Industrial Software:
Business Value, Agility & Resilience

Frank J. Furrer
Dr. sc. techn. ETH-Zürich

Ringvorlesung TU Dresden WS 2014/2015
Montag, 26. Januar 2015

V1.7 /18.01.2015

Frank J. Furrer

- Dipl. El. Ing. ETH Zurich (1970)
- Ph.D. Technical Sciences ETH Zurich (1974)
- Management Consultant: «Very Large Software Systems»
- EC Expert (Evaluator & Reviewer): «Safety-Critical Systems»
- TUD: «Future-Proof Software Systems» (WS13/14, WS14/15)

frank.j.furrer@bluewin.ch / http://st.inf.tu-dresden.de/teaching/fps

Line of Thought

Context: A world of software

⇩

Vision: Future-Proof Software-Systems

⇩

Strategy: Managed Evolution

⇩

Implementation: Industrial Software

⇩

Foundation: Architecture Principles

⇩

Aging: Architecture Erosion & Technical Debt

http://www.wissenswertes.biz

Line of Thought

Context: A world of software

⇩

Vision: Future-Proof Software-Systems

⇩

Strategy: Managed Evolution

⇩

Implementation: Industrial Software

⇩

Foundation: Architecture Principles

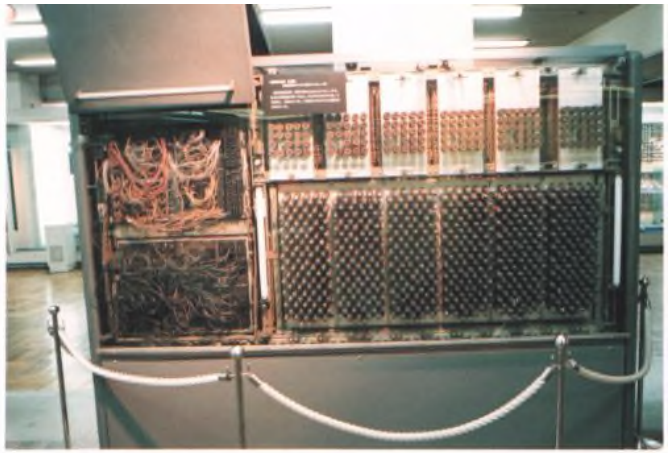
⇩

Aging: Architecture Erosion & Technical Debt

http://www.wissenswertes.biz

Industrial Software: Business Value, Agility & Resilience

UNIVAC 120 (1953)



http://web.kyoto-inet.or.jp/people/s-oga/oldcom/index.html

Decimal storage: 120 digits
„Program“: Wired Panel

+ 60 years

CRAY Titan (2013)

http://www.eweek.com/servers/slideshows

Data storage: 710 Terabytes RAM
30 Petabytes Disk
Computing Power: 17,59 Petaflops

http://www.motorbase.com/picture/by-id/437941611

Land Rover 1953: SLOCs = 0
(SLOC = Source Lines of Code)

http://myauto24.blogspot.ch/2013/

Mercedes S-Class 2013: SLOCs ≈100 Million

❶ Software has evolved from a system *building block*

to a major *industrial asset*

with a large financial *investment*

and tremendous impact on the *business opportunities*

❷ Many software modules have a long life span.

They need to be maintained and evolved over many years/decades

Therefore: Our software needs to be **future-proof**

(Definition follows later)

UNIVAC 120 (1953)

CRAY Titan (2013)

http://web.kyoto-inet.or.jp/people/s-oga/oldcom/index.html

https://www.mercedes-benz.com/en/mercedes-benz/innovation

http://www.motorbase.com/picture/by-id/437941611

http://myauto24.blogspot.ch/2013/

> 2015: 80% der technischen Innovation

= Software

Land Rover 1968: SLOCs ~0

Mercedes S-Class 2013: SLOCs ~100 Million

(SLOC = Source Lines of Code)

Line of Thought

Context: A world of software

⇩

Vision: Future-Proof Software-Systems

⇩

Strategy: Managed Evolution

⇩

Implementation: Industrial Software

⇩

Foundation: Architecture Principles

⇩

Aging: Architecture Erosion & Technical Debt

http://www.wissenswertes.biz

Software-Systems: **Impact Factors**

„The three devils of systems engineering are:

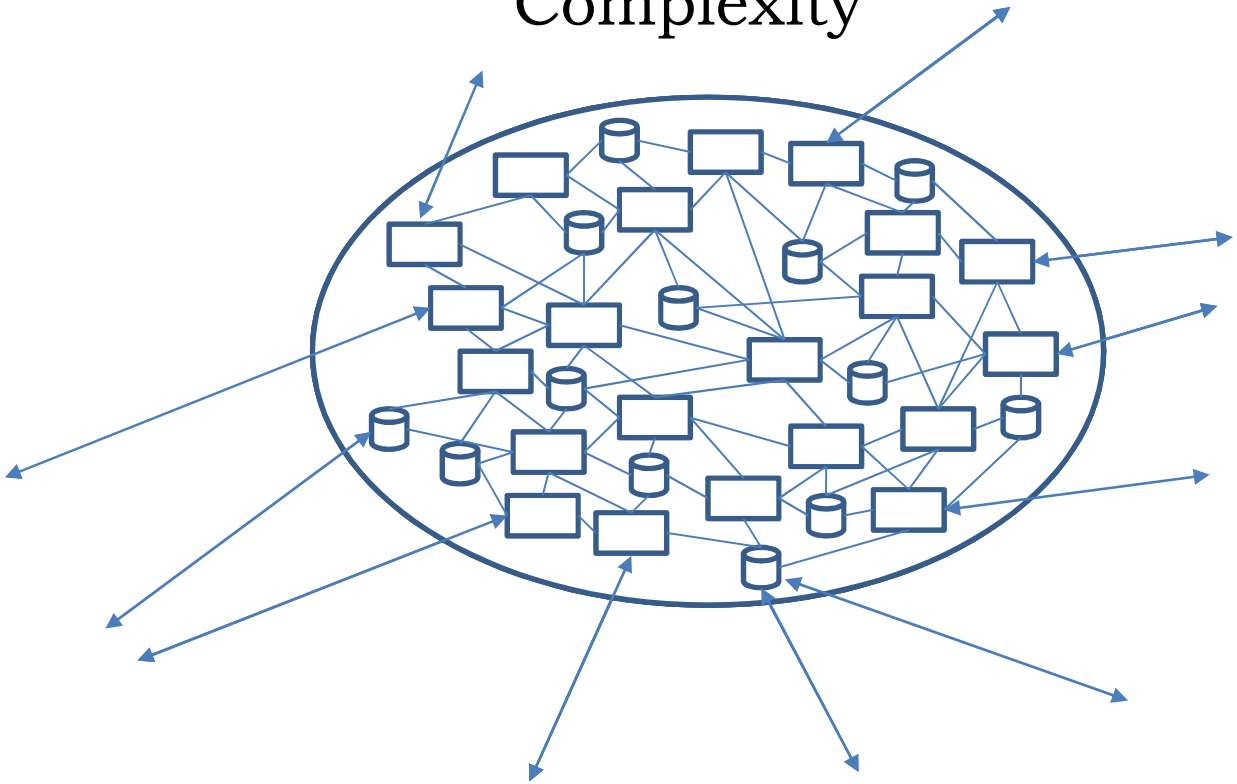
- Complexity,
- Change,
- Uncertainty"

Anonymous

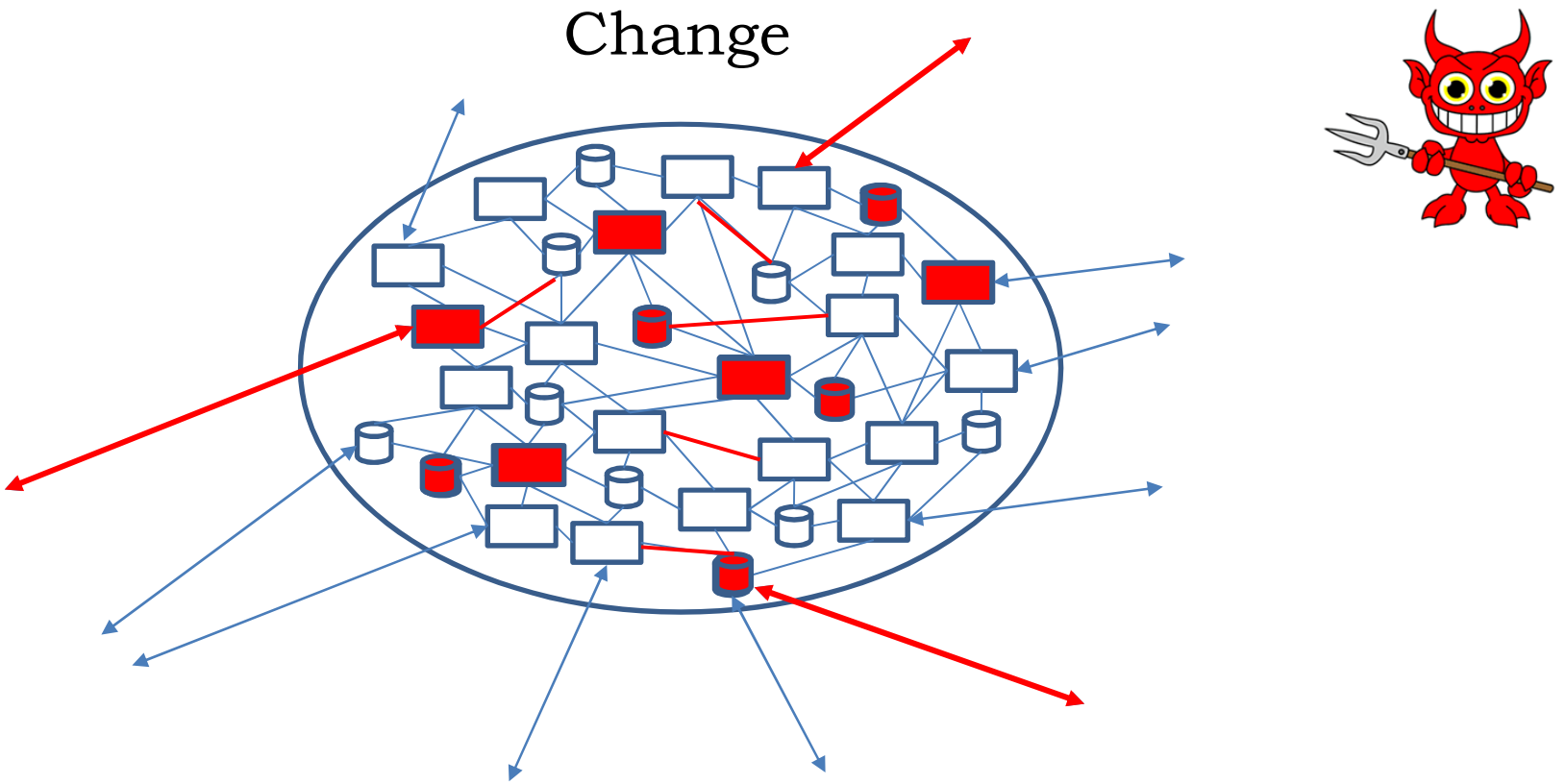
What do they do to our software?

How can we fight them?

Complexity

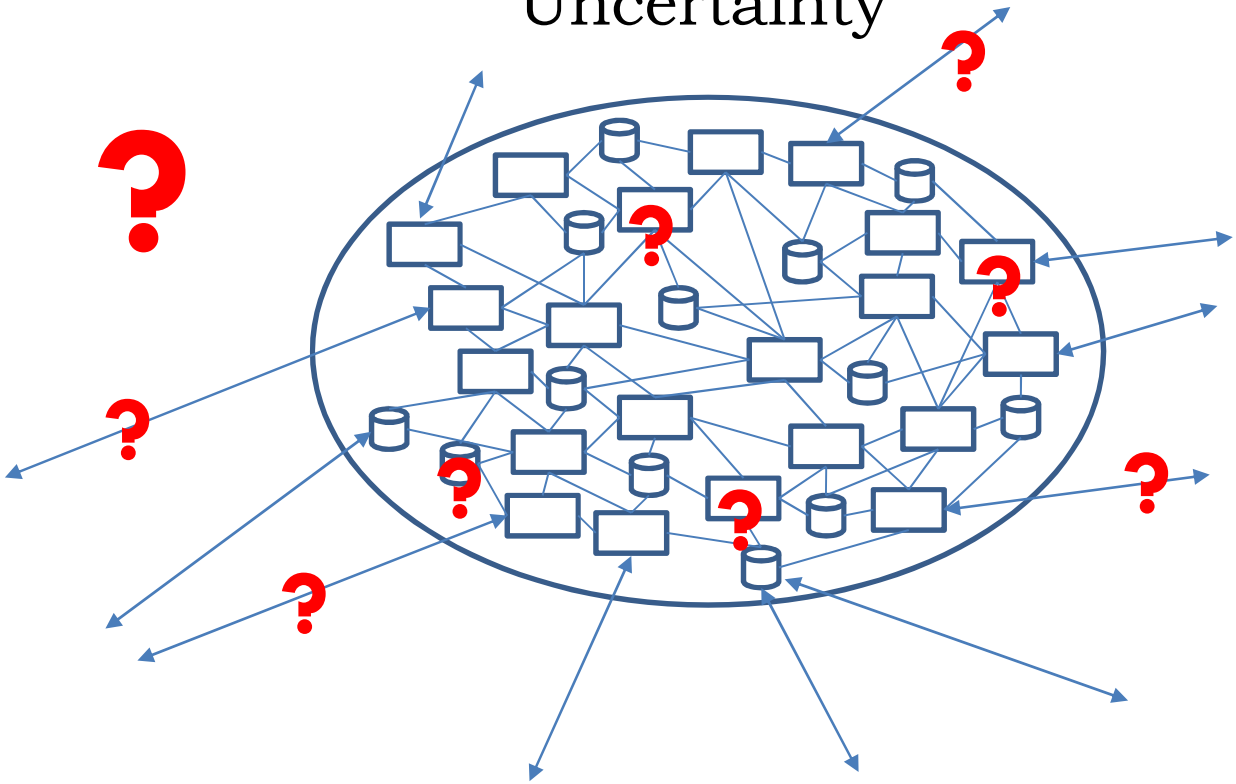


"**Complexity** is that property of an IT-system which makes it difficult to formulate its overall behaviour, even when given complete information about its parts and their relationships"



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

Uncertainty



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

How can we successfully fight them?

Complexity

Change

Uncertainty

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

Vision: Future-Proof Software-Systems

Definition:

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

http://juliecoleman.org

Vision: Future-Proof Software-Systems

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

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

Definition:

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

Best value for the parameters 'money' and 'time-to-market'
→ **Agility**

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

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

Future-Proof Software-Systems: **Primary Characteristics**

Business Value

Definition:

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

Agility

Resilience

Domain-specific **Quality Properties**

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

Primary Characteristics:

- Business Value
- Agility
- Resilience

«continuous improvement»

Secondary Characteristics:

- Non-functional properties:
  - Reusability
  - Hardware Resource Consumption
  - Adherence to industry-standards
  - etc.

«as good as necessary»

Business Value

http://blog.enfocussolutions.com

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

Business Value (of a software development) =

The opportunity to gain an **advantage** for the business

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

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:	$(1+0.8)^{-1}$	$(1+0.8)^{-2}$	$(1+0.8)^{-3}$	$(1+0.8)^{-4}$	$(1+0.8)^{-5}$
	1.08	**1.17**	**1.26**	**1.36**	**1.47**
	+ 222‘000 €	+ 230‘000 €	+ 182‘000 €	+ 205‘000 €	+ 186‘000 €

+ 1‘025‘000 €

Agility

Metric: **Size²/(TtM*DeVC)**

http://giaonthemove.com

Agility =

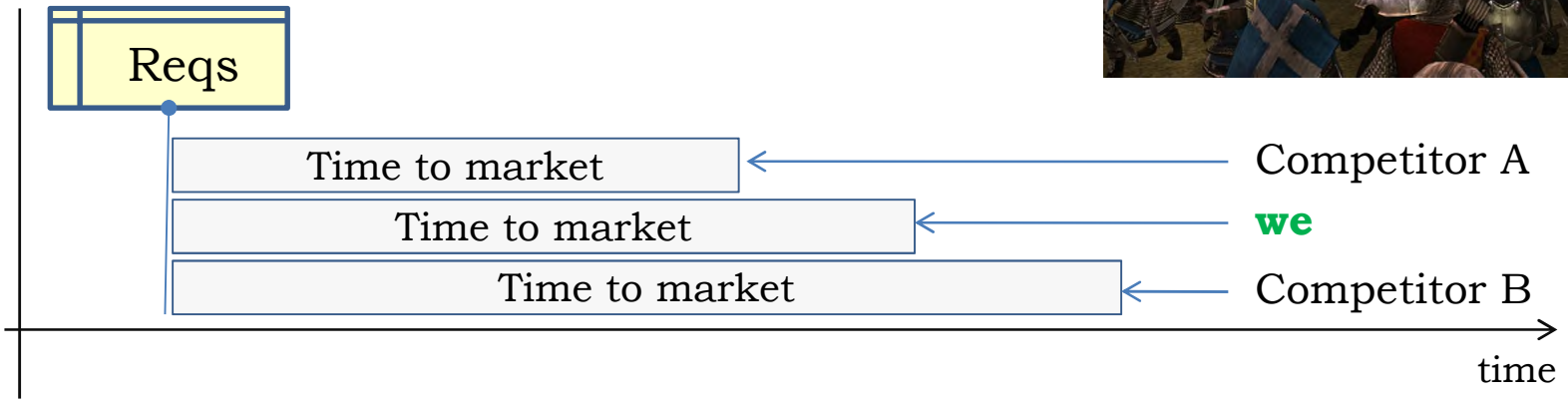
The **capability** to develop new functionality with a

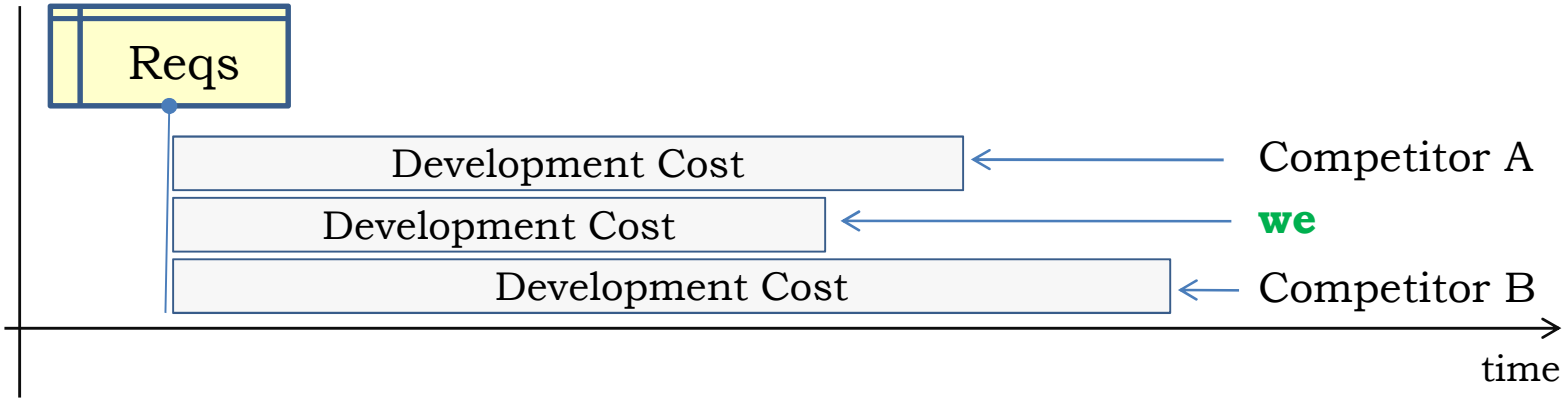
- short time-to-market
- reasonable development cost

Important note: This capability is a property of an *organization*!

Agility

Why is **agility** so important?



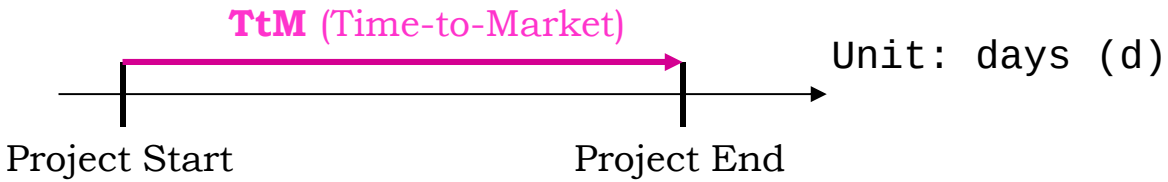


Agility

Agility Metric:

Amount of functionality:
Functional Size

Size Unit: #UCP or #FP



$$\text{Agility} = \frac{(\sum \text{Size}_i)^2}{\sum \text{TtM}_i * \sum \text{DevC}_i}$$

Unit: #UCP/(days*k€)

Resilience

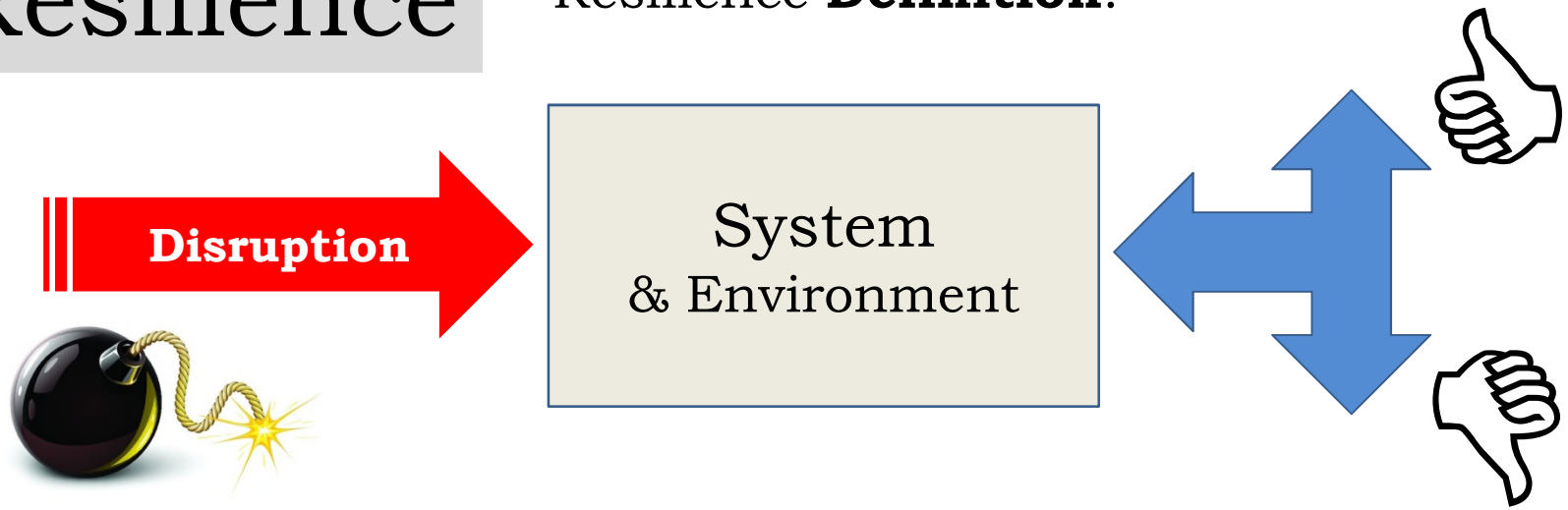
Why *resilience* as a primary characteristic of future-proof software-systems?

⇒ The world has become a dangerous place for software

http://bookroar.blogspot.ch/2012/05/most-dangerous-snakes-in-world.html

Resilience

Resilience **Definition**:

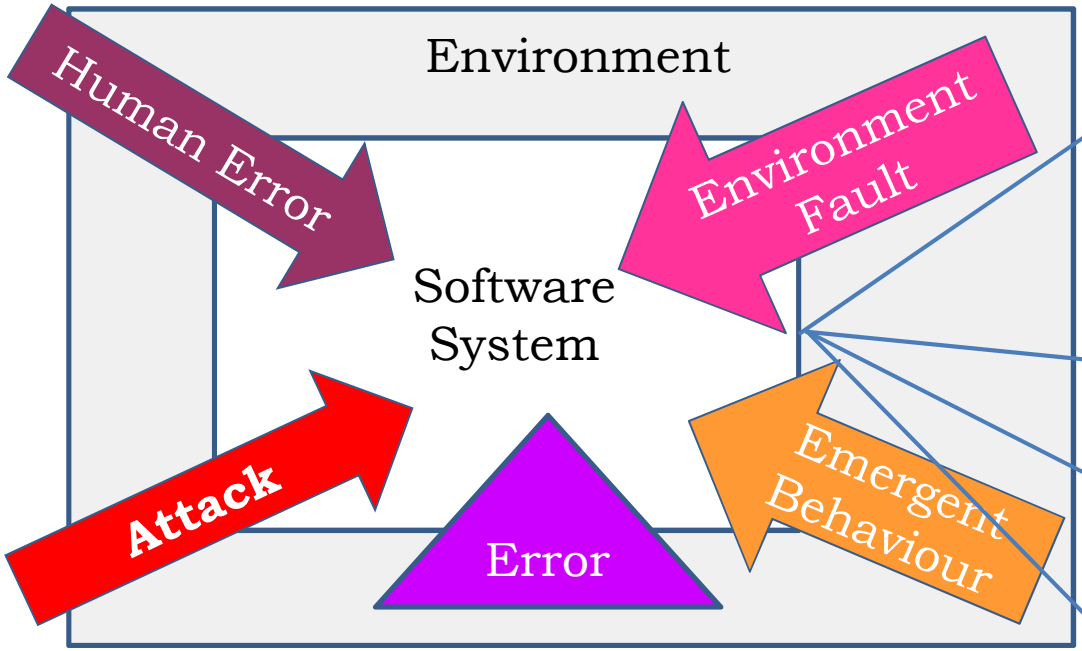


Resilience is the *capability* of a system with specific characteristics before, during and after a *disruption* to *absorb* the disruption, *recover* to an acceptable level of performance, and *sustain* that level for an acceptable period of time

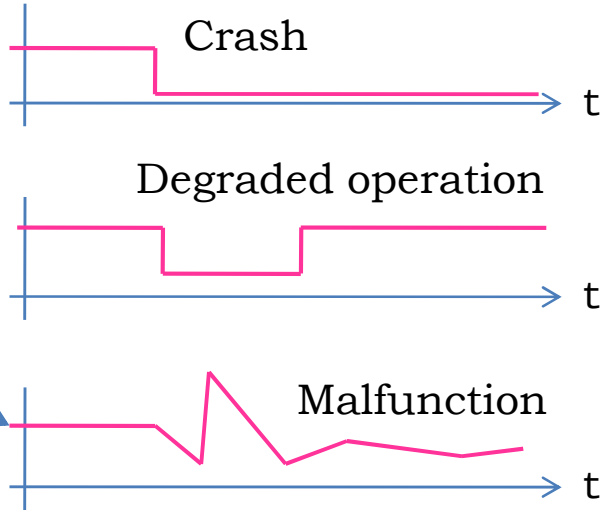
- *Before* – Allows anticipation and corrective action to be considered
- *During* – How the system survives the impact of the disruption
- *After* – How the system recovers from the disruption

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

Resilience



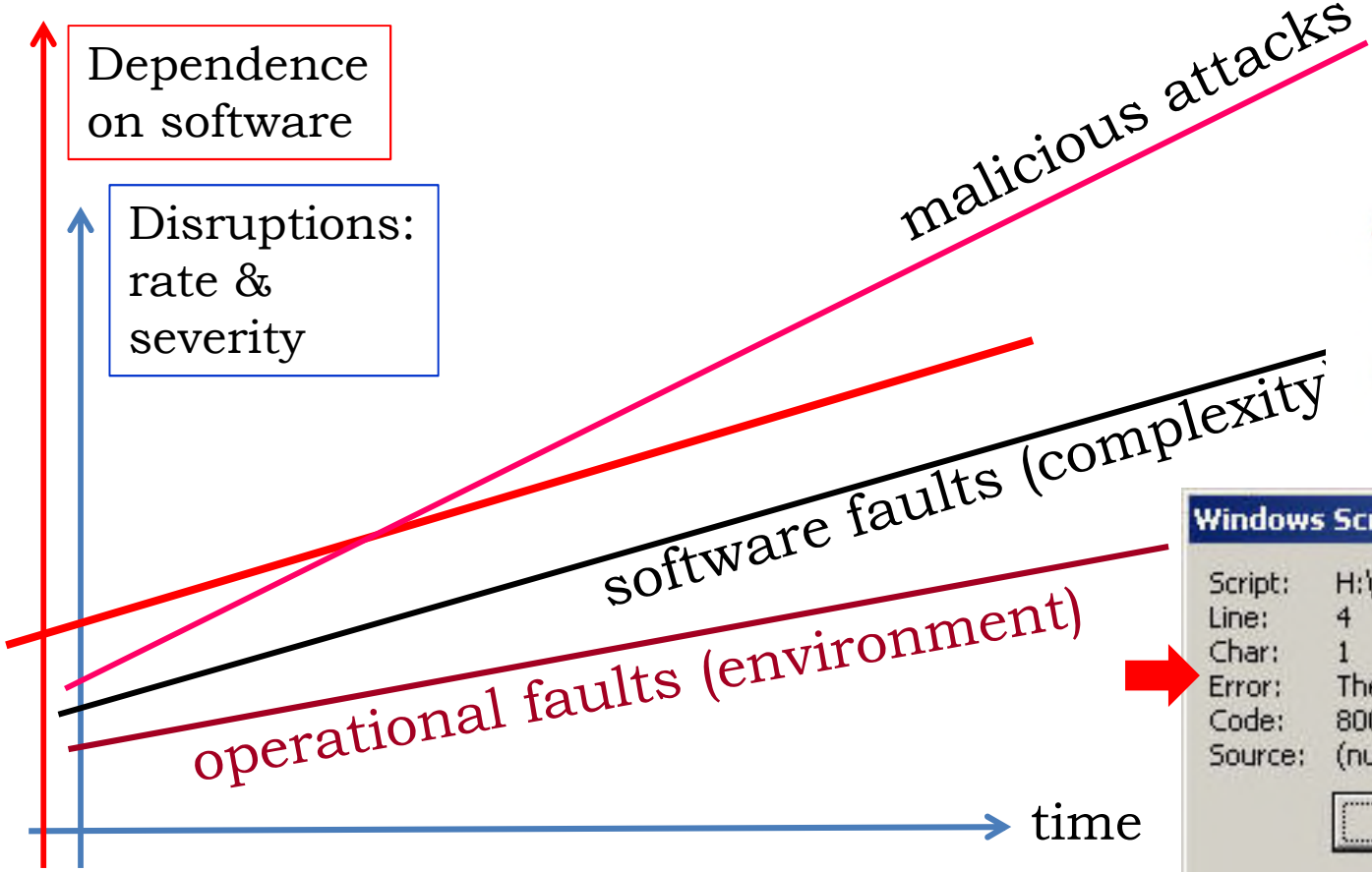
https://soundcloud.com

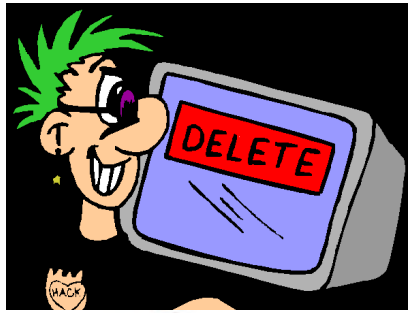


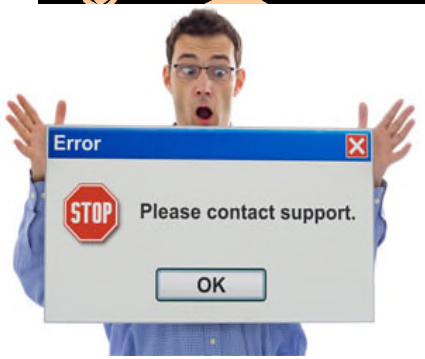
http://kanto.stripes.com

Resilience

Why is **resilience** so important?







http://blog.qatestlab.com

http://www.computerperformance.co.uk

Resilience

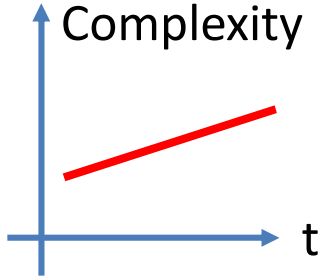
< 2010 → > 2010

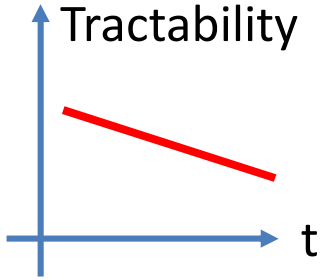
Disruptive Incidents
1. Xxx
2. Yyy
3. Zzz

System

Safety Case

predict





Resilient System

adjust

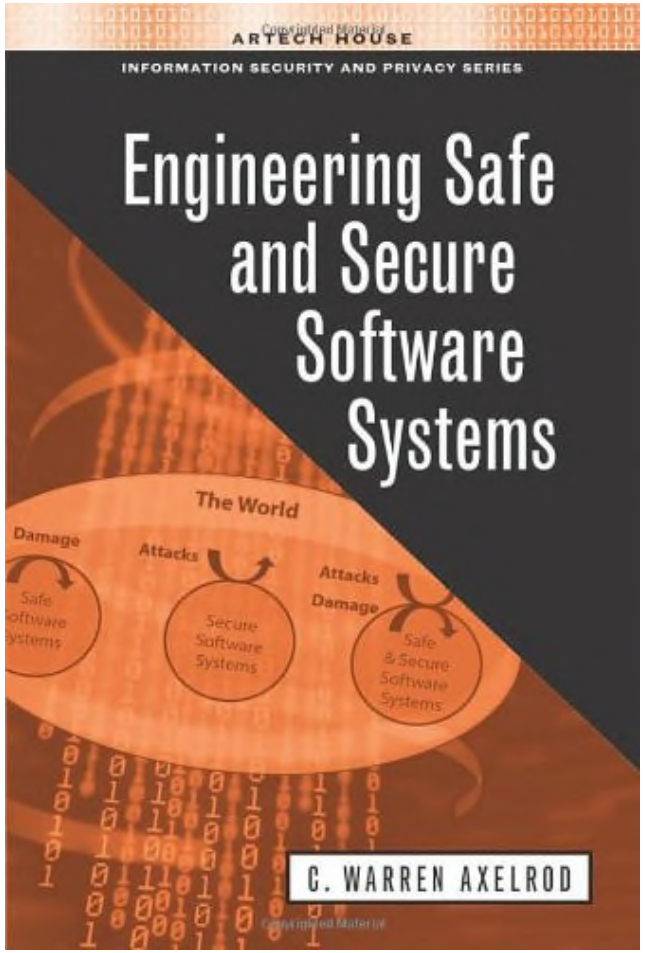
Resilience

«While the general concept of safety and reliability is understood by most parties, the specialty of software safety and reliability is not»

(Debra S. Hermann)

http://www.motivationalmemo.com

Pflicht-lektüre!



C. Warren Axelrod, 2013
ISBN 978-1-60807-472-3

Line of Thought

Context: A world of software

⇩

Vision: Future-Proof Software-Systems

⇩

Strategy: Managed Evolution

⇩

Implementation: Industrial Software

⇩

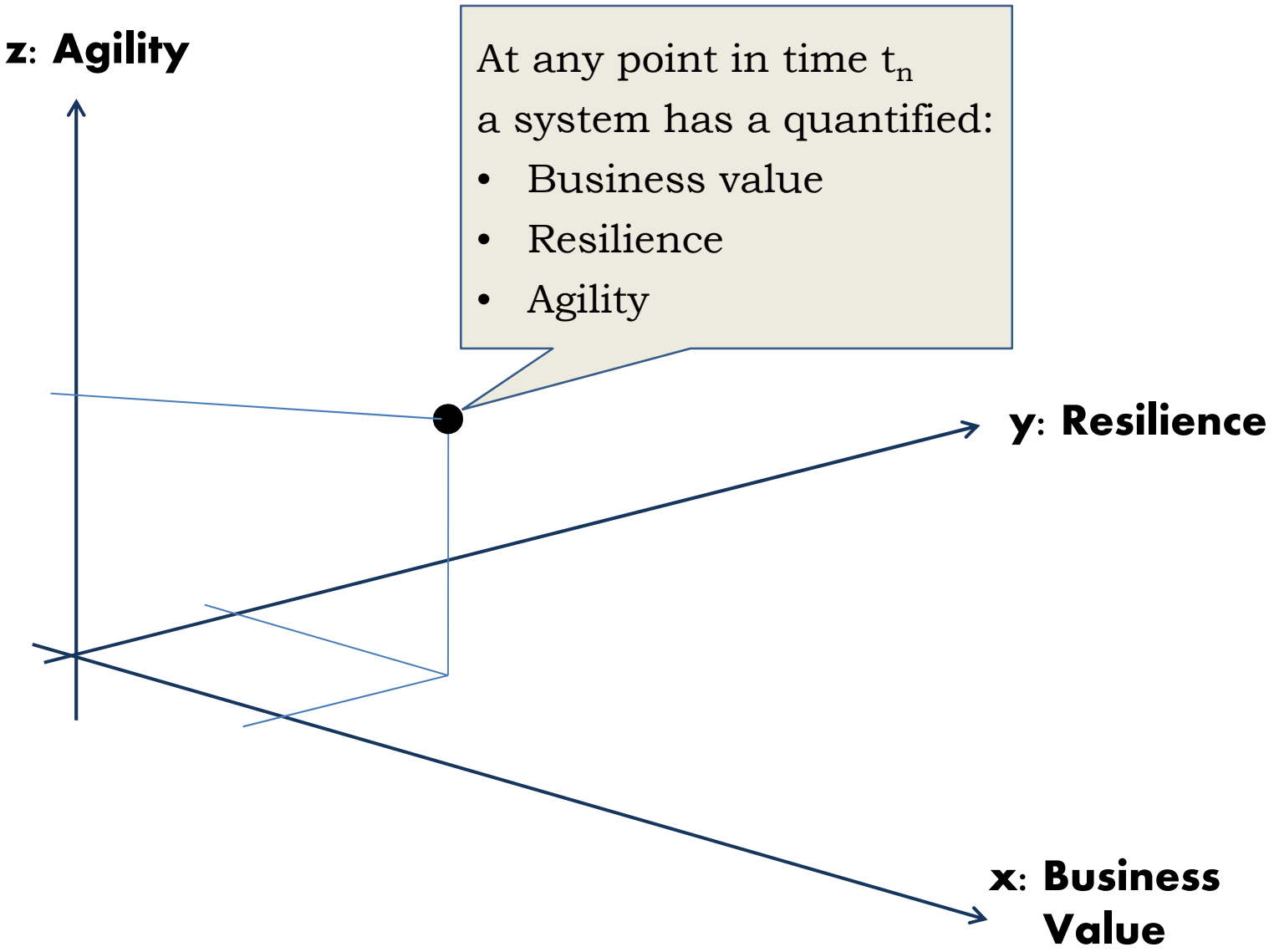
Foundation: Architecture Principles

⇩

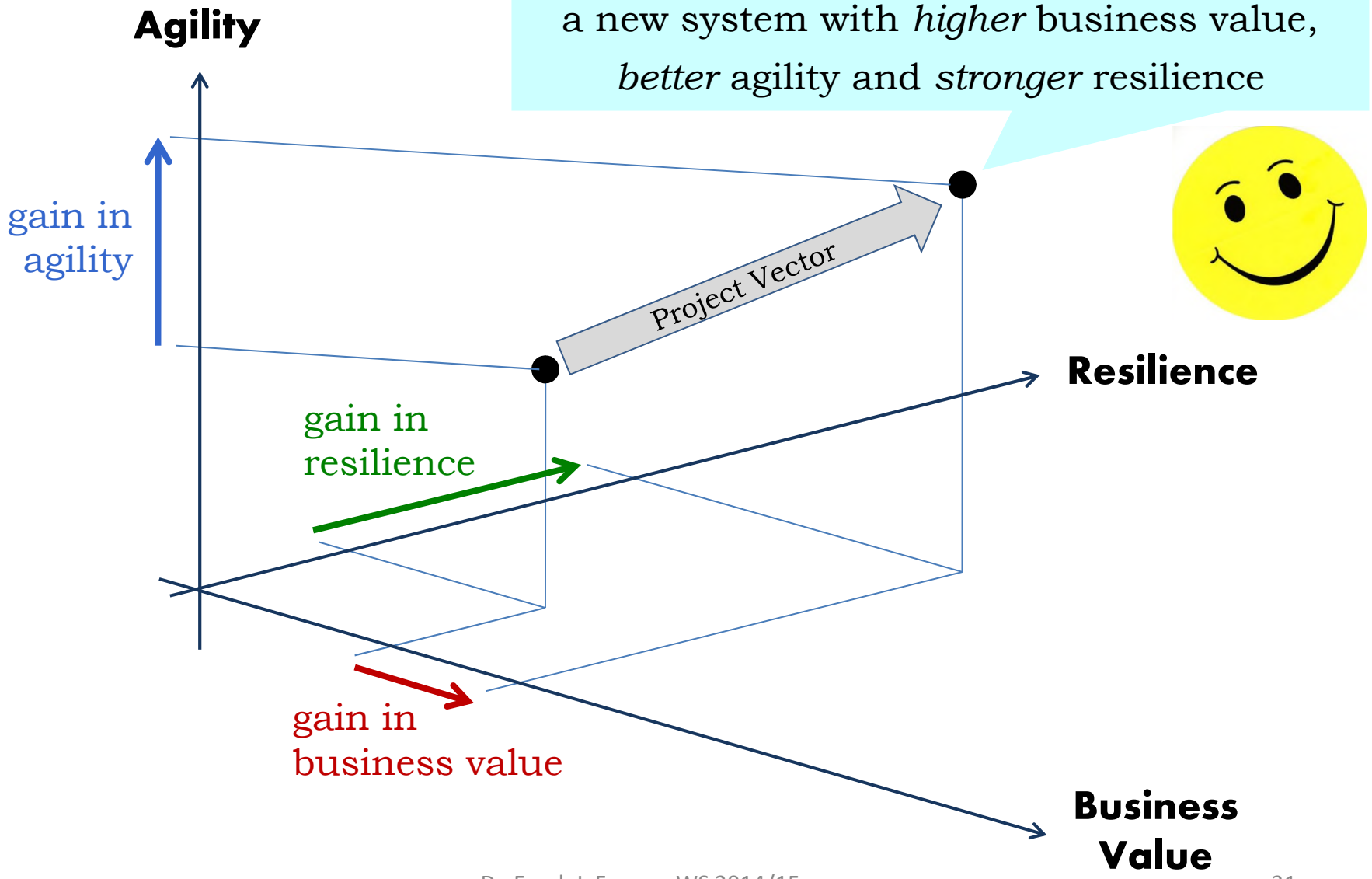
Aging: Architecture Erosion & Technical Debt

http://www.wissenswertes.biz

Future-Proof Software-Systems: **Managed Evolution** Coordinate System

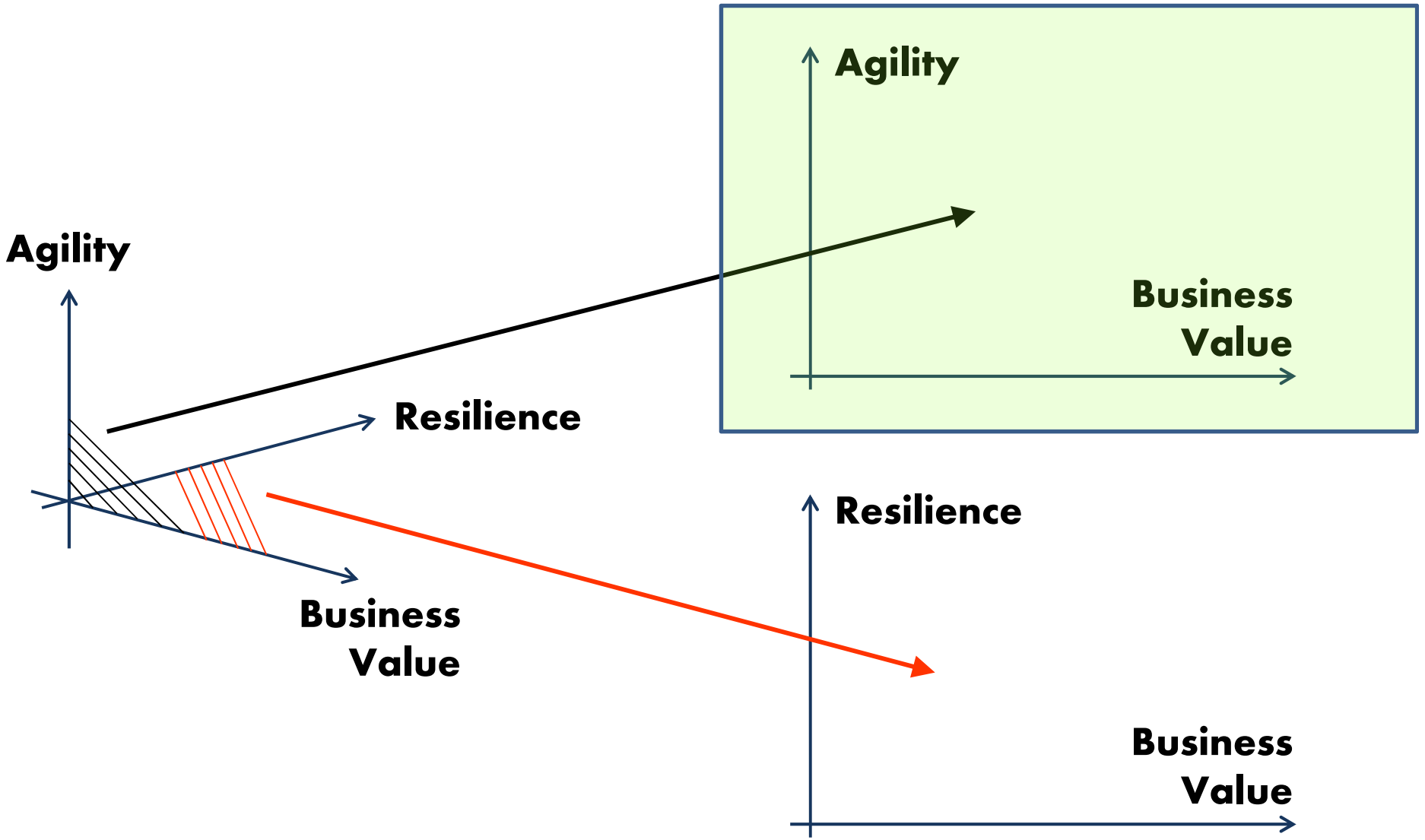


The project has transformed the system into a new system with *higher* business value, *better* agility and *stronger* resilience

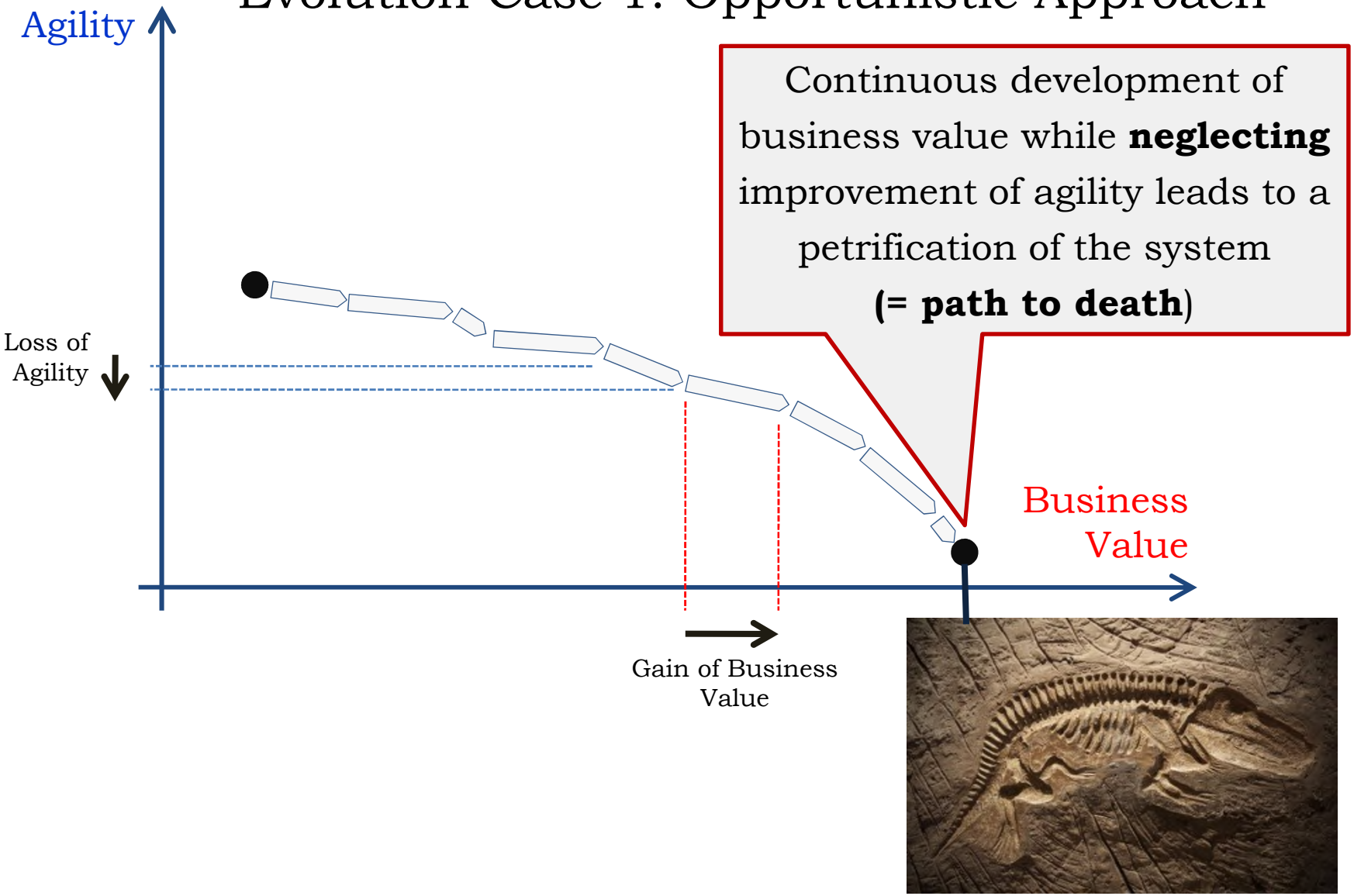


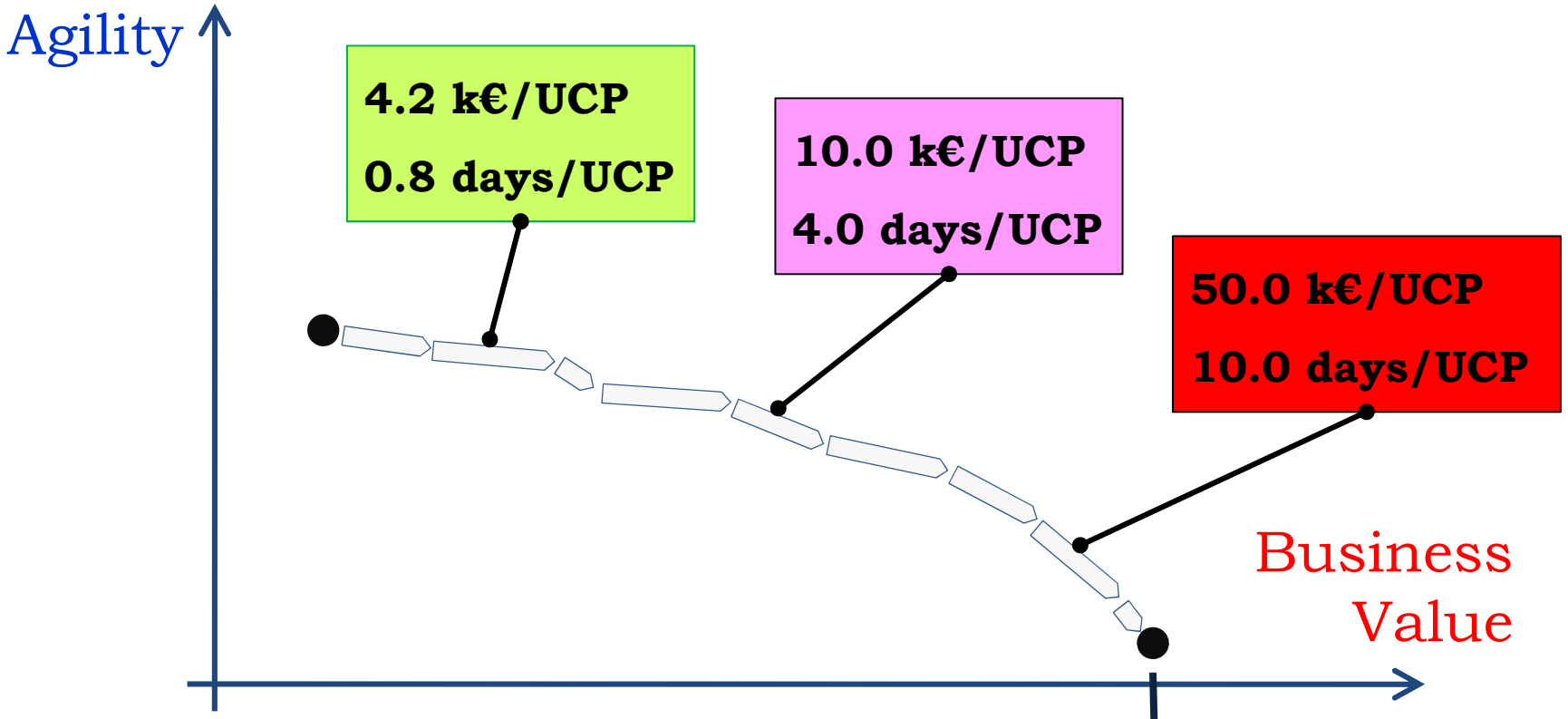
The project has transformed the system into a new system with *higher* business value, *lower* agility and sligthly *weaker* resilience

Industrial Software: Business Value, Agility & Resilience



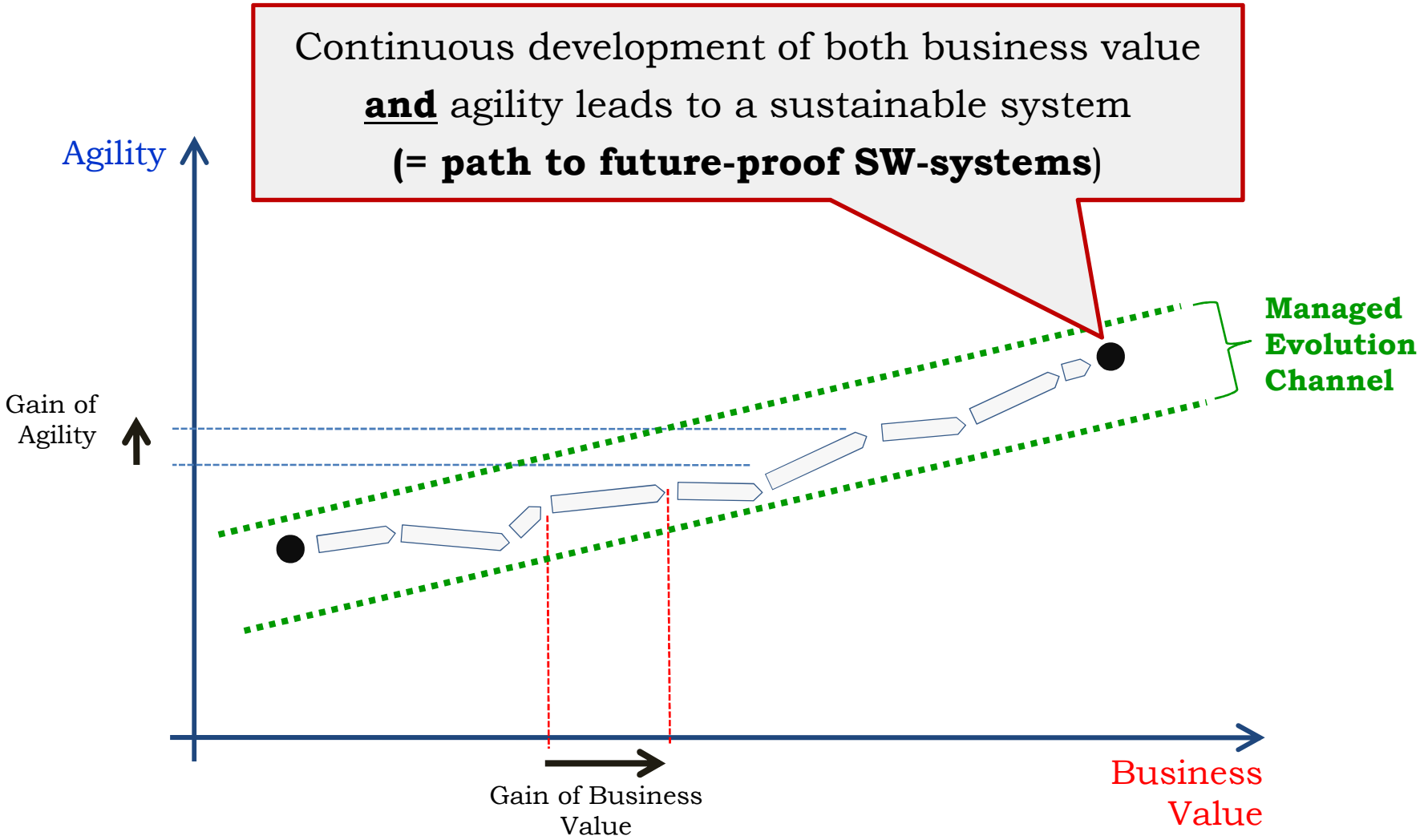
Evolution Case 1: Opportunistic Approach



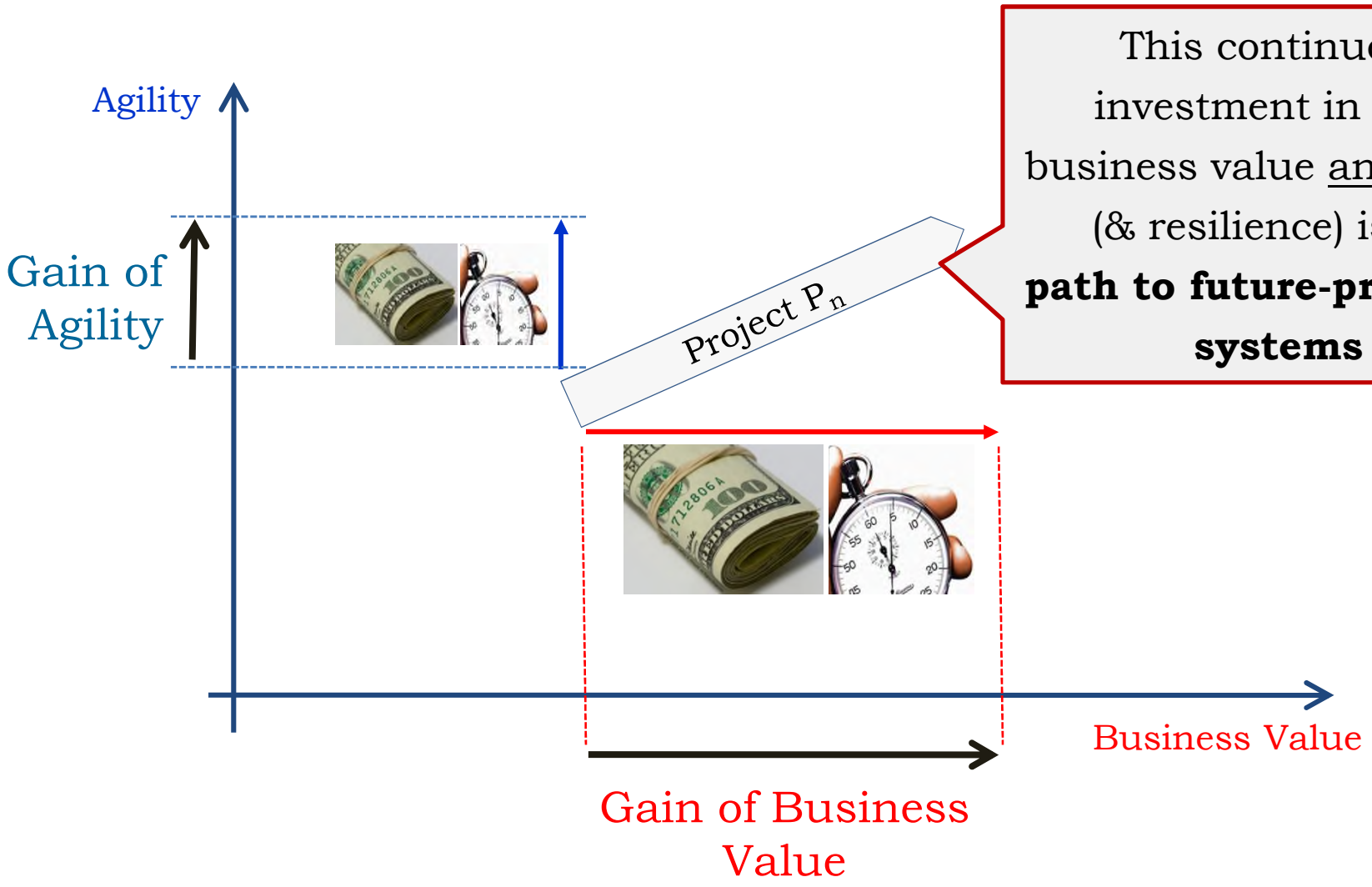


Trajectory Case 1:
Opportunistic Evolution

Evolution Case 2: Managed Evolution



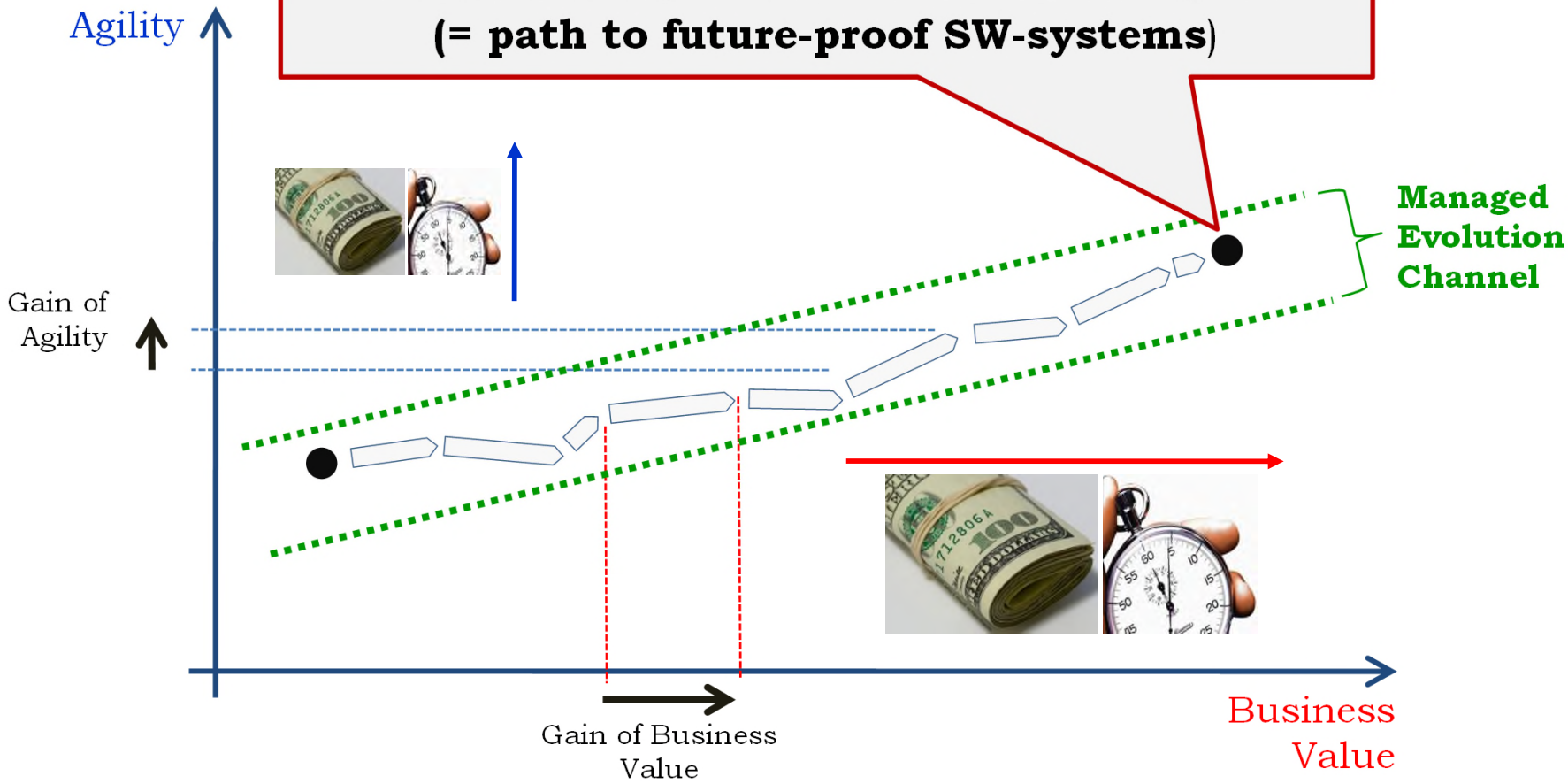
Evolution Case 2: Managed Evolution



This continuous investment in both business value and agility (& resilience) is the **path to future-proof SW-systems**

Evolution Case 2: Managed Evolution

Only continuous development of both business value **and** agility leads to a sustainable system
(= path to future-proof SW-systems)



Line of Thought

http://www.wissenswertes.biz

Context: A world of software

⇩

Vision: Future-Proof Software-Systems

⇩

Strategy: Managed Evolution

⇩

Implementation: Industrial Software

⇩

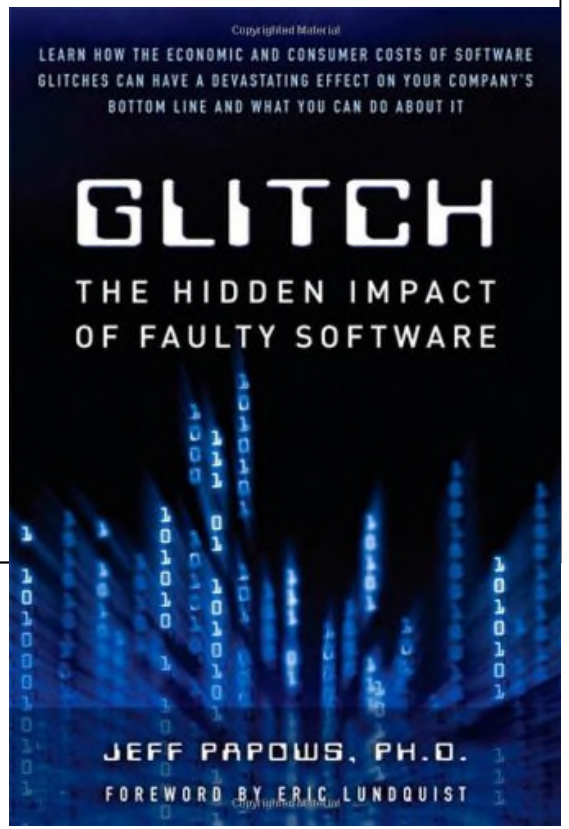
Foundation: Architecture Principles

⇩

Aging: Architecture Erosion & Technical Debt

Software – especially *faulty* software – has an enormous impact on people and society:

- Functionality in all areas of life and work
- Tremendous business opportunities
- Risks in safety-critical systems
- Legal & regulatory consequences
- Product liability
- etc.



Jeffrey Papows, 2010
ISBN 978-0-132-16063-6

http://www.motivationalmemo.com

Pflicht-lektüre!

Software must be specified, developed, maintained and evolved according to *industrial methods and processes*

⇒ **Industrial Software**

Industrial Software:

7 laws

1. Clear business – IT alignment
2. Architecture-centric
3. Strategy-controlled (Evolution, Reuse, Product Lines, …)
4. Unambiguous specification of requirements (functionality and non-functional properties)
5. Powerful, accepted, enforced development process
6. Modelled (~ formal)
7. Metrics √

Industrial Software: Business Value, Agility & Resilience

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

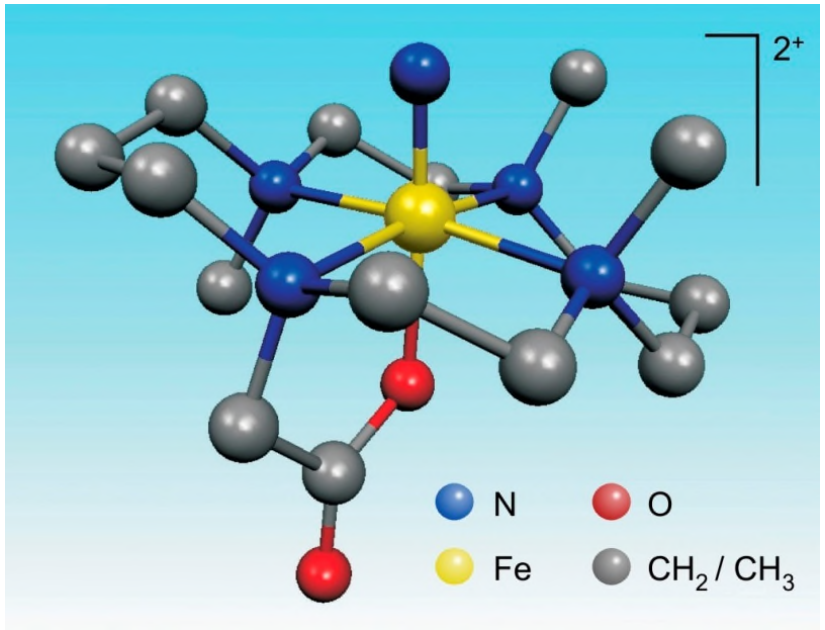
http://www.0lll.com/architecture-exhibitions/?gal=24

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?

Why is structure important?

What determines structure?



http://www.news.wisc.edu/newsphotos/ironVI.html

The tower of babel by Pieter Bruegel the Elder (1563)

Structure is the basis for ordered, managed evolution

Architecture !

Architecture Definition:

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

[IEEE-SA Standards Board, Standard IEEE 1471-2000]

www.rfc123.com

Architecture Importance:

The architecture of a system determines:

- The success of coping with complexity
- The effort for extensions
- Many non-functional quality properties (Safety, security, …)
- The probability of system development failures
- The viability of compromises
- Early (and dependable) stakeholder commitments

Line of Thought

http://www.wissenswertes.biz

Context: A world of software

⇩

Vision: Future-Proof Software-Systems

⇩

Strategy: Managed Evolution

⇩

Implementation: Industrial Software

⇩

Foundation: Architecture Principles

⇩

Aging: Architecture Erosion & Technical Debt

1843 – today:

Software engineering matured from a „black art“
(mastered by few, experienced, talented individuals only)
to an accepted engineering discipline

www.123rf.com/photo_9353072

http://wordpress.com

Today - future:

Software engineering progresses to a mature, industrial production – based on formal, proven and validated principles and processes

Architecture Principles

Architecture Principles:

Fundamental insights – formulated as rules – how a good software-system should be built

Architecture Principles:

→ highly valuable architecture knowledge in proven & easily accessible form

→ teachable & enforcable

→ the foundation for the design, implementation and evolution of future-proof software systems

Example: Architecture Principle A2

A2

Architecture Principle A2:

Partitioning, Encapsulation & Coupling

1. Partition the functionality and data into encapsulation units according to their coherence and cohesion (thus minimizing dependencies)
2. Isolate the encapsulation units by strictly hiding any internal details. Allow access to functionality and data only through stable, well specified interfaces governed by contracts
3. Minimize the impact of dependencies between the encapsulation units by using adequate coupling mechanisms

Justification: These 3 principles minimize the number and the impact of dependencies. The resulting system therefore offers the least resistance to change, because any change affects the smallest possible number of system elements. A low resistance to change corresponds to high agility.

Line of Thought

Context: A world of software

⇩

Vision: Future-Proof Software-Systems

⇩

Strategy: Managed Evolution

⇩

Implementation: Industrial Software

⇩

Foundation: Architecture Principles

⇩

Aging: Architecture Erosion & Technical Debt

http://www.wissenswertes.biz

Architecture Erosion

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

Architecture Erosion:

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

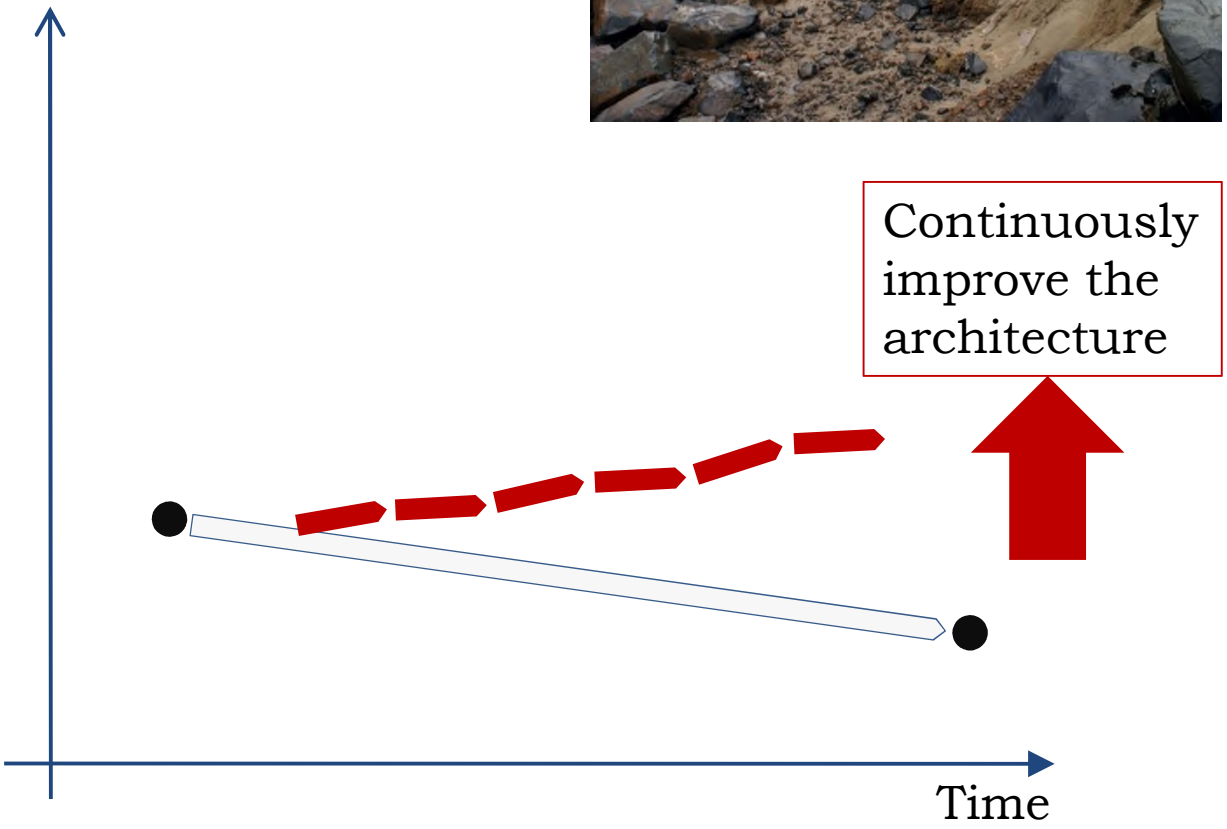
- SW Paradigm changes (e.g. SOA)
- New laws & regulations
- New standards (e.g. interoperability)
- Accumulation of mistakes + shortcuts (e.g. breaking partitions)
- Sloppy system extensions
- Introduction of new architecture principles

… and some more

Architecture Erosion

Degeneration of Quality Properties:

- Business Value
- Agility
- Resilience
- Std Conformance
- Reusability
- etc.



Technical Debt

Technical Debt:

Technical Debt is the accumulation of violations of best practices in a software system:

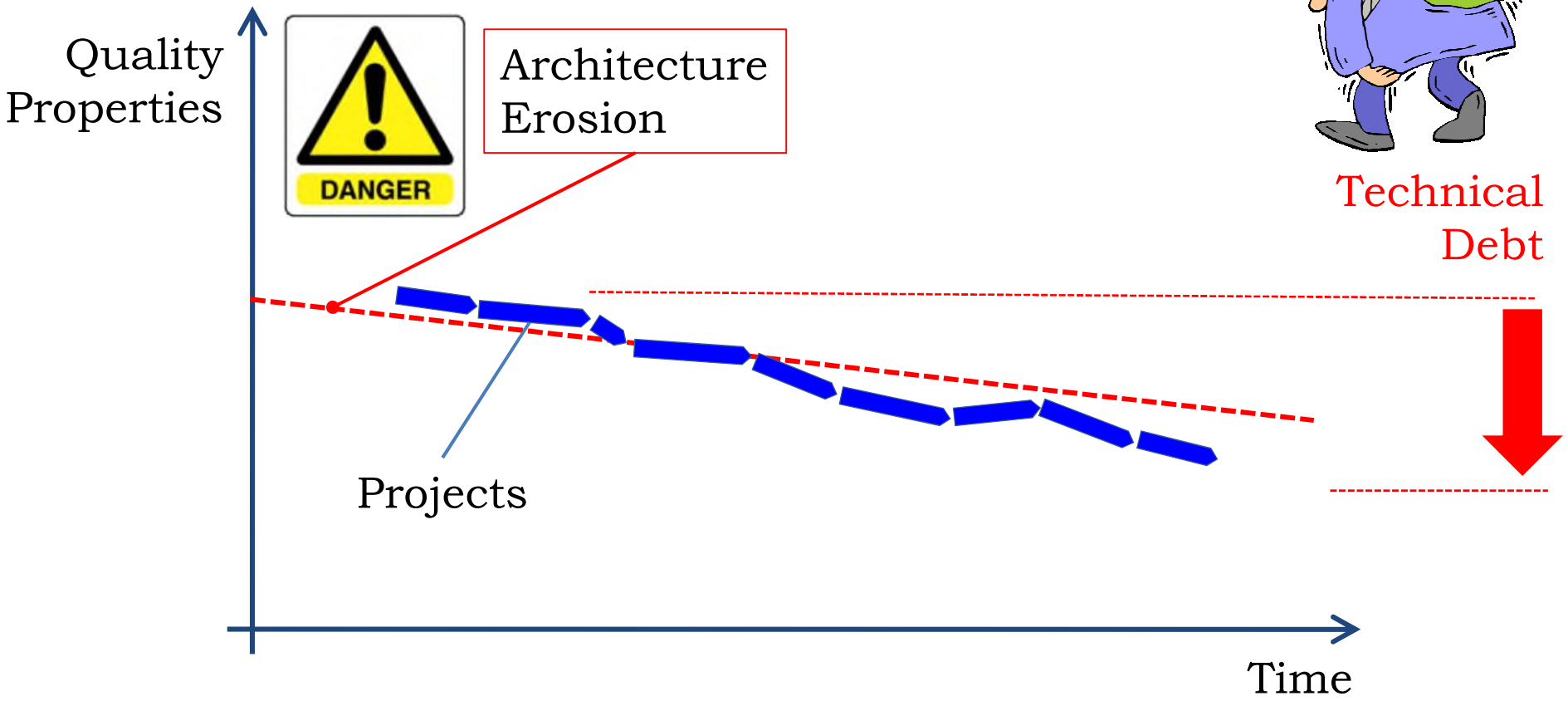
- Architecture Erosion
- Disruptive technology
- Dead code (missed explementations)
- Redundancy (code and data)
- Progress in software-engineering (e.g. programming languages)
- Careless or skipped upgrades
- Management mistakes (time-to-market over SW-engineering)

… and some more

Technical Debt

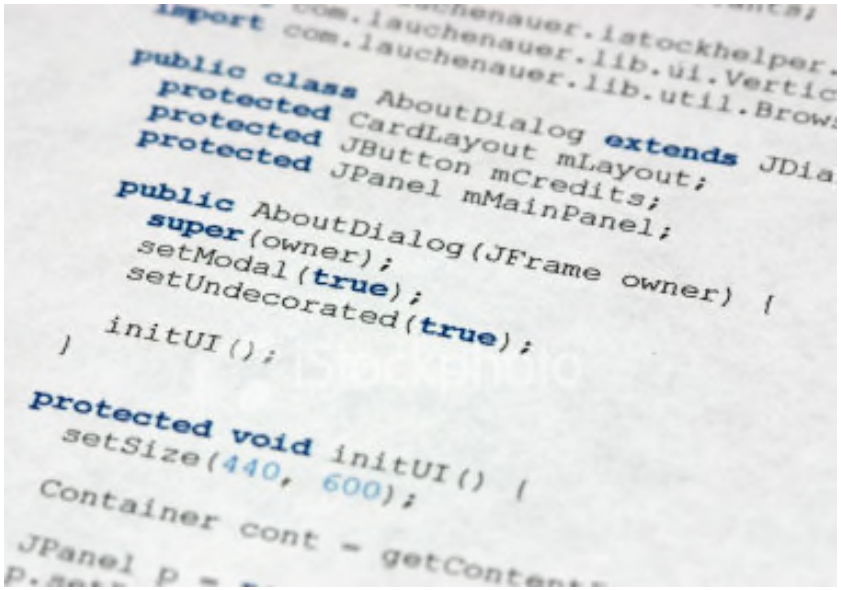
Technical Debt

Technical Debt



Technical Debt

http://dandev91.wordpress.com/



Cost of one source line of embedded systems code:

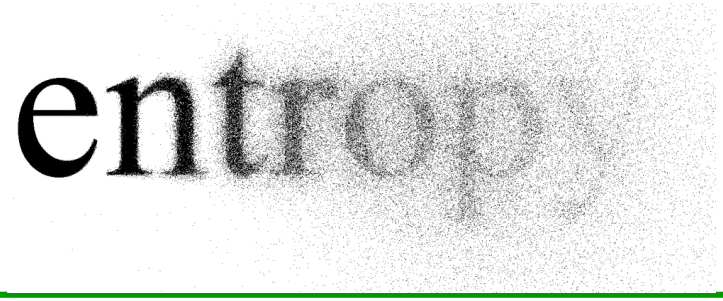
€ 15.00 ... € 40.00

Average Technical Debt in each source line of embedded systems code:

€ 2.70

[Deloitte Consulting LLP: Tech Trends 2014]

Conclusions



http://larvalsubjects.wordpress.com

The force of entropy means that *disorder* is the only thing that happens automatically and by itself.

If you want to create a completely *ad-hoc IT architecture*, you do not have to lift a finger.

It will happen *automatically* as a result of day-to-day IT activity.

Richard Hubert: **Convergent Architecture**, 2002. ISBN 978-0-471-10560-2

Conclusions

Architecture Principles

Agility
Resilience

Gain of Agility & Resilience

Managed Evolution Channel

Project modifying the system

Gain of Business Value

Business Value

Conclusions

Architecture Principles

Agility
Resilience

Gain of Agility & Resilience

Managed Evolution Channel

Project modifying the system

Gain of Business Value

Business Value

References: **Business Value** (1/2)

Reference
Stephan Murer, Bruno Bonati, Frank J. Furrer: **Managed Evolution – *A Strategy for Very Large Information Systems*** Springer-Verlag, Berlin Heidelberg, 2011, ISBN 978-3-642-01632-5
Michael A. Cusumano: **Staying Power – *Six Enduring Principles for Managing Strategy & Innovation in an Uncertain World*** Oxford University Press, New York, USA, 2010. ISBN 978-0-19-921896-7
Olivier L. de Weck, Daniel Roos, Christopher L. Magee: **Engineering Systems – *Meeting Human Needs in a Complex Technological World*** MIT Press, Cambridge, USA, 2011. ISBN 978-0-262-01670-4
George Fairbanks: **Just Enough Software Architecture – *A Risk-Driven Approach*** Marshall & Brainerd, Boulder CO, USA, 2010. ISBN 978-0-9846181-0-1
Mario Godinez, Eberhardt Hechler, Klaus Koening, Steve Lockwood, Martin Oberhofer, Michael Schroeck: **The Art of Enterprise Information Architecture: A Systems-Based Approach for Unlocking Business Insight** Addison Wesley Publishing Inc., USA, 2010. ISBN 978-0-13-703571-7
Jeffrey Papows: **Glitch – *The Hidden Impact of Faulty Software*** Prentice Hall Inc., USA, 2010. ISBN 978-0-132-16063-6

References: **Business Value** (2/2)

Reference
Theo J.W. Renkema **:** **The IT Value Quest – *How to Capture the Business Value of IT-Based Infrastructure*** John Wiley & Sons, Inc., Chichester, UK, 2000. ISBN 978-0-471-98817-0
Roger Gutbrod, Christian Wiele: **The Software Dilemma – *Balancing Creativity and Control on the Path to Sustainable Software*** Springer-Verlag, Heidelberg, 2012. ISBN 978-3-642-27235-6
Luke Hohmann: **Beyond Software Architecture – *Creating and Sustaining Winning Solutions*** Pearson Education, Addison-Wesley, Boston, USA, 2003. ISBN 978-0-201-77594-8
Gerrit Muller: **Systems Architecting – *A Business Perspective*** CRC Press (Taylor & Francis), Boca Raton, FL, USA, 2012. ISBN 978-1-4398-4762-6
Dan Remenyi, Arthur Money, Michael Sherwood-Smith: **The effective measurement and management of IT costs and benefits** Butterworth-Heinemann, Oxford UK, 2nd edition, 2000. ISBN 0-7506-4420-6
Jeanne W. Ross, Peter Weill, David C. Robertson: **Enterprise Architecture as Strategy – *Creating a Foundation for Business Execution*** Harvard Business Review Press, USA, 2006. ISBN 978-1-5913-9839-4

References: **Agility**

Reference
Barry Boehm, Richard Turner: **Balancing Agility and Discipline – *A Guide for the Perplexed*** Pearson Education, Addison-Wesley, Boston, USA, 2004. ISBN 978-0-321-18612-5
Richard de Neufville, Stefan Scholtes: **Flexibility in Engineering Design** MIT Press, Cambridge, USA, 2011. ISBN 978-0-262-01623-0
James Coplien, Gertrud Bjornvig: **Lean Architecture for Agile Software Development** John Wiley & Sons, Inc., Chicester UK, 2010. ISBN 978-0-470-68420-7
Fred A. Cummins: **Building the Agile Enterprise – *with SOA, BPM and MBM*** Morgan Kaufmann (Elsevier), Amsterdam, 2009. ISBN 978-0-12-374445-6
Jez Humble, David Farley: **Continuous Delivery – *Reliable Software Releases through Build, Test, and Deployment Automation*** Pearson Education (Addision-Wesley), Boston, USA, 2011. ISBN 978-0-321-60191-9
Bertrand Meyer: **Agile! – *The Good, the Hype and the Ugly*** Springer Verlag, Berlin und Heidelberg, 2014. ISBN 978-3-3190-5154-3
Dean Leffingwell: **Scaling Software Agility – *Best Practices for Large Enterprises*** Pearson Education (Addison-Wesley), Boston, USA, 2007. ISBN 978-0-321-45819-3

References: **Resilience** (1/2)

Reference
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
Erik Hollnagel, Jean Pariès, David D. Woods, John Wreathall (Editors): **Resilience Engineering in Practice – *A Guidebook*** Ashgate Publishing Ltd., Farnham, UK, 2011. ISBN 978-1-4724-2074-9
Erik Hollnagel : **FRAM: The Functional Resonance Analysis Method – *Modelling Complex Socio-Technical Systems*** Ashgate Publishing Ltd., Farnham, UK, 2012. ISBN 978-1-4094-4551-7
Michael Howard, David LeBlanc: **Writing Secure Code – *Practical Strategies and Techniques for Secure Application Coding in a Networked World*** Microsoft Press, Redmond, USA, 2003. ISBN 0-7356-1722-8
Clifford J. Berg: **High-Assurance Design – *Architecting Secure and Reliable Enterprise Applications*** Addison-Wesley, N.J., USA, 2006. ISBN 0-321-37577-7
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

References: **Resilience** (2/2)

Reference
C. Warren Axelrod: **Engineering Safe and Secure Software Systems** Artech House, Norwood, USA, 2013. ISBN 978-1-60807-472-3
Stuart Anderson, Massimo Felice: **Emerging Technological Risk – Underpinning the Risk of Technology Innovation** Springer-Verlag, London, UK, 2012. ISBN 978-1-4471-2142-8
Nancy G. Leveson: **Engineering a Safer World – *Systems Thinking applied to Safety*** MIT Press, Cambridge MA, USA, 2011. ISBN 978-0-262-01662-9
Mark S. Merkow, Lakshmikanth Raghavan: **Secure and Resilient Software Development** CRC Press, Taylor & Francis Group, Boca Raton, USA, 2010. ISBN 978-1-4398-2696-6
Kim Zetter: **Countdown to Zero Day – *Stuxnet and the Launch of the World's First Digital Weapon*** Crown Publishing, 2014. ISBN 978-0-7704-3617-9
Drew Chapman: **The Pattern of Fear – *Paranoia is all in the Mind*** Penguin Books, London, UK, 2013. ISBN 978-1-405-91287-7

References: **System & Software Architecture** (1/2)

Reference

Eric Evans:
Domain-Driven Design – *Tackling Complexity in the Heart of Software*
Pearson Education, Addison-Wesley, Boston, USA, 2004. 7th printing 2006.
ISBN 978-0-321-12521-5

Richard de Neufville, Stefan Scholtes:
Flexibility in Engineering Design
MIT Press, Cambridge, USA, 2011. ISBN 978-0-262-01623-0

Frederik Ahlemann, Eric Stettiner, Marcus Messerschmidt, Christine Legner (Editors):
Strategic Enterprise Architecture Management – *Challenges, Best Practices, and Future Developments*
Springer-Verlag, Berlin Heidelberg, 2012. ISBN 978-3-642-24222-9

Eric J. Braude, Michael E. Bernstein:
Software Engineering – *Modern Approaches*
John Wiley & Sons, Inc., New York, USA, 2nd edition, 2011. ISBN 978-0-471-69208-9

Ian Gorton
Essential Software Architecture
Springer-Verlag, Berlin Heidelberg, 2006. ISBN 978-3-540-28713-1

David Greefhorst, Erik Proper:
Architecture Principles – *The Cornerstones of Enterprise Architecture*
Springer Verlag, Heidelberg, Berlin, 2011. ISBN 978-3-642-20278-0

References: **System & Software Architecture** (2/2)

Reference
Chris Sterling: **Managing Software Debt – *Building for Inevitable Change*** Pearson Education, Addison-Wesley, N.J., USA, 2011. ISBN 978-0-321-55413-0
Alexander Kossiakoff, William N. Sweet, Samuel J. Seymour, Steven M. Biemer: **Systems Engineering – *Principles and Practice*** John Wiley & Sons, Inc., Hoboken, N.J., USA, 2nd edition 2001. ISBN 978-0-470-40548-2
Olivier L. de Weck, Daniel Roos, Christopher L. Magee: **Engineering Systems – *Meeting Human Needs in a Complex Technological World*** MIT Press, Cambridge, USA, 2011. ISBN 978-0-262-01670-4
Eric J. Braude, Michael E. Bernstein: **Software Engineering – *Modern Approaches*** John Wiley & Sons, Inc., New York, USA, 2nd edition, 2011. ISBN 978-0-471-69208-9
Deloitte Consulting LLP: ***How to Reverse Your Technical Debt*** Tech Trends 2014: Inspiring Disruption, June 18, 2014 Downloadable from: http://www.castsoftware.com/castresources/materials/recorded/061814/How_To_Reverse_Your_Technical_Debt.pdf [last accessed: 16.8.2014]

https://yourdailygerman.wordpress.com

Questions please ?

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