Summary of Lecture 16.10.2019

http://englishskills.se

… Very condensed summary of the 30.10.2019 lecture

https://de.fotolia.com

Future-Proof Software-Systems: Summary 16.10.2019

Summary 30.10.2019

Dependability

Resilience

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/

General resilience principles

https://www.npmis.com

Engineering Profession: Resilience Engineer

Domain-specific properties

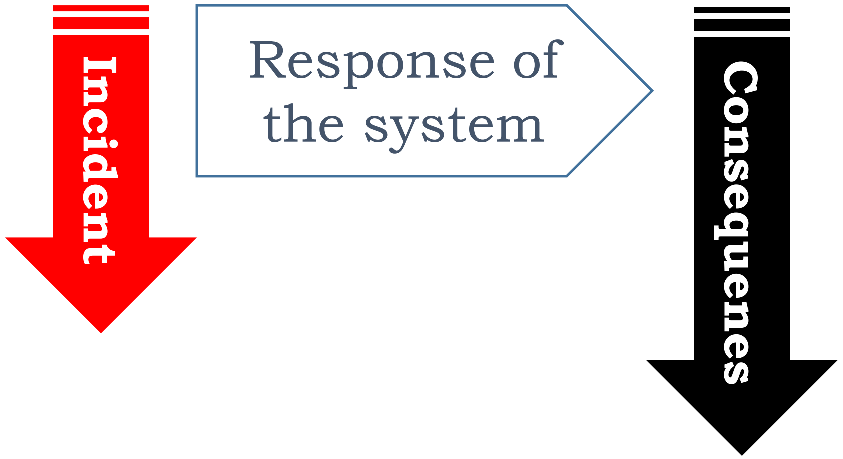
Domain-specific dependability properties are the defense against specific threats:

- Security,
- Safety,
- etc.

https://www.npmis.com

Risk-based specific dependability principles

Summary 30.10.2019



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

Summary 30.10.2019

General (Overarching) Architecture Principles for Resilience

- R1: Policies
- R2: Vertical Architectures
- R3: Fault Containment Regions
- R4: Single Points of Failure
- R5: Multiple Lines of Defense
- R6: Fail-Safe States
- R7: Graceful Degradation
- R8: Dependable Foundation (Infrastructure)
- R9: Monitoring

⇒ Part 4

http://www.tralphium.net

Summary 30.10.2019

User trust **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).

Car:

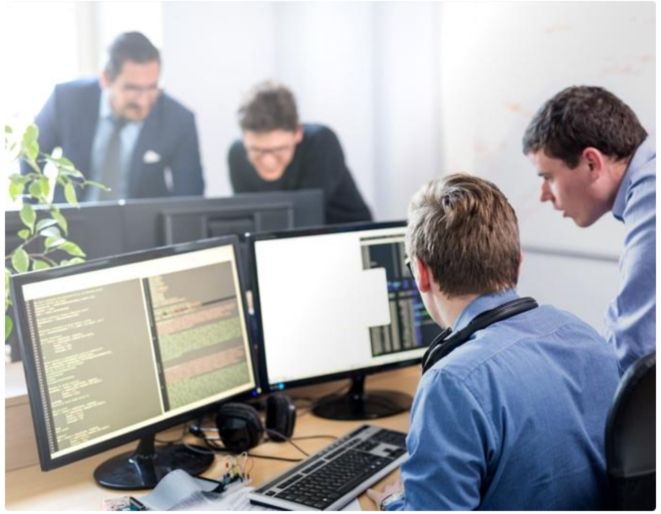
- *safety* (= no accidents)
- *security* (= no hostile influence)
- *reliability* (= no engine failures on the motorway)
- *conformance* to all laws and regulations

Trustworthiness expectations
=
Application domain

Summary 30.10.2019

Technical Debt

«We know it is a shortcut – … we will fix it later»



https://www.sourceseek.com

… is the direct way to (software) hell

https://www.behance.net

Definition:

Technical debt in an IT-system is the result of all those necessary things that you choose *not to do now*, but will impede future evolution if left undone

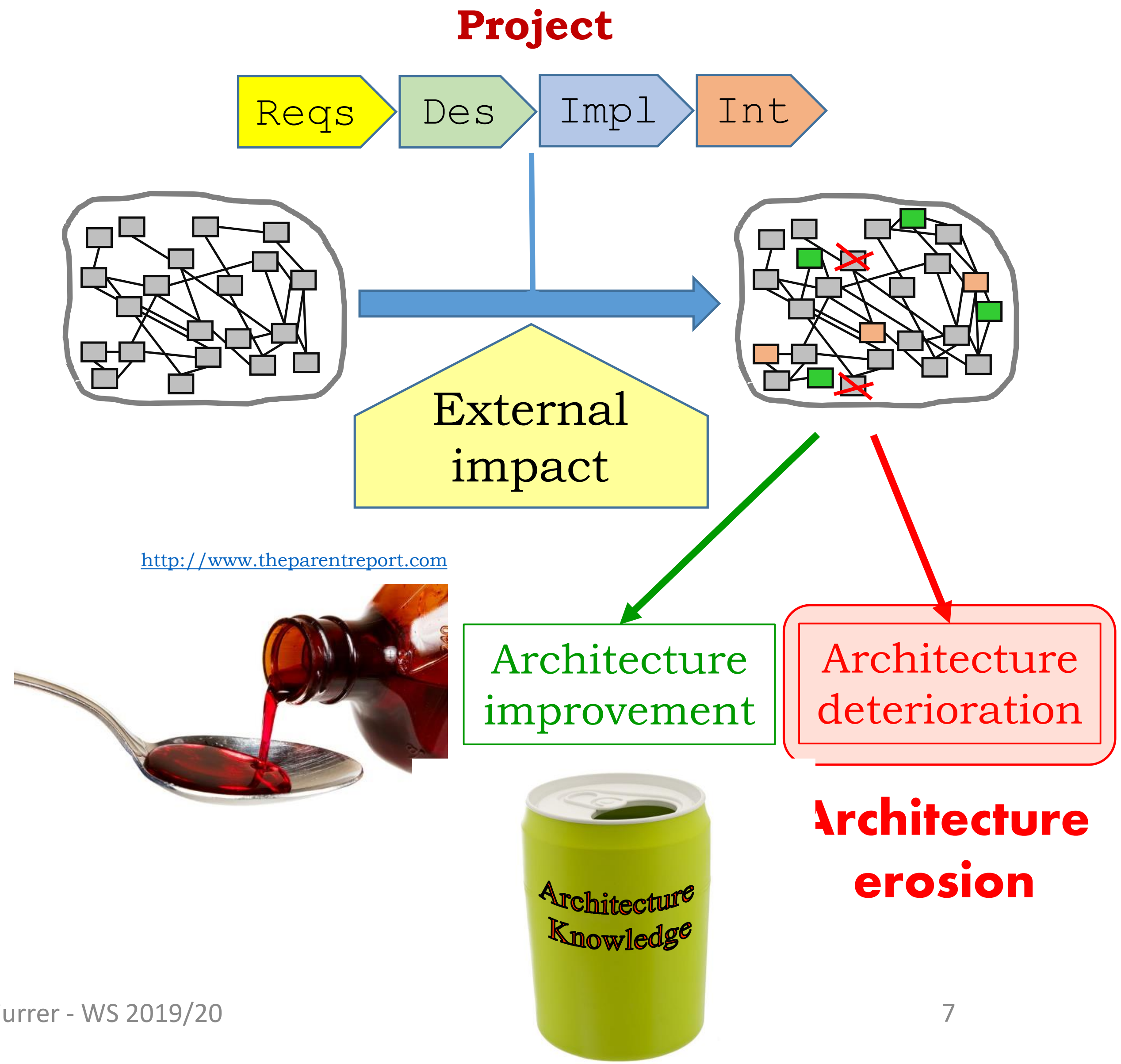
Ward Cunningham, 2007

Summary 30.10.2019

Architecture Erosion

Definition: Architecture Erosion

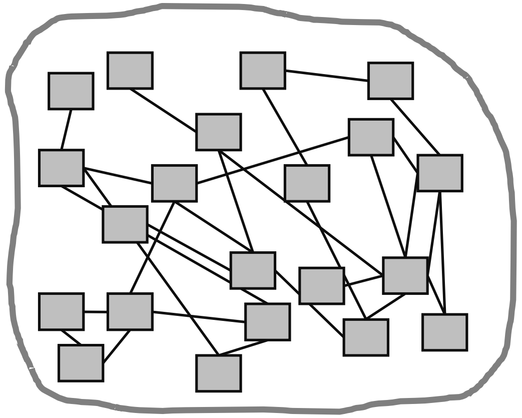
Architecture erosion is the process where an initially well-designed, adequate architecture of a software-system is **gradually destroyed** by the activities of evolution and maintenance of the software-system.



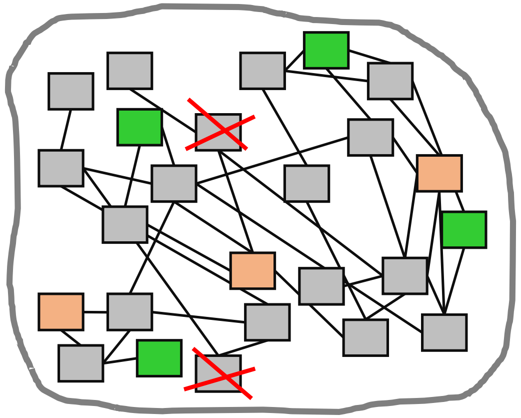
http://www.theparentreport.com

Summary 30.10.2019

Future-Proof Software-Systems



Evolution: Software Life-Cycle



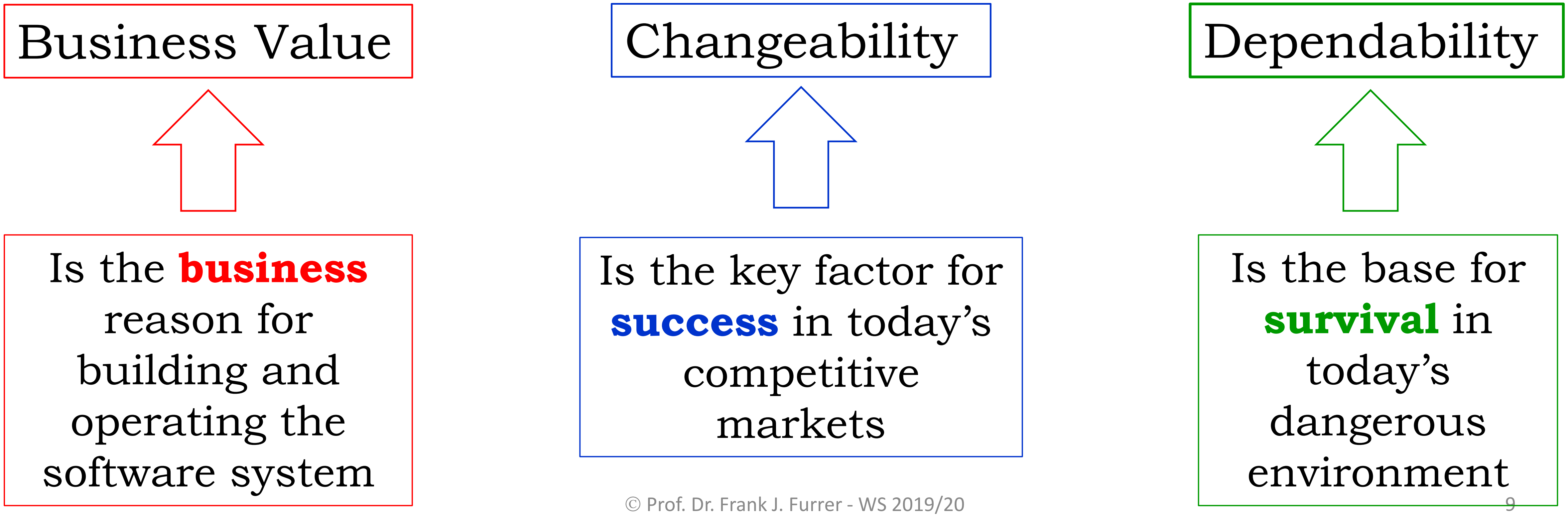
Continuous software-system improvement: We need **three positive powers**

Good architects

Summary 30.10.2019

Future-Proof Software-Systems

Primary Properties:



Business Value

Changeability

Dependability

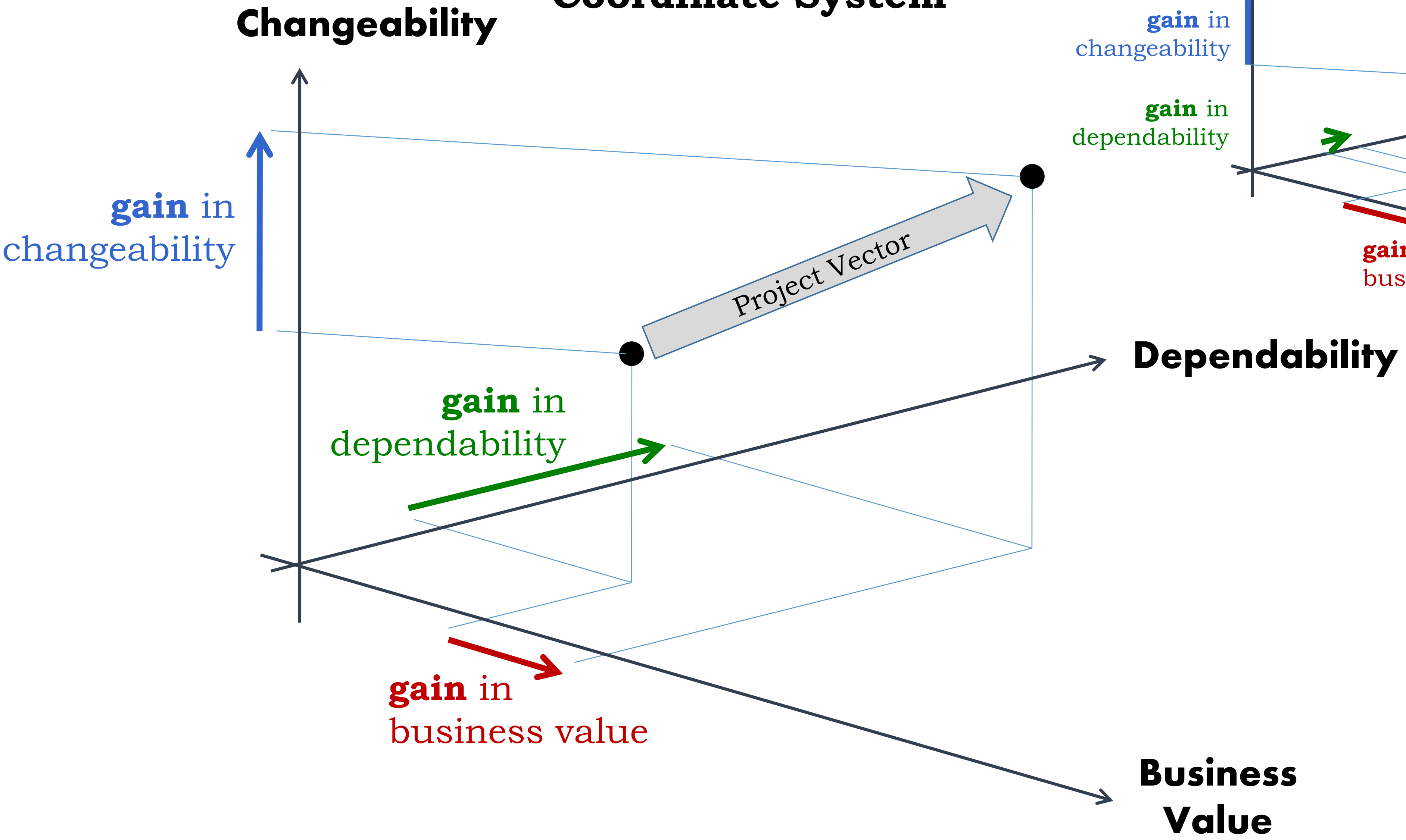
Is the **business** reason for building and operating the software system

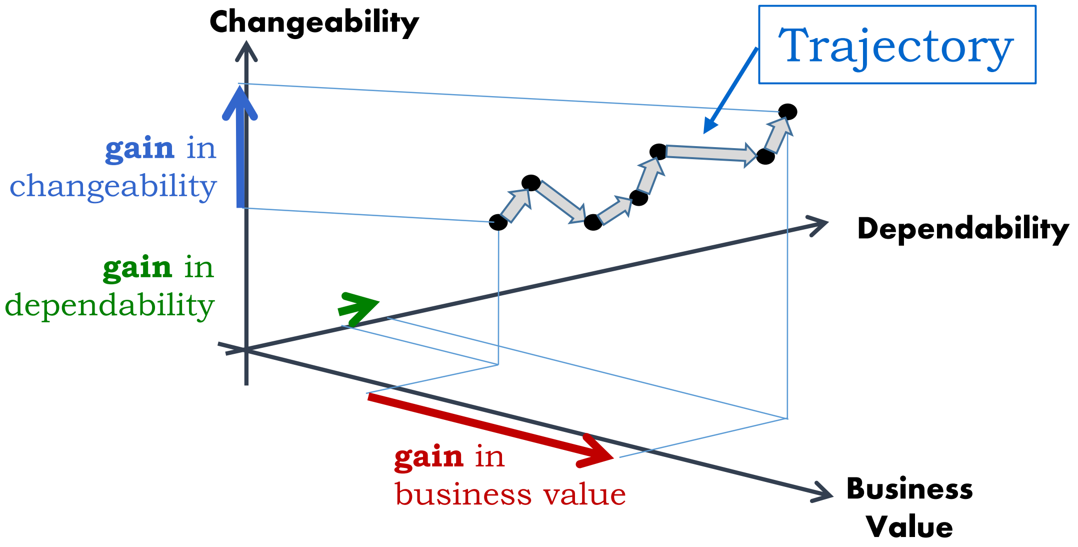
Is the key factor for **success** in today’s competitive markets

Is the base for **survival** in today’s dangerous environment

Summary 30.10.2019

Managed Evolution Coordinate System



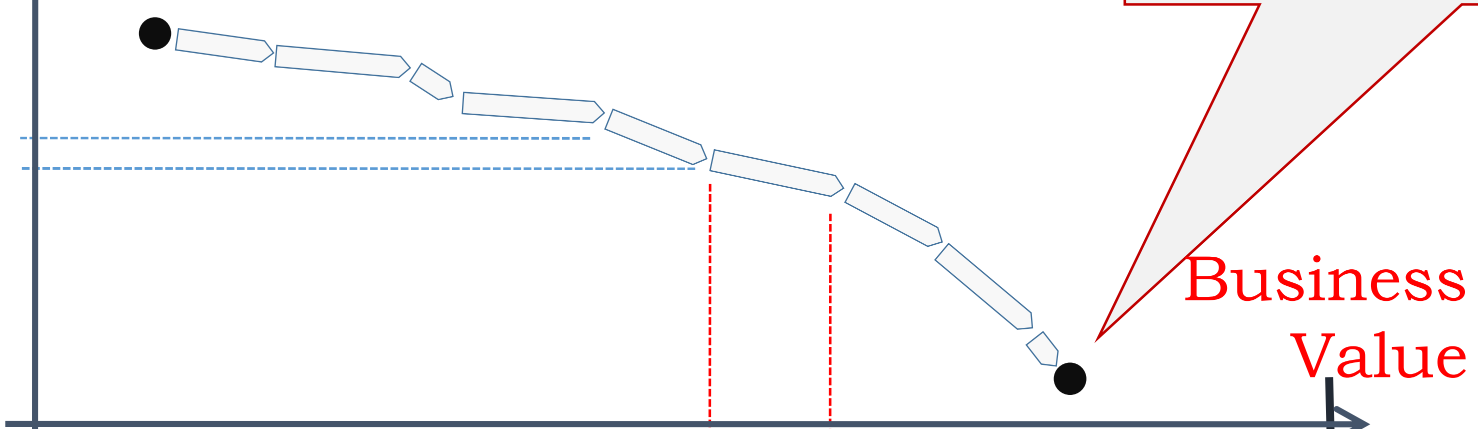


Summary 30.10.2019

Changeability

Continuous development of business value while **neglecting** improvement of agility leads to a petrification of the system

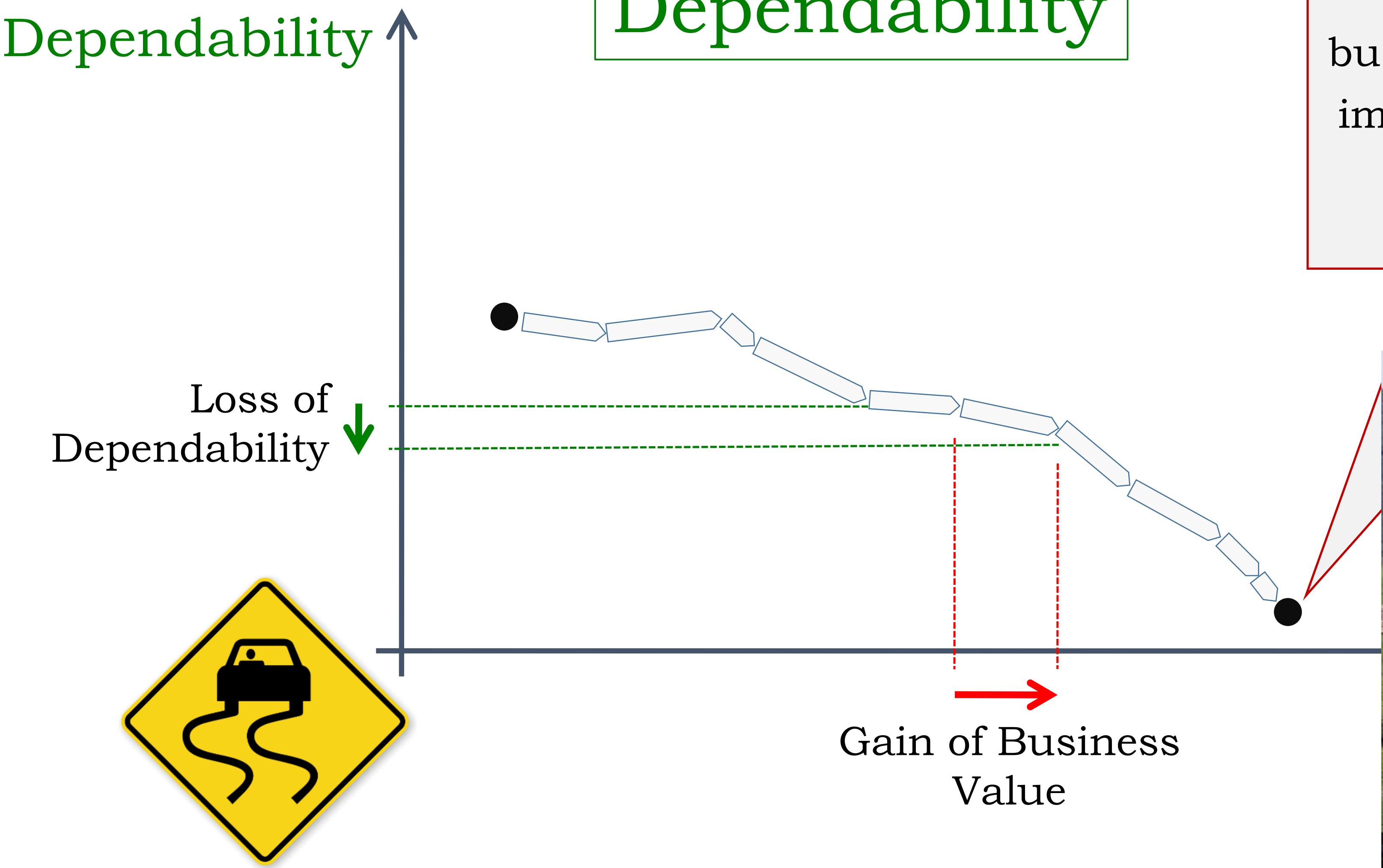
(**= path to death**)



Trajectory Case 1:
Opportunistic Evolution

Summary 30.10.2019

Dependability

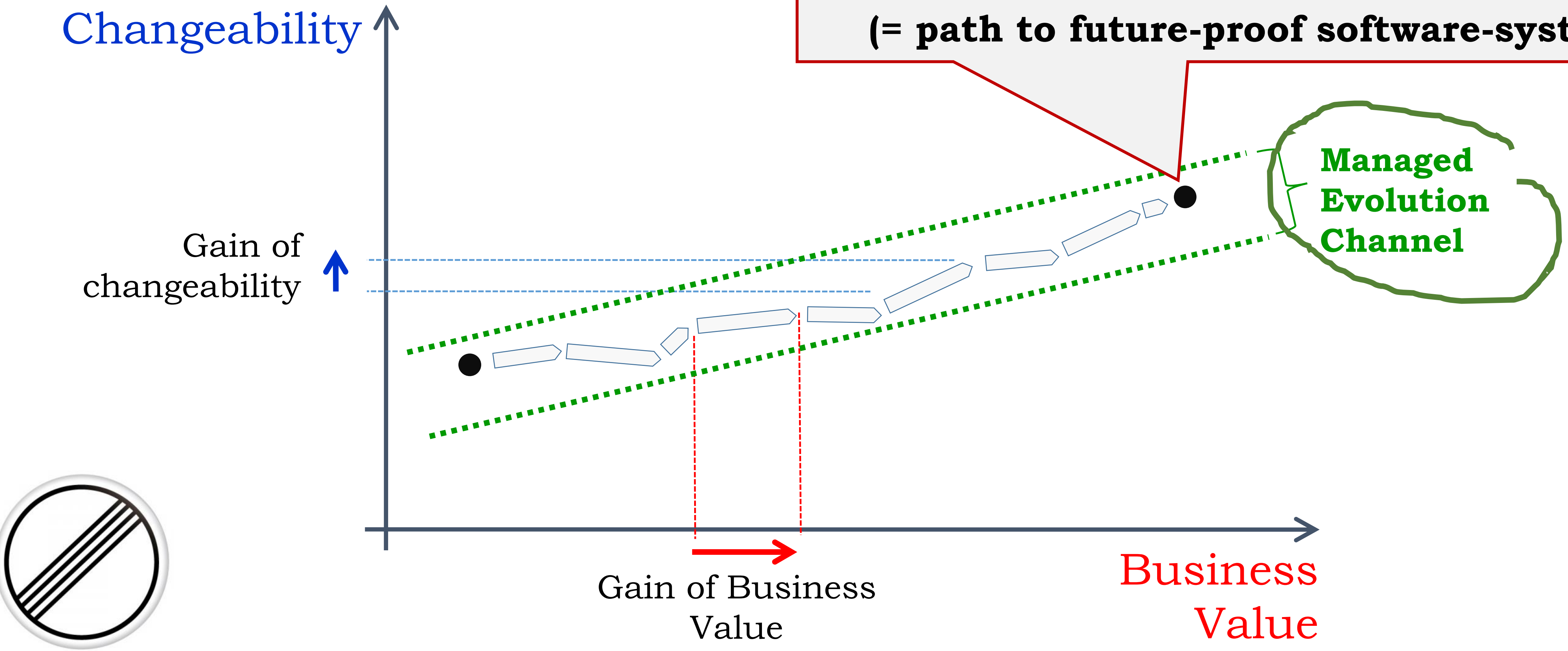


Continuous development of business value while **neglecting** improvement of resilience leads to an undefendable system (= **path to death**)

Trajectory Case 1:
Opportunistic Evolution

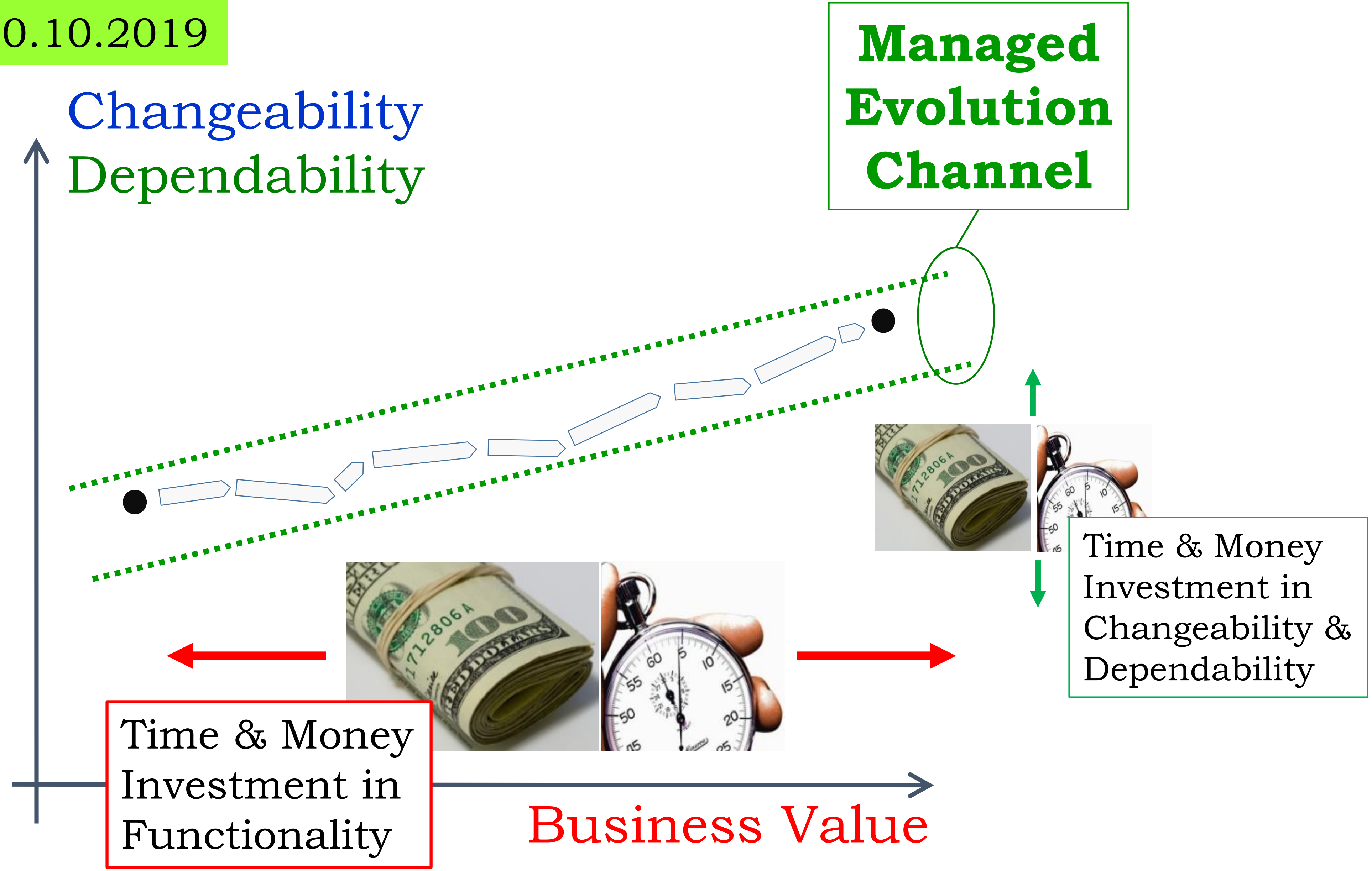
Summary 30.10.2019

Continuous development of both business value **and** changeability leads to a sustainable system **(= path to future-proof software-systems)**



Trajectory Case 2:
Managed Evolution

Summary 30.10.2019



Changeability
Dependability

Managed Evolution Channel

Time & Money Investment in Changeability & Dependability

Time & Money Investment in Functionality

Business Value

Future-Proof Software-Systems: Summary 16.10.2019

Summary 30.10.2019

https://spin.atomicobject.com

The execution of the *managed evolution strategy* assures:

1. The optimum generation of **business value**
2. The continuous improvement of **changeability**
3. The reliable increase in **dependability**
4. The guarantee of the other **quality attributes**

⇒ therefore: The sustainable increase of the **value** of the software

Summary 30.10.2019

Is there an obstacle to managed evolution?

☑ YES

□ NO

http://wohleranzeiger.ch/seilziehen/index.html

Business People

CIO & IT-Architects

Conflict of Interests: Time-to-Market, Development Cost vs. Clean implementation

Business wants:

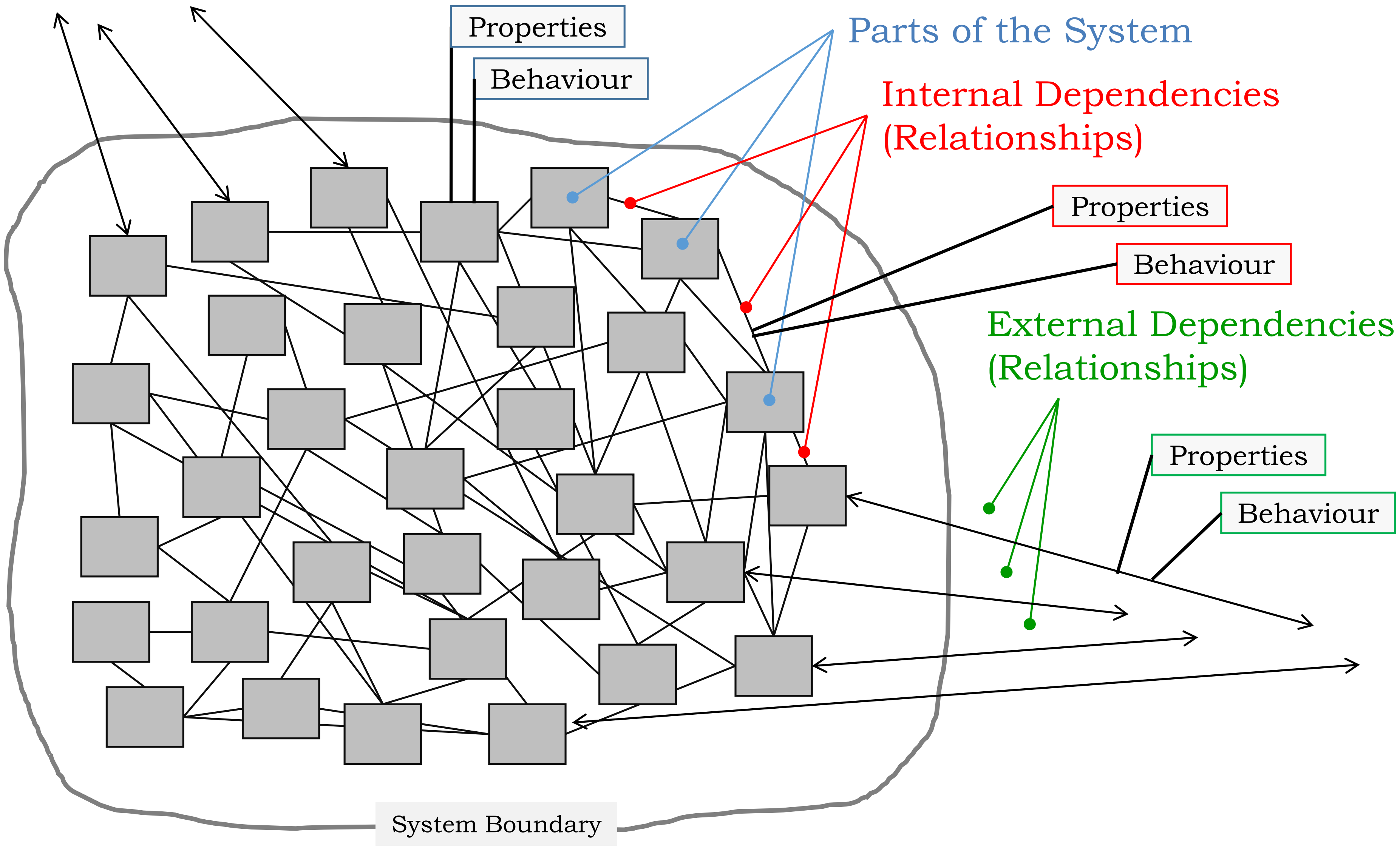
- (Very) short time to market
- Low cost
- Only essential functionality
- Newest technology

CIO & Architecture want:

- Improving Changeability
- Improving Dependability
- Limit growth in complexity
- No technical debt & architecture erosion

Summary 30.10.2019

Definition: **IT Architecture**



IT Architecture

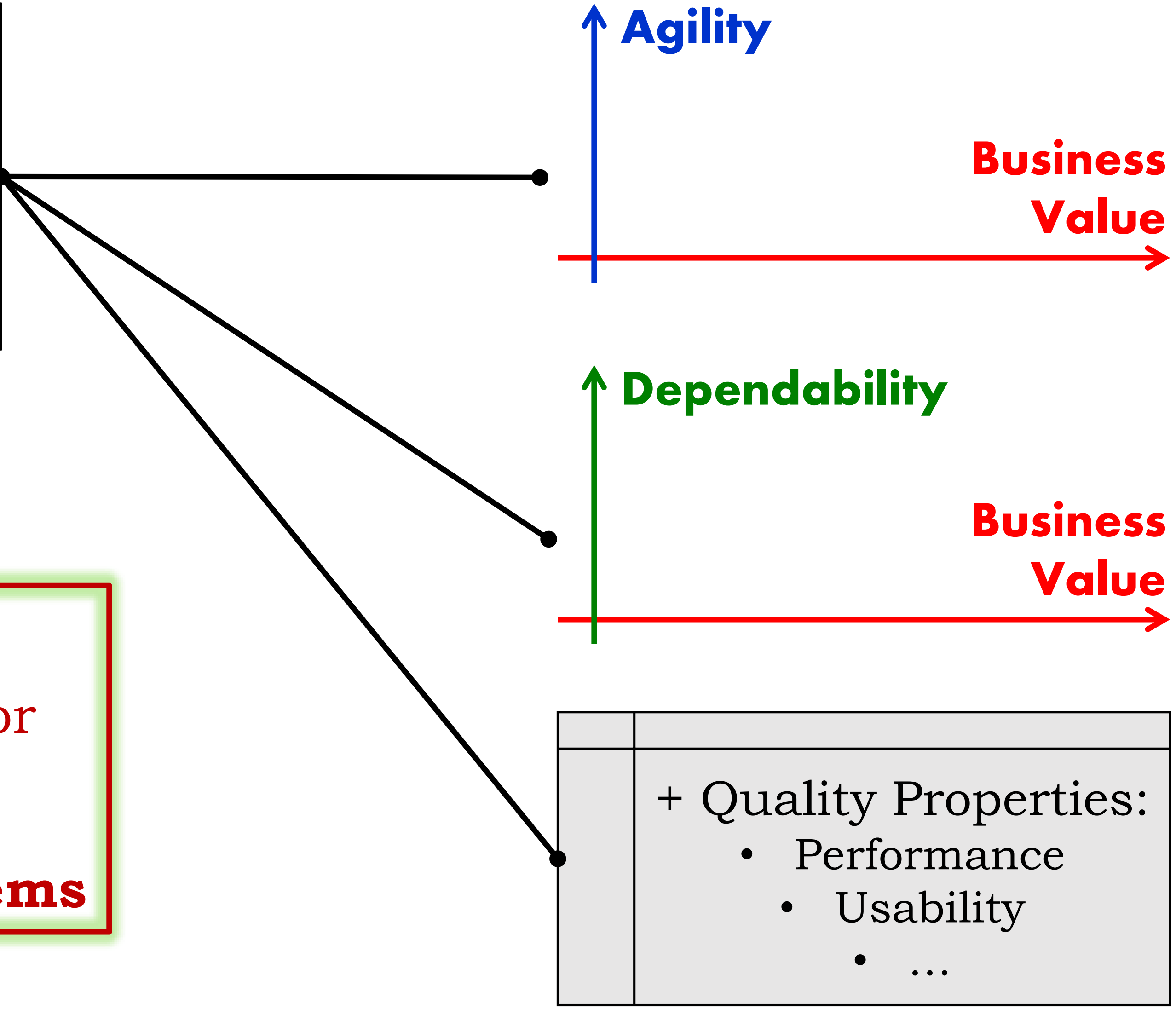
Definition:

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

[adapted from IEEE00]

Summary 30.10.2019

The structure of the system – i.e. its **architecture** – determines to a large extent the properties of the system



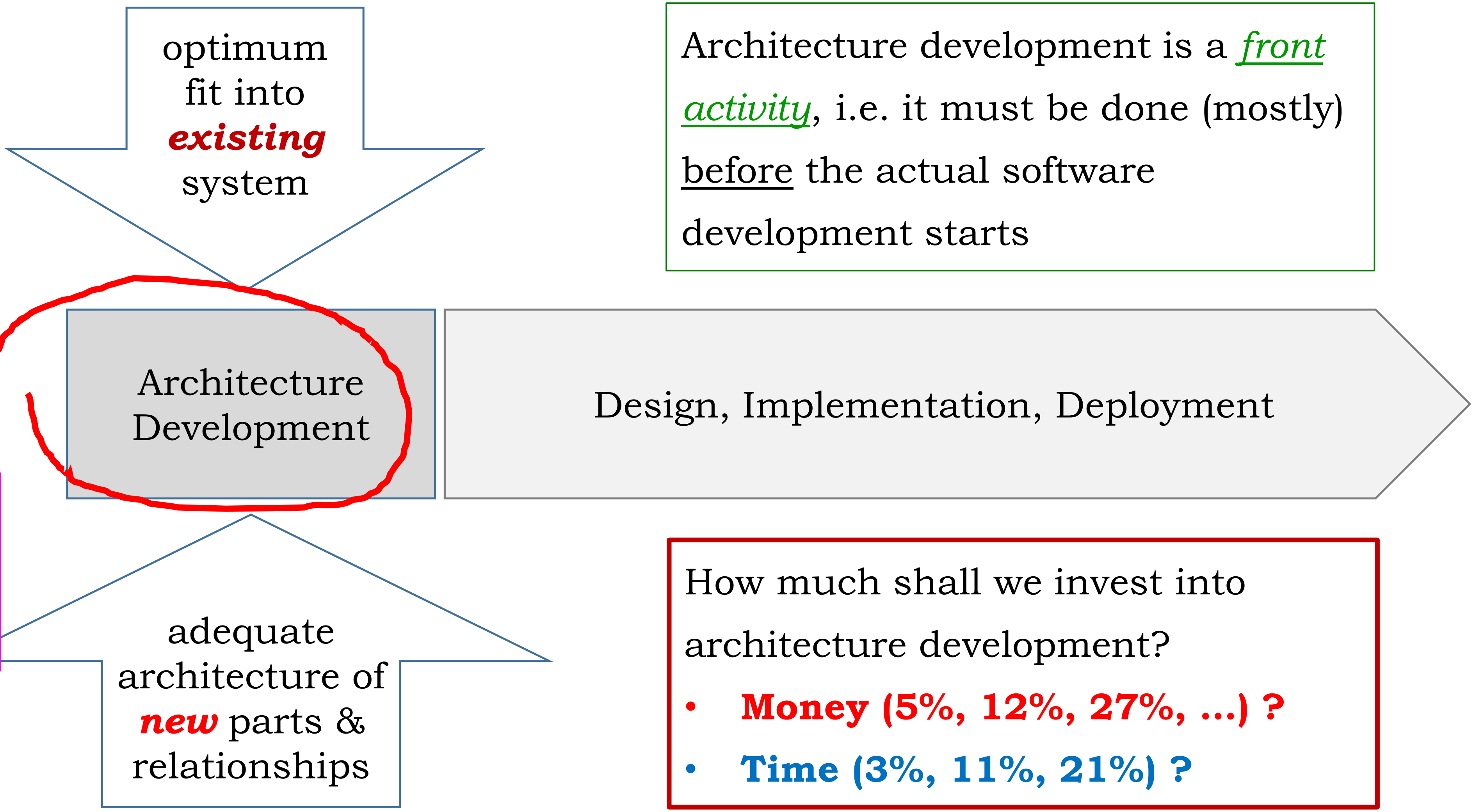
+ Quality Properties:
- Performance
- Usability
- …

Architecture is the most important factor for **future-proof software-systems**

Future-Proof Software-Systems: Summary 16.10.2019

Summary 30.10.2019

https://www.peninsulamission.org



optimum fit into ***existing*** system

Architecture Development

Design, Implementation, Deployment

adequate architecture of ***new*** parts & relationships

Architecture development is a ***front activity***, i.e. it must be done (mostly) before the actual software development starts

How much shall we invest into architecture development?

- **Money (5%, 12%, 27%, ...) ?**
- **Time (3%, 11%, 21%) ?**

Neglecting architecture is planned failure

https://openpuma.wordpress.com

Summary 30.10.2019

… Continue with Part 2