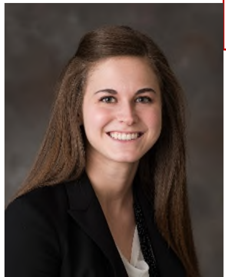


Authentication

Authenticator Pattern

[Schuhmacher, ISBN 978-0-470-85884-4]

Step 2: Use



User Name
Password

Subject

ID

request

1..*

1

checks

1

Authentication Information

1

AUTHENTICATOR

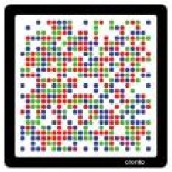
1

creates

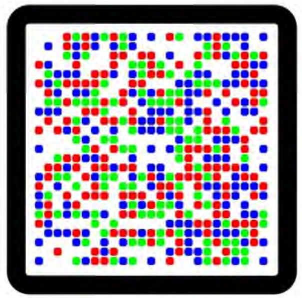
1..*

Proof of Identity

28e91f00aH



http://www.cam.ac.uk



http://jcrs.uol.com.br

Authorization

Authorization:
Allow or **deny** access
to a computer resource

http://stonehousesigns.com

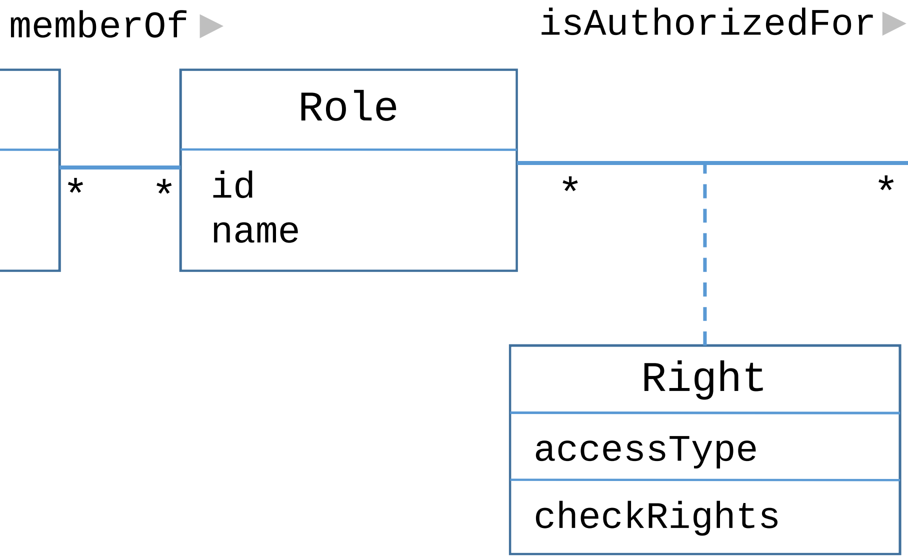
RBAC Pattern
[Fernandez, ISBN 978-1-119-99894-5]

Full Access with Errors Pattern
[Schuhmacher, ISBN 978-0-470-85884-4]

Authorization

RBAC Pattern

[Fernandez, ISBN 978-1-119-99894-5]

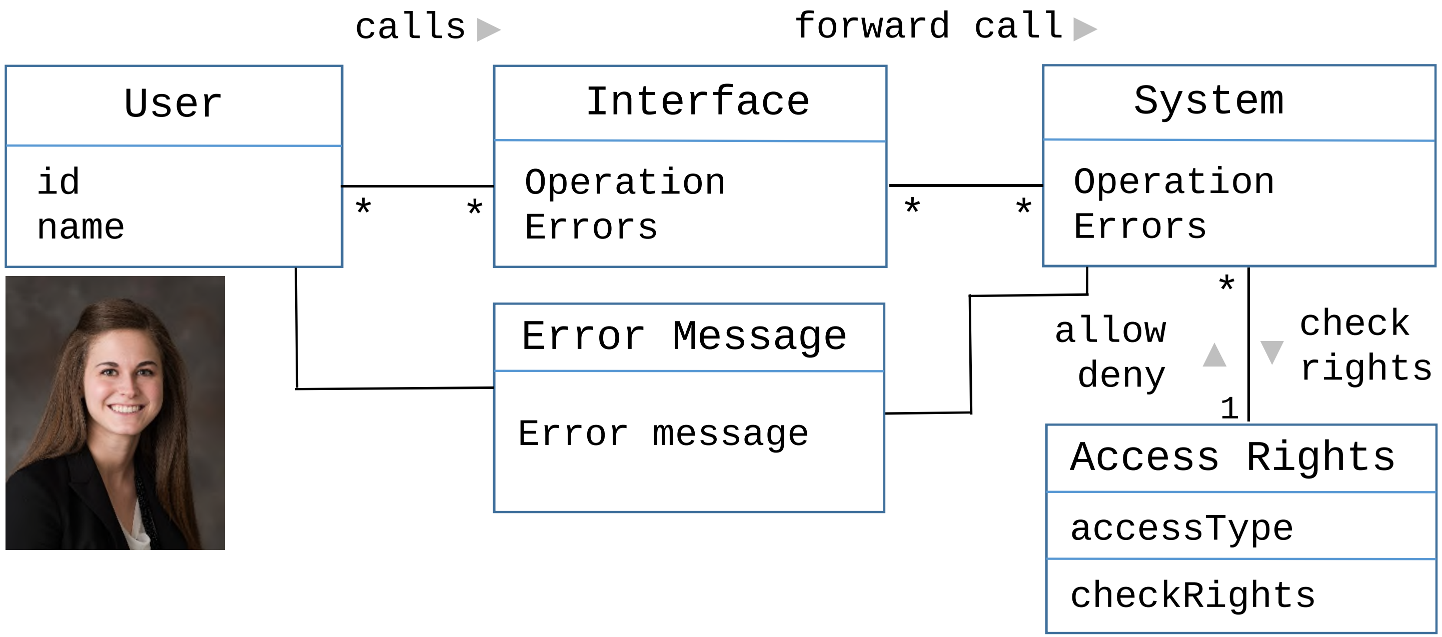


The User and Role classes describe registered users and their predefined roles.
Users are assigned to roles, roles are given rights according to their functions

Authorization

Full Access with Errors Pattern

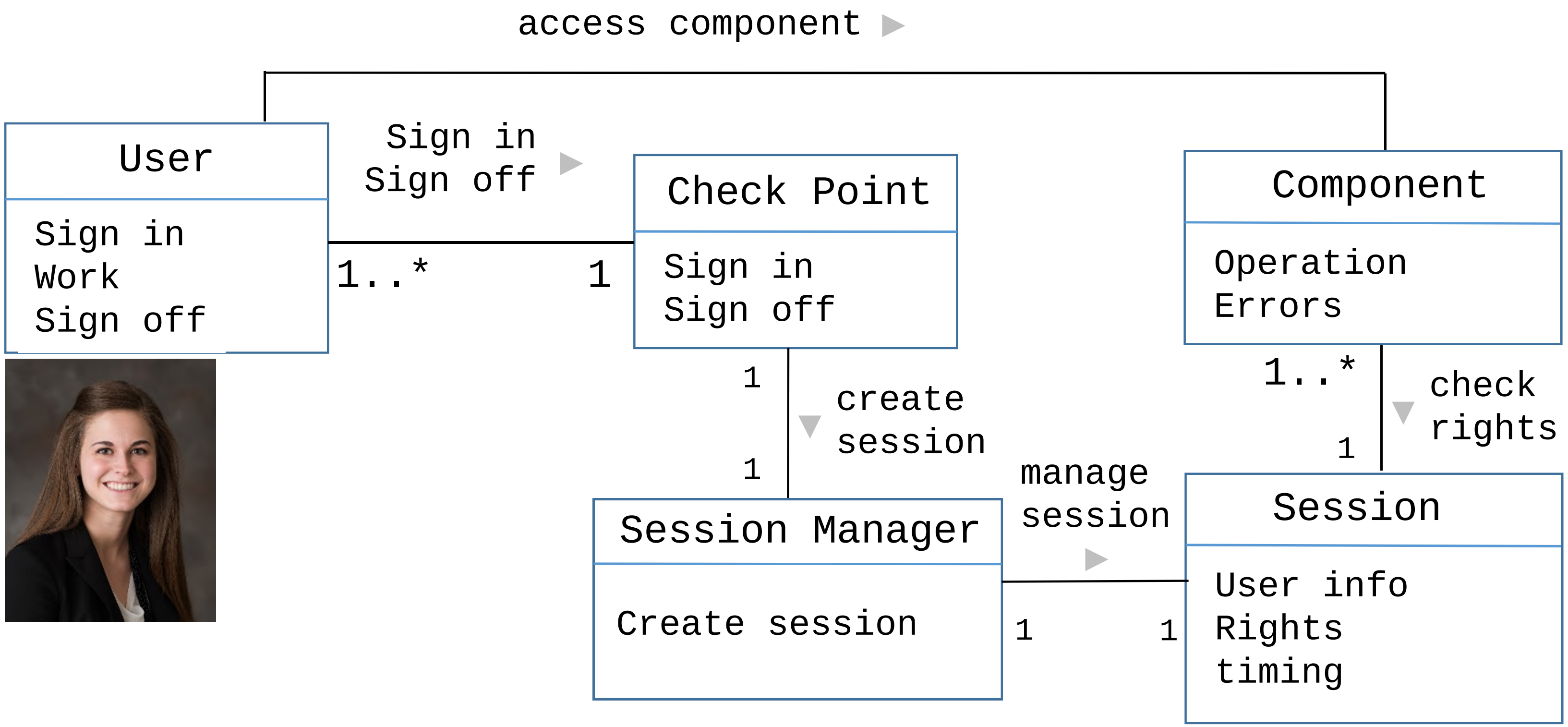
[Schuhmacher, ISBN 978-0-470-85884-4]

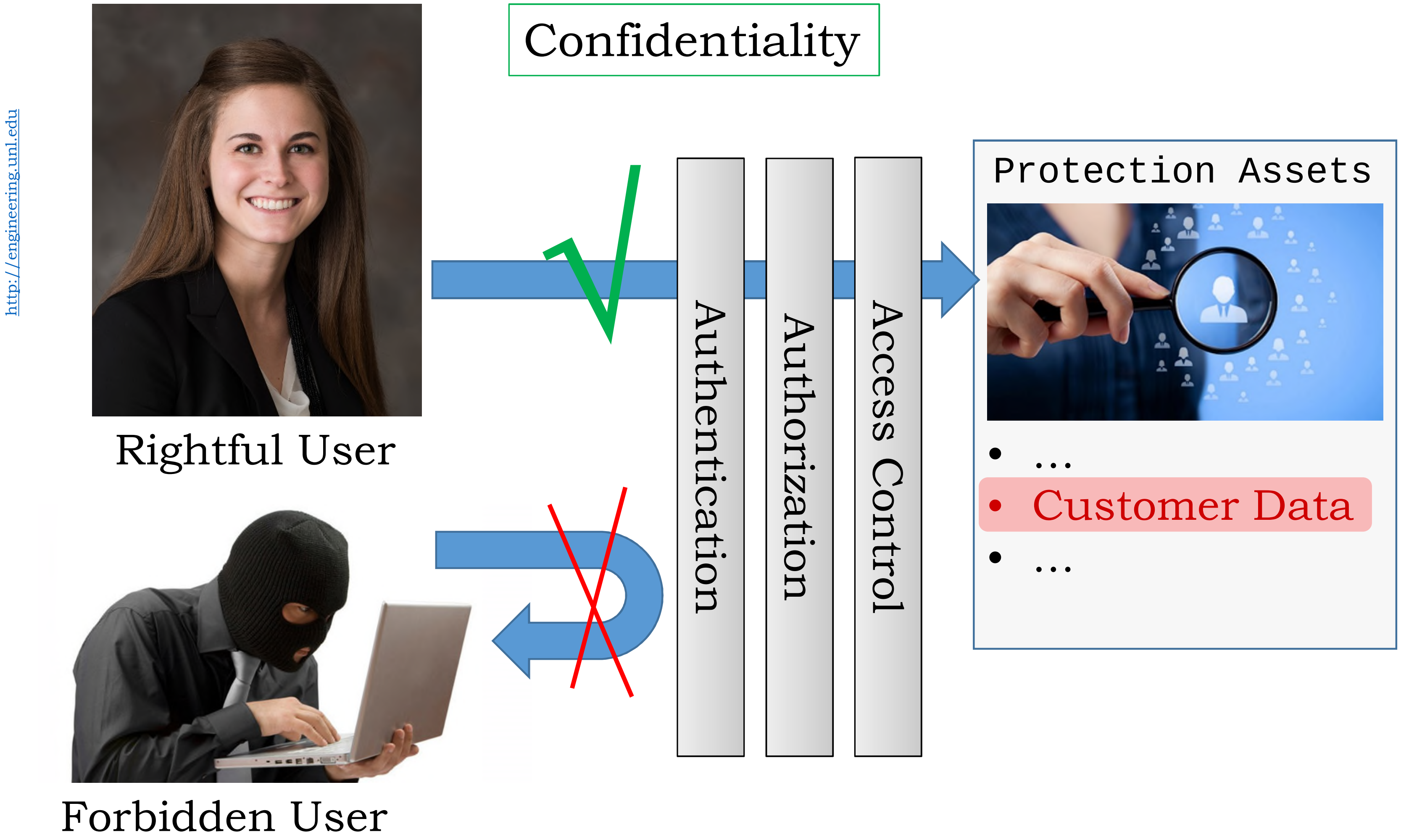


Access Control

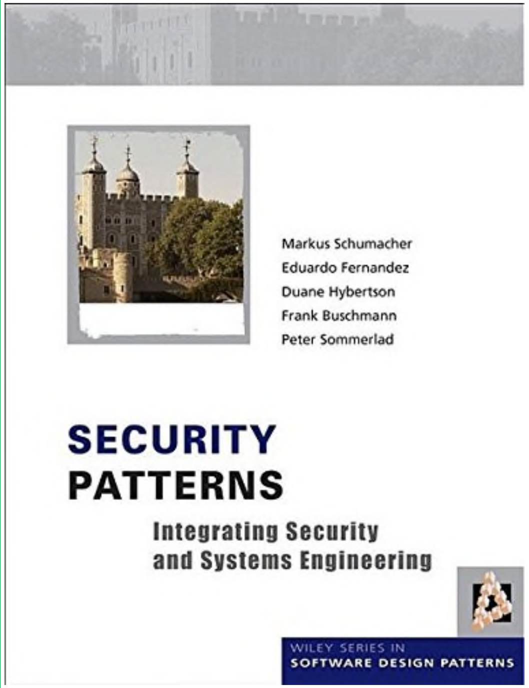
Security Session Pattern

[Schuhmacher, ISBN 978-0-470-85884-4]



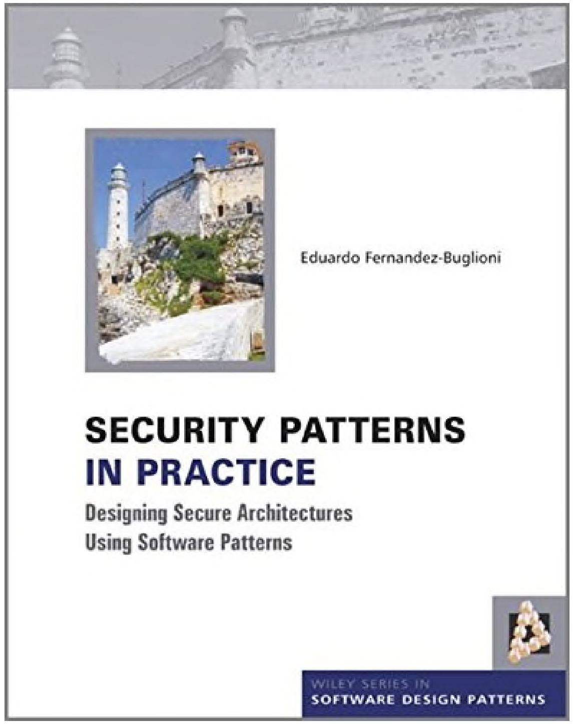

Confidentiality
http://engineering.unl.edu
Rightful User
Forbidden User
Authentication
Authorization
Access Control
Protection Assets
• ...
• Customer Data
• ...

Textbook



Markus Schumacher, Eduardo Fernandez-Buglioni, Duane Hybertson, Frank Buschmann, Peter Sommerlad:
Security Patterns: *Integrating Security and Systems Engineering*
John Wiley, USA, 2005. ISBN 978-0-470-85884-4

Textbook



Eduardo Fernandez-Buglioni:
Security Patterns in Practice: *Designing Secure Architectures Using Software Patterns*
Wiley Inc., 2013. ISBN 978-1-119-99894-5

https://vstracking.com

Dependability Property: Integrity

http://www.phdconstruction.com/

Enterprise Record Archive

Each enterprise is obliged by law to generate records and to keep them available *unaltered* for 10 years

Protection Assets

- Electronic Records (Archive)

Threats & Dangers

- Record modification

Protection Means

- Integrity protection

http://www.nelbaliproperty.com

Data Integrity

How can you assure and prove that a *digital document* has not been altered?

Digital Data Integrity Protection Technology:
Hashing & Digital Signatures

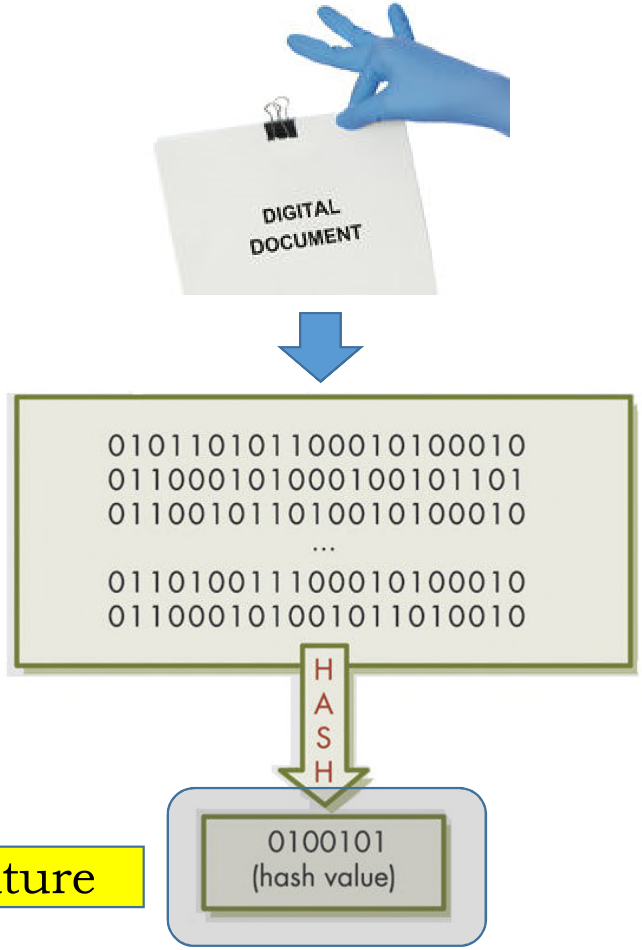
Hashing & *Digital Signatures* are a secure technique to preserve the integrity of a digital document over long periods of time

https://www.tsl.texas.gov/

Data Integrity

http://www.nelbaliproperty.com

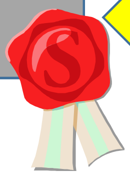
Hashing & Digital Signatures



http://de.dreamstime.com

http://www.3gforensics.co.uk

Document ID
Time Stamp
Hash Value



Digital Signature

Data Integrity Assurance Pattern

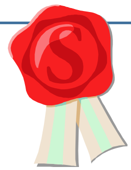
Document/ Records Generation
Document Production

hash & sign ▼

 # Signature Server
Sign document #

deliver ▼

ARCHIVE
Secure document store

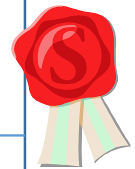


◀ # key

Certification Agency [CA]
signature

▶ # key

Document/ Records Retrieval
Document Access



▲ retrieve

 # Integrity Check
Check document sign

retrieve ▶

automatic retrieve ▼

Integrity Check
Periodic Document Integrity Checking

exception ▶

Integrity Violation Handling
Procedures

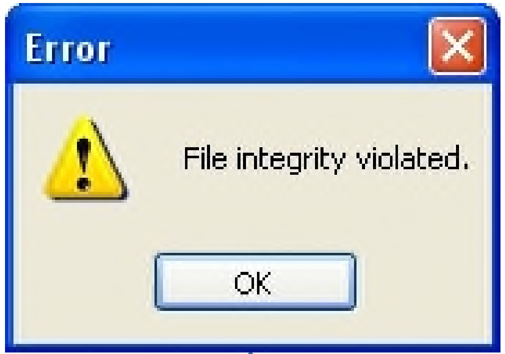
Document Integrity

http://de.dreamstime.com

https://dzone.com

 # Signatures:
- Documents
- Files
- e-Mails
- Messages
- …

Check # signature



Document Integrity

Why are #signatures secure?

RSA-Algorithm

Key generation

r56Hj35mNhrw90ksT

ENcryption Key

SE83k45bNh23Mi14L

DEcryption Key

Check

√

Document Integrity

Why are #signatures secure?

a) Calculation of **EN**cryption key from **DE**cryption key «impossible»

SE83k45bNh23Mi14L

DEcryption Key

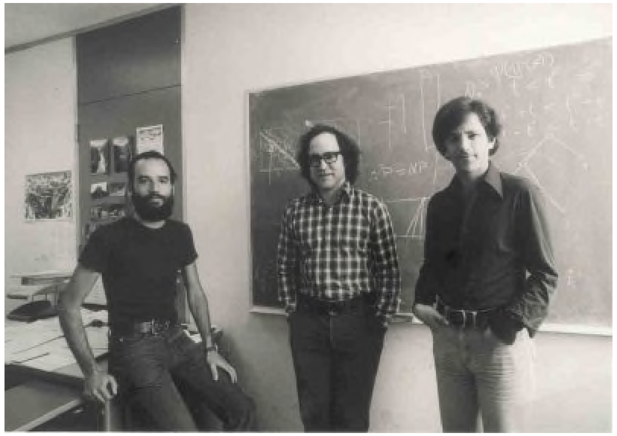
r56Hj35mNhrw90ksT

ENcryption Key

http://info.mide.com

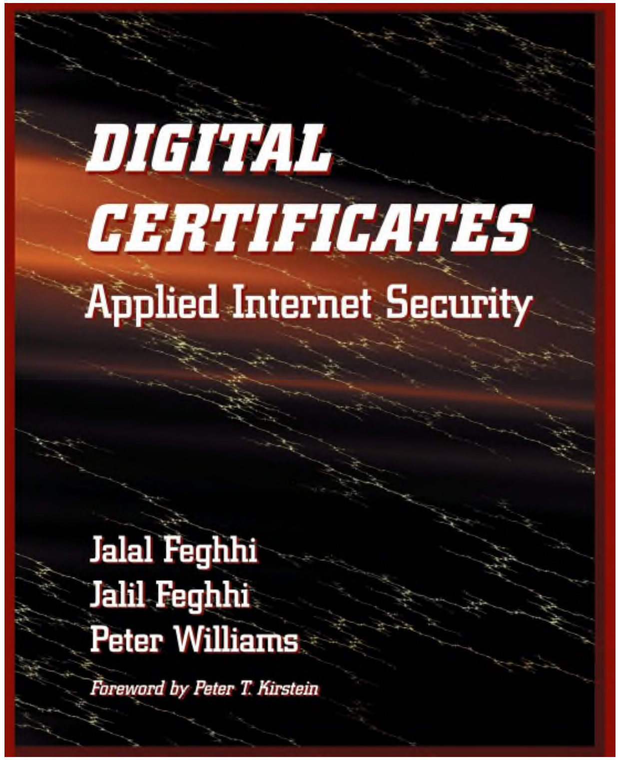
RSA Public Key Algorithm

b) **EN**cryption key kept ***secure***



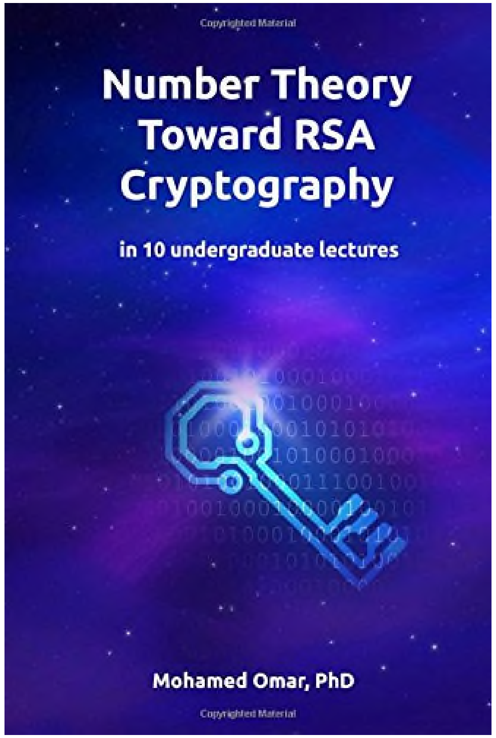
http://imps.mcmaster.ca

Textbook



Jalal Feghhi, Jalil Feghhi, Peter Williams:
Digital Certificates: *Applied Internet Security*
Addison-Wesley Longman, Amsterdam, 1998.
ISBN 0-201-30980-7

Textbook



Mohamed Omar:
Number Theory Toward RSA Cryptography - *in 10 Undergraduate Lectures*
CreateSpace Independent Publishing Platform,
2017. ISBN 978-1-9784-5746-1

https://vstracking.com

Dependability Property: Availability

Availability

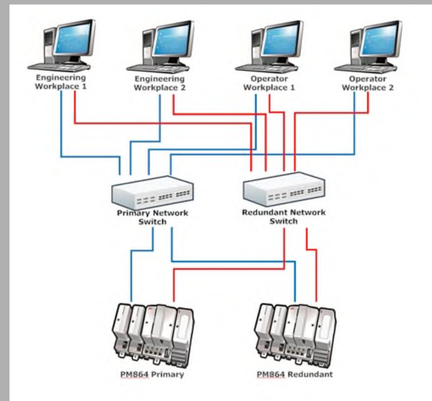
Protection Assets

- Computer/Network system

Threats & Dangers

- HW/SW failure

Protection Means



- Fault Tolerance

Availability

Percentage of time a computer system's information and functionality is *ready* for the intended use.

The math behind availability:

$$\boxed{\text{Availability} = \frac{\text{Uptime}}{\text{Uptime} + \text{Downtime}} \quad [\%]}$$

The math behind availability:

$$\text{Availability} = \frac{\text{Uptime}}{\text{Uptime} + \text{Downtime}} \quad [\%]$$

Example:

Uptime per day: 23.9 hours = **1’434 min**

Downtime per day: 0.1 hours = **6 min**

Availability = 1’434 / (1’434 + 6) = 0.99583 **[99,583 %]**

http://www.datacenterjournal.com

The «9» notation: «Three nines»

Availability Techniques:

Technology:

- Redundancy: Standby/switchover (hot/cold standby)
- Monitoring: early failure detection
- Fallback: Revert to old software release
- Reroute/Network reconfiguration
- Degraded operation

Processes:

- Planned downtimes (Sunday 02:00 – 02:30)
- Fast human intervention

Example:

∅ -50%/+400%

Business Load

\> 1’000 changes/day

Intended Changes

~ 10 disruptions/h

Disruptions

60’000 Servers

2’000 Routers

10’000 Business Databases

12’000 Business Applications

90’000 Workstations

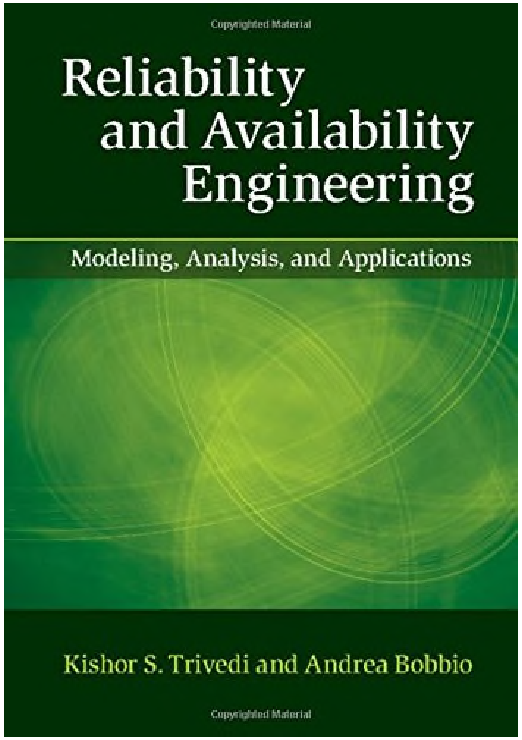
Large Computing Infrastructure

Required Availability:

99.9 %

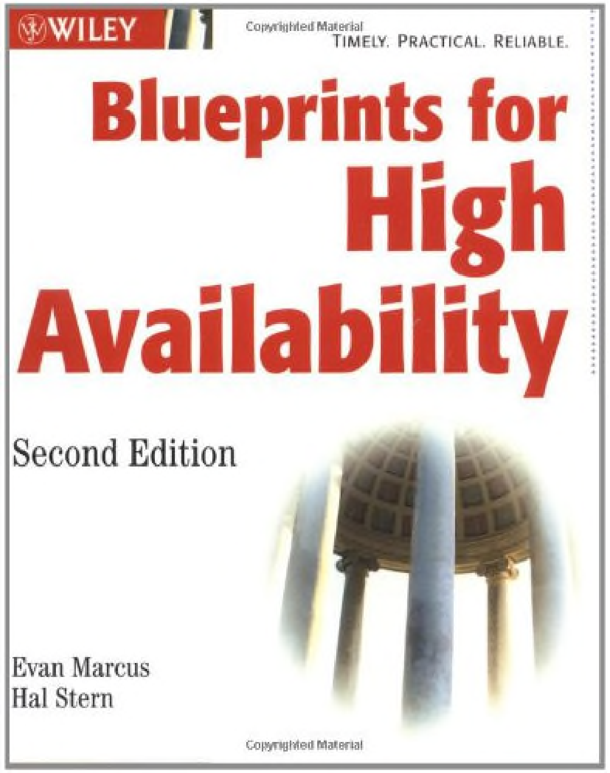
NOT including planned downtime

Textbook



Kishor S. Trivedi, Andrea Bobbio:
Reliability and Availability Engineering:
Modeling, Analysis, and Applications
Cambridge University, 2017. ISBN 978-1-107-09950-0

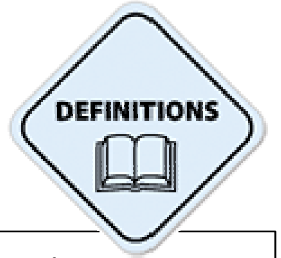
Textbook



Evan Marcus, Hal Stern:
Blueprints for High Availability
John Wiley & Sons, USA, 2nd edition, 2003. ISBN 978-0-471-43026-1

Dependability Property: Real-Time Capability

http://www.watermarklearning.com

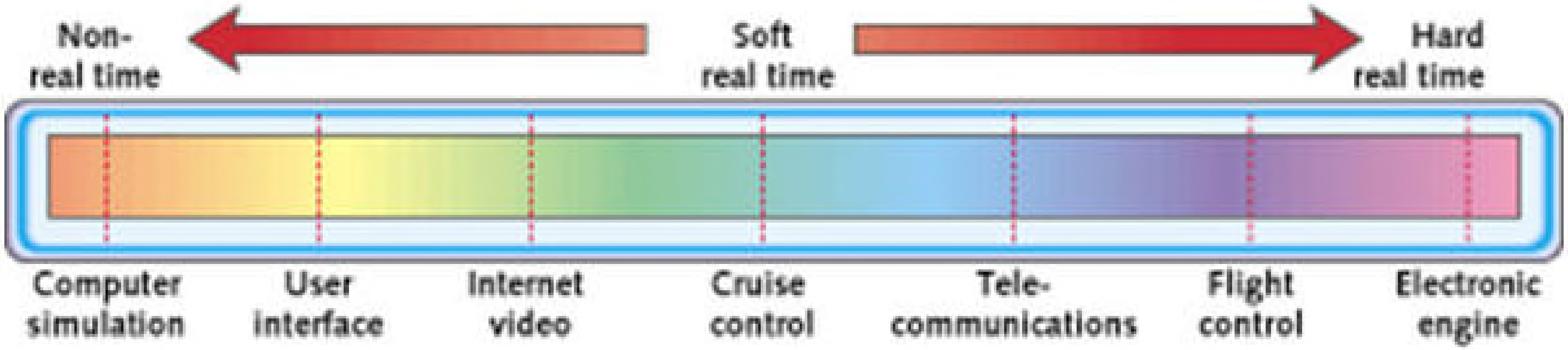


Real-time computing (RTC), or reactive computing describes hardware and software systems subject to a ***real-time constraint***, for example from event to system response.
Real-time systems must guarantee response within specified time constraints, often referred to as ***deadlines***

https://en.wikipedia.org/wiki/Real-time_computing

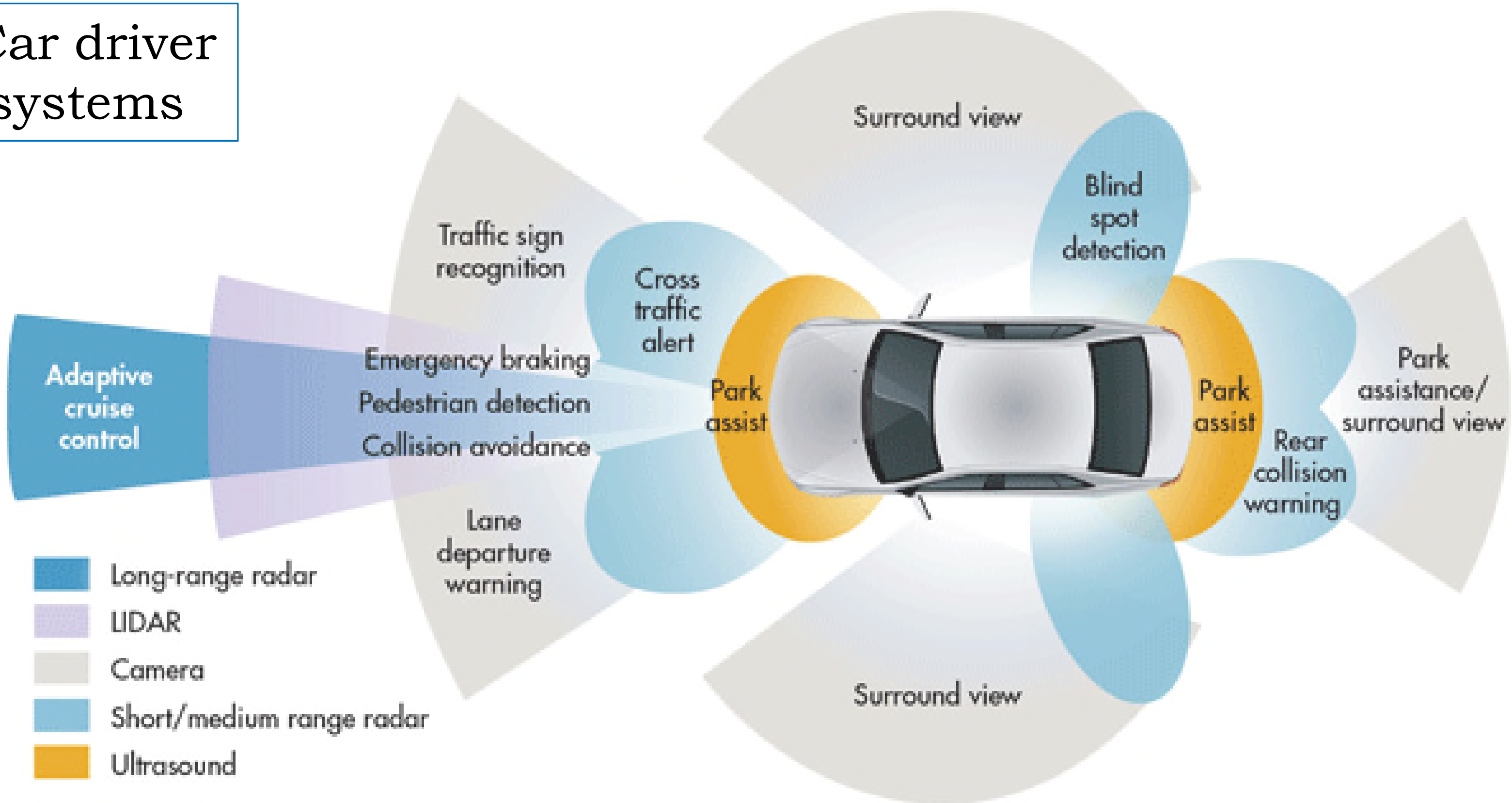
DEFINITIONS

Real-time capability is the capability to react to events in a *predictable*, *guaranteed* time



https://www.logicsupply.com

Example: Car driver assistance systems

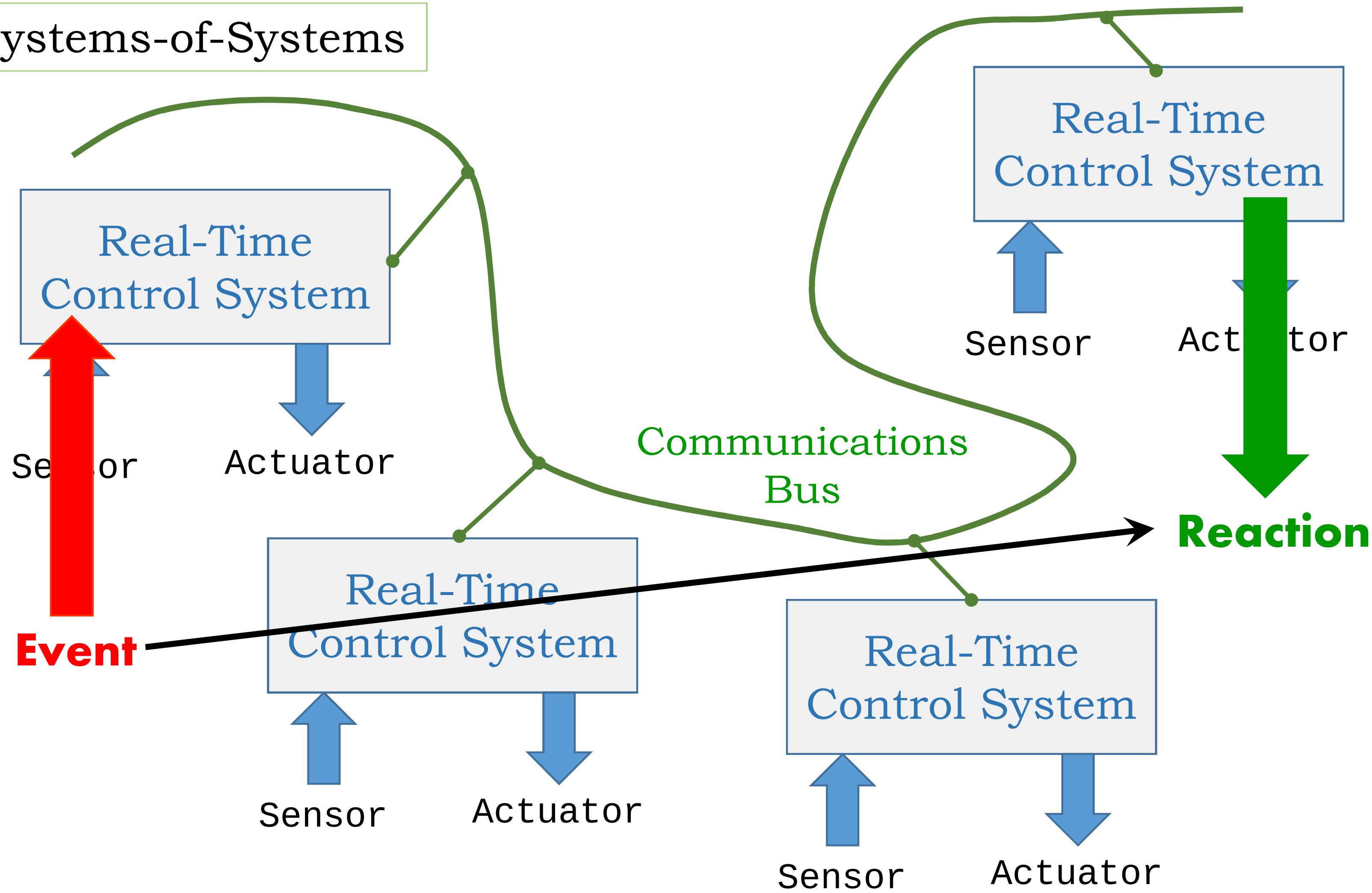


http://electronicdesign.com

Data acquisition ⇒ processing ⇒ event ⇒ decision ⇒ reaction

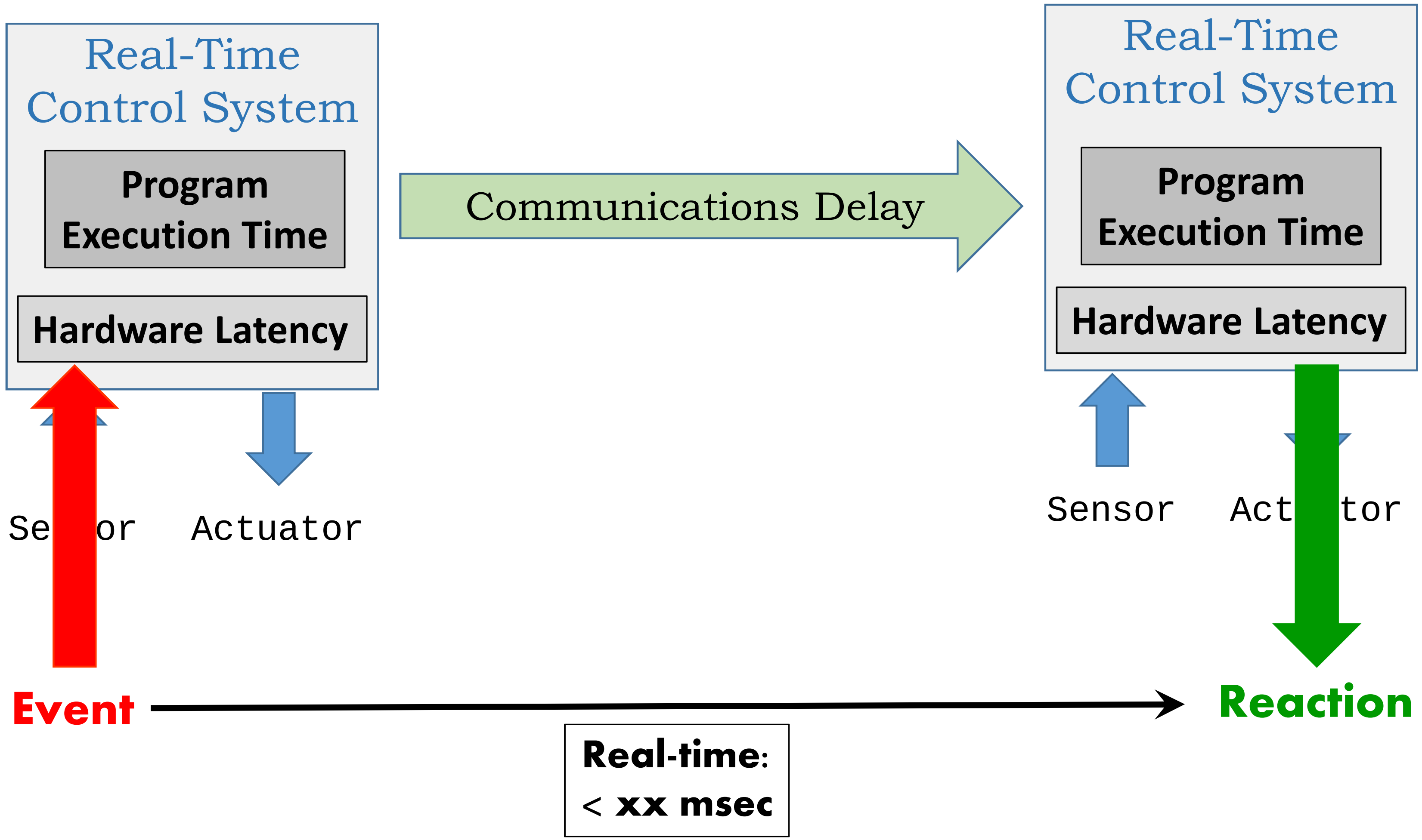
Real-time: max. xx msec

Real-world: Systems-of-Systems

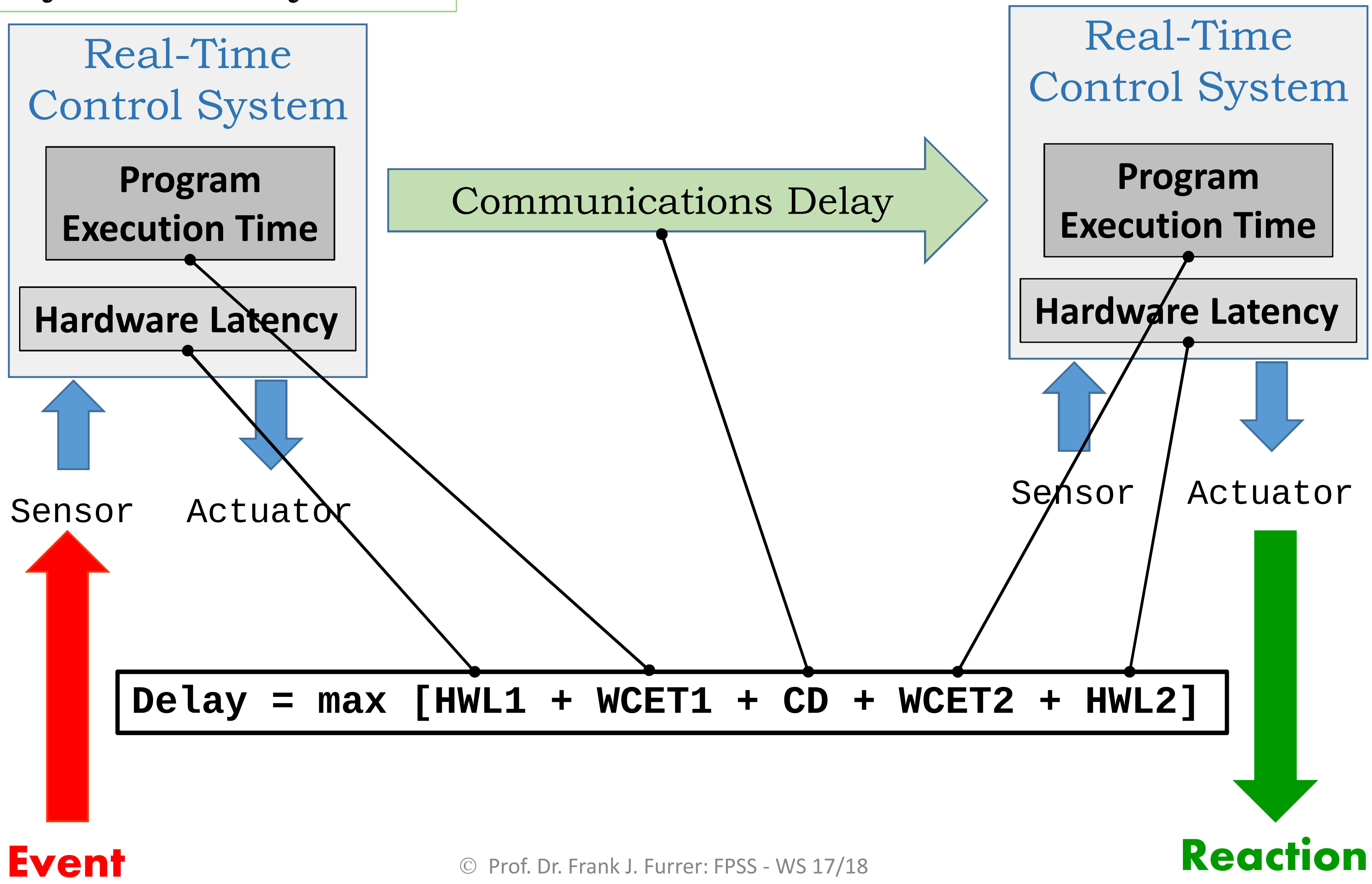


Real-time: < xx msec

Real-world: Systems-of-Systems



Real-world: Systems-of-Systems



Delay = max [HWL1 + WCET1 + CD + WCET2 + HWL2]

Real-world: Systems-of-Systems

Delay = max [HWL1 + WCET1 + CD + WCET2 + HWL2]

Communication Delay:
Max: xx sec
Exact: xx sec

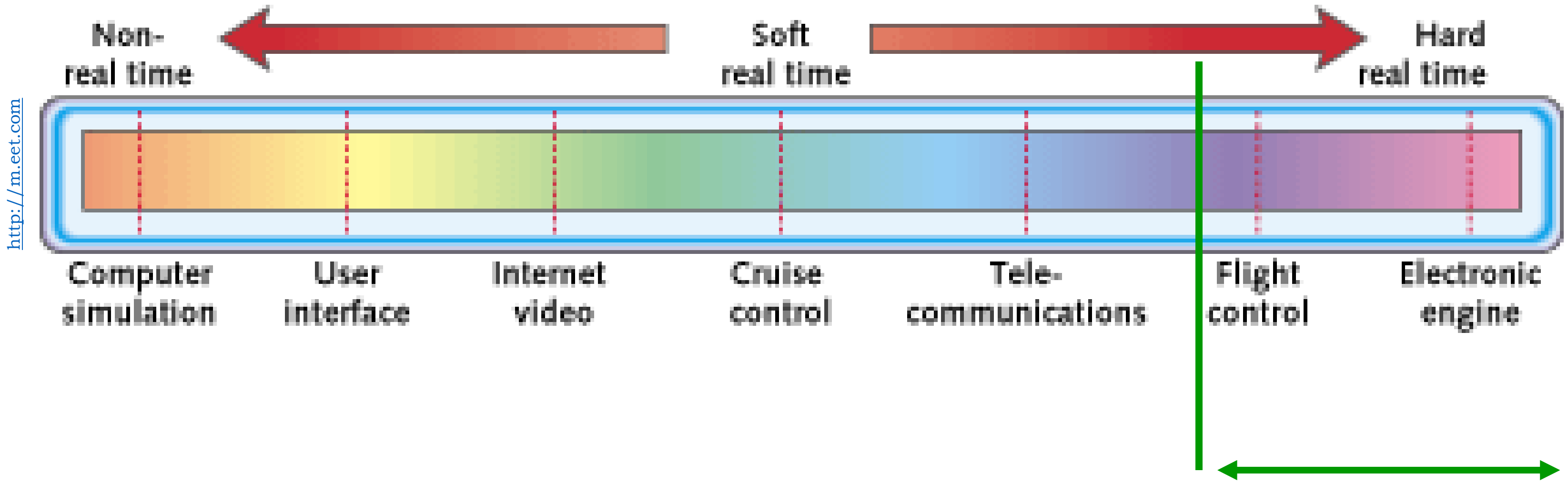
WCET:
Worst Case Execution Time
Max: xx sec

Hardware Latency Time:
Max: xx sec

Extremely **difficult** to estimate and guarantee

Real-world: Systems-of-Systems

Delay = max [HWL1 + WCET1 + CD + WCET2 + HWL2]

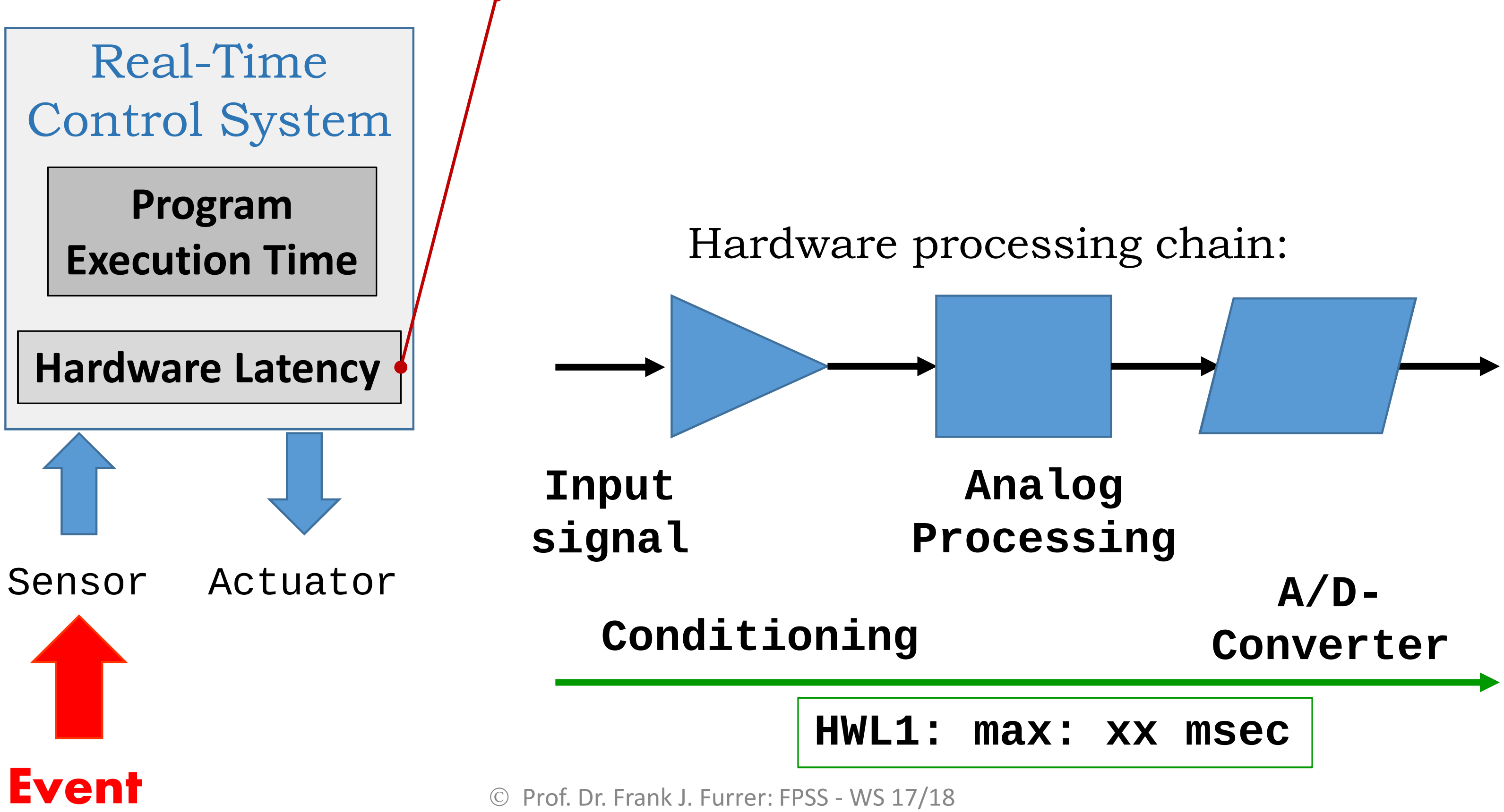


Real-time systems must guarantee response within specified time constraints ***under all operating conditions***

Safety-critical systems

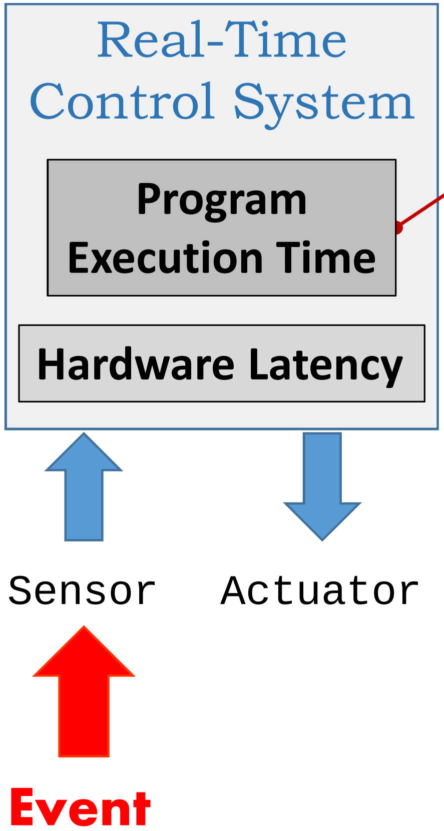
Real-world: Systems-of-Systems

Delay = max [HWL1 + WCET1 + CD + WCET2 + HWL2]



Real-world: Systems-of-Systems

Delay = max [HWL1 + $\boxed{\text{WCET1}}$ + CD + WCET2 + HWL2]



WCET: Worst Case Program Execution Time

DEFINITIONS

The worst-case execution time (**WCET**) of a computational task is the ***maximum length of time*** the task could take to execute on a specific hardware platform

Real-world: Systems-of-Systems

Delay = max [HWL1 + WCET1 + CD + WCET2 + HWL2]

Program Execution Time

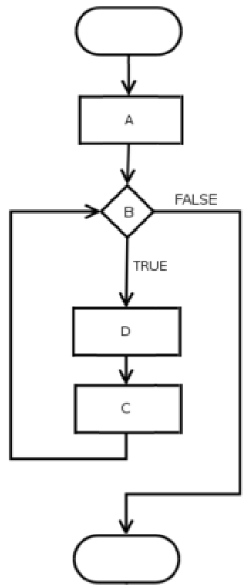
WCET: Worst Case Program Execution Time

```
#include <stdio.h>
#include <stlib.h>
#include <string.h>
#include <fcntl.h>
#include <sys/shm.h>
#include <sys/stat.h>

int main()
{
/* the size (in bytes) of shared memory object */
const int SIZE 4096;
/* name of the shared memory object */
const char *name = "OS";
/* strings written to shared memory */
const char *message_0 = "Hello";
const char *message_1 = "World!";

/* shared memory file descriptor */
int shm_fd;
/* pointer to shared memory obect */
void *ptr;

   /* create the shared memory object */
   shm_fd = shm_open(name, O_CREAT | O_RDRW, 0666);

   /* configure the size of the shared memory object */
   ftruncate(shm_fd, SIZE);

   /* memory map the shared memory object */
   ptr = mmap(0, SIZE, PROT_WRITE, MAP_SHARED, shm_fd, 0);

   /* write to the shared memory object */
   sprintf(ptr,"%s",message_0);
   ptr += strlen(message_0);
   sprintf(ptr,"%s",message_1);
   ptr += strlen(message_1);

   return 0;
}
```

Figure 3.17 Producer process illustrating POSIX shared-memory API.

https://www.cs.uic.edu

for(A;B;C)
D;

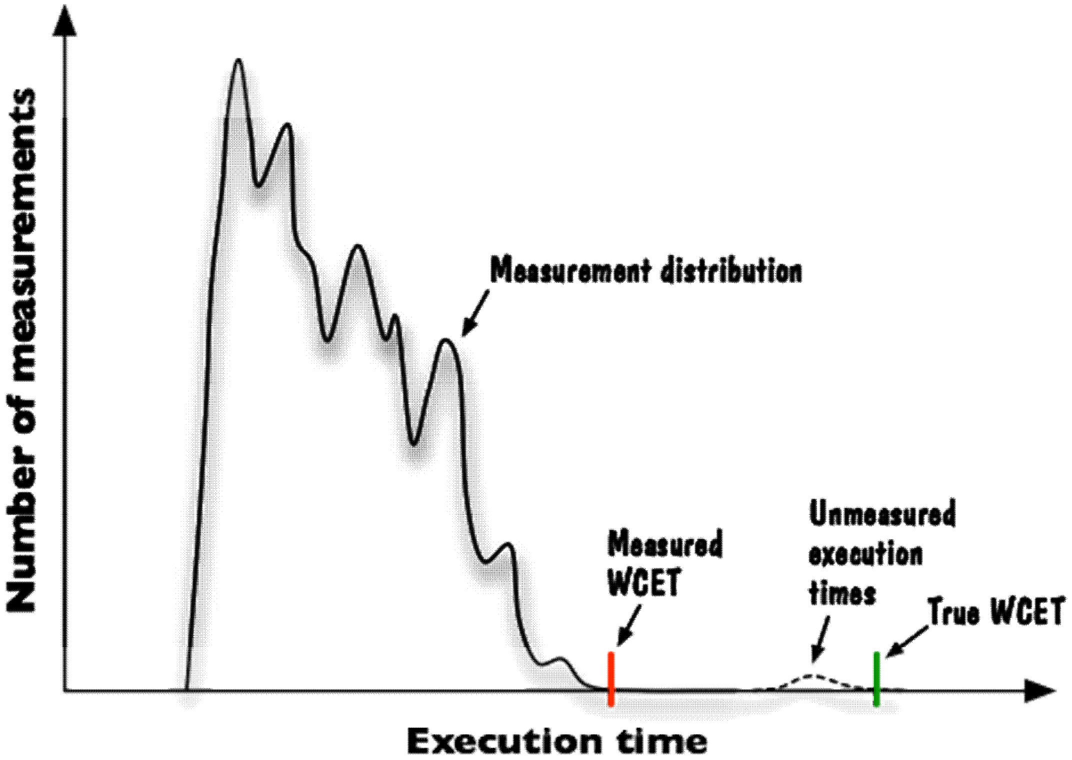


WCET:
Longest
possible path trough the program
$\Rightarrow$ $msec_{max}$

Delay = max [HWL1 + WCET1 + CD + WCET2 + HWL2]

Program Execution Time

WCET: Worst Case Program Execution Time



http://volta.sourceforge.net

Many methods & tools for the WCET determination exist
⇒ Very important parameter for hard real-time systems!

Real-world: Systems-of-Systems

Delay = max [HWL1 + WCET1 + CD + WCET2 + HWL2]

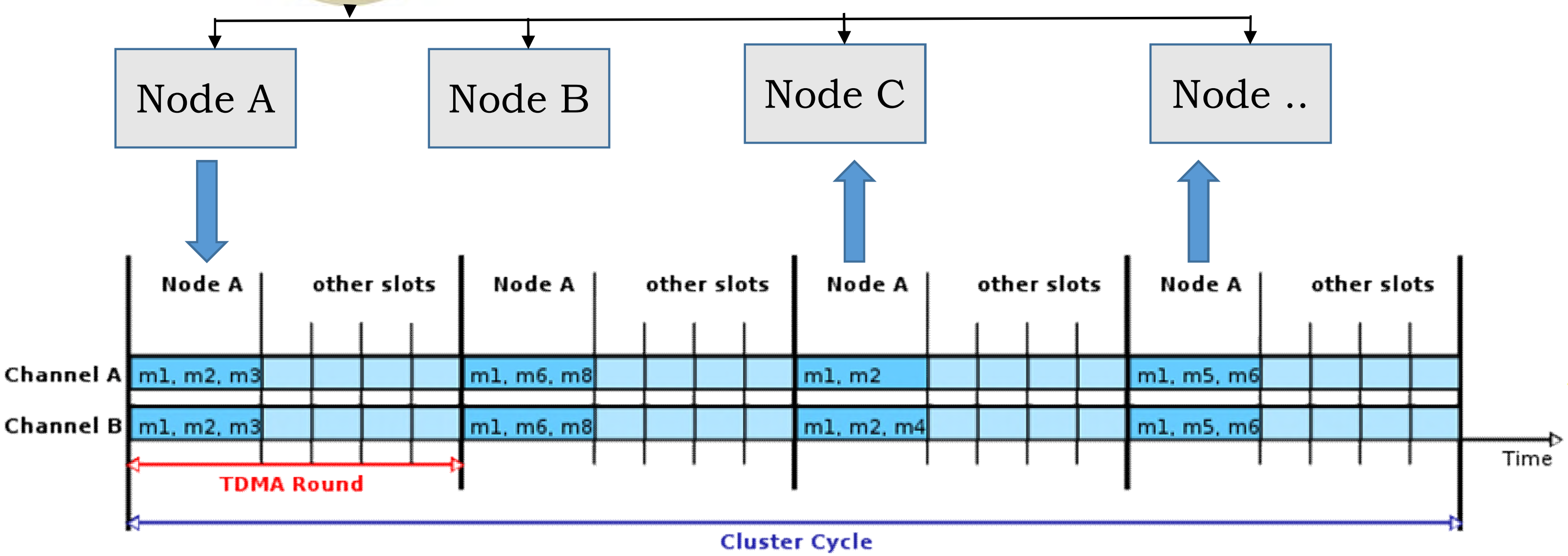
Communications Delay

Communication Delay:
Max: xx sec
Exact: xx sec

Real-time bus: e.g. TTA, Flexray

Example: Real-time Bus (Time-Triggered Architecture **TTA**)

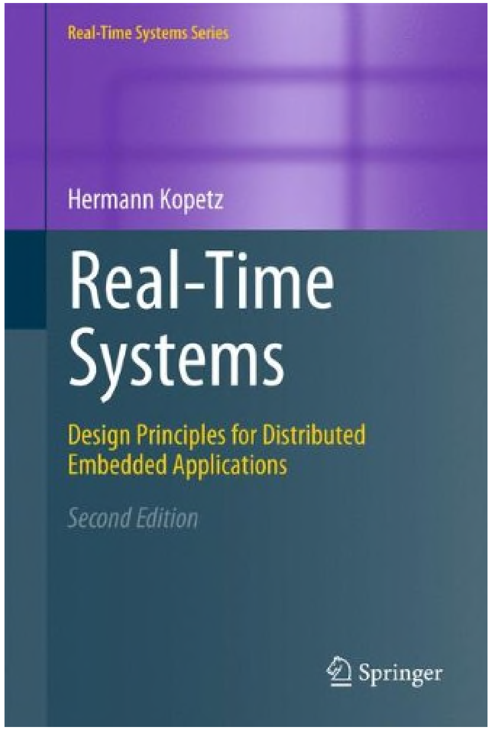
Master Clock:
`Clock Synchronization Algorithm`



Guaranteed real-time behaviour of the channel

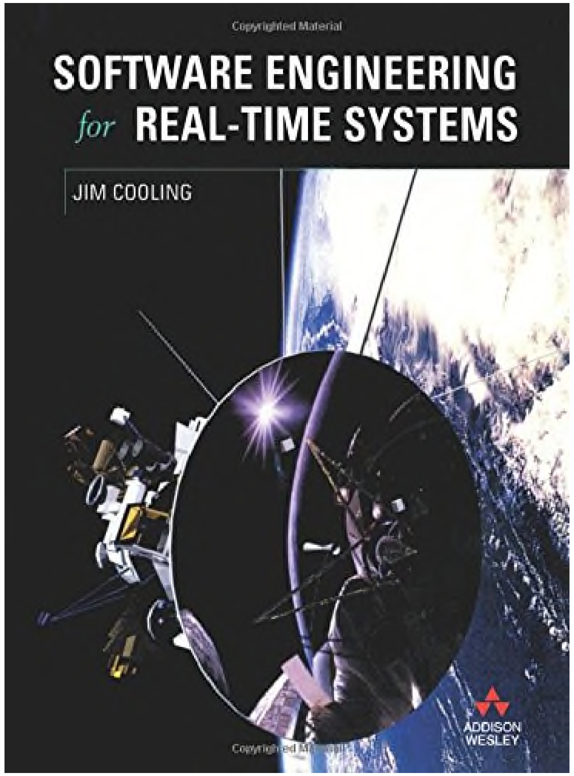
The messages are transported in exactly defined and assigned time slots, based on precise clock synchronization in all nodes

Textbook



Hermann Kopetz:
Real-Time Systems: *Design Principles for Distributed Embedded Applications*
Springer-Verlag, 2nd edition, 2011. ISBN 978-1-441-98236-0

Textbook



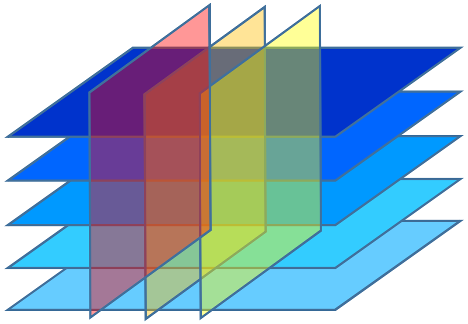
Jim Cooling
Software Engineering for Real-Time Systems
Addison Wesley, USA, 2002. ISBN 978-0-201-59620-5

Dependability Engineering and Methodology

- New project
- System extension

Architecture team

Reqs, Specs



Changeability architecting

Dependability architecting

Development, Implementation, Deployment, Operation

Principles
Patterns
Frameworks

Consistency assurance, quality checking, validation, verification

Dependability Methodology

DEFINITIONS

Methodology:

A system of *principles* and *rules* from which specific methods or procedures may be derived to interpret or solve different problems within the scope of a ***particular discipline***

Note: Unlike an algorithm, a methodology is not a formula but a set of practices.

http://www.businessdictionary.com

Particular discipline

= ***Building dependable systems***

Dependability Methodology

Part 1: Dependability Taxonomy

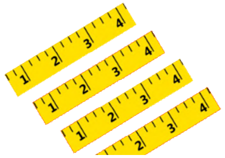
Quality property scorecard

#	System Quality Property	Weight 0: irrelevant 10: highest importance
	Primary Characteristics	
1	Business Value	**10**
2	Agility	**10**
Resilience:		
3	Safety	9
4	Fault-Tolerance	9
5	Compliance to laws & regulations	9
6	Integrity (Sensor Data)	9
7	Availability	8
8	Security	7
9	Diagnosability	6
	Secondary Characteristics	
10	Resources (Memory, CPU, ...)	8
11	Compliance to industry-standards	7
12	Usability (User Interfaces)	9
etc		

Specific Taxonomy = Domain-dependent

- Resilience
  - Security
    - Confidentiality
    - Integrity
    - Availability
  - Safety
  - Certifiability
  - Business Continuity
  - Accountability
  - …

Metrics



Protection measures

- Authentication
- Authorization
- Access Control

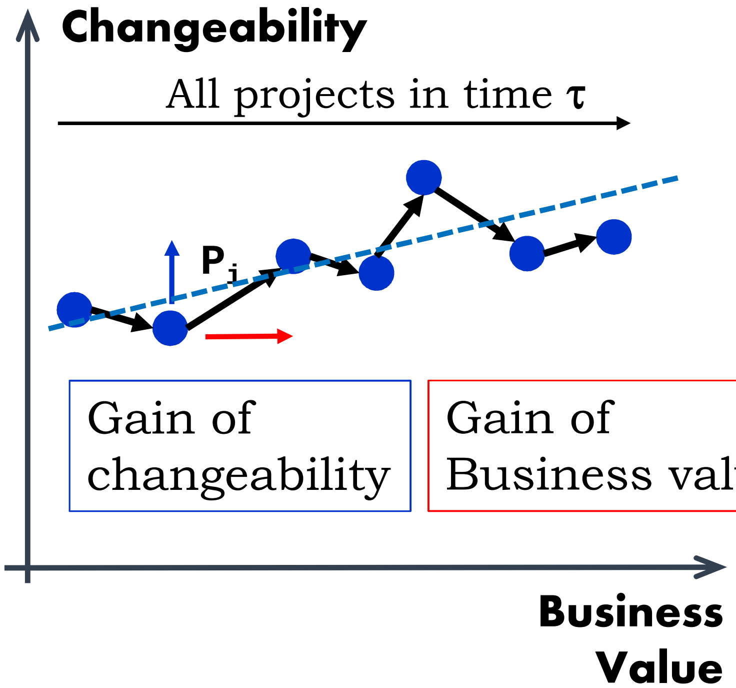
Risk analysis

Application-Domain specific

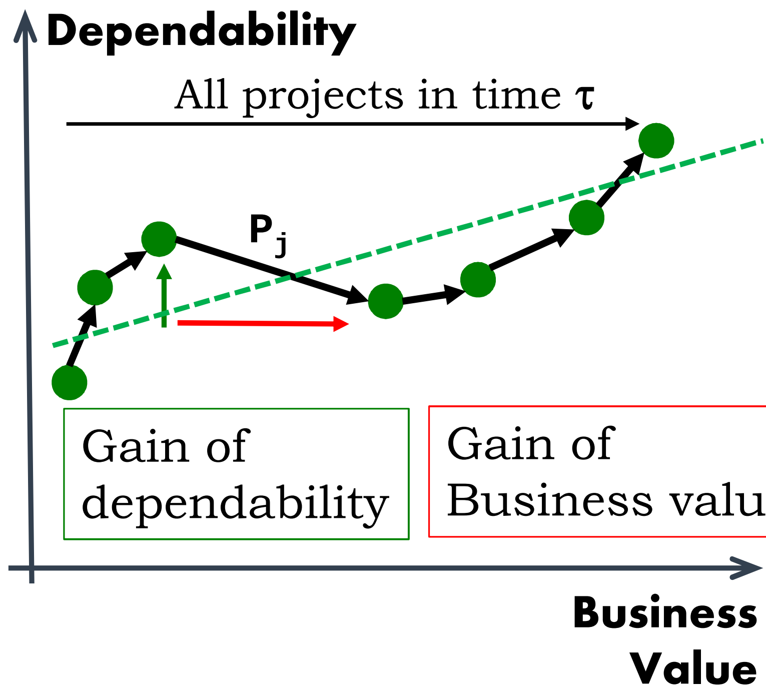
Dependability Methodology

Part 2: Dependability Strategy

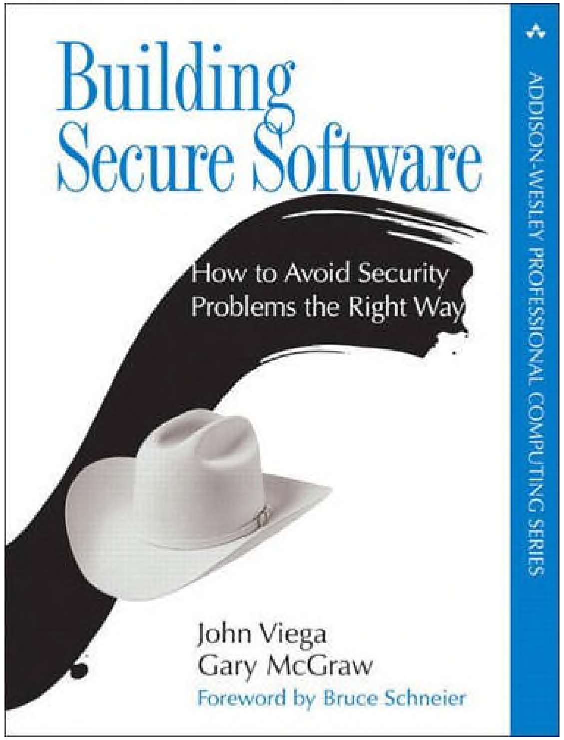
Changeability Evolution Trajectory



Dependability Evolution Trajectory

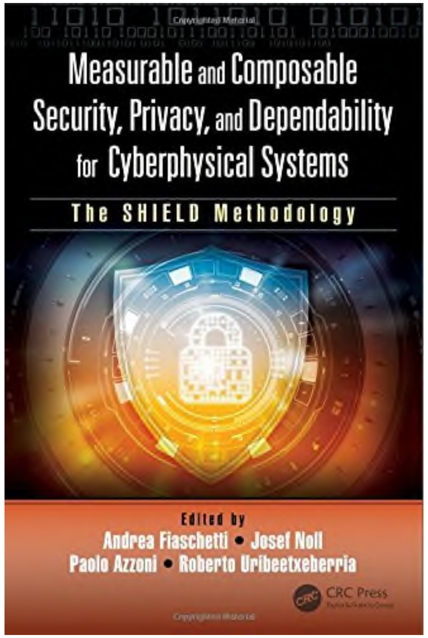


Textbook



John Viega, Gary R. McGraw:
Building Secure Software: *How to Avoid Security Problems the Right Way*
Addison-Wesley Educational Publishers Inc, USA, 2006. ISBN 978-0-321-42523-2

Textbook



Andrea Fiaschetti, Josef Noll, Paolo Azzoni, Roberto Uribeetxeberria:
Measurable and Composable Security, Privacy, and Dependability for Cyberphysical Systems: *The Shield Methodology*
CRC Press, Taylor & Francis, USA, 2018. ISBN 978-1-138-04275-9

Part 4B