Summary of Lecture 01.11.2017

http://englishskills.se

Summary 01.11.2017

Architects

Development Teams

Managed Evolution

«Software Quality»

Evolution: Add, change, delete

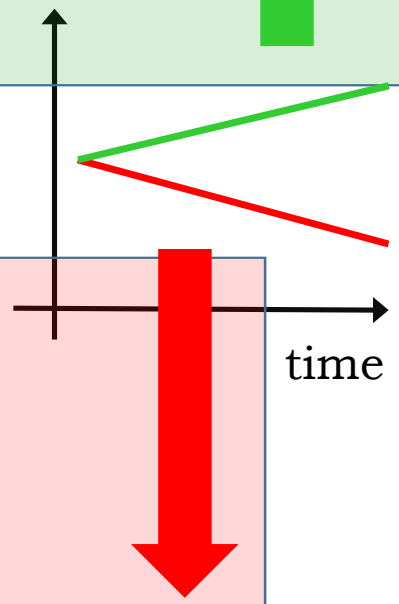
Complexity

Change

Uncertainty

Technical Debt

Architecture Erosion



Summary 01.11.2017

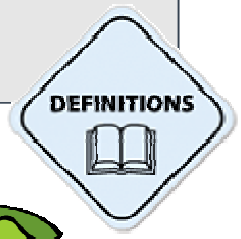
Causes of Technical Debt:

- Architecture Erosion
- Disruptive technology
- **Accumulation of mistakes + shortcuts (e.g. breaking partitions)**
- Dead code (missed explementations)
- **Bad (or ignored) programming best practices & guidelines**
- **Violation of Architecture Principles, e.g. unmanaged redundancy**
- Deferred refactoring
- Progress in software-engineering (e.g. programming languages)
- Careless or skipped upgrades
- **Missing or bad documentation**

… and some more

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)



Summary 01.11.2017

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

DEFINITIONS

Architecture Erosion:

Any IT-architecture is continuously *degenerating* due to many factors:

- Accumulation of technical debt
- SW Paradigm changes (e.g. SOA)
- New laws & regulations
- New standards (e.g. interoperability standards)
- New technology platforms (e.g. Web Services)
- Introduction of new architecture principles
- Complexity increase
- New malicious activities

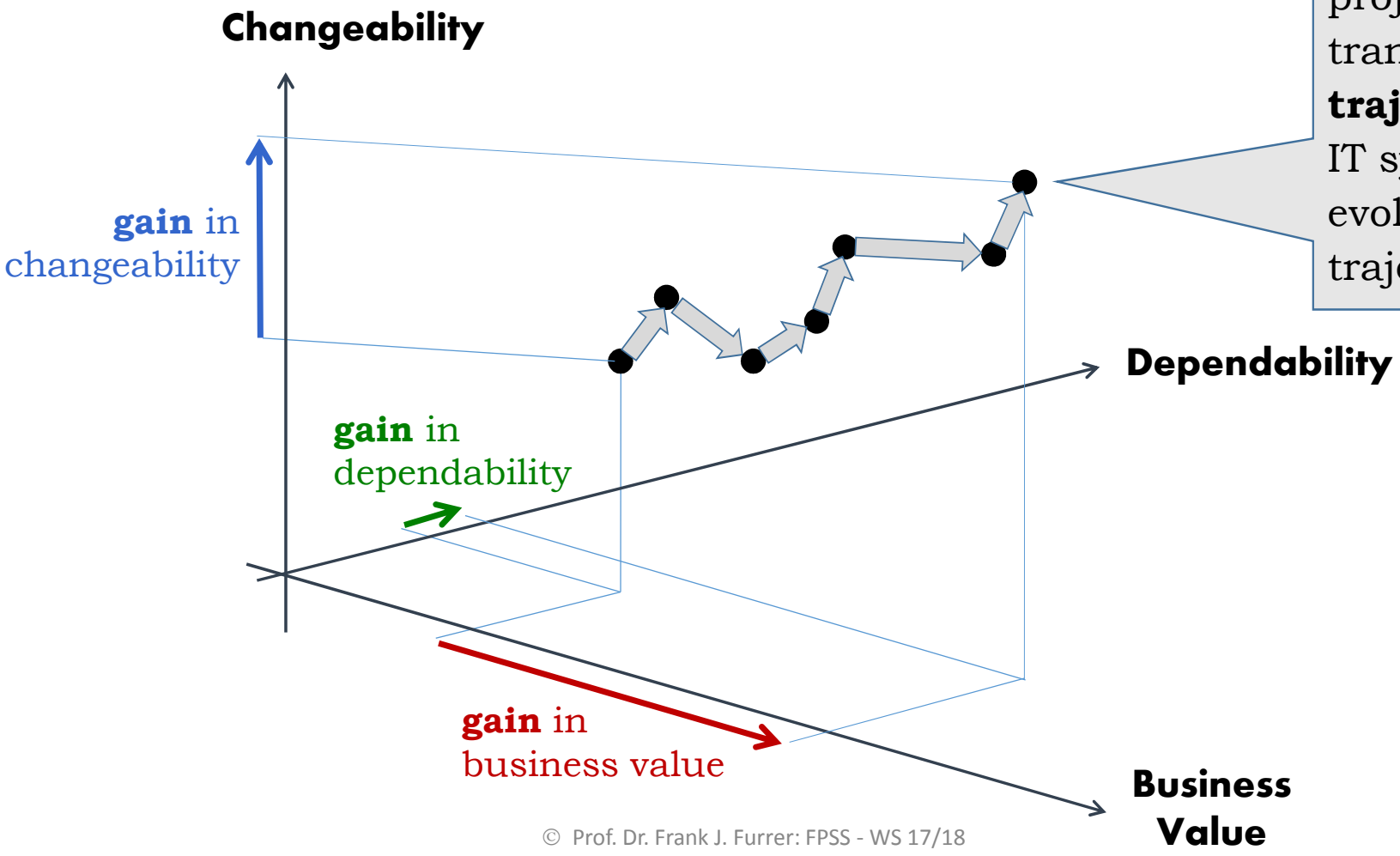
… and some more

http://thoreau.colonial.net/Students/EricksonHoyt/erosion

Summary 01.11.2017

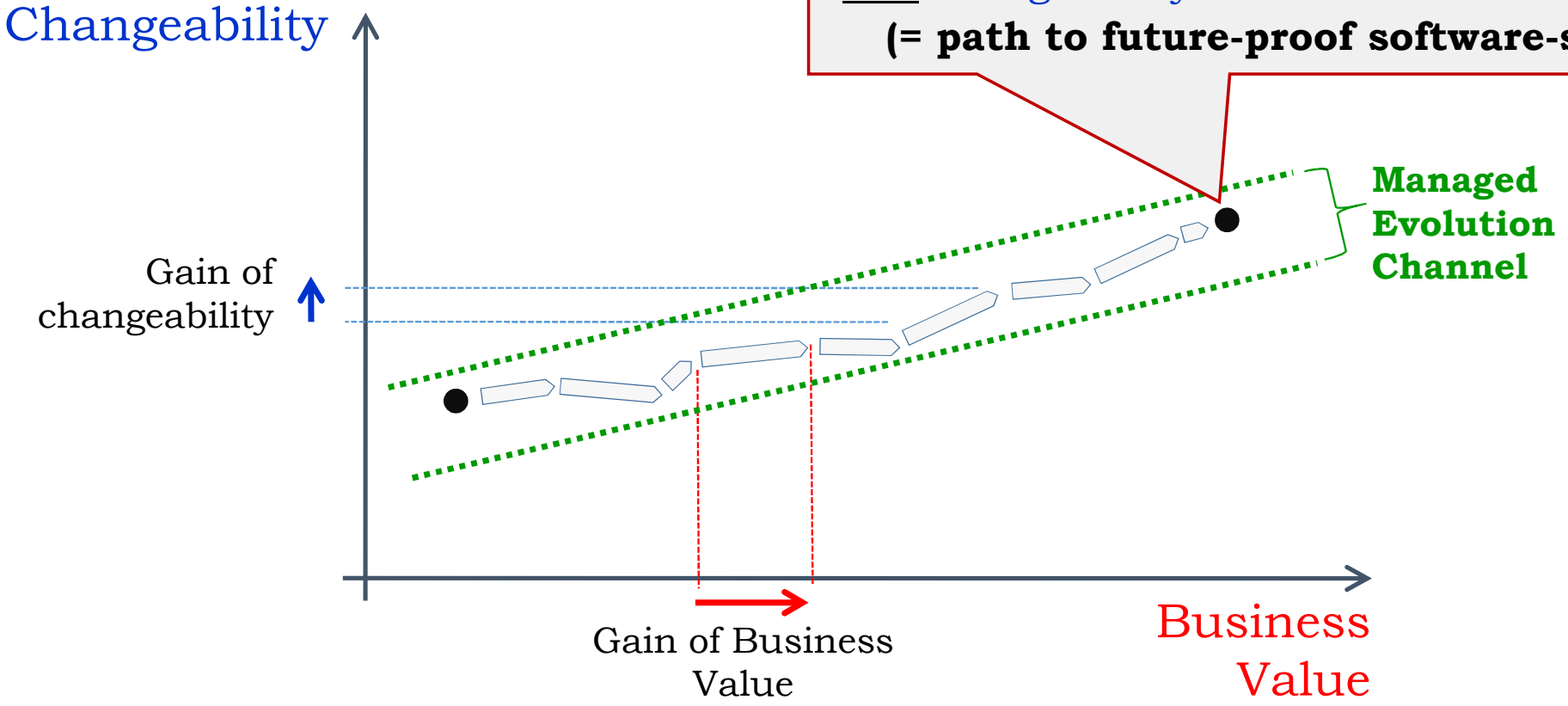
Evolution Trajectory = Sequence of Projects

A sequence of projects builds a transformation **trajectory** of the IT system (= the evolution trajectory)



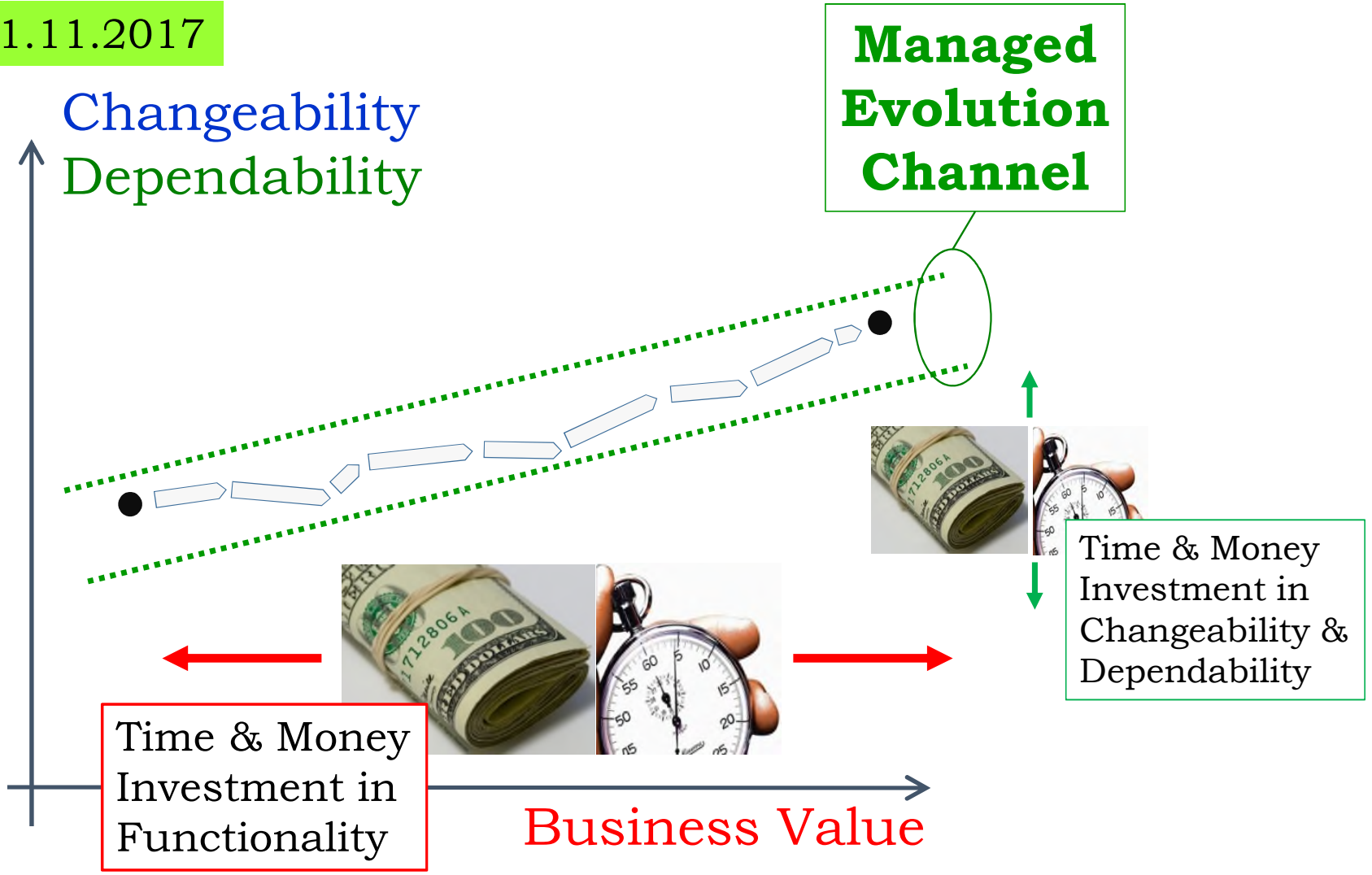
Summary 01.11.2017

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



Note: same for dependability

Summary 01.11.2017



Summary 01.11.2017

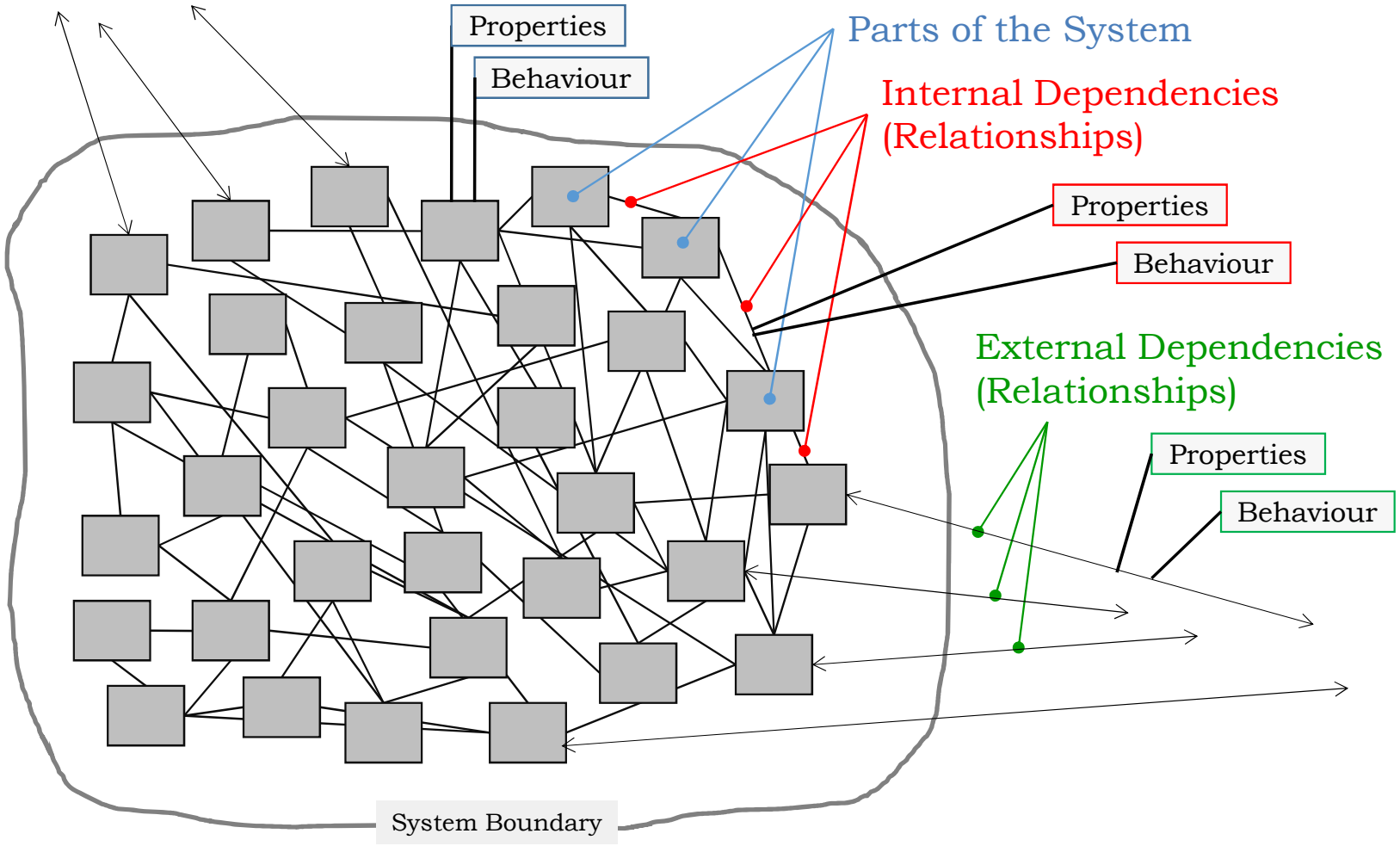
Importance of Architecture

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

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

Which structure is easier to expand and evolve?
Which structure has the better properties, e.g. quality of life?
Which structure is future-proof (expandable)?

Summary 01.11.2017



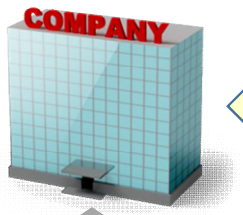
Definition:
IT Architecture

"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"

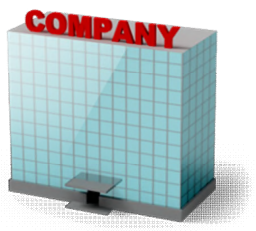
[adapted from IEEE00]

Summary 01.11.2017

Architecture Levels



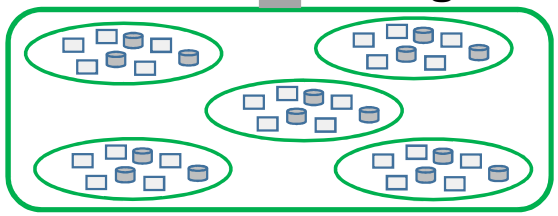
Systems-of-Systems



Enterprise-Architecture

SoS-Architecture

Integration

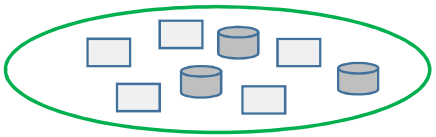


Application Landscape

Application Landscape Architecture

Integration

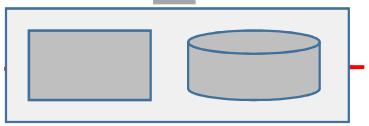
Technology-**in**dependence



Application

Application-Architecture

Integration



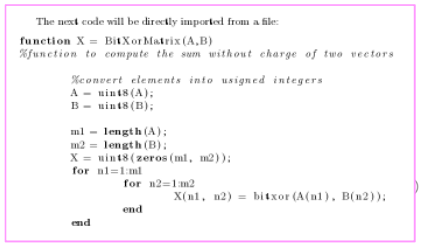
Component

Component-Architecture

Component-Design

Integration

Technology-dependence



```
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
```

Program Module

Program-/Module-Design

Summary 01.11.2017

System/Software Engineering/Development Process

http://www.ovocreatives.com

http://stevendwood.com

Sum of all decisions

IT Architect

Summary 01.11.2017

Good Architecture:

- Manages essential **complexity**
- Minimizes accidental **complexity**
- Provides optimal **changeability** (= minimum resistance to change, DevC, TtM)
- Enables **dependability** and other quality properties
- Reduces the impact of **uncertainty**
- «Fun to work»

Bad Architecture:

- Difficult to **understand**, maintain and evolve
- Messy **dependencies** («far effects»)
- **Erosion**: «Path to Death»
- Entangled quality properties (Orthogonality)
- Demotivating, «overpriced» work

Summary 01.11.2017

Part 3 **Part 4** **Part 5**

Managed Evolution

Knowledge

Architects

Evolution: Add, change, delete

Development Teams

Good Architecture

Future-Proof Software-Systems

Summary 01.11.2017

Questions

+

Clarifications

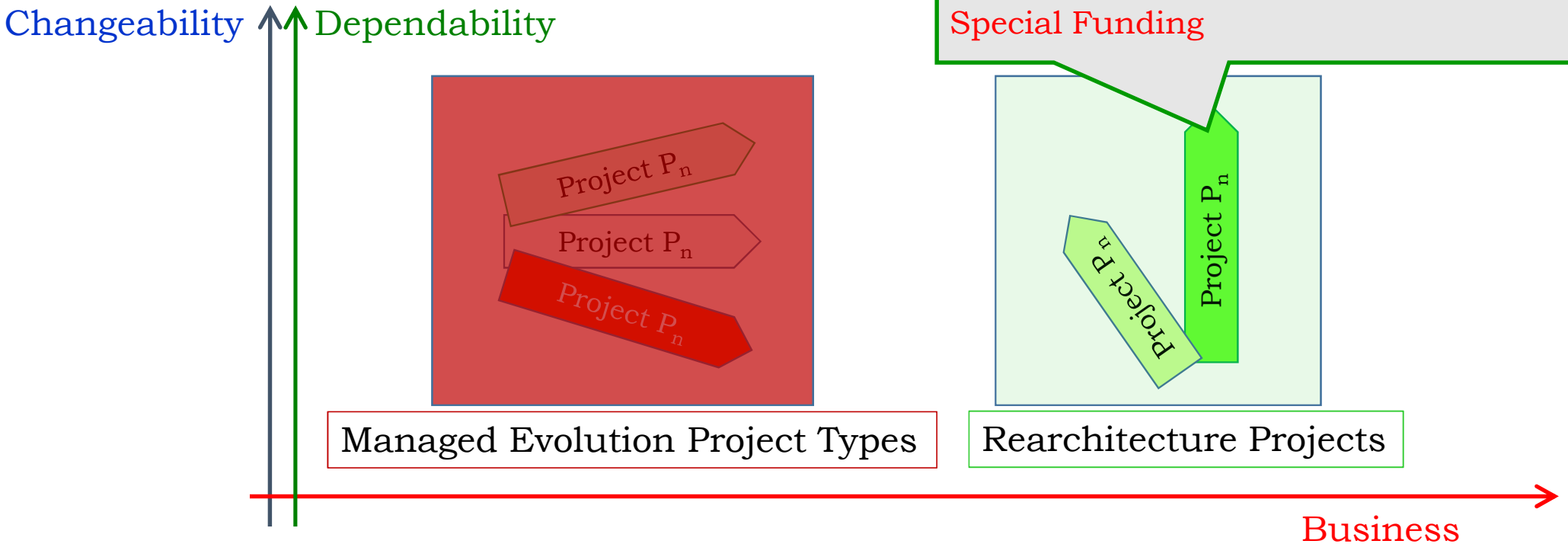
Summary 01.11.2017

Project Types

Purpose:

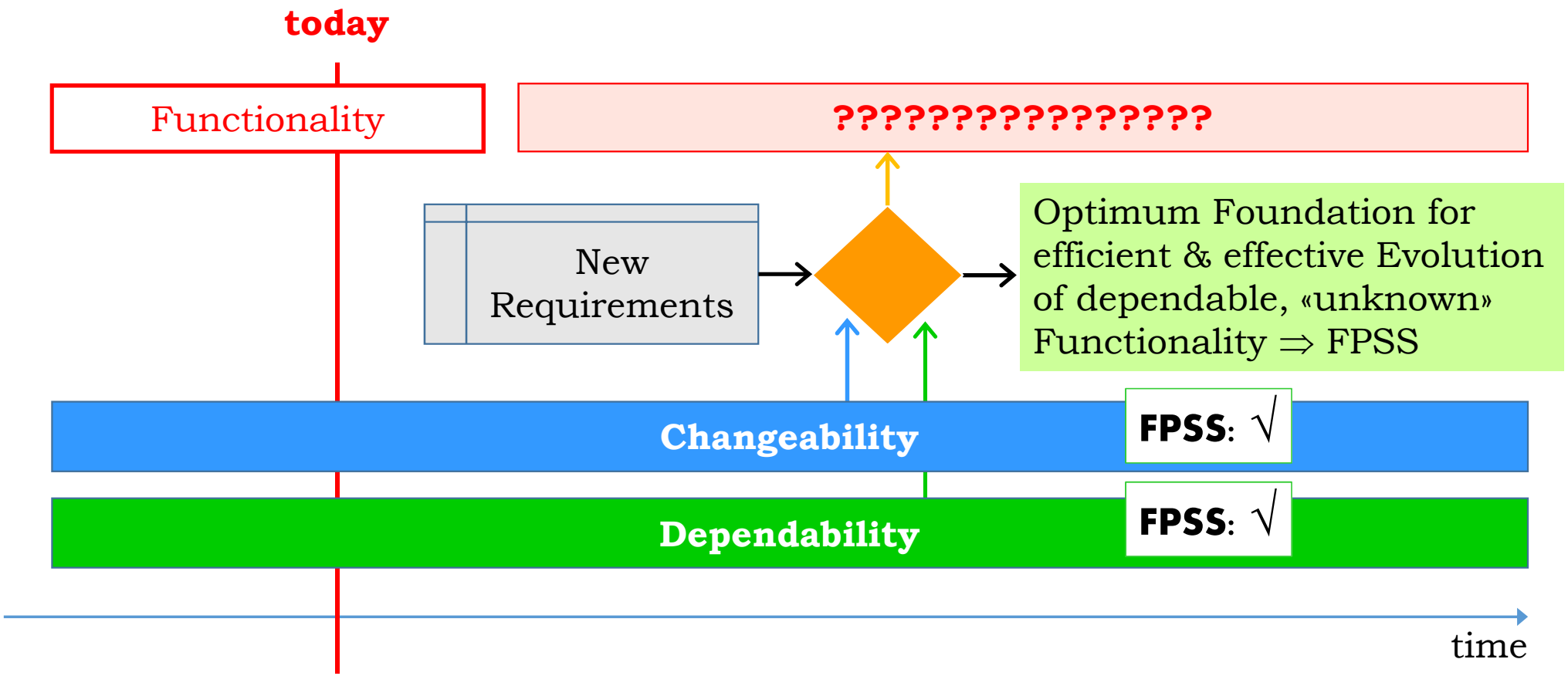
- Eliminate Technical Debt
- Rearchitecting
- Refactoring
- Reengineering

Special Funding



Summary 01.11.2017

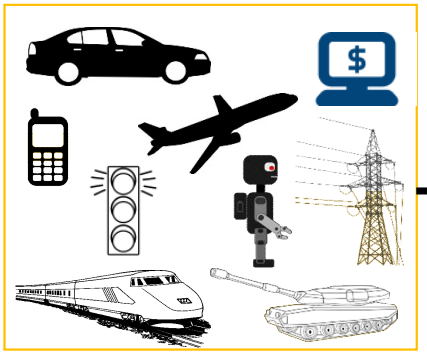
What is future-proof?



Summary 01.11.2017

Dependability Metric

Application Domain



Dependability **expectations**

Dependability **properties**

General Property

Specific Properties

Resilience:

- **Fault-Tolerance** → Submetric
- **Graceful degradation** → Submetric
- **...**

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