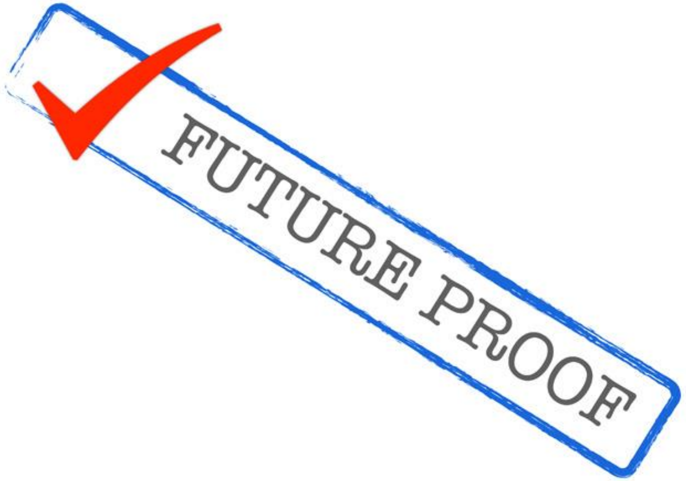


https://static1.squarespace.com

Future-Proof Software-Systems (FPSS)

Part 4B: Architecting for Dependability

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

Version 0.1

https://vstracking.com

Dependability Property: Availability

Availability

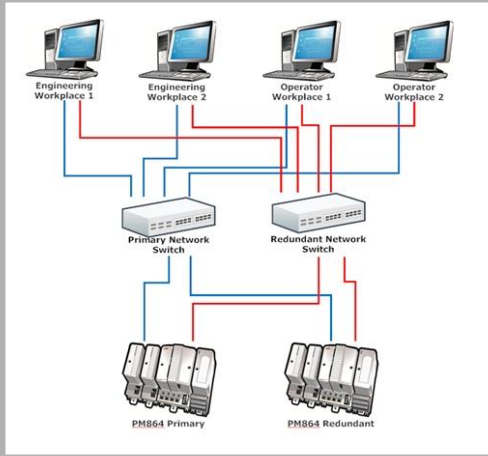
Protection Assets

- 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:

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

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%

> 1’000 changes/day

~ 10 disruptions/h

Business Load

Intended Changes

Disruptions

60’000 Servers

2’000 Routers

10’000 Business Databases

12’000 Business Applications

90’000 Workstations

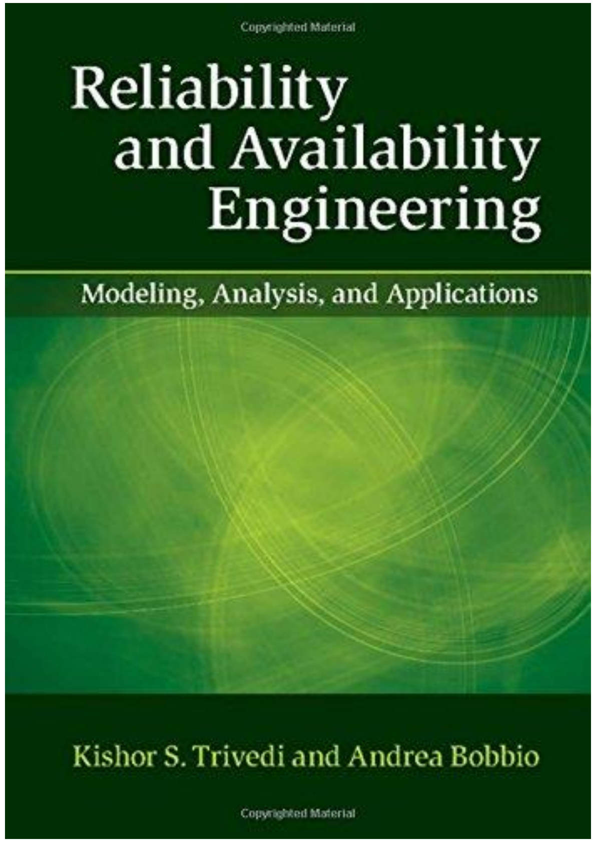
Large Computing Infrastructure

Required Availability:

99.9 %

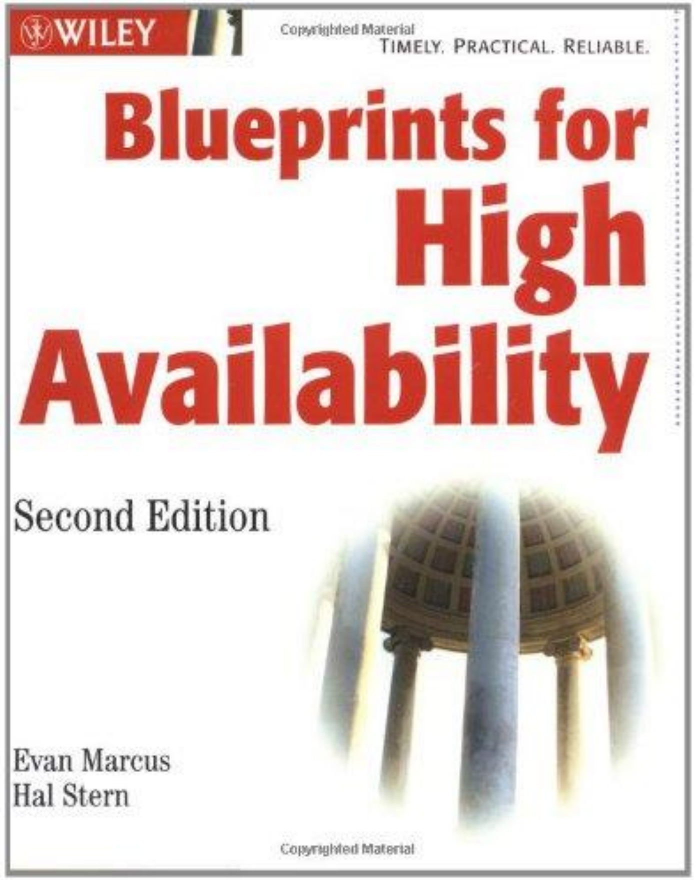
NOT including planned downtime

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

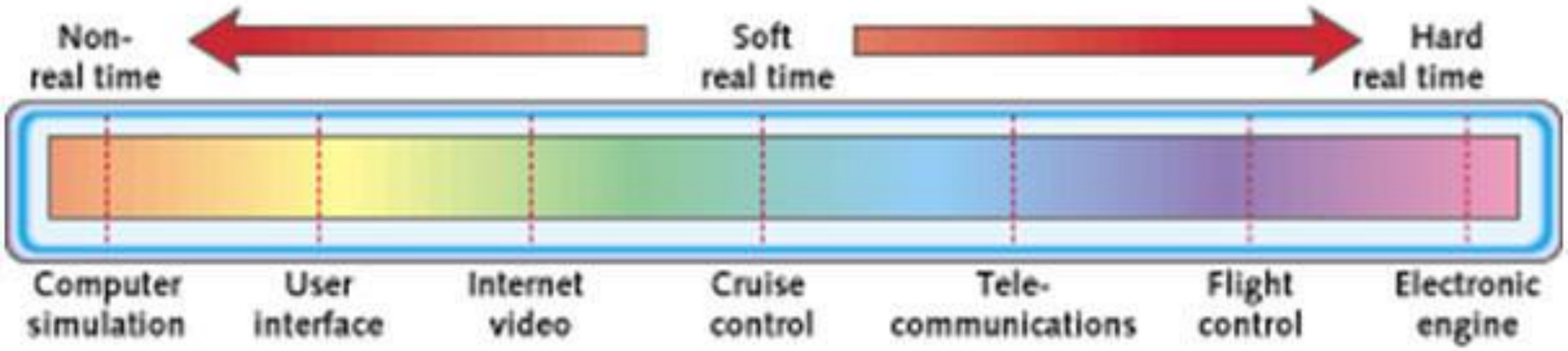
DEFINITIONS

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

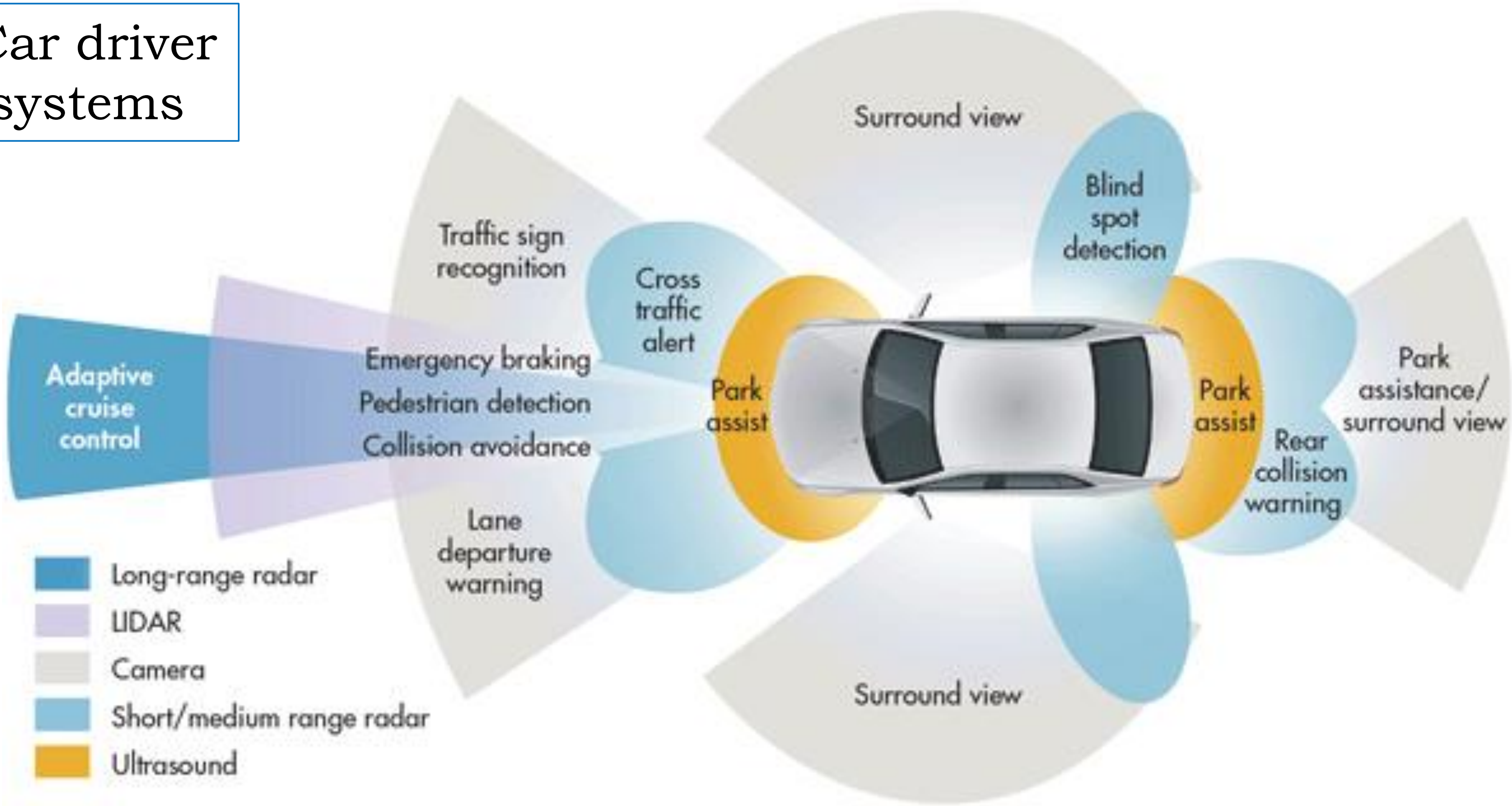
Real-time capability is the capability to react to events in a *predictable*, *guaranteed* time



https://www.logicsupply.com

Example: Car driver assistance systems

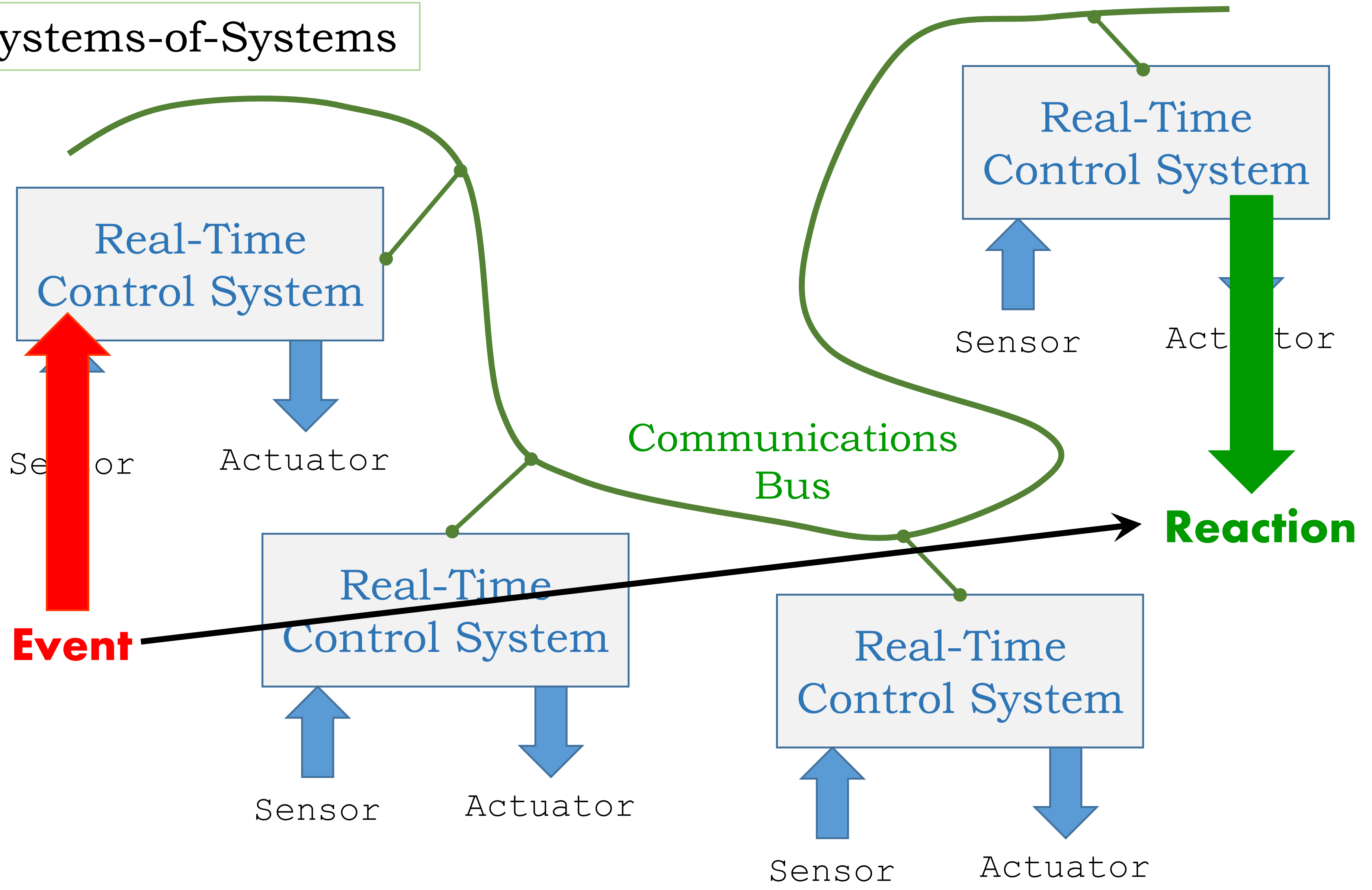
http://electronicdesign.com



Data acquisition ⇒ processing ⇒ event ⇒ decision ⇒ reaction

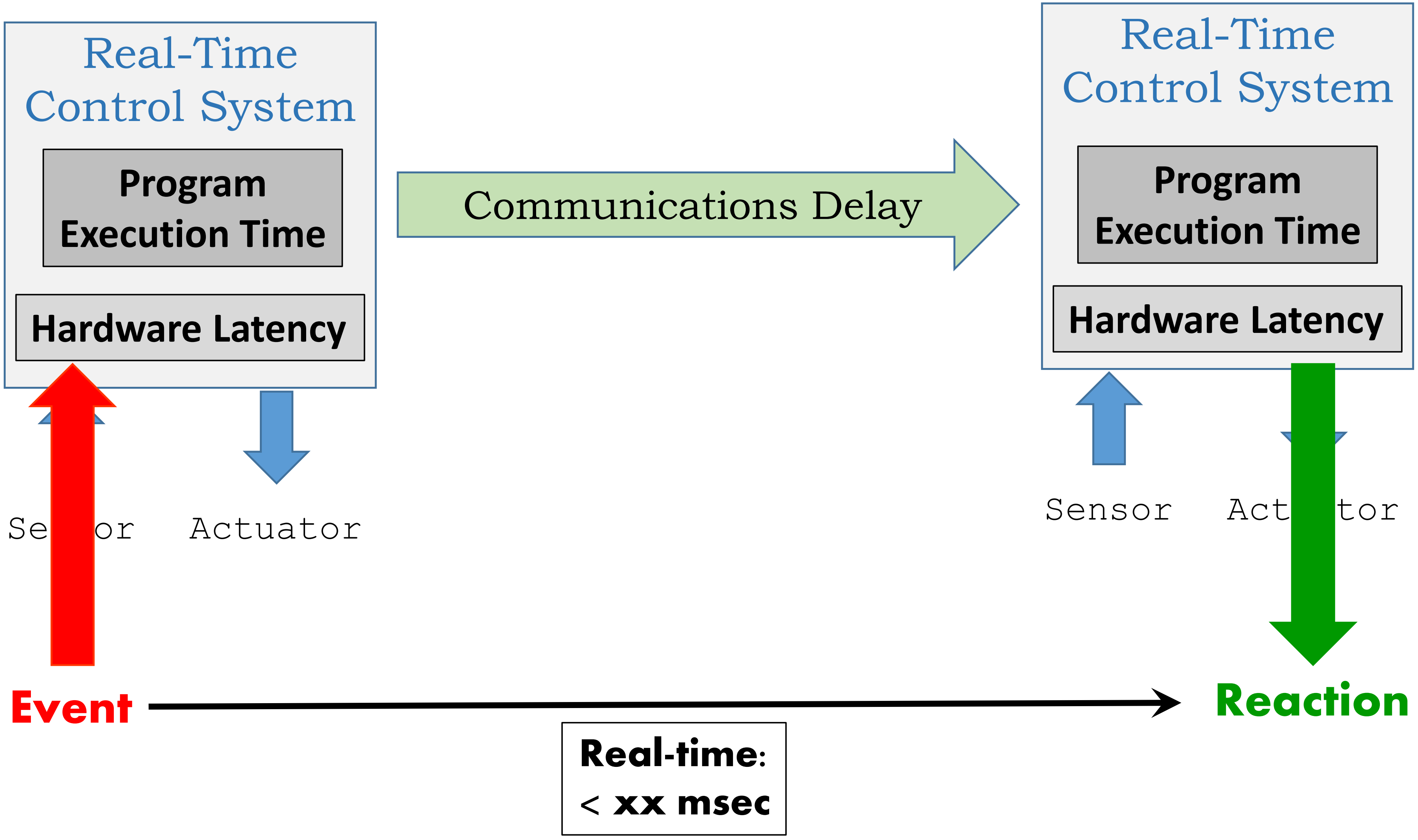
Real-time: max. xx msec

Real-world: Systems-of-Systems

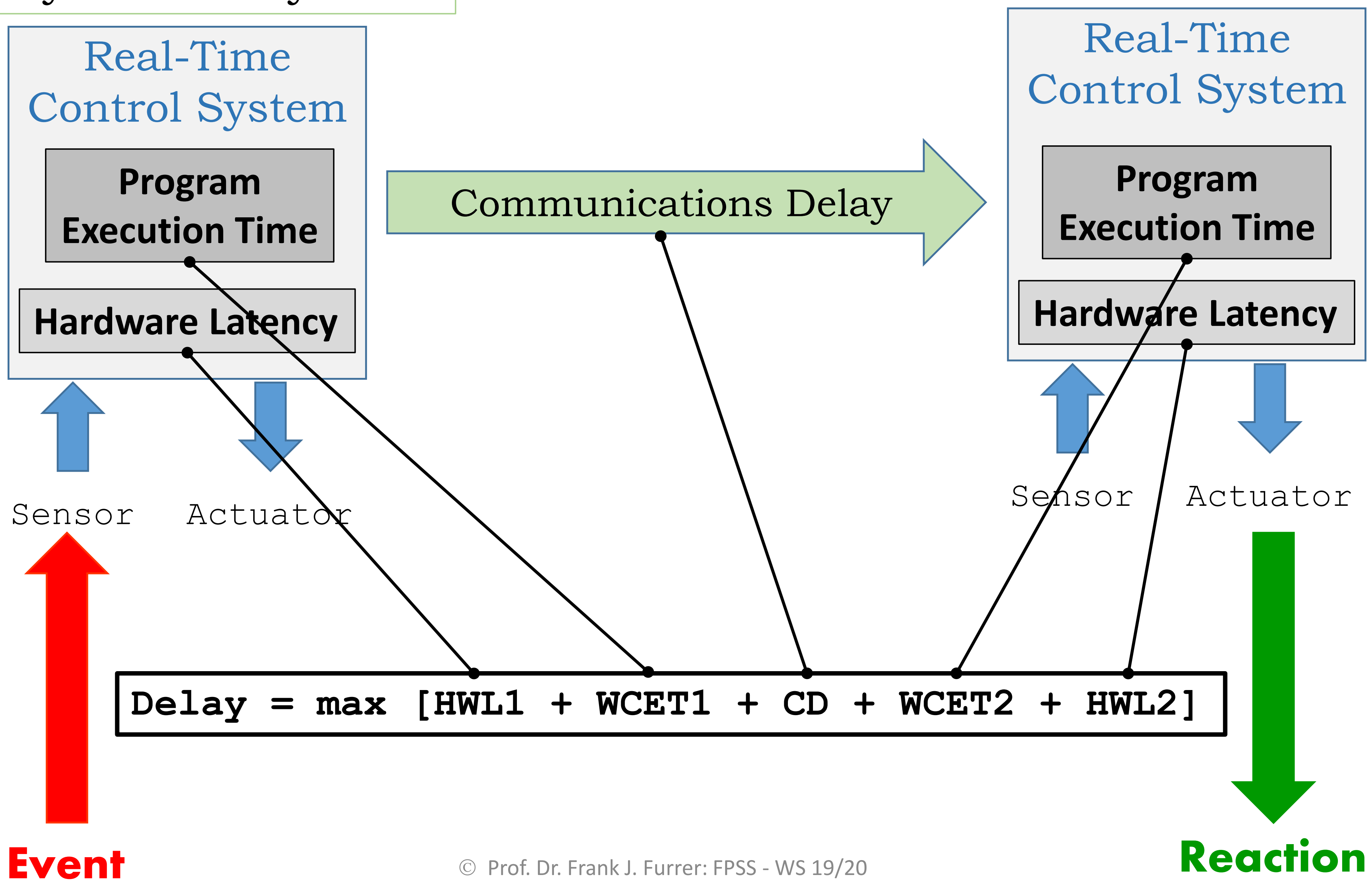


Real-time:
< xx msec

Real-world: Systems-of-Systems



Real-world: Systems-of-Systems



Real-world: Systems-of-Systems

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

Communication Delay:
Max: xx sec
Exact: xx sec

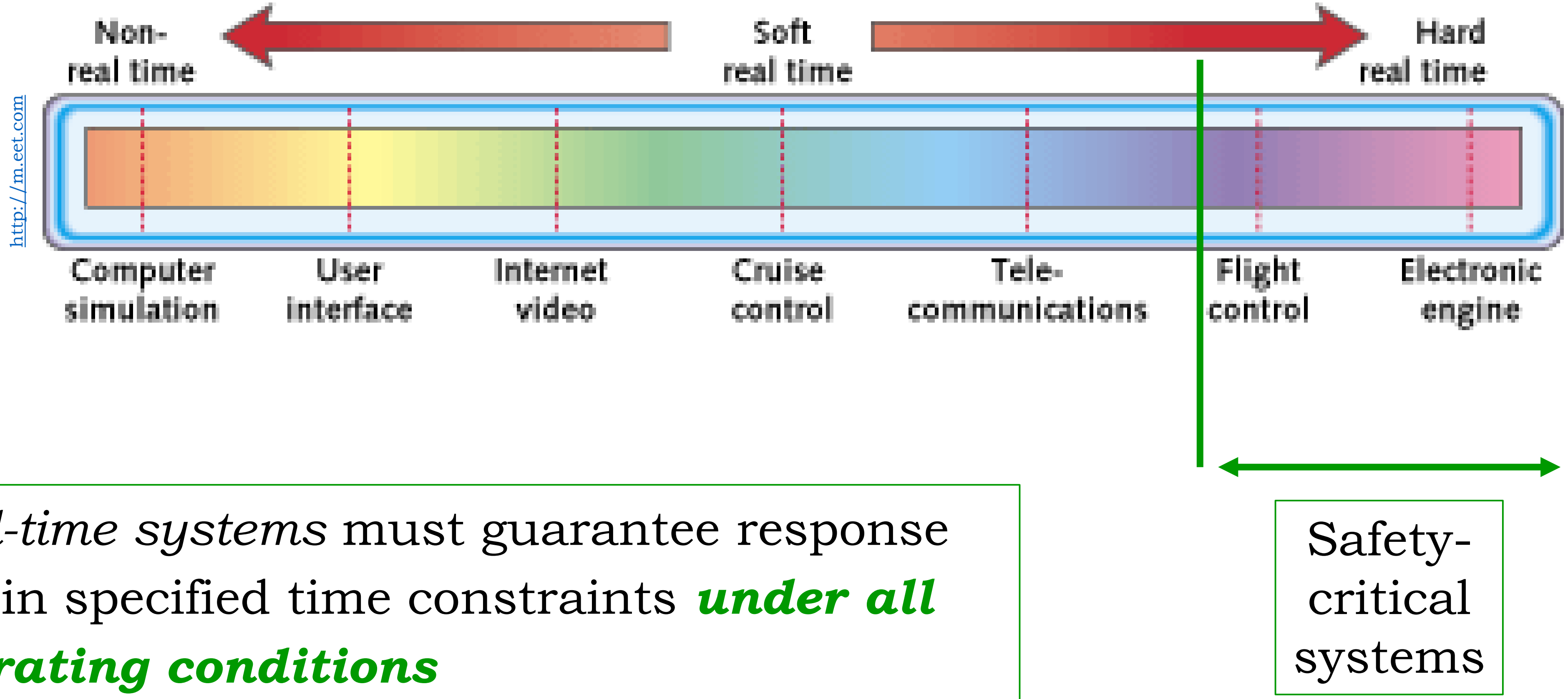
WCET:
Worst Case Execution Time
Max: xx sec

Hardware Latency Time:
Max: xx sec

Extremely **difficult** to estimate and guarantee

Real-world: Systems-of-Systems

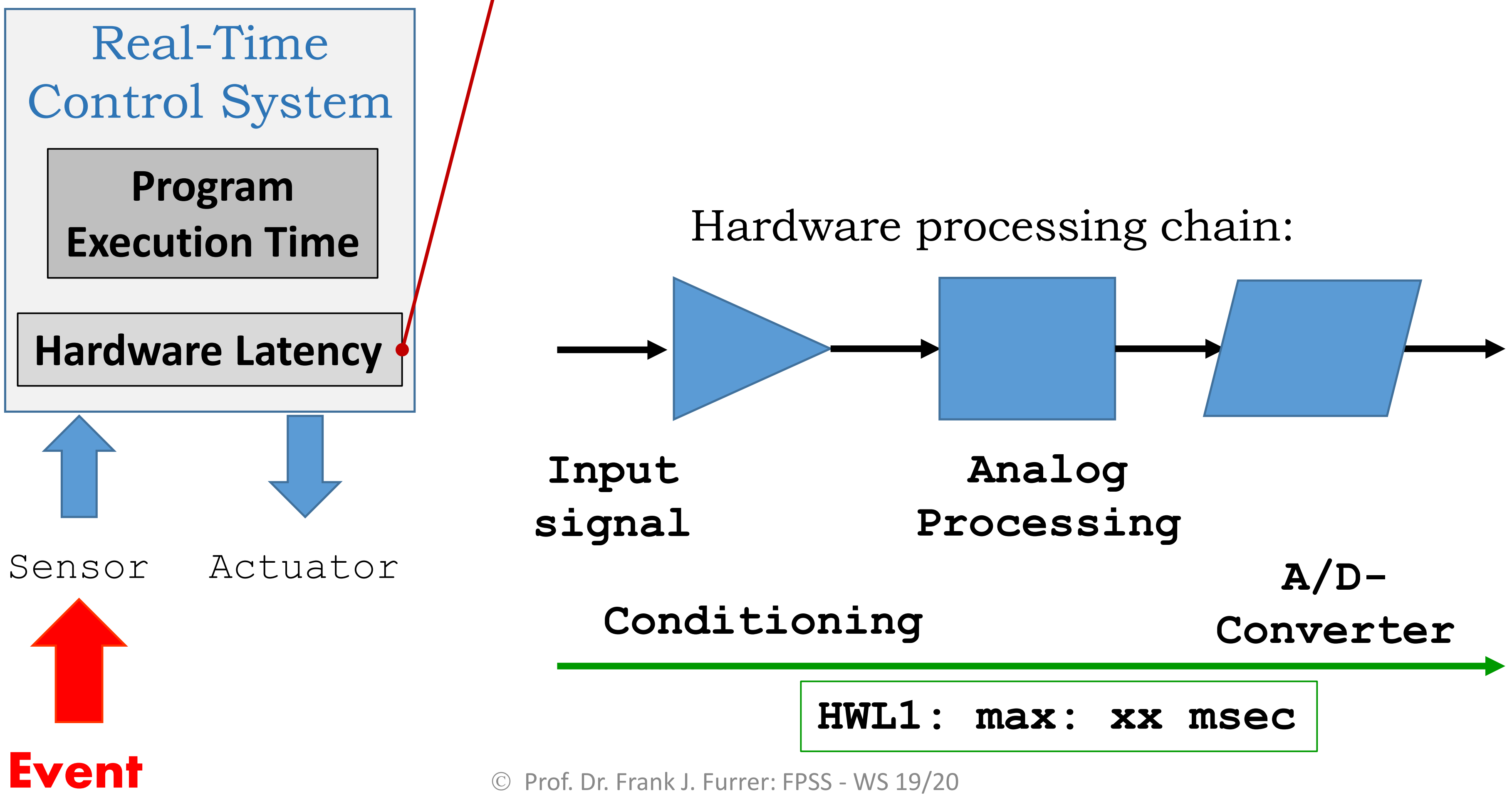
```
Delay = max [HWL1 + WCET1 + CD + WCET2 + HWL2]
```



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

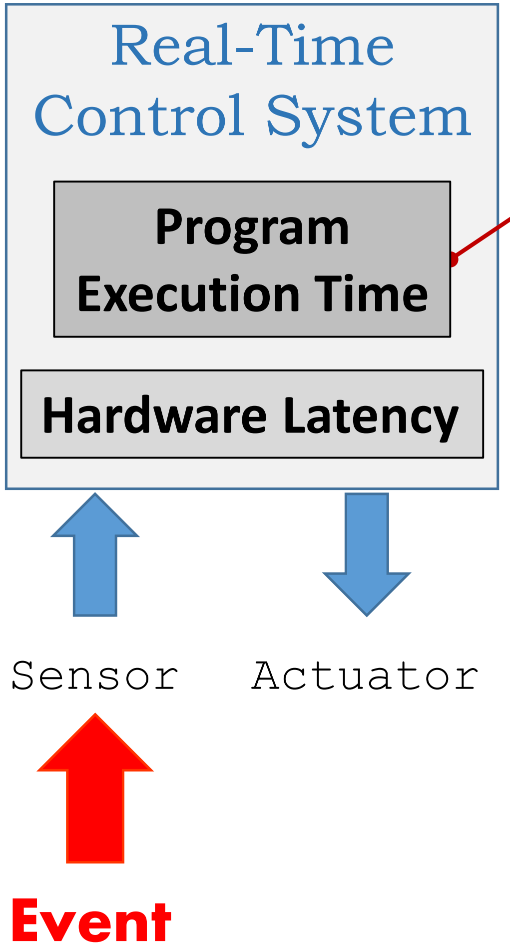
Real-world: Systems-of-Systems

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



Real-world: Systems-of-Systems

Delay = max [HWL1 + 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]

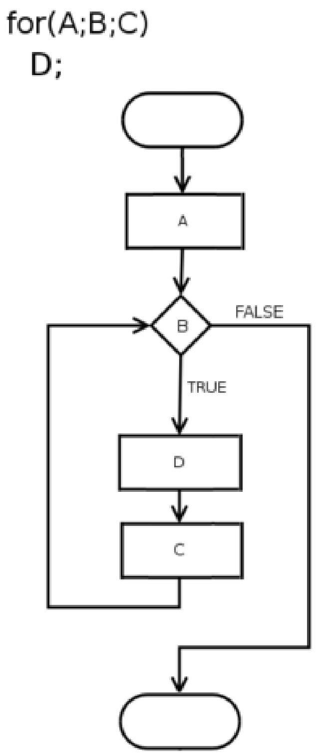
Program Execution Time

WCET: Worst Case Program Execution Time

https://www.cs.uic.edu

```
#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.

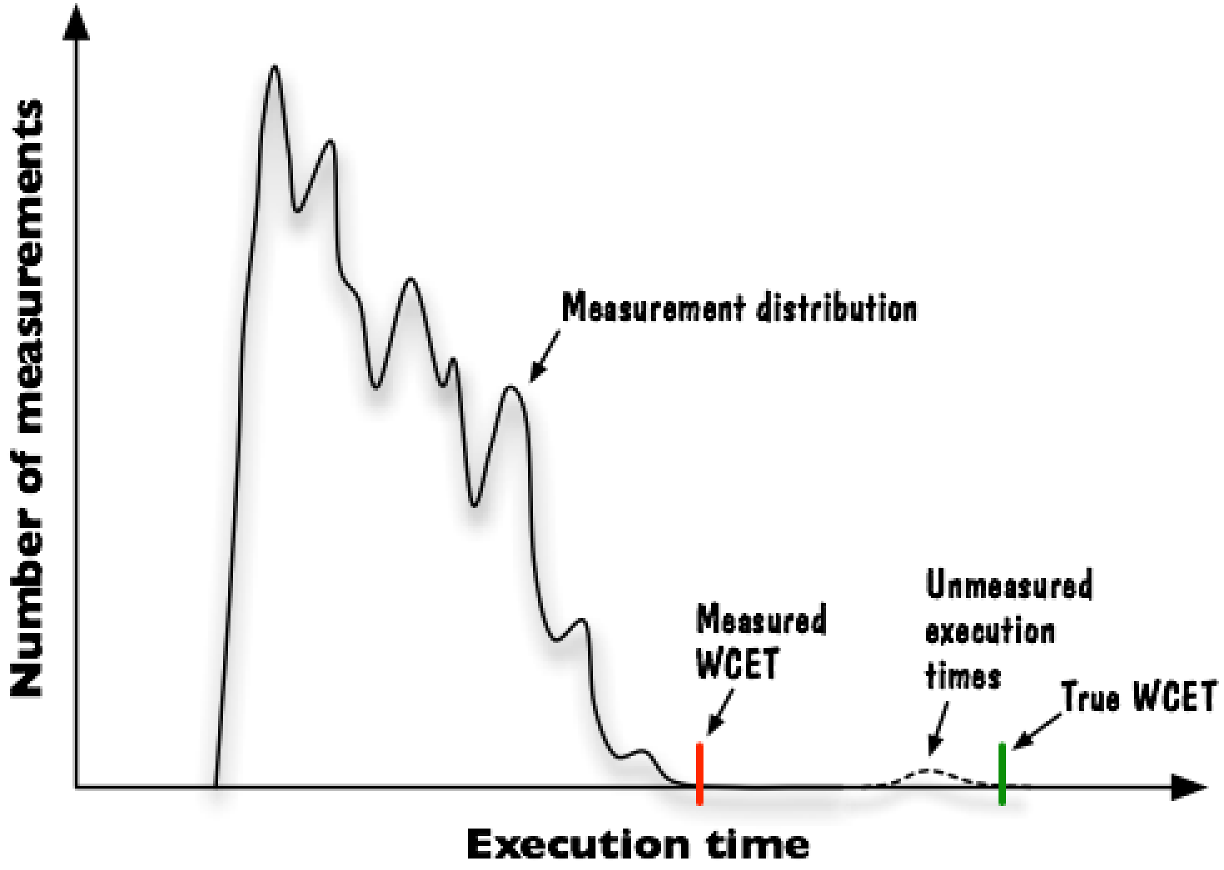


WCET:
Longest possible path trough the program
$\Rightarrow$ $msec_{max}$

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]

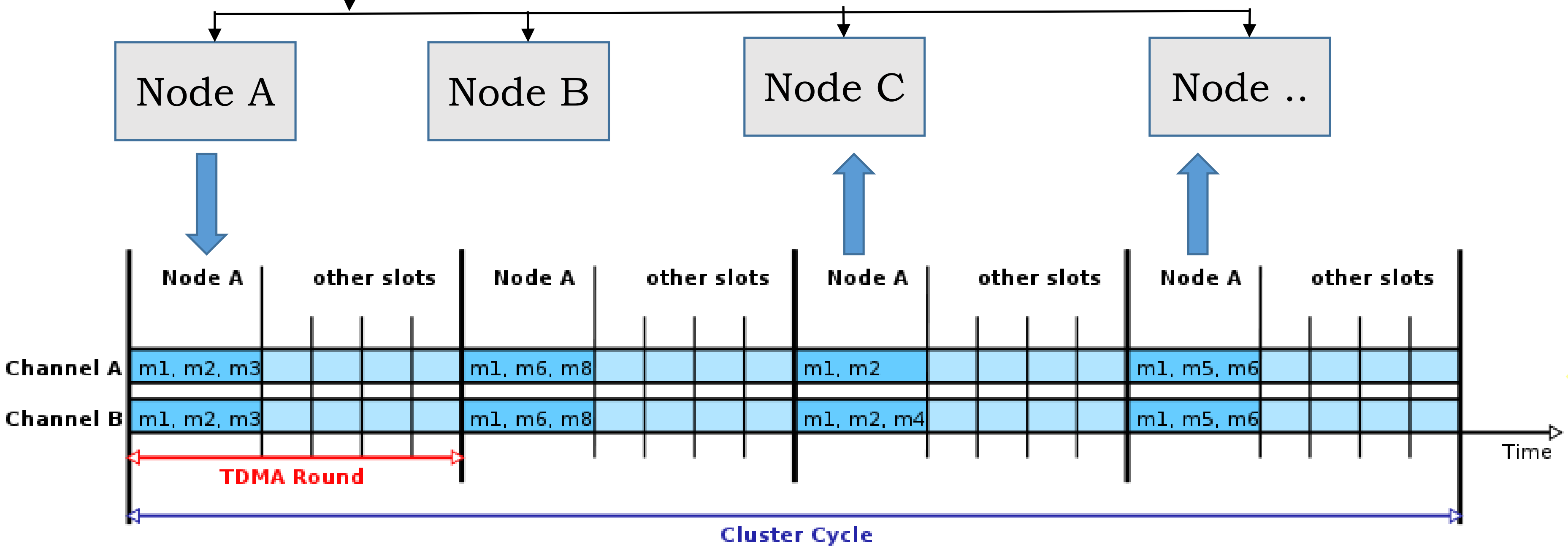
Communications Delay

Communication Delay:
Max: xx sec
Exact: xx sec

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

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

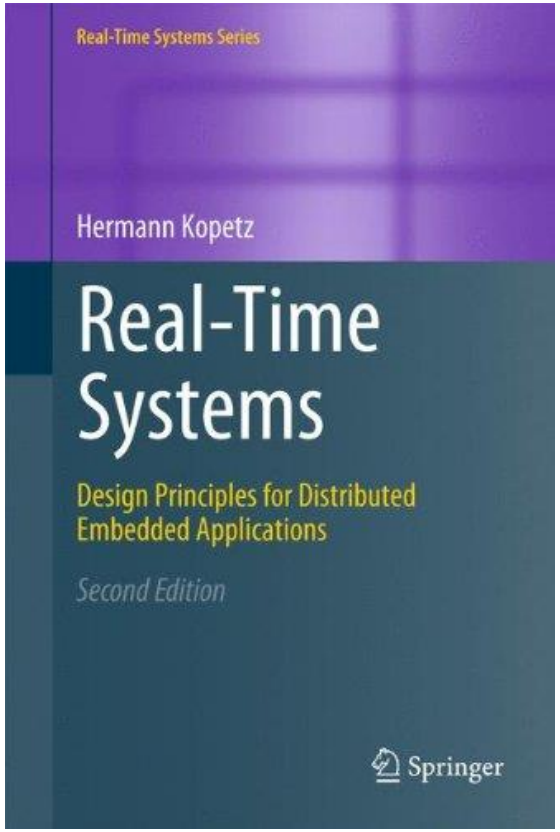
Master Clock:
`Clock Synchronization Algorithm`



Guaranteed real-time behaviour of the channel

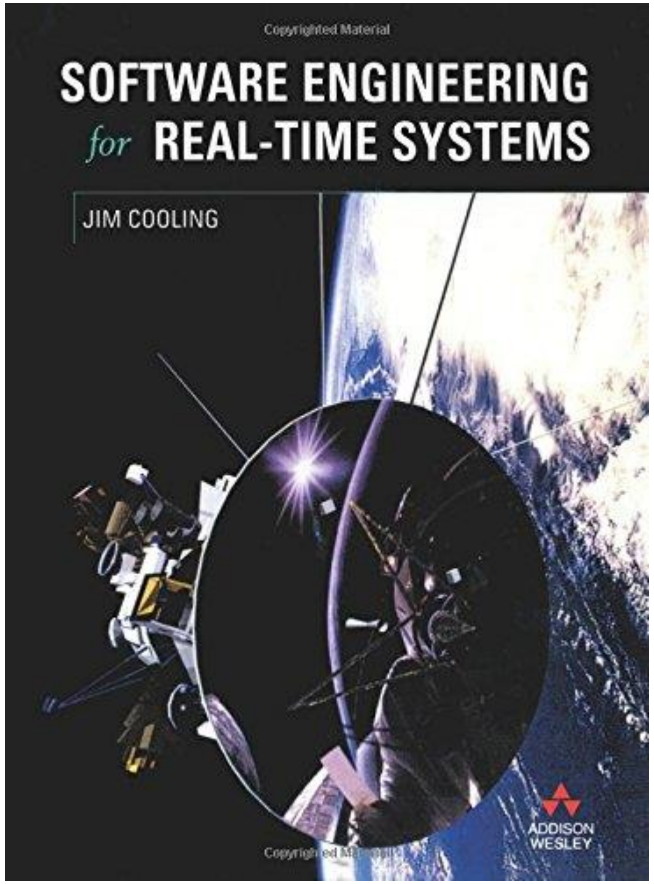
The messages are transported in exactly defined and assigned time slots, based on precise clock synchronization in all nodes

Textbook



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

Textbook



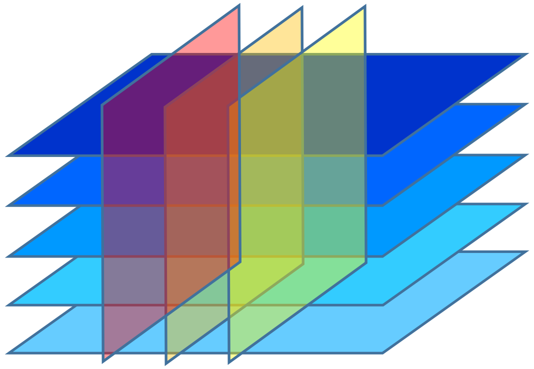
Jim Cooling
Software Engineering for Real-Time Systems
Addison Wesley, USA, 2002. ISBN 978-0-201-59620-5

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***

<u>*Note*</u>*: 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

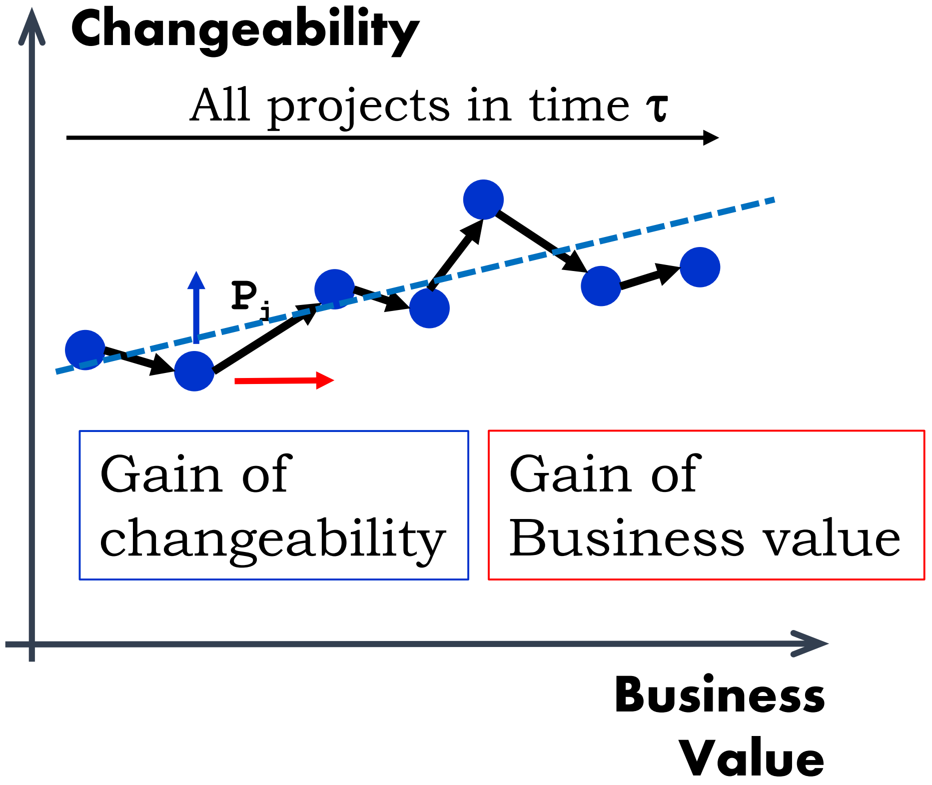
Application-Domain specific

Risk analysis

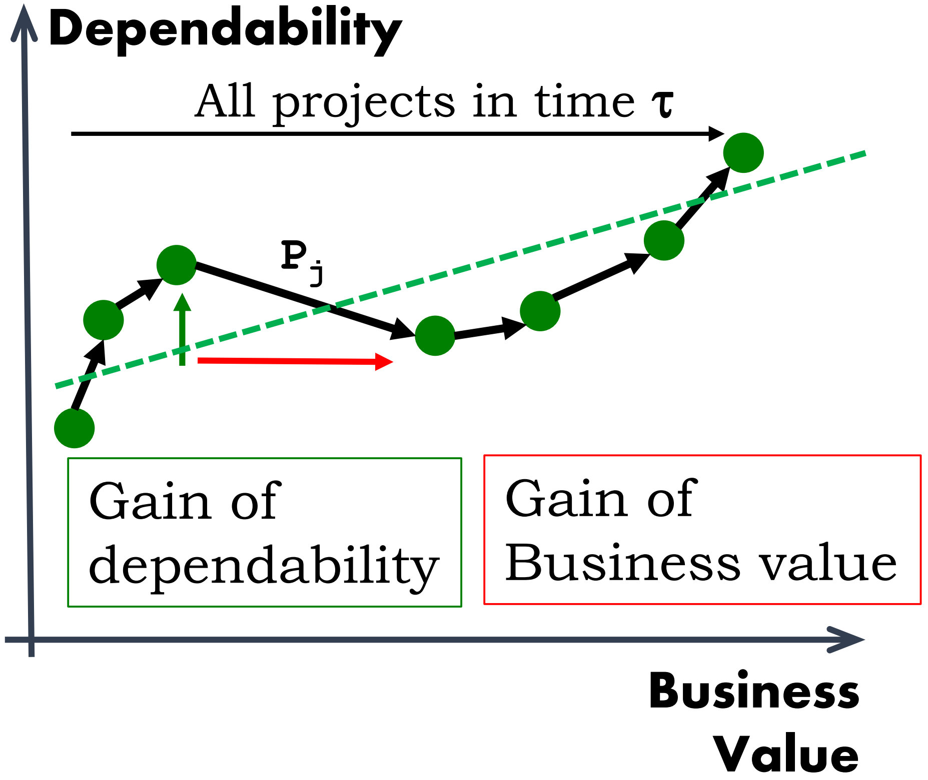
Dependability Methodology

Part 2: Dependability Strategy

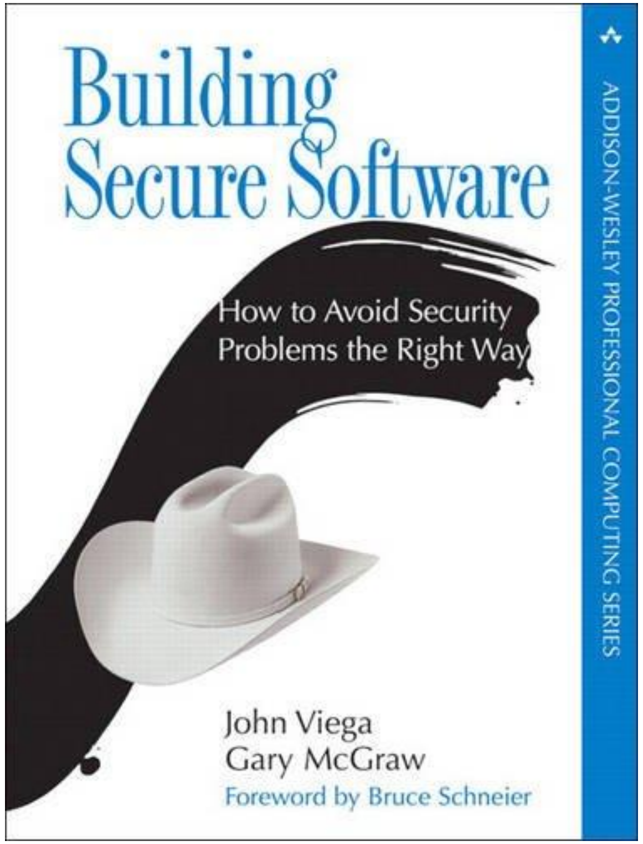
Changeability Evolution Trajectory



Dependability Evolution Trajectory

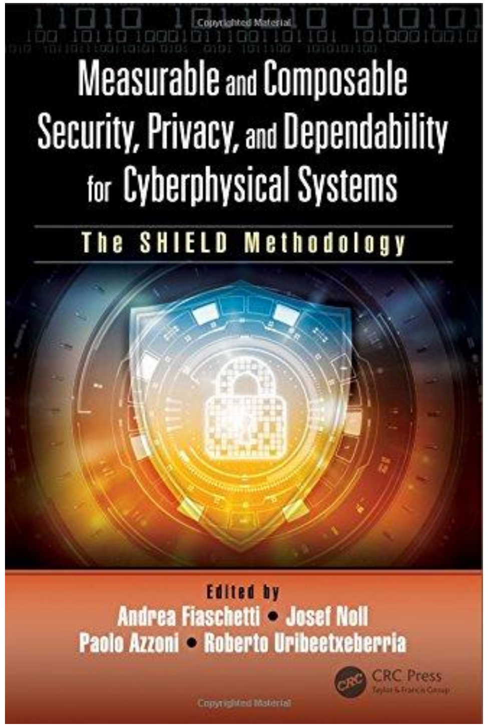


Textbook



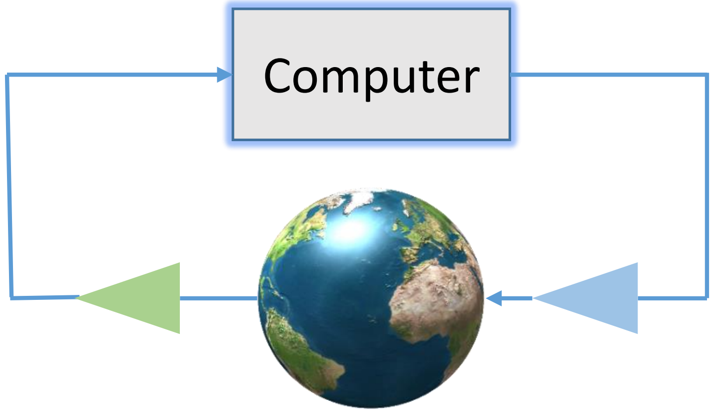
John Viega, Gary R. McGraw:
Building Secure Software: *How to Avoid Security Problems the Right Way*
Addison-Wesley Educational Publishers Inc, USA, 2006. ISBN 978-0-321-42523-2

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

Risk Management



Fault, Failure

Attack, Intrusion

Risk

= Inherent **property** of cyber-physical systems

Risk Management

= Decisive part of systems engineering

Risk Management Basics

Risk → Vulne-rability → Impact → **Damage**

Threat → Vulne-rability

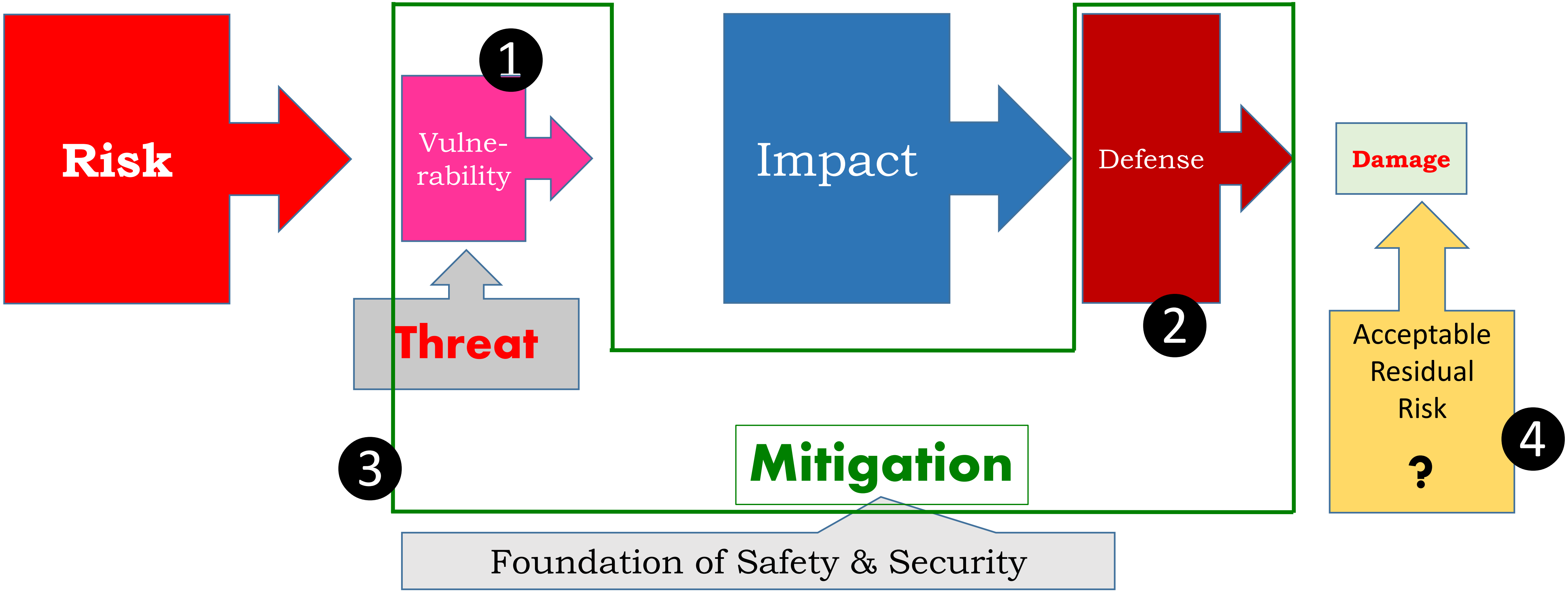
Unsafe flight condition

stalling

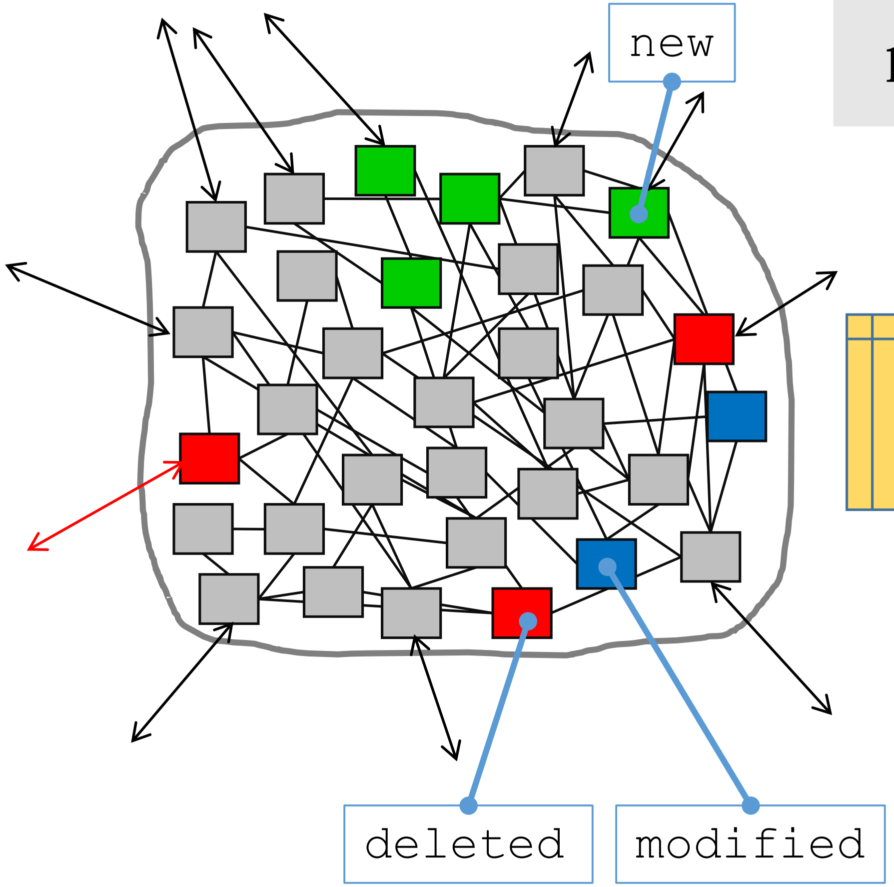
MCAS
Maneuvering **C**haracteristics **A**ugmentation **S**ystem
overrules pilots

crash

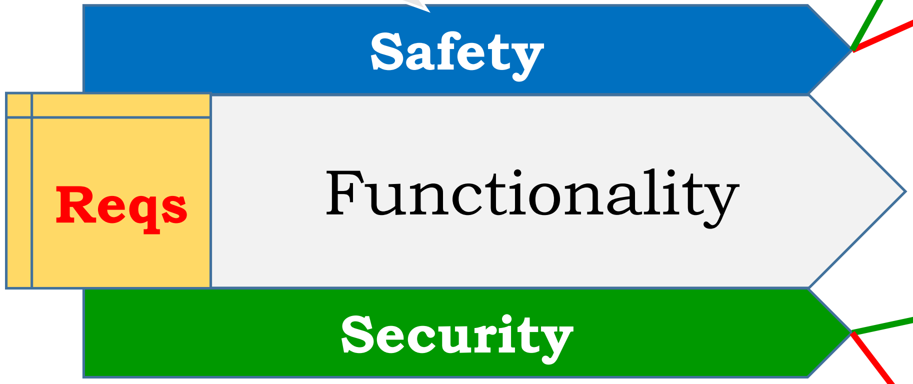
Risk Management Basics



System extension
→ Project

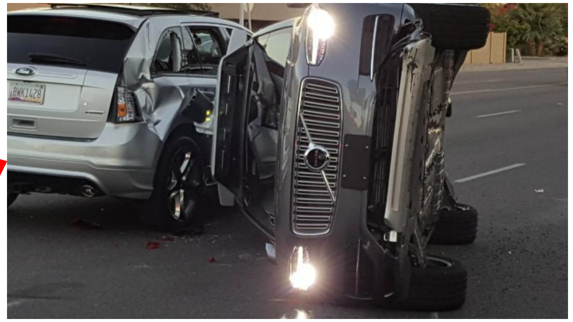


Safety and Security concerns/requirements have higher priority then functionality



Risk Management


https://www.indiatimes.com


https://bgr.com

https://www.seoblog.com


https://news.softpedia.com

Identification of Risks → **Risk**

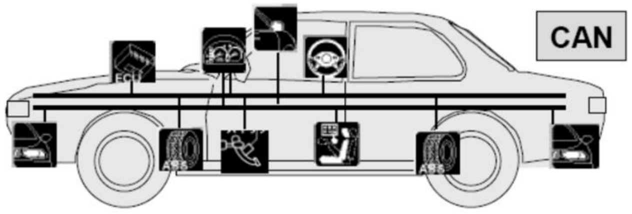
Search for Vulnerabilities → Vulne-rability

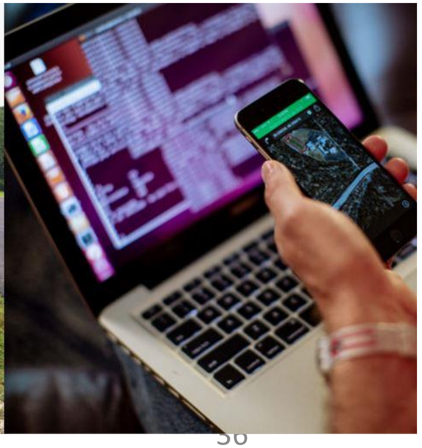
Predict Impact → Impact

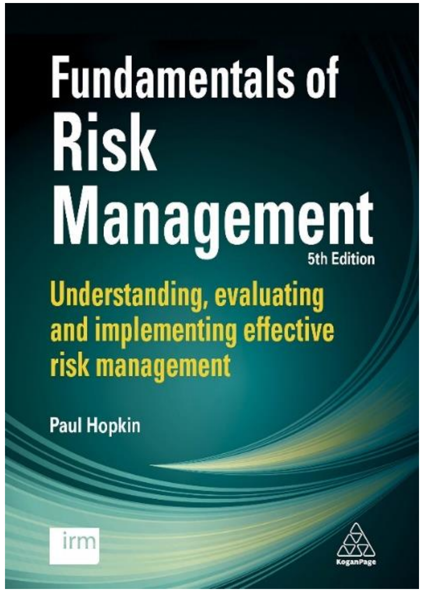
Assess Damage & Probability → **Damage**

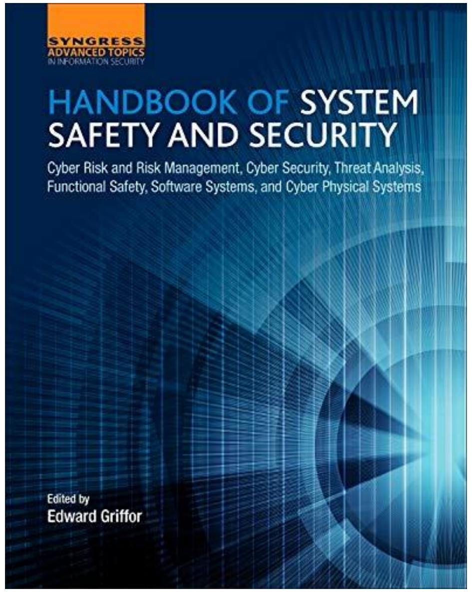
Miti-ga-tion,

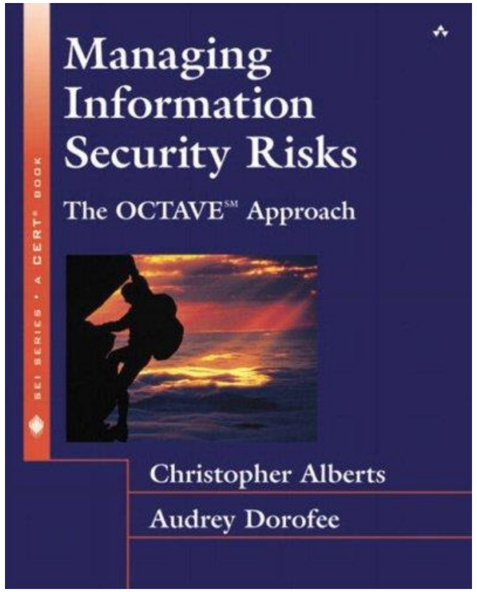
Pro-tec-tion







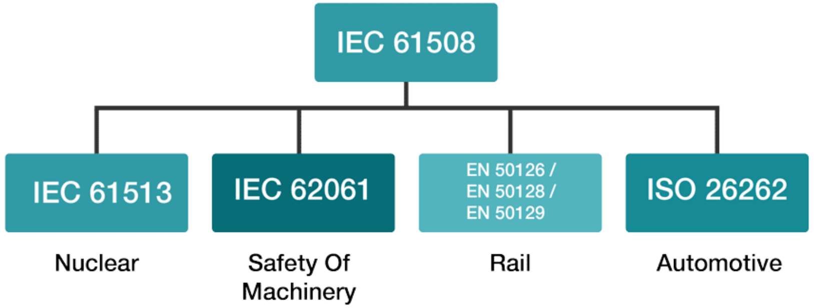




A significant number of risk management **methodologies** exist

Many industries are based on risk management **standards**

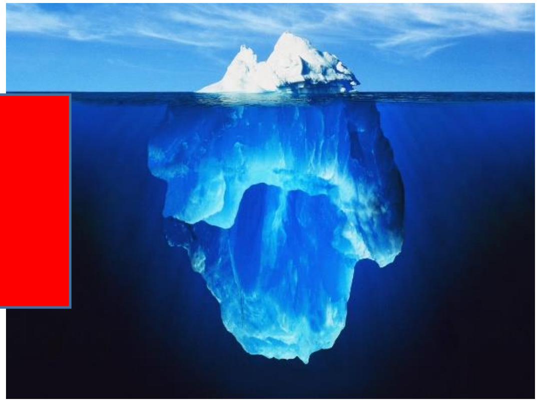
Companies have their own set of methodologies & standards



https://www.fool.com

http://inadinaofset.com

Identified (known) Risks



http://ontheiob.45things.com

Hidden (unknown) Risks

http://www.clker.com

✓ Mitigation
✓ Protection

✓ Generic Protection Measures

❖ Safety Engineer
❖ Security Engineer

https://www.123rf.com

Safety Requirements

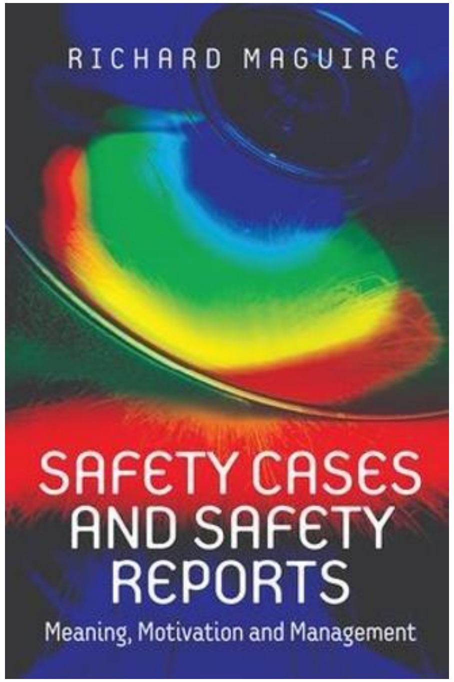
Security Requirements

Risk Management Process

Approval Certification

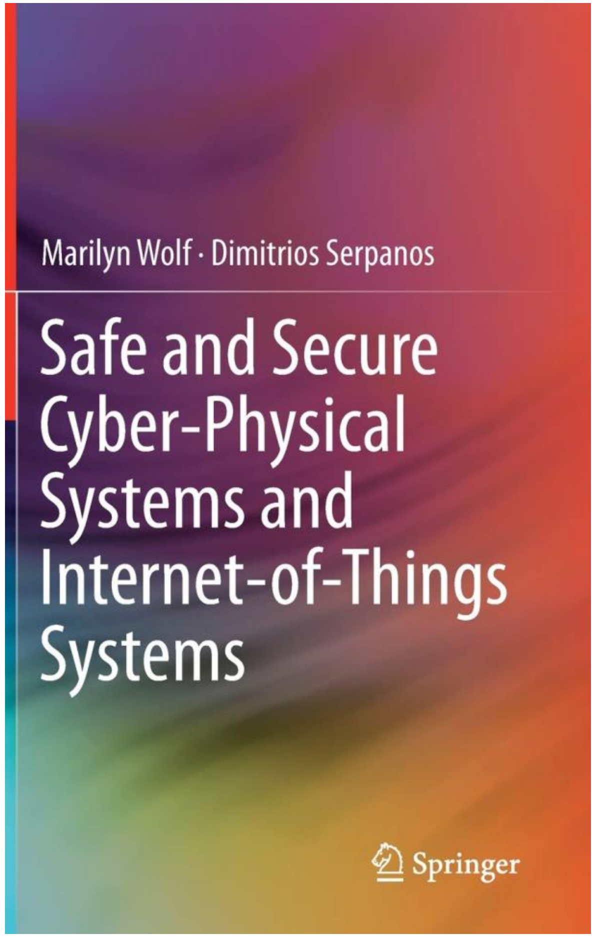
√

Risk Management Report [Safety Case]

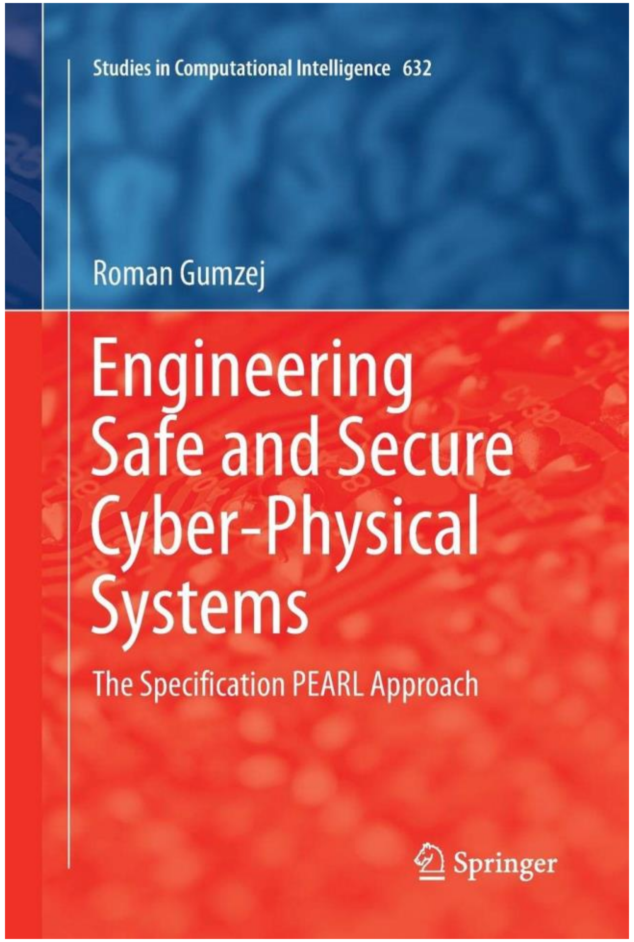


Richard Maguire: **Safety Cases and Safety Reports – Meaning, Motivation and Management**
Taylor & Francis Ltd (CRC Press), USA, 2017
ISBN 978-1-138075320

https://www.witn.com



Marilyn Wolf, Dimitrios Serpanos:
Safe and Secure Cyber-Physical Systems and Internet-of-Things Systems
Springer-Verlag, 1st edition 2020
ISBN 978-3-030-25807-8



Roman Gumzej:
Engineering Safe and Secure Cyber-Physical Systems – *The Specification PEARL Approach*
Springer-Verlag, 1st edition 2016
ISBN 978-3-319-80454-5

Part 4B

https://www.askideas.com

… it was a pleasure working with you !

frank.j.furrer@bluewin.ch

frank.furrer@mailbox.tu-dresden.de