Future-Proof Software-Systems

(Zukunftsfähige Software-Systeme)

Frank J. Furrer

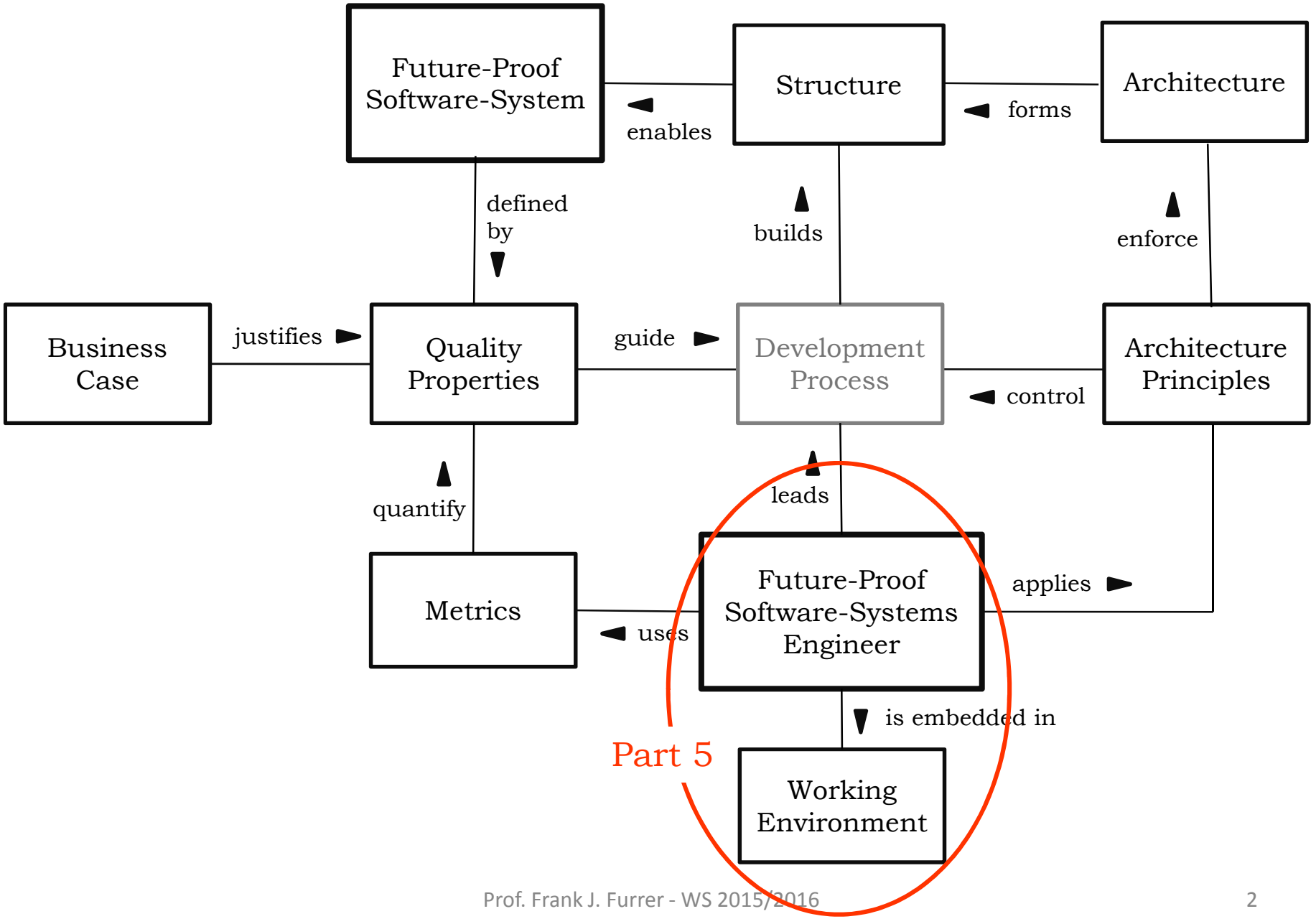
Dr. sc. techn. ETH-Zürich

TU Dresden WS 2015/2016

Part 5: The Future-Proof Software-System Engineer

V0.2 / 25.01.2015

Future-Proof Software-Systems: Lecture Structure



Contents (Part 5)

- The **Responsibility** and **Role** of the Future-Proof Software-Systems Engineer
- The **Skills** and Personality of the Future-Proof Software-Systems Engineer
- The **Working Context** of the successfull Future-Proof Software-Systems Engineer

Context

The following guidelines for a *future-proof software-systems engineer/architect* have been developed and proven in:

- Companies developing significant amounts of software (for their own use or for customers) → In-house SW-development
- Companies which strongly depend on their (long-lived) software
- Fairly large companies (SW development staff > 500 people, organization structure)

... but the material is important for your personal skill set !

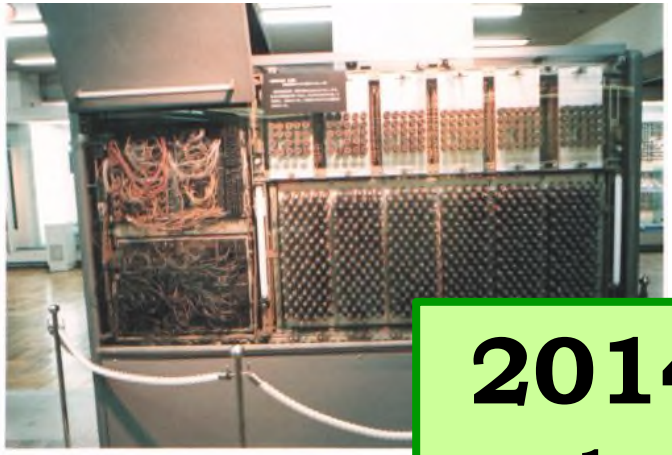
Repetition: Importance and impact of software

… etc.

```
public HelloWorld() {

 System.out.println("Hello World");
 /*  the native method name is 2000 characters */
 HelloWorldHelloWorld...();
}

public static void main(String arg[]) {
 HelloWorld hw = new HelloWorld();
}

//  the native method name is 2000 characters //
public native void HelloWorldHelloWorld...(); }
```

Future-Proof Software-Systems: Software Importance

UNIVAC 120 (1953)



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

Decimal storage: 120 d
„Program“: Wired Pane

CRAY Titan (2013)

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

ytes RAM
ytes Disk
Petaflops

2014: ≈ 80% of the technical innovation in cars is due to software

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

Software is one of the most important key success factors for today's and tomorrow's products and services

http://www.change-management.com

Software determines (to a large extent):

- ✓ Functionality
- ✓ Quality Properties, such as safety, security, availability, integrity, performance etc.
- ✓ Competitiveness
- ✓ Revenue generation
- ✓ Innovation capacity
- ✓ Intellectual Property Rights (→ IPR protection)

Future-Proof Software-Systems: Software Importance

What is the **foundation** of good software ?

⇒ The underlying architecture !

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?

What is a good software-architecture ?

A good software architecture

generates a *structure* that enables

the *management* of complexity, change and uncertainty

with the least effort, with acceptable risk

and with specified quality properties

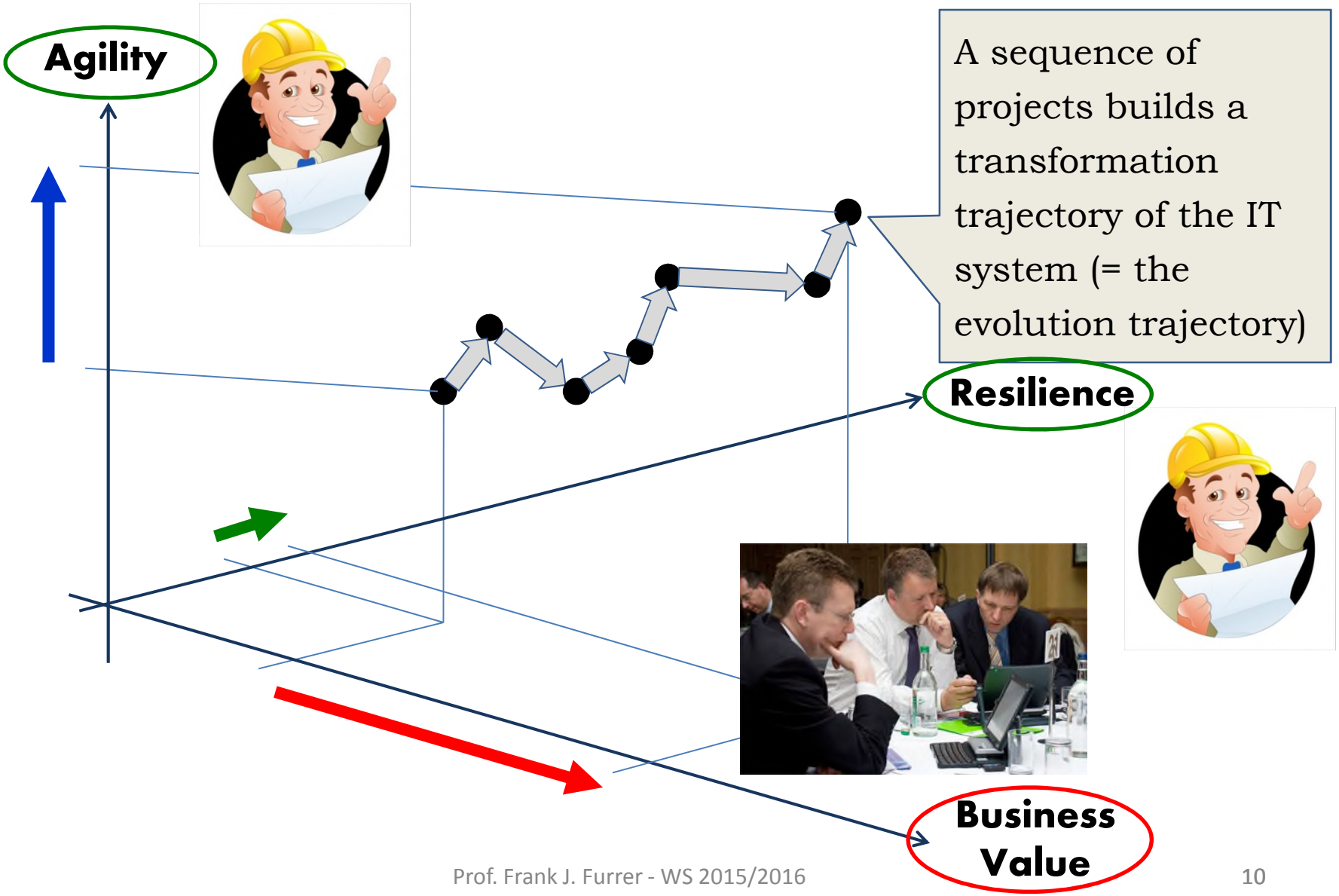
Agility **Resilience**

How do we develop and maintain good software-architecture ?

By consistently applying and enforcing proven

architecture principles **Foundation**

Future-Proof Software-Systems: **Managed Evolution Trajectory**

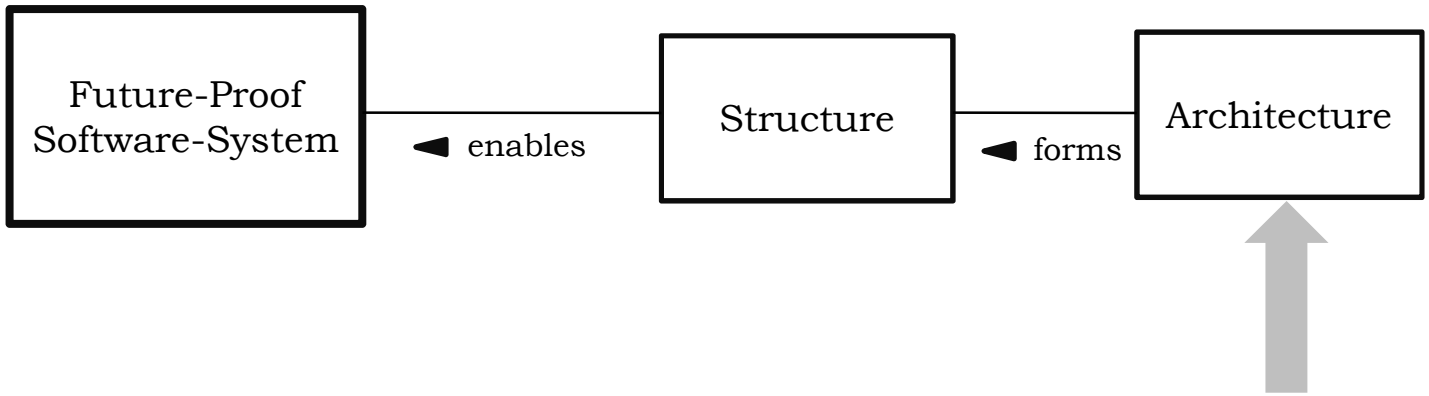


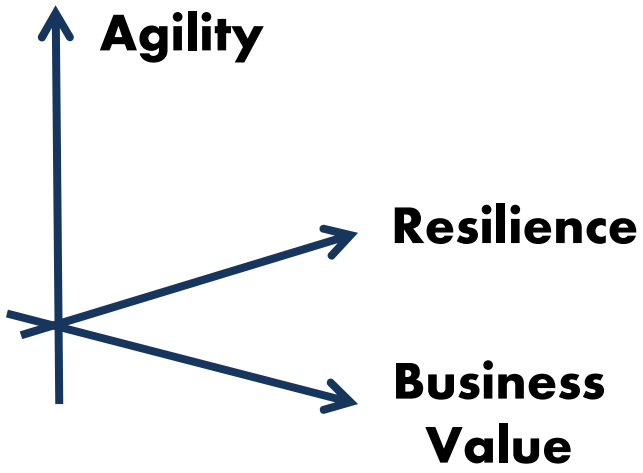
Who is responsible for a good software-architecture ?

⇐ **YOU !**

The future-proof software-system architect !

Future-Proof Software-Systems: **Software Architecture**

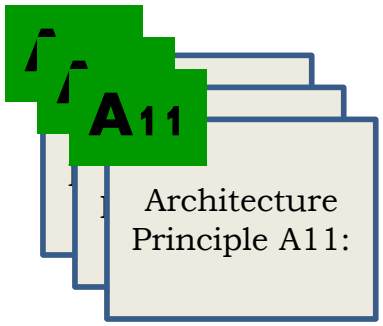




The **software-architecture** determines to a large extent the **agility**, **resilience** (and other quality properties)

⇒ Key role of the architect!

The Way to Future-Proof Software-Systems: Basic needs



Principles and guidelines for future-proof software-systems architecture

Knowledgeable and respected future-proof software-systems engineers

http://www.amisinsurance.com

Dedicated and committed management

Lecture: Part 5

Working **Context**

www.123rf.com

Stakeholders

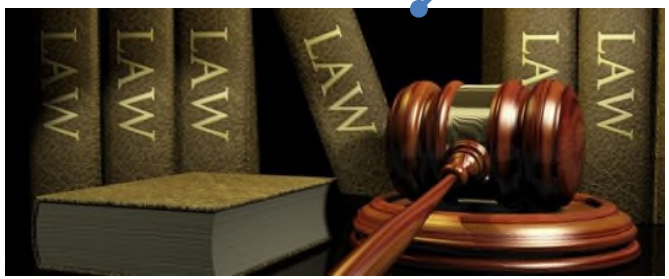
http://www.telco2.net/blog

Business People

Project P_n

Project P_n

Regulators



http://www.rifreedom.org

Colleagues

Future-Proof Software-Systems: Context

Knowledge → Emotions → Enforcement

www.clipartpanda.com

Architecture Principles

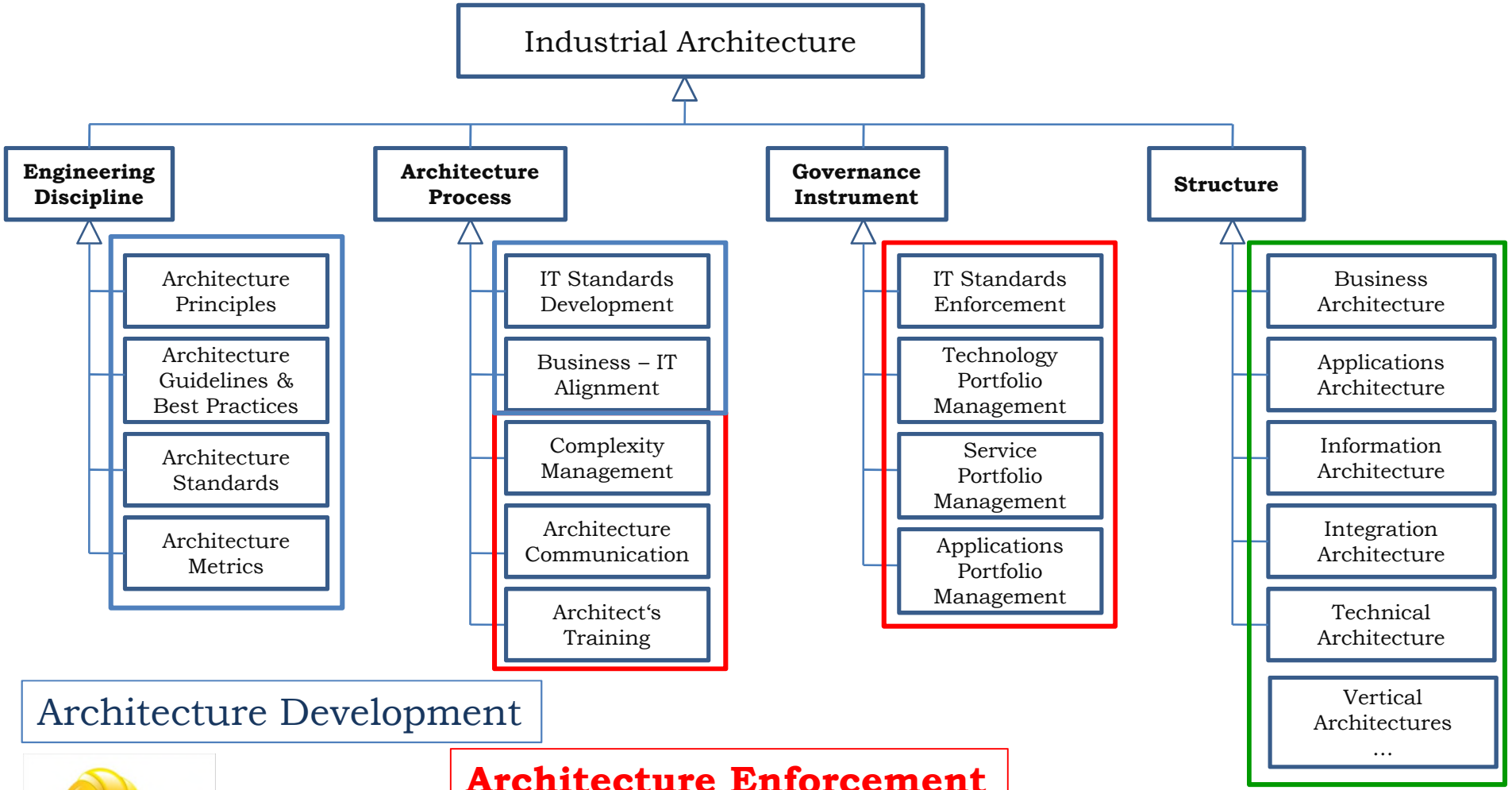
www.bestclipartblog.com

www.clipartlord.com

Architecture Tools

www.clipartpanda.com

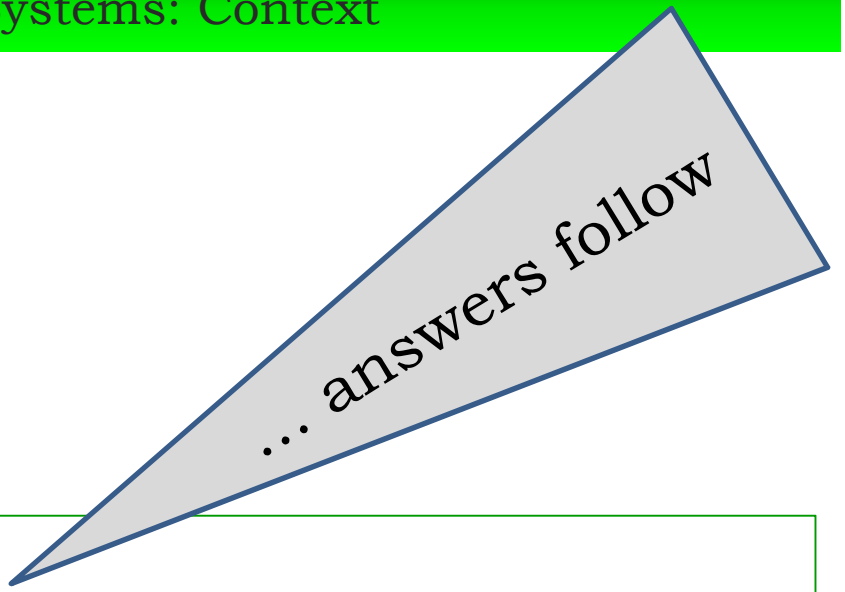
Future-Proof Software-Systems: Context



Architecture Development

Architecture Enforcement

Result: Structure („Architecture“)



How do you become

a valuable and successful

future-proof software-systems engineer ?

References:

References	
Murer11	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
Fairbanks10	George Fairbanks: **Just Enough Software Architecture – *A Risk-Driven Approach*** Marshall & Brainerd, Boulder CO, USA, 2010. ISBN 978-0-9846181-0-1
Bass13	Len Bass, Paul Clements, Rick Kazman: **Software Architecture in Practice** SEI-Series (Pearson Education), Addison-Wesley, N.J., USA, 3rd edition, 2013. ISBN 978-0-321-81573-6
Albin03	Stephen T. Albin: **The Art of Software Architecture** Wiley Publishing Inc., Indiana, USA, 2003. ISBN 978-0-8493-0440-7
Braude11	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
Evans06	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
Gorton06	Ian Gorton **Essential Software Architecture** Springer-Verlag, Berlin Heidelberg, 2006. ISBN 978-3-540-28713-1
Greefhorst11	David Greefhorst, Erik Proper: **Architecture Principles – *The Cornerstones of Enterprise Architecture*** Springer Verlag, Heidelberg, Berlin, 2011. ISBN 978-3-642-20278-0
Spinellis09	Diomidis Spinellis, Georgios Gousios (Editors): **Beautiful Architecture – *Leading Thinkers Reveal the Hidden Beauty in Software Design*** O'Reilly Media, USA, 2009. ISBN 978-0-596-51798-4

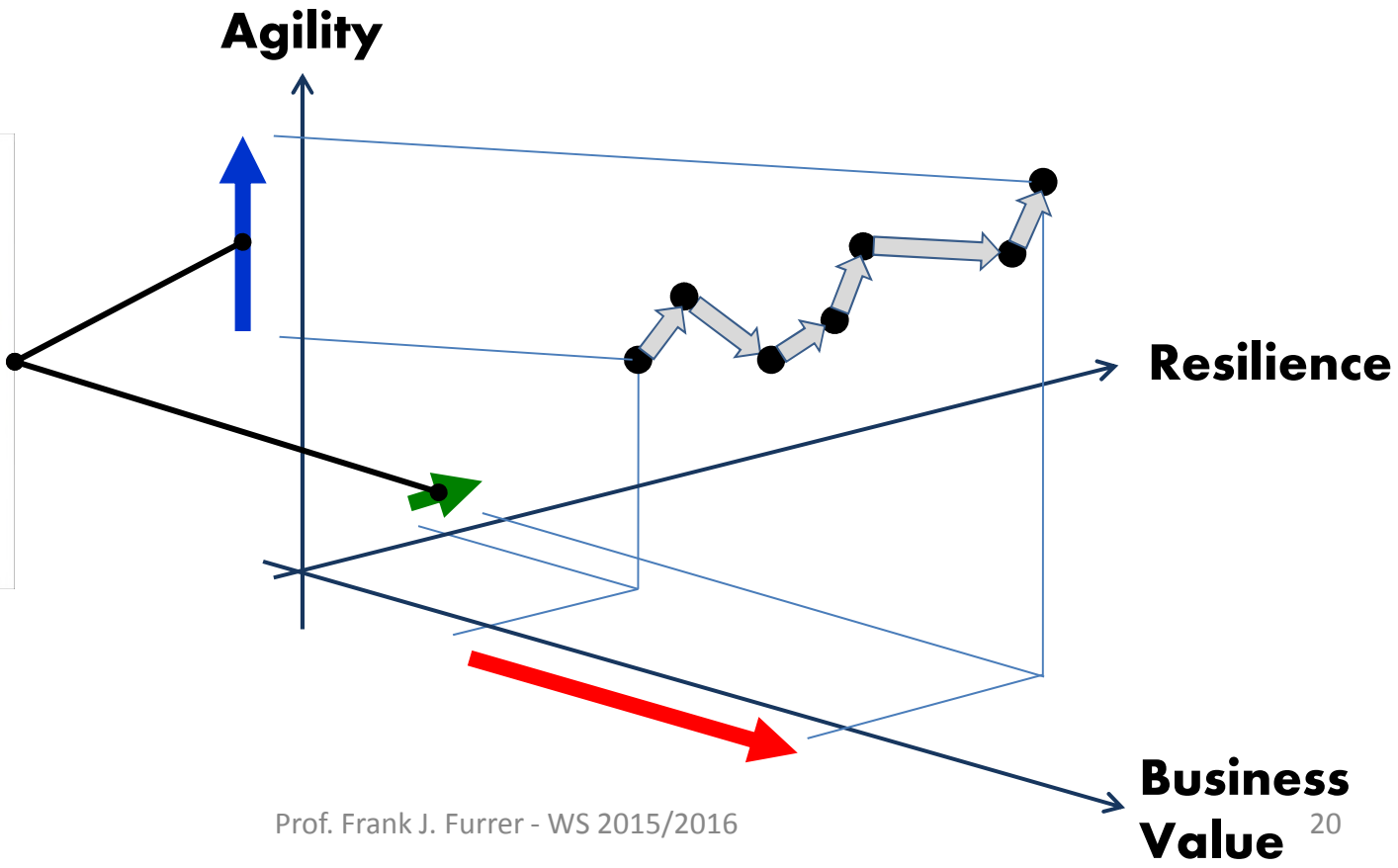
Future-Proof Software-Systems:

The Responsibility and Role of the Future-Proof Software-Systems Engineer

Responsibility

Responsibility:

Develop, maintain and enforce an *adequate IT-architecture* to guarantee the required *quality properties* of the system, especially the continuous increase in *agility* and *resilience*



Conflicting Interests:

Architecture wants:
- Good fit into the existing system
- Refactoring to improve architectural quality
- Limit growth in complexity
- Use proven technologies

Project P_n

Project P_n

www.123rf.com

http://www.telco2.net/blog

Project Team wants:
- Budget & Time compliance
- „Shortest“ path to solution
- Minimum external intervention
- Least constraints

Business wants:
- Short time to market
- Low cost
- Only essential functionality
- Newest technology

Conflicting interests:

Responsibility:

Develop, maintain and enforce an *adequate IT-architecture* to guarantee the required *quality properties* of the system, especially the continuous increase in *agility* and *resilience*

- Architecture Principles
- Cooperative Teams
- Good Processes
- Strong Governance

Build it !

Future-Proof Software-System

Future-Proof Software-Systems:

The Responsibility and Role of the Future-Proof Software-Systems Engineer

Role

Responsibility:

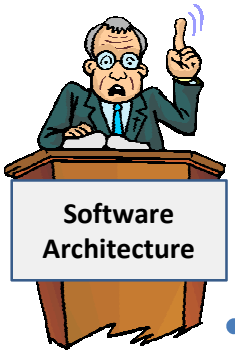
Develop, maintain and enforce an *adequate IT-architecture*

How ?
Challenges ?
Help ?
Risks ?

Decisive:
Knowledge, personal & social skills of the architect

Future-Proof Software-Systems Engineer: **Role** („4-in-1“)

http://www.bible.ca



Missionary:

Untiringly preach the value and necessity of good IT-architecture

http://www.rlfreedom.org

Lawmaker:

Consistently develop and maintain an adequate, powerful set of IT-architecture principles

❶

❷

❸

❹

Consultant:

Be a competent, useful and fair consulting partner to all projects

Enforcer:

Insist on the consequent, complete and correct implementation of the IT-architecture principles and standards

Future-Proof Software-Systems Engineer: „5th Role“

„Magician“

http://de.123rf.com

Architecture Topic Map

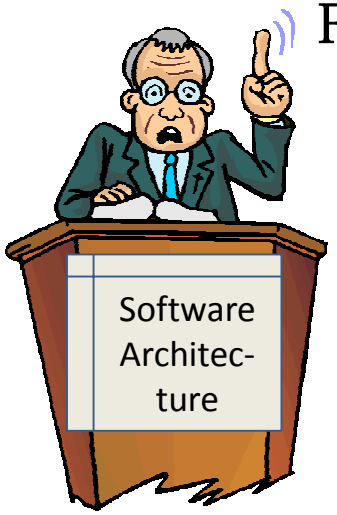
Functionality

Architecture-Quality:
- **Agility**
- **Resilience**
- Resources
- Operation cost
- Performance
- …

Architecture-Greatness:
- **Simplicity**
- **Elegance**

Future-Proof Software-Systems

Engineer: **Position**



http://www.bible.ca

Missionary

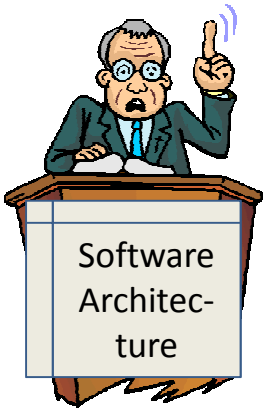
http://www.rjfreedom.org

Lawmaker

Consultant

Enforcer

Future-Proof Software-Systems Engineer: **Position**



Missionary

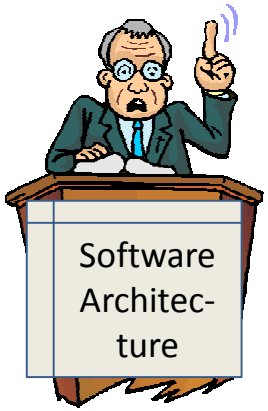
Lawmaker

Consultant

Enforcer

A true, believable, reliable and consistent positioning is the key to an IT architect's success

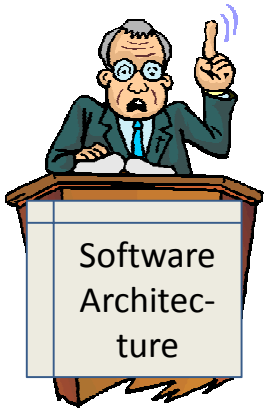
Future-Proof Software-Systems: Responsibility and Role



How can you successfully play all these roles?

https://essayhelp.org/

Future-Proof Software-Systems: Responsibility and Role



Missionary

Lawmaker
(Architecture Principles)

Consultant

Enforcer

Knowledge,
Experience,
Authority,
Communi-
cation skills

Knowledge,
Precision,
Farsighted-
ness,
Restriction

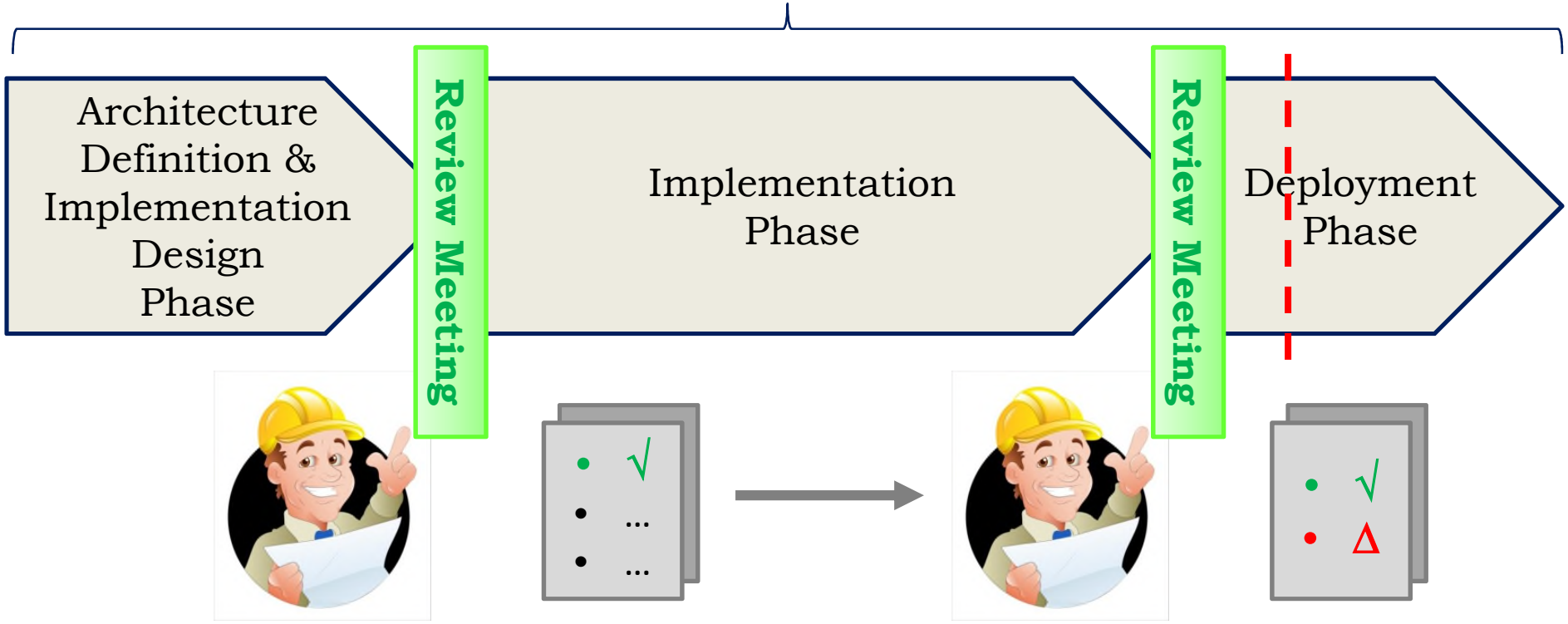
Fairness,
Engagement,
Teamspirit,
Committment

Transparency,
Fairness,
Reliability,
Consequence

Architecture Principles and Standards **Enforcement**:

Architecture Principles and Standards **Enforcement**

Project Team
(Colleagues)



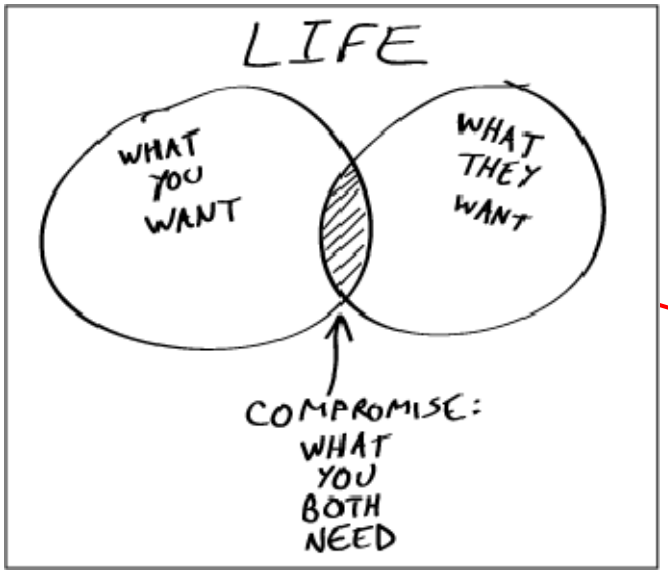
Architecture Principles and Standards **Enforcement**

consult, convince, negotiate

force, constrain, repress

Strategy of a successful IT-architect

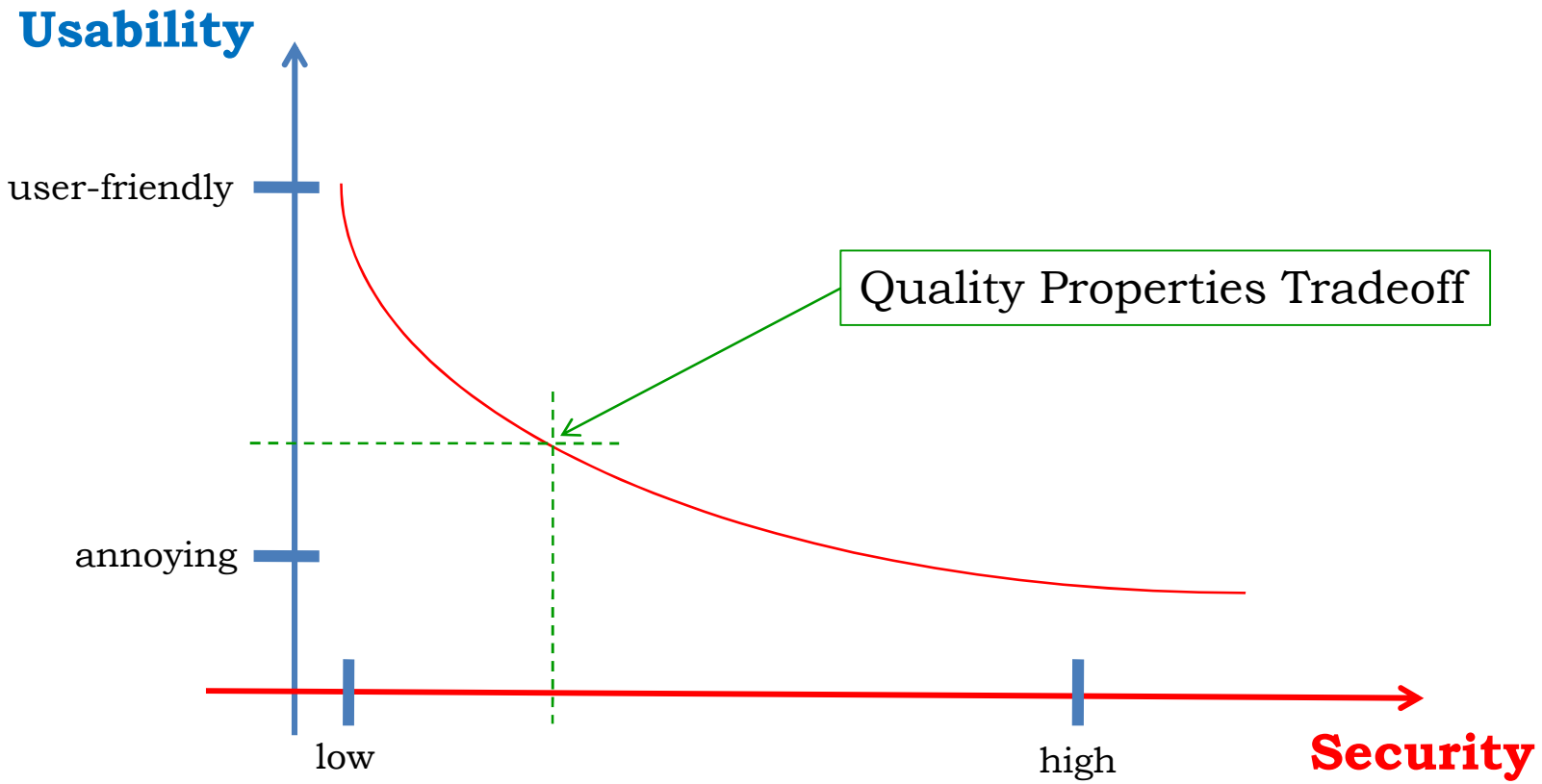
compromise



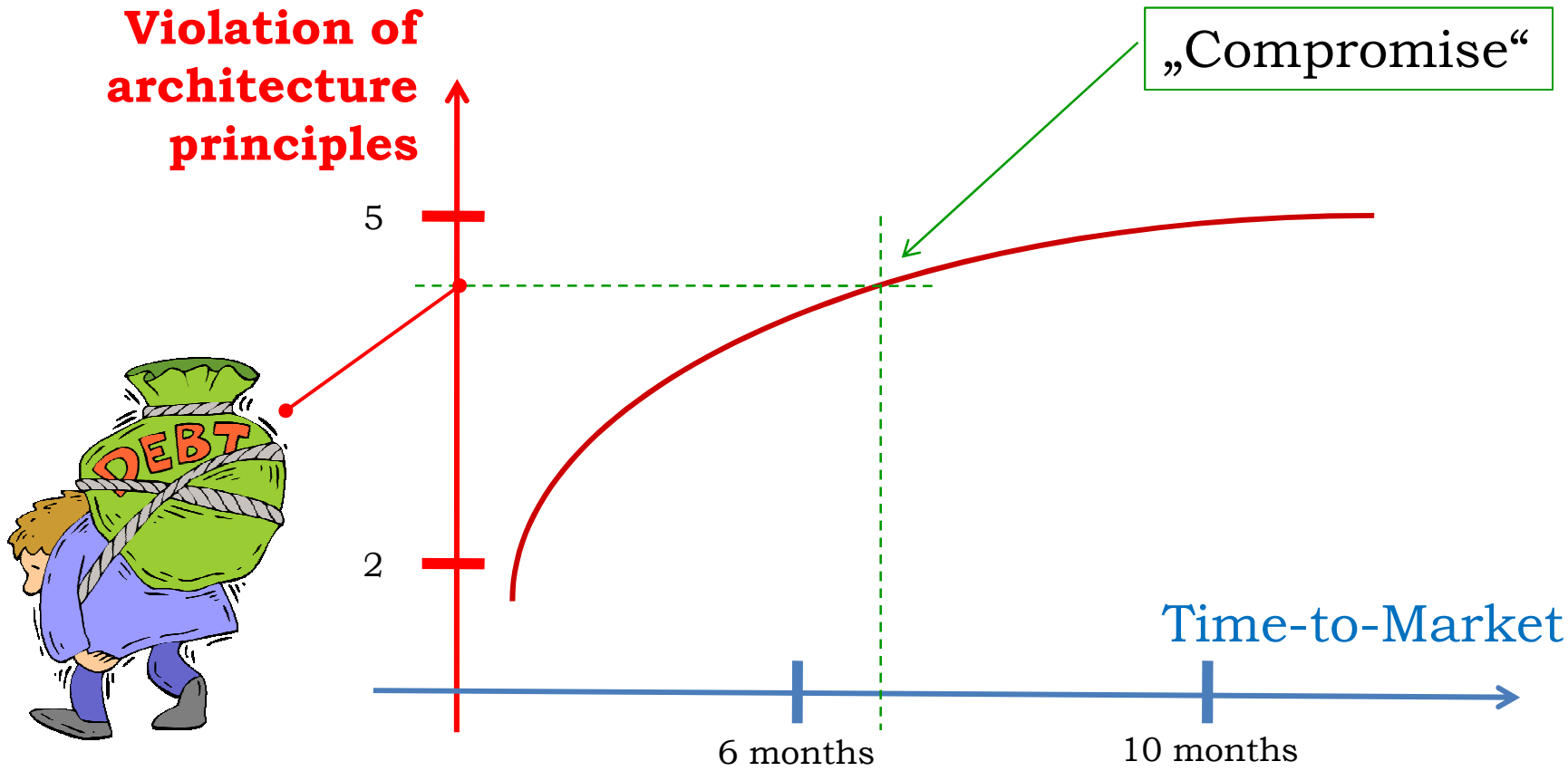
But: Manage the technical debt!

Example 1: Quality Properties **Tradeoffs**

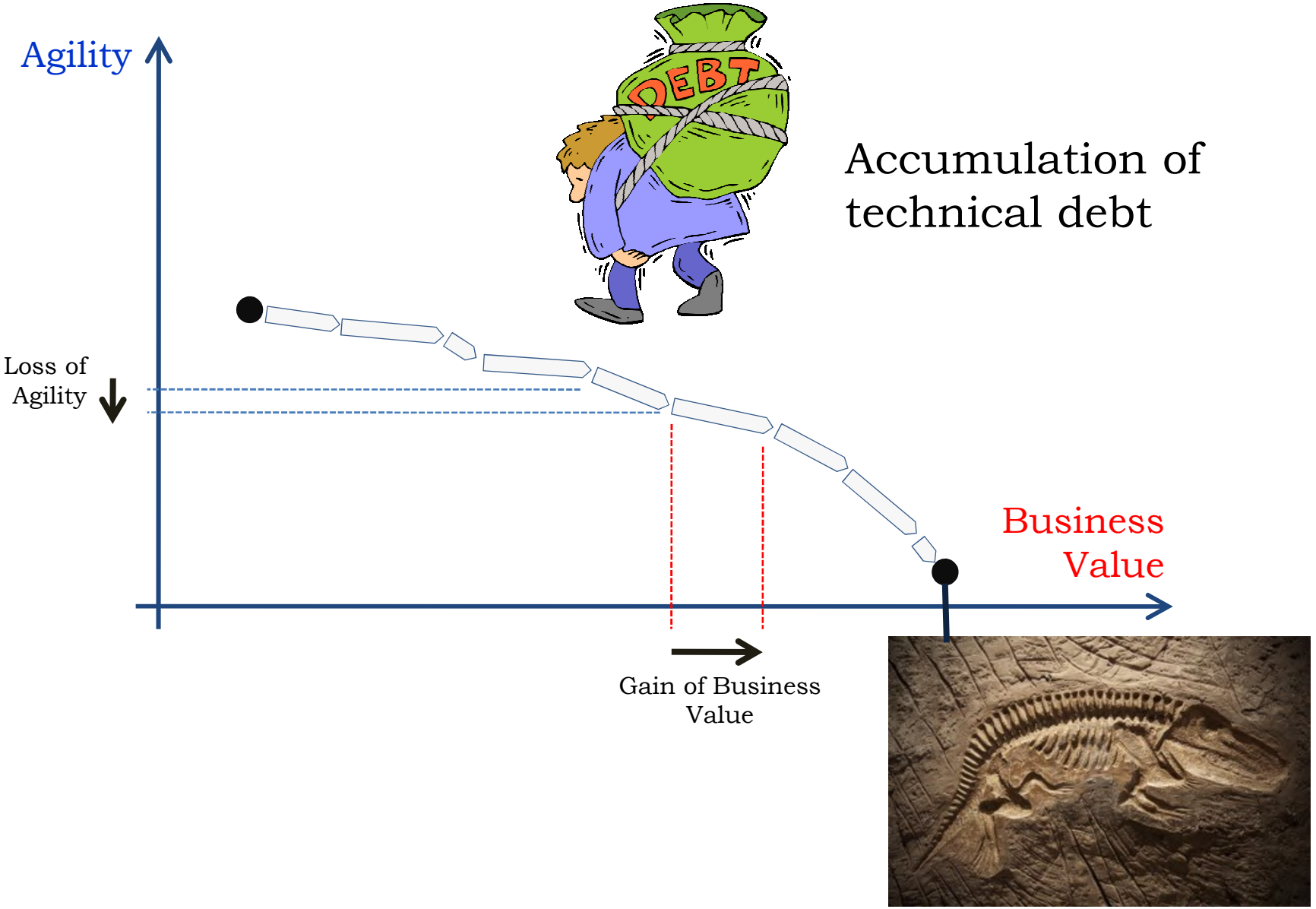
Financial On-Line Transaction System



Example 2: Time-to-Market Shortcuts



Technical debt!


Agility
DEBT
Accumulation of technical debt
Loss of Agility
Business Value
Gain of Business Value

Future-Proof Software-Systems: Responsibility and Role

Role:

Stakeholders

Business Analyst/ Requirements Engineer

Architects and Designers

Programmers, Tester, etc.

- New Business Requirements
- Implementation Specifications
- Architecture Development
- Software Design
- Software Implementation

consulting

review

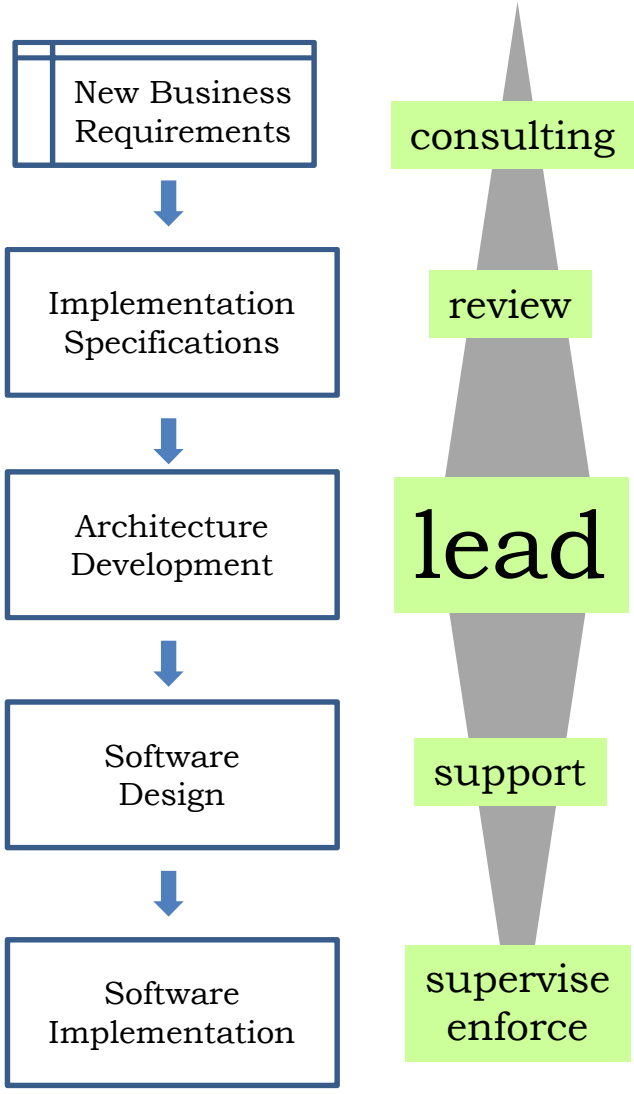
lead

support

supervise enforce

future-proof software-system engineer

Role:



future-proof software-systems engineer

http://www.amisinsurance.com

A *dedicated* and *committed* management

√

Summary

The responsibility of the future-proof software-systems engineer is to design, maintain and evolve an *adequate IT-architecture* which guarantees the required *quality properties* for a specific application area

The future-proof software-systems engineer has „4+1 *roles*“: Missionary, Lawmaker, Consultant, Enforcer + „Magician“

The future-proof software-systems engineer must demonstrate a careful and consequent *balance* between the „4+1 roles“

Whenever the future-proof software-systems engineer agrees to an architectural compromise, *technical debt* is generated

The future-proof software-systems engineer must carefully keep track of technical debt and enforce its reduction/elimination

References:

References	
DeWeck11	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
Eeles10	Peter Eeles, Peter Cripps: **The Process of Software Architecting** Pearson Education (Addison-Wesley), Boston, USA, 2010. ISBN 978-0-321-35748-9
Boehm04	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
Axelrod13	C. Warren Axelrod: **Engineering Safe and Secure Software Systems** Artech House, Norwood, USA, 2013. ISBN 978-1-60807-472-3
Coplien10	James Coplien, Gertrud Bjornvig: **Lean Architecture for Agile Software Development** John Wiley & Sons, Inc., Chicester UK, 2010. ISBN 978-0-470-68420-7
Fairbanks10	George Fairbanks: **Just Enough Software Architecture – *A Risk-Driven Approach*** Marshall & Brainerd, Boulder CO, USA, 2010. ISBN 978-0-9846181-0-1
Kossiakoff11	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
Lattanze09	Anthony J. Lattanze: **Architecting Software Intensive Systems – *A Practicioner's Guide*** Auerbach Publications, Taylor & Francis Group, LLC, 2009. ISBN 978-1-4200-4569-7
Sewell02	Marc T. Sewell, Laura M. Sewell: **The Software Architect's Profession – *An Introduction*** Prentice Hall PTR, New Jersey, USA, 2002. 978-0-13-060796-7

Future-Proof Software-Systems:

The Skills and Personality of the Future-Proof Software-Systems Engineer

Skills

Skill:

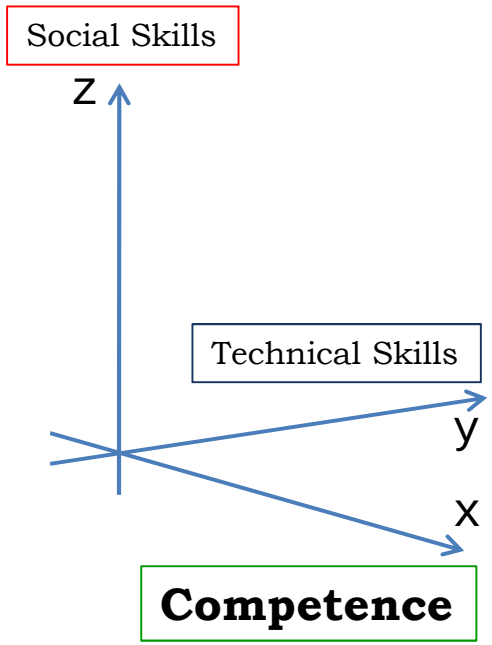
The ability to do something well

[The New Oxford Dictionary of English]

http://www.inman.com

Skills Coordinate System

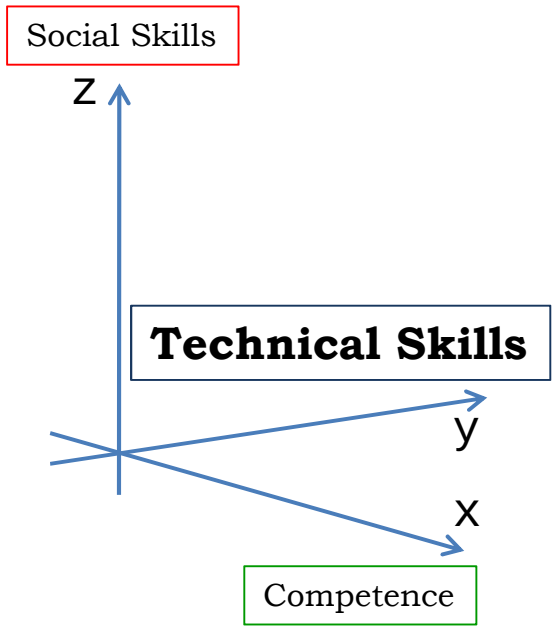
Skills: **Competence**



(Professional) Competence

- IT (architecture) knowledge √
- IT (practical) experience
- State-of-the-Art knowledge (broad, hardware, software, processes)
- Technology mastering (HW & SW)
- Business knowledge
- Innovation capability
- Vision

Skills: **Technical Skills**



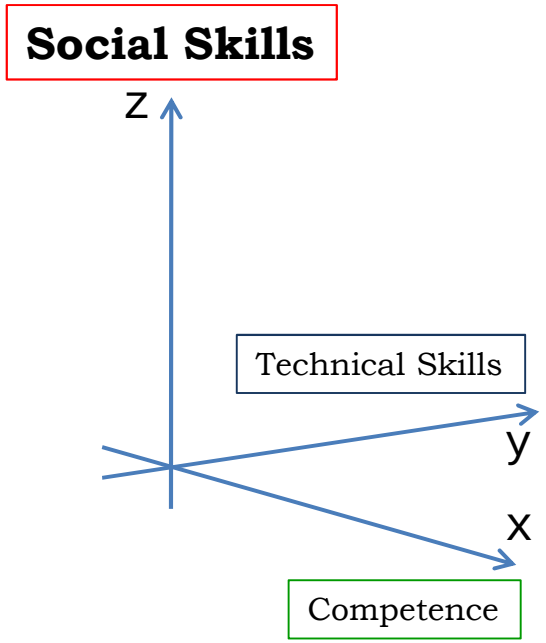
Technical Skills

- Communication skills (speech & writing)
- Presentation skills (oral, graphical & writing)
- Logical reasoning capability
- Efficiency & effectiveness
- Languages
- „Architecture Feel“ (Simp & beauty)

Efficiency:
Doing the things right

Effectiveness:
Doing the right things

Skills: **Social Skills**

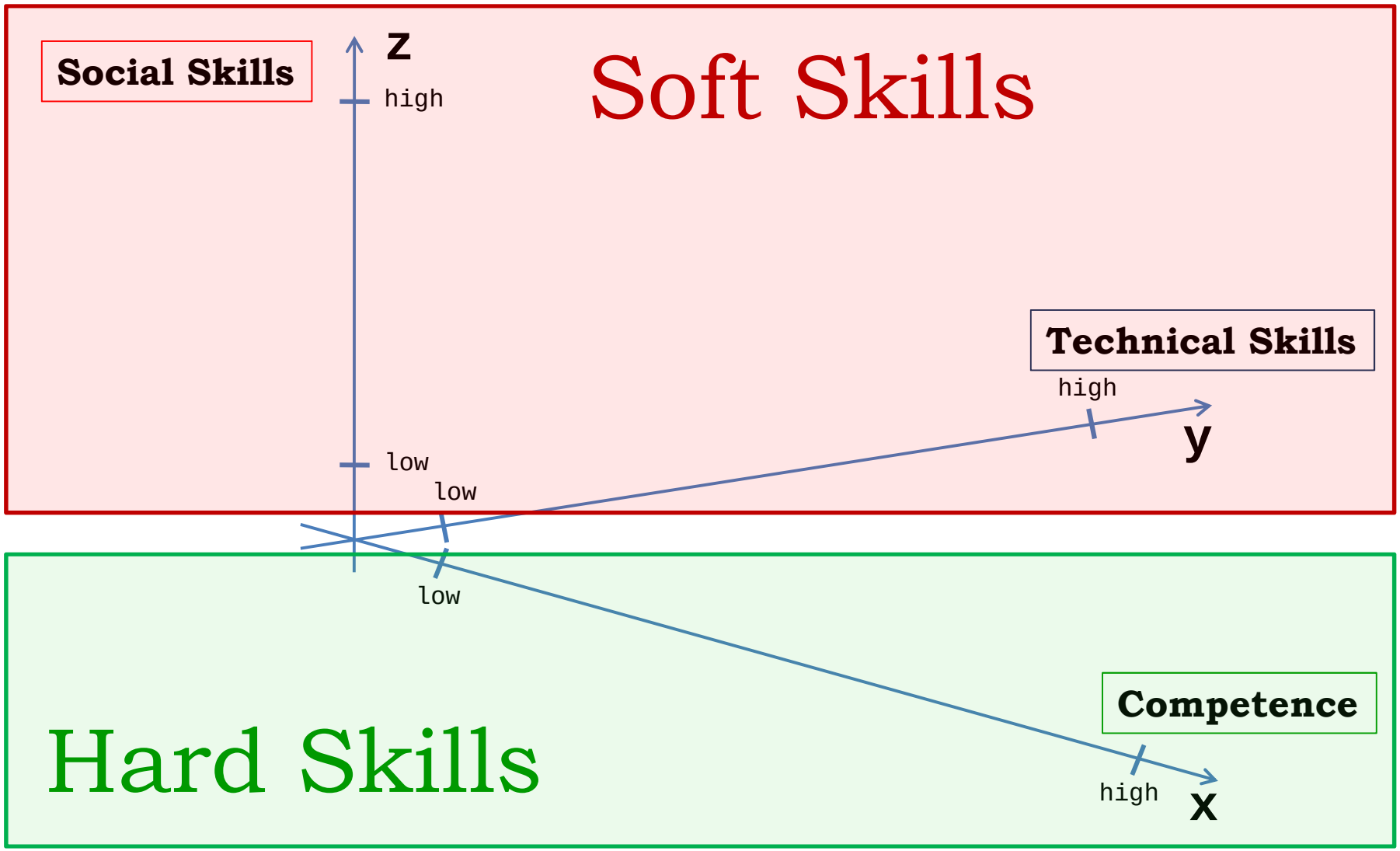


Social Skills

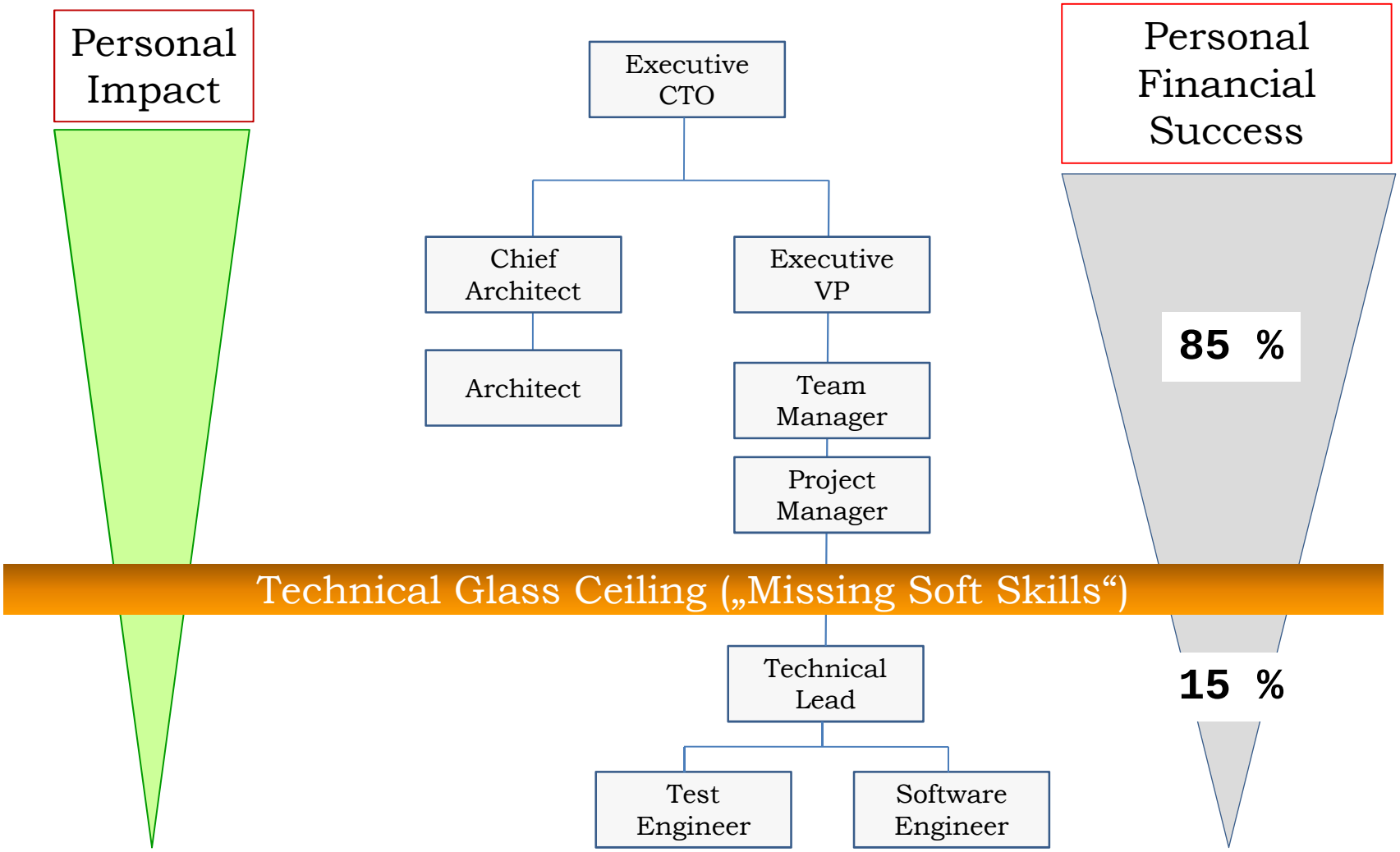
- Negotiation skills
- Persuasion capability
- People interaction capability
- Enthusiasm
- Leadership
- Life-long learning
- Socializing/Networking
- Team Work
- Honesty (Ethics)
- Work-life balance

http://samingersoll.com/life-work-balance/

Skills Coordinate System



Hard Skills ↔ *Soft* Skills: Which are more important?



Dale Carnegie, 1937
ISBN 978-1-4391-9919-0

Dave Hendricksen, 2012, ISBN 978-0-321-71729-0

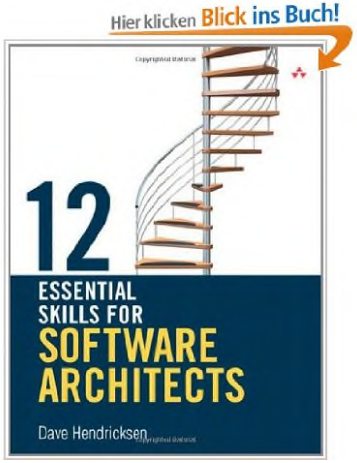
Hard Skills ↔ *Soft* Skills: Which are more important?

The „future-proof software-systems engineer“:

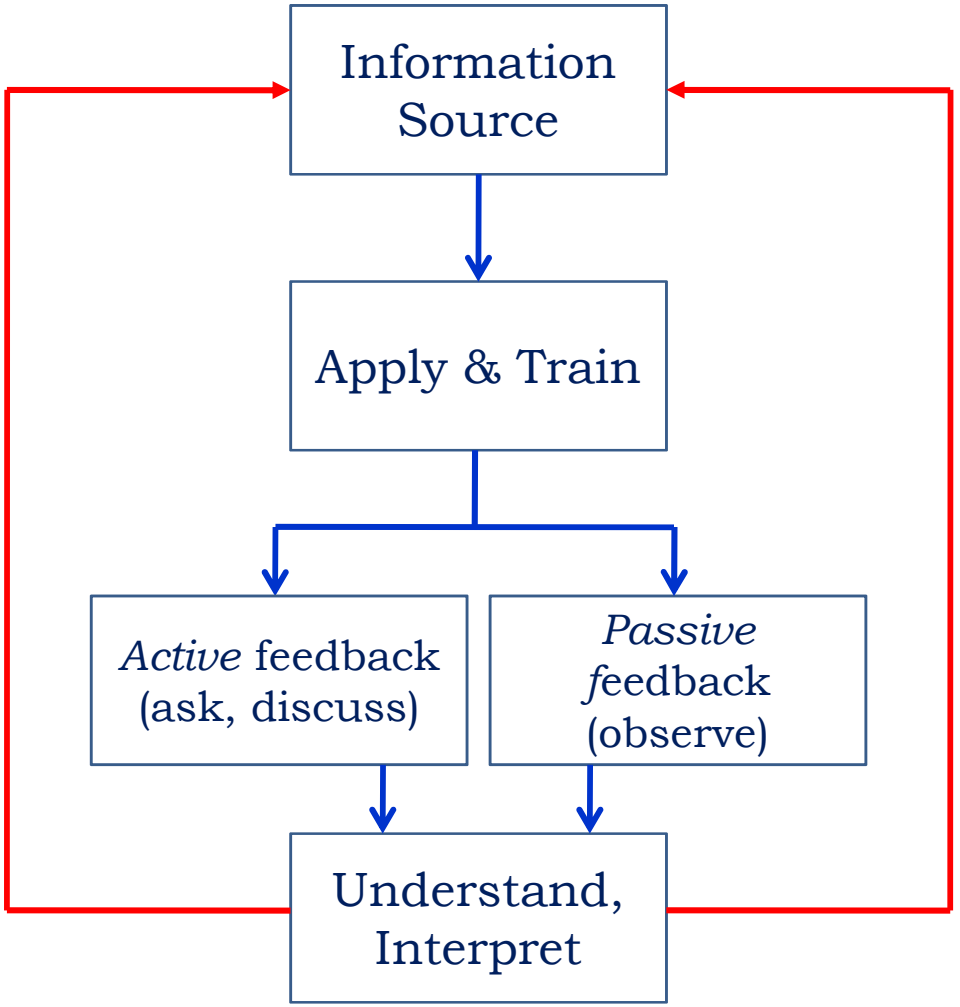
„*Hard skills* help us qualify for a job;

Soft skills dictate our career growth“

[Wushow Chou, 2013, ISBN 978-1-118-52178-6]



How can we learn *Soft* Skills?

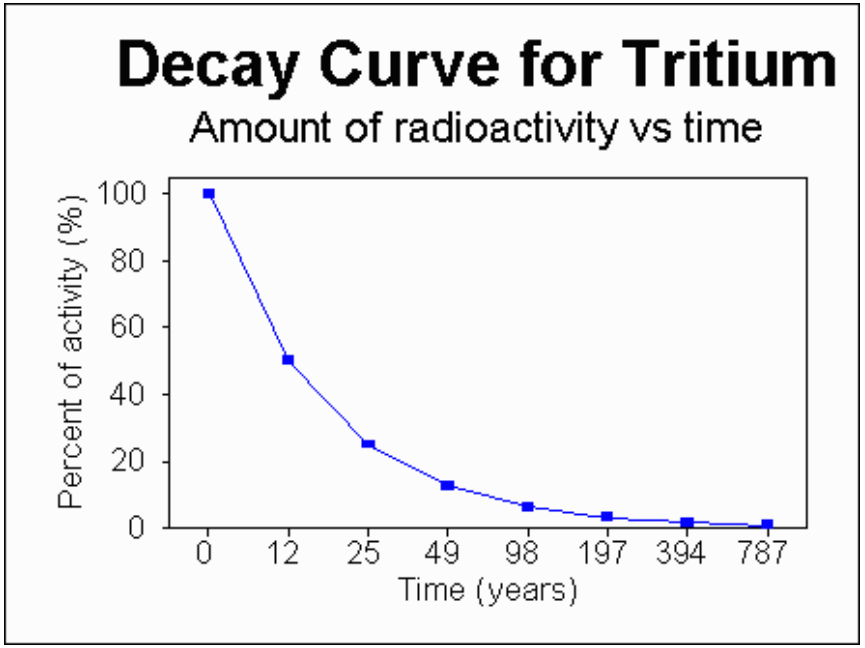


Life-Long Learning:

„*Half-Life* of IT-Engineering Knowledge“

Which is the half-life of IT-engineering knowledge ?

Def: The time-span after which *half* of your current IT-knowledge has become obsolete



http://wwwchem.csustan.edu/chem3070/3070m04.htm

Field	Half-life (in years)
Physics	**13**
Mathematics	**9**
Economics	**9**
Computer Science	**6**

Future-Proof Software-Systems:

The Skills and Personality of the Future-Proof Software-Systems Engineer

Personality

Personality

Personality:

The combination of characteristics or qualities

that form an individual‘s distinctive character

[The New Oxford Dictionary of English]

Personality

„The fundamental principle behind any *soft skill* is to cultivate the perception in other people‘s minds that they can gain and benefit by engaging with us“

[Wushow Chou 2013, ISBN 978-1-118-52178-6]

http://www.signalpatternslabs.com

Personality

Photo Credit: Silvia Furrer

http://en.wikipedia.org/Great_Horned_Owl

Courage

Wisdom

Fighting Spirit

Mediation Capability

http://www.wildanimalfightclub.com

http://www.taringa.net

… and – most important:

Honesty
(Ehrlichkeit)

(Professional) Competence:
Your professional advice must be (provably) correct and believable, as well as realistic

Behaviour:
Your behaviour must be truthful, fair and human in all situations

http://warrencampdesign.com

Praising and Reprimanding

http://www.mindtools.com

http://footage.shutterstock.com

Praise:
- honest
- precise
- no „..., but ...“
- (can be) personal

„Your design of the module ABC is clear and elegant.
I like it“

Reprimand:
- true
- precise
- fair
- constructive
- never personal

„You did not take into consideration that a suitable data structure is already existing“

Software Engineering **Ethics**

http://courses.planetizen.com/course/planning-ethics

ACM/IEEE: Software Engineering Code of Ethics and Professional Practice (© 1999)

1. PUBLIC - Software engineers shall act consistently with the public interest.
2. CLIENT AND EMPLOYER - Software engineers shall act in a manner that is in the best interests of their client and employer consistent with the public interest.
3. PRODUCT - Software engineers shall ensure that their products and related modifications meet the highest professional standards possible.
4. JUDGMENT - Software engineers shall maintain integrity and independence in their professional judgment.
5. MANAGEMENT - Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance.
6. PROFESSION - Software engineers shall advance the integrity and reputation of the profession consistent with the public interest.
7. COLLEAGUES - Software engineers shall be fair to and supportive of their colleagues.
8. SELF - Software engineers shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession.

http://www.acm.org/about/se-code

Summary

The succesfull future-proof software-systems engineer has excellent (professional) *competence* and highly developped *soft skills*

Missing (or insufficient) soft skills greatly limit (or inhibit) both the effectiveness and also the career chances of the future-proof software-systems engineer

The future-proof software-systems engineer do *life-long learning* (half-time of computer science knowledge ≈ 6 years!)

The personality of the succesfull future-proof software-systems engineer features *courage*, *wisdom*, *fighting spirit* and *mediation capability*

Unconditional *honesty* and adherence to *ethics* is a fundamental characteristic

References:

References	
Selinger04	Carl Selinger: **Stuff You Don't Learn in Engineering School – *Skills for Success in the Real World*** John Wiley & Sons, Inc., Hoboken, N.J, USA, 2004. ISBN 0-471-65576-7
Hendricksen12	Dave Hendricksen: **12 Essential Skills for Software Architects** Pearson Education, Addison-Wesley, J.J., USA, 2012. ISBN 978-0-321-71729-0
Johnson07	Steven Johnson: **The IT Professional's Business and Communications Guide** Wiley Publishing Inc., Indianapolis, USA, 2007. ISBN 978-0-470-12635-6
Carnegie10	Dale Carnegie: **How to Win Friends and Influence People** Pocket Publishing, New York, USA, 2010 (first published 1937). ISBN 978-1-4391-9919-0
Chou13	Wushow „Bill" Chou: **Fast-Tracking Your Career – *Soft Skills for Engineering and IT Professionals*** IEEE Press, John Wiley & Sons, Inc., N.J., USA, 2013. ISBN 978-1-118-52178-6
ACM/IEEE99	ACM/IEEE: ***Software Engineering Code of Ethics and Professional Practice*** Version 5.2, 1999. Downloadable from: http://www.acm.org/about/se-code [last accessed: 1.11.2013]

Future-Proof Software-Systems:

The Working Context of the successfull Future-Proof Software-Systems Engineer

Responsibility:

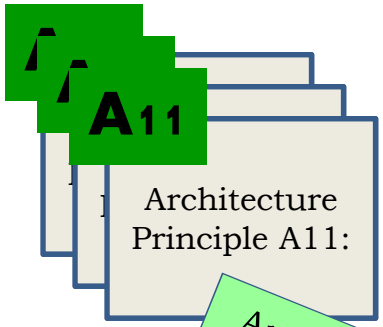
Develop, maintain and enforce an *adequate IT-architecture*

to guarantee the required *quality properties* of the system,

especially the continuous increase in *agility* and *resilience*

… only possible with an adequate working context

Context: Impact

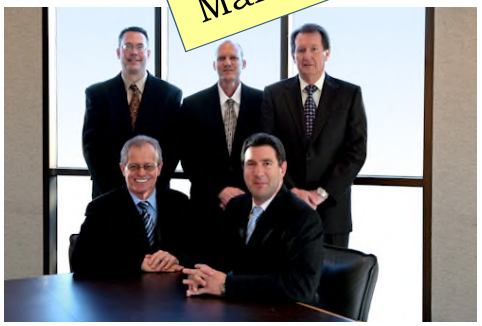


Architecture Knowledge

http://www.midcontinent.org/press/press_rivets.html

Legacy Restrictions

Management Expectations



http://sgs-uae.com

3 devils

„The three devils of systems engineering:

- Complexity,
- Change,
- Uncertainty"

Anonymous

Successfull Working Context: **Elements**

http://www.telco2.net/blog

Collaborative Business

❶

❷

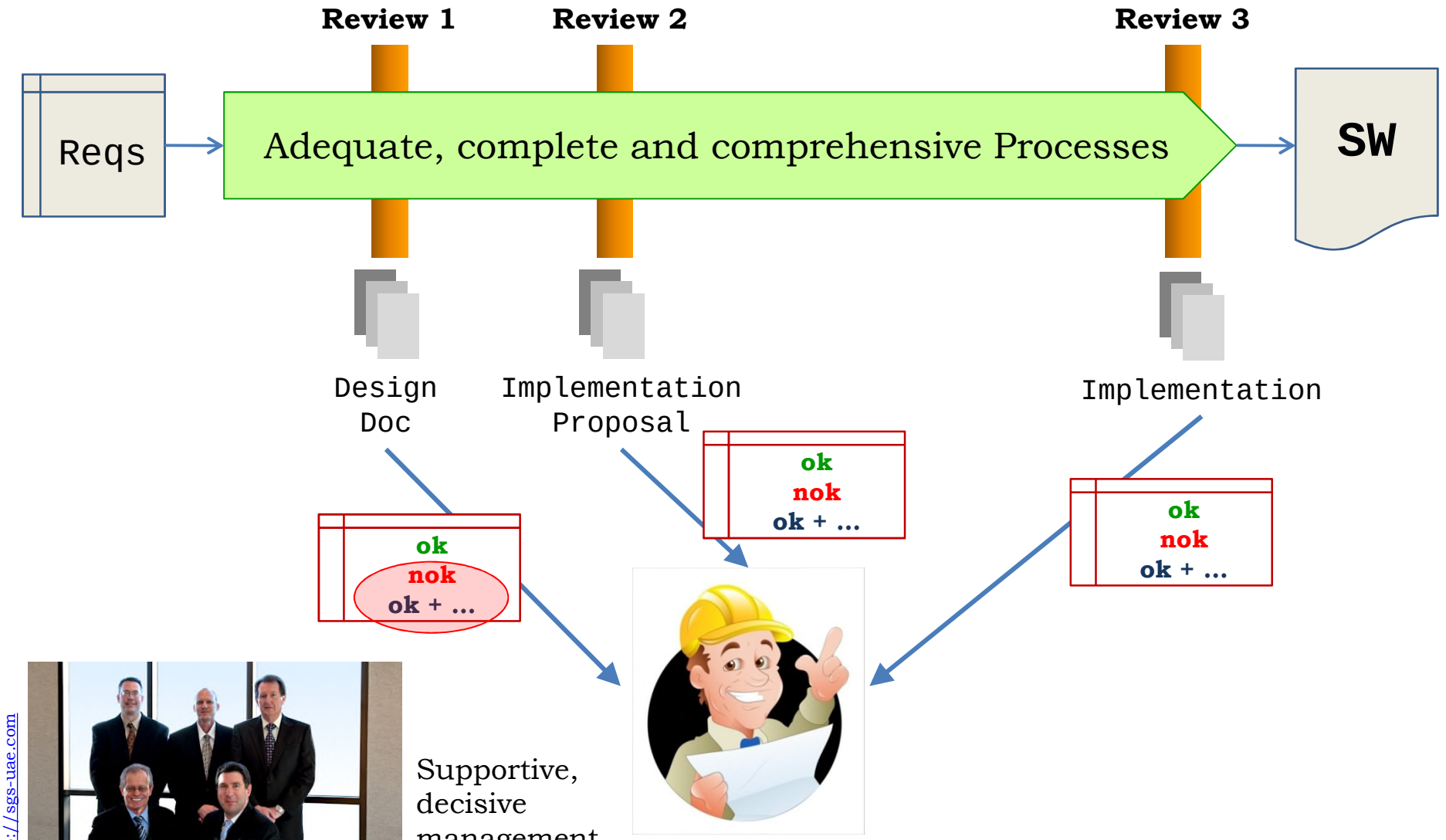
http://sgs-uae.com

Supportive, decisive management

❸

Reqs → Adequate, complete and comprehensive Processes → SW

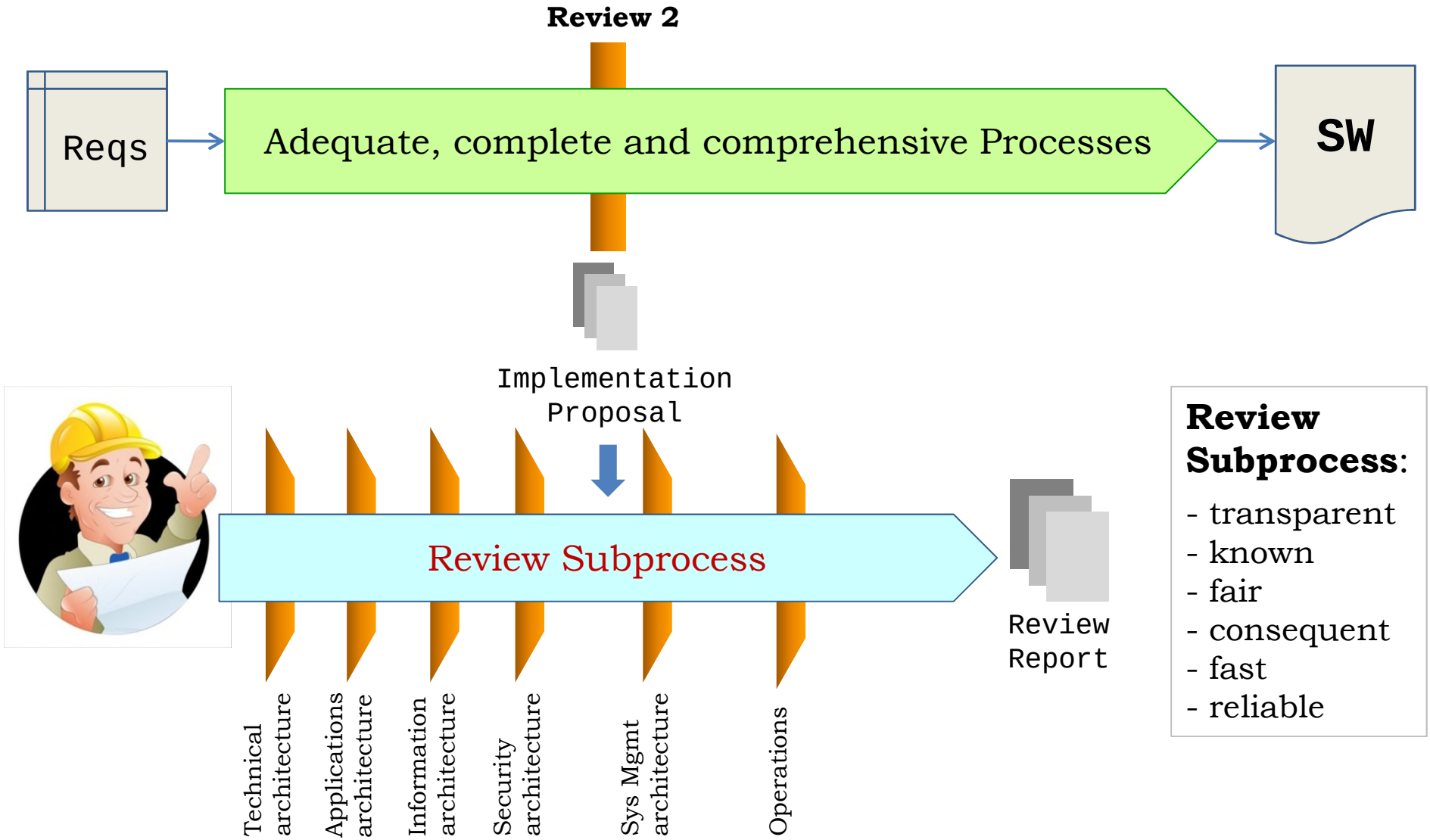
Successfull Working Context: **Process**



http://sgs-uae.com

Supportive, decisive management

Successfull Working Context: **Review Procedure**



Review Subprocess:

- transparent
- known
- fair
- consequent
- fast
- reliable

Future-Proof Software-Systems: Working Context

Your **Relationships**:

Review Subprocess

Review Report

ok
nok
ok + ...

www.123rf.com

Project Teams

Team Spirit

Respect, Trust

http://www.telco2.net/blog

Business

Cooperation

Transparency

http://sgs-uae.com

Peer Architects

IT Management

Example: Financial Institution – Review Report (1/2)

IT Architecture Project Review Board of the 27.05.2009

Review Summary

	OK	OKA	NOT OK	Reasons for NOK
Evaluation Architecture		X		

Architecture Reviewers	Date	Findings	Condition	Deadline
Hans Muster Peter Beispiel Jürg Modell	19.05.2009	The proposed data migration concept leads to unmanaged data redundancy	Propose a new data migration concept which completely eliminates data redundancy	PO

Exceptions, accepted deviation from standards
none

Conditions of Previous Reviews	Y	N	I	Comment / Statement
Have the conditions of the previous review(s) been met? If conditions have not been met, discuss further actions with KSCD			**X**	No open conditions

Example: Financial Institution – Review Report (2/2)

Part 2: General setup, integration/delineation

Nr.	PC	PO	RO	Legend: Y = Yes / N = No / I = Irrelevant	Y	N	I	Comment / Statement
G02	**X**	**X**	**(X)**	Integration into the overall system/ avoidance of redundancies ▪ Are the boundaries/interactions with other projects, processes, domains, applications and infrastructures clear and appropriate? ▪ Are potential redundancies, overlaps e.g. concerning infrastructures, services reasonable? Justified? Accepted? ▪ Is there a mix of old and new architecture? Reasonable? Justified?				
G03		**X**	**X**	Migration to standard architecture / phase out of obsolete architecture/standards: ▪ Are the necessary actions for a migration to standard architecture planned, described and appropriate? ▪ Is it documented how/when old architecture will be phased out?				
G04	**(X)**	**X**	**(X)**	Options: ▪ Are the proposed options appropriate and complete? ▪ Is the proposed option reasonable?				
G05	**(X)**	**X**	**X**	Risks: ▪ Have all risks relevant for architecture been identified? ▪ Have they been mitigated accordingly?				
etc.								

Review Details

Context for successful architecture work:

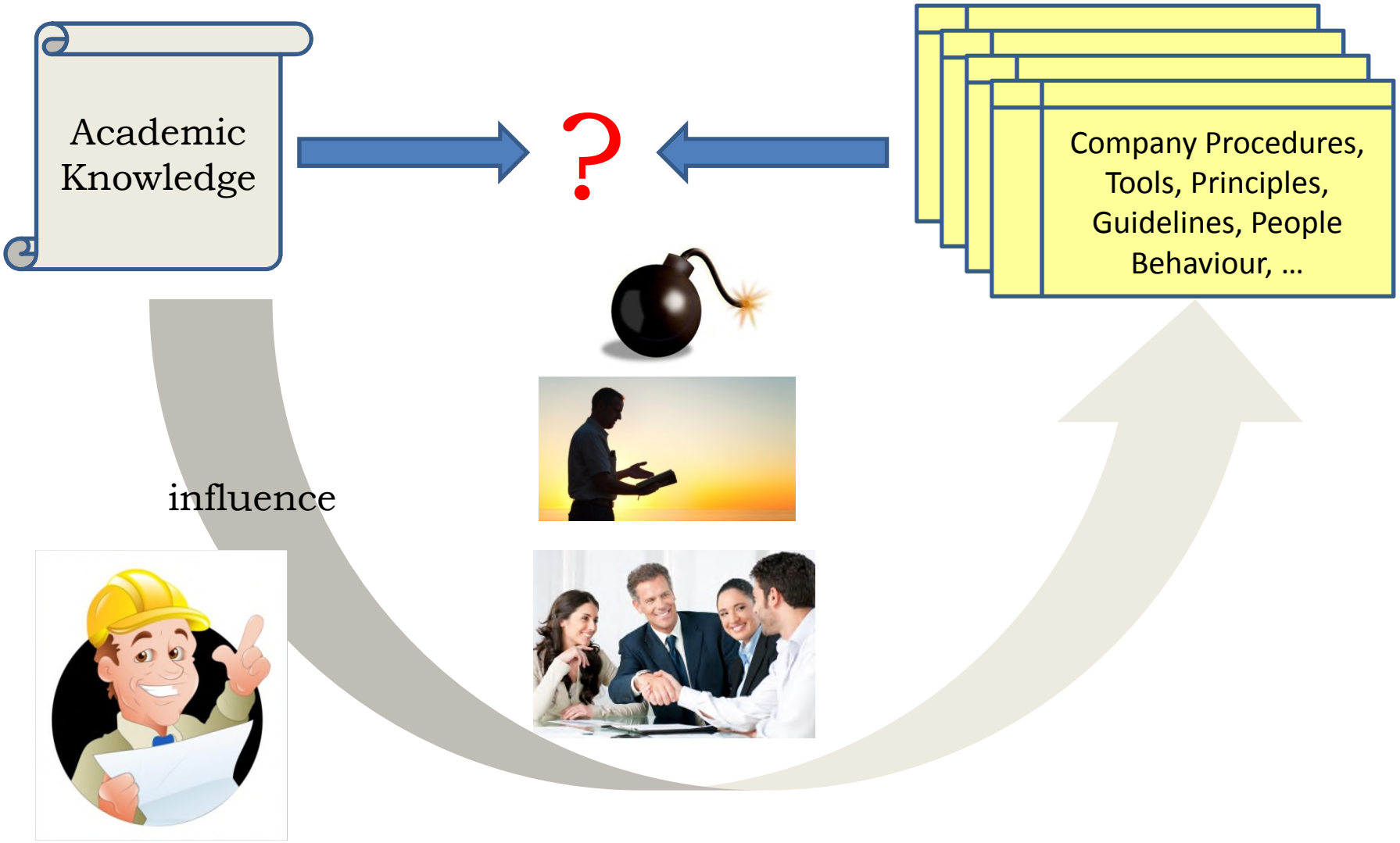
- ✓ An architecture-aware company culture
- ✓ A supporting (top) management
- ✓ Principles, standards and guidelines
- ✓ Accepted managed evolution strategy
- ✓ Adequate processes with good governance

Future-Proof Software-Systems: Working Context

http://sportsfitnessnetwork.com

The (positive) impact of the future-proof software-systems engineer relies on **good relationships** with project teams, business partners, peer architects and IT management

Transition to **Industry**



Summary

The context of the future-proof software-systems engineer is a *tension field* between architecture knowledge, management expectations, legacy systems restrictions and the 3 devils of systems engineering (complexity, change and uncertainty)

The context enabling the success of the future-proof software-systems engineer includes: (1) Collaborative business partners, (2) a supportive, decisive management, and (3) adequate, complete and comprehensive processes

The (positive) impact of the future-proof software-systems engineer relies on **good relationships** with project teams, business, peer architects and IT management

References:

References	
Murer11	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
El-Haik10	Basem S. El-Haik, Adnan Shaout: **Software Design for SIX-SIGMA – *A Roadmap for Excellence*** John Wiley & Sons, Inc., Hoboken, N.J., USA, 2010. ISBN 978-0-470-40546-8
Ahlemann12	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
DeMarco97	Tom DeMarco: **The Deadline – *A Novel About Project Management*** Dorset House Publishing, N.Y., USA, 1997. ISBN 978-0-932633-39-2
Ebert12	Christof Ebert: **Global Software and IT – *A Guide to Distributed Development, Projects, and Outsourcing*** (IEEE Series) John Wiley & Sons, Inc., N.Y., USA, 2012. ISBN 978-0-470-63619-0
Eeles10	Peter Eeles, Peter Cripps: **The Process of Software Architecting** Pearson Education (Addison-Wesley), Boston, USA, 2010. ISBN 978-0-321-35748-9
Endres03	Albert Endres, Dieter Rombach: **A Handbook of Software and Systems Engineering – *Empirical Observations, Laws and Theories*** Pearson Education Ltd., Harlow UK, 2003. ISBN 978-0-321-15420-7
Knittel06	Susanne Knittel-Ammerschuber: **Architecture: The Element of Success – *Building Strategies and Business Objectives*** Birkhäuser Verlag, Basel, CH, 2006. ISBN 978-3-76437465-5
Rodin00	Robert Rodin: **Free, Perfect, and Now – *Connecting the Three Insatiable Customer Demands*** Touchstone, Simon & Schuster, New York, USA, 2000. ISBN 978-0-684-85022-2

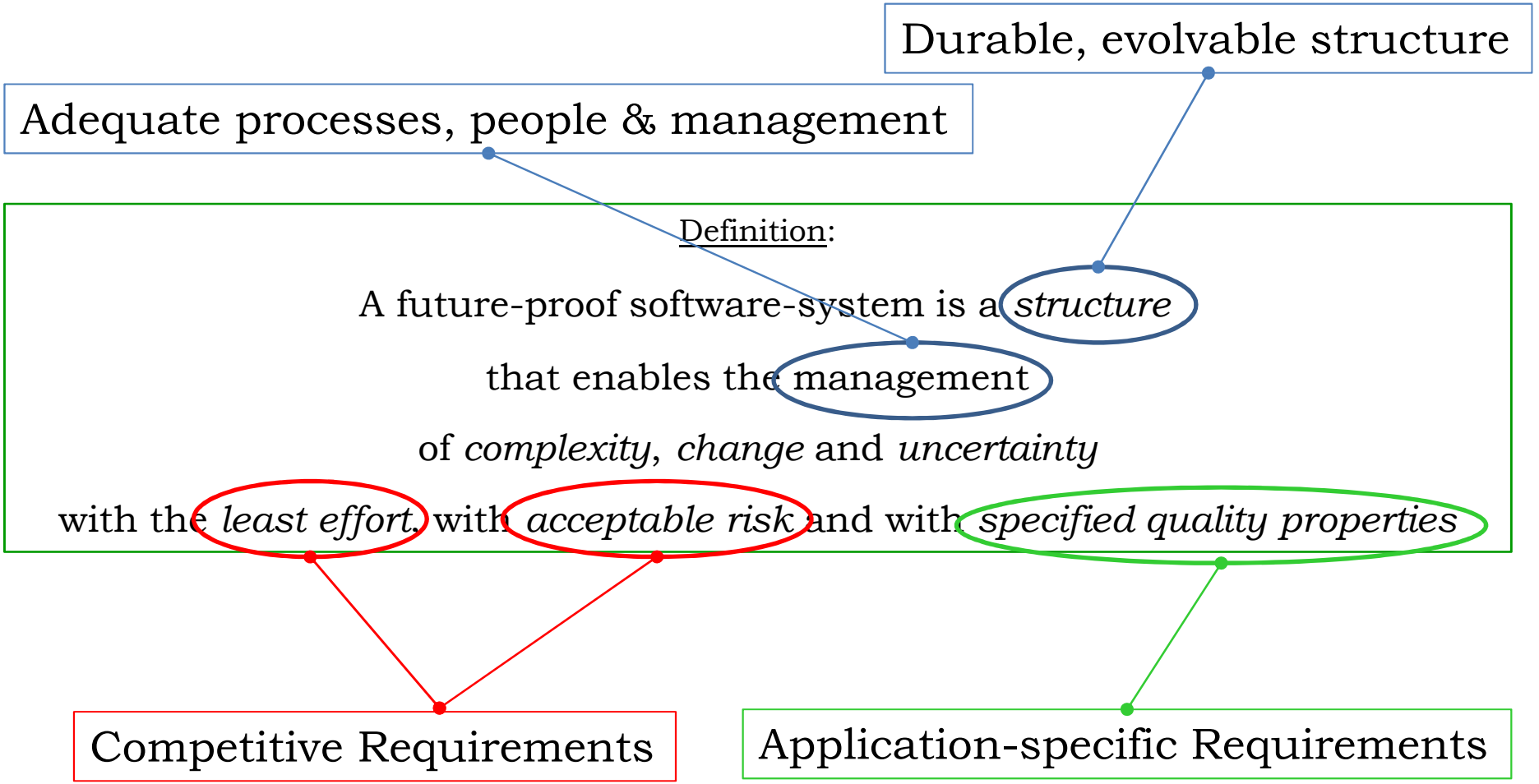
Future-Proof Software-Systems:

Finally:

Do We now have a Future-Proof Software-System?

(… are we there?)

Reminder: What do we really want?



Future-Proof Software-Systems: Are we there?

Business Pressure:
- Short Time-to-Market
- Low Development Cost

Reminder:

What do we really want?

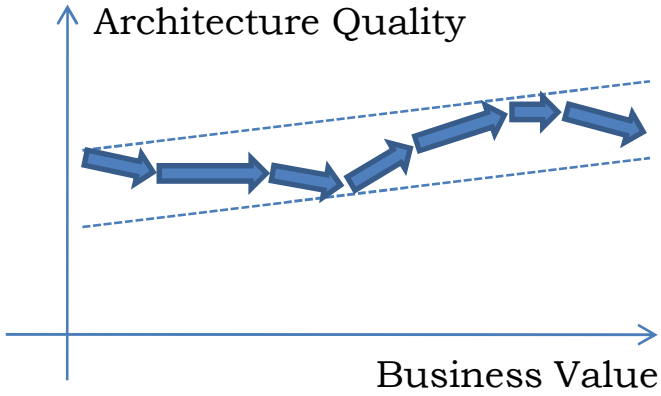
Future-Proof Software-Systems Engineer

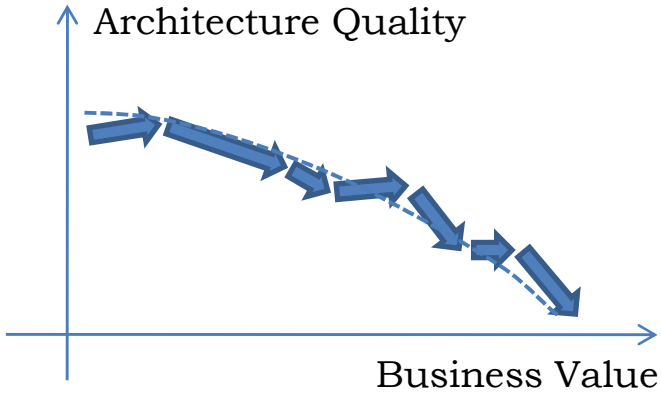
- Functionality

Architecture-Quality:
- Agility
- Availability
- Security
- Safety
- …
- Simplicity
- Elegance

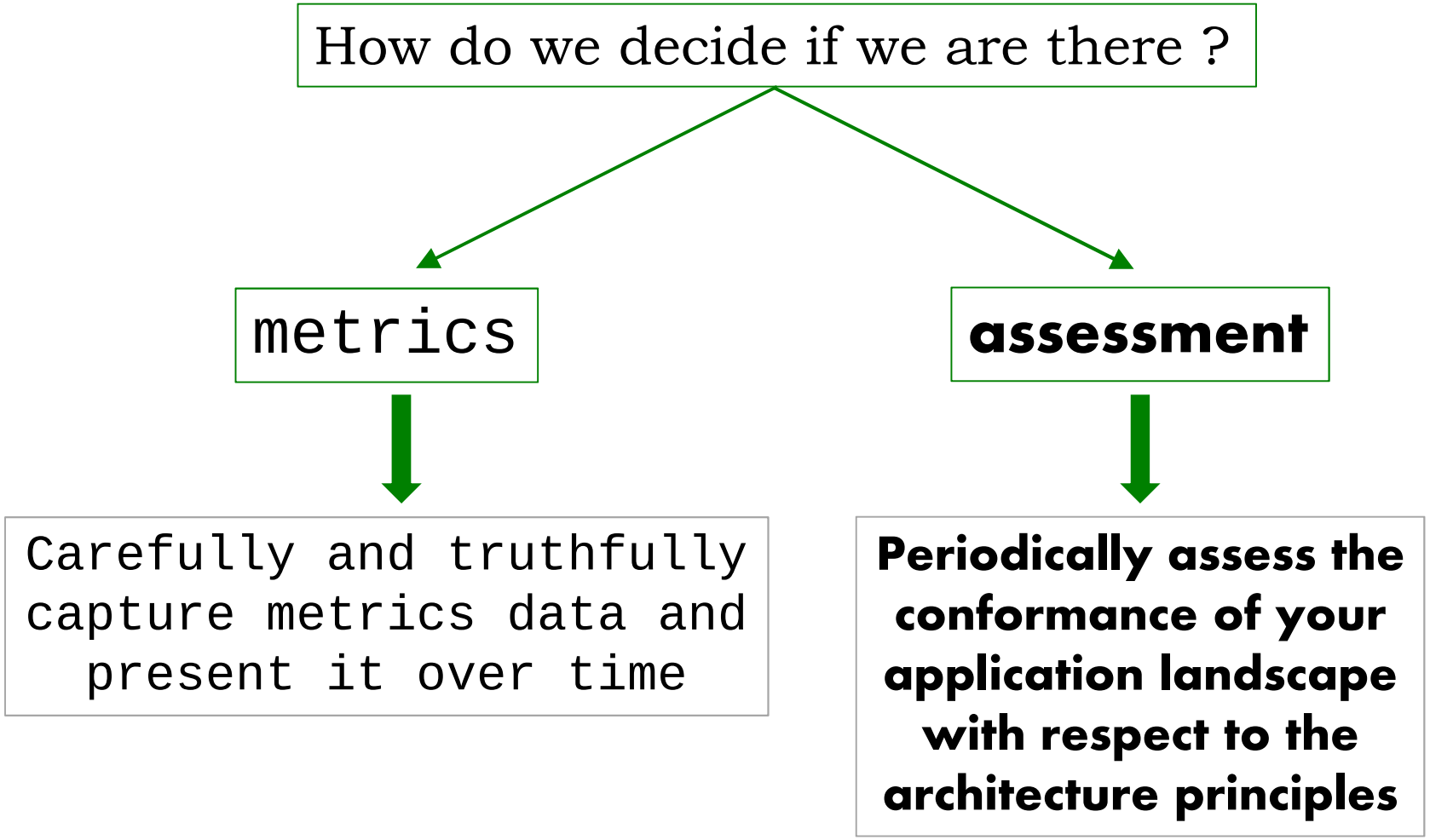
Success

Failure





Future-Proof Software-Systems: Are we there?



Business Value Metric: NPV (Net Present Value)

$$\mathbf{NPV} = \sum_{n} \frac{\mathrm{Benefit}_{\mathrm{year\text{-}n}}}{(1 + i)^n} - I$$

Agility Metric:

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

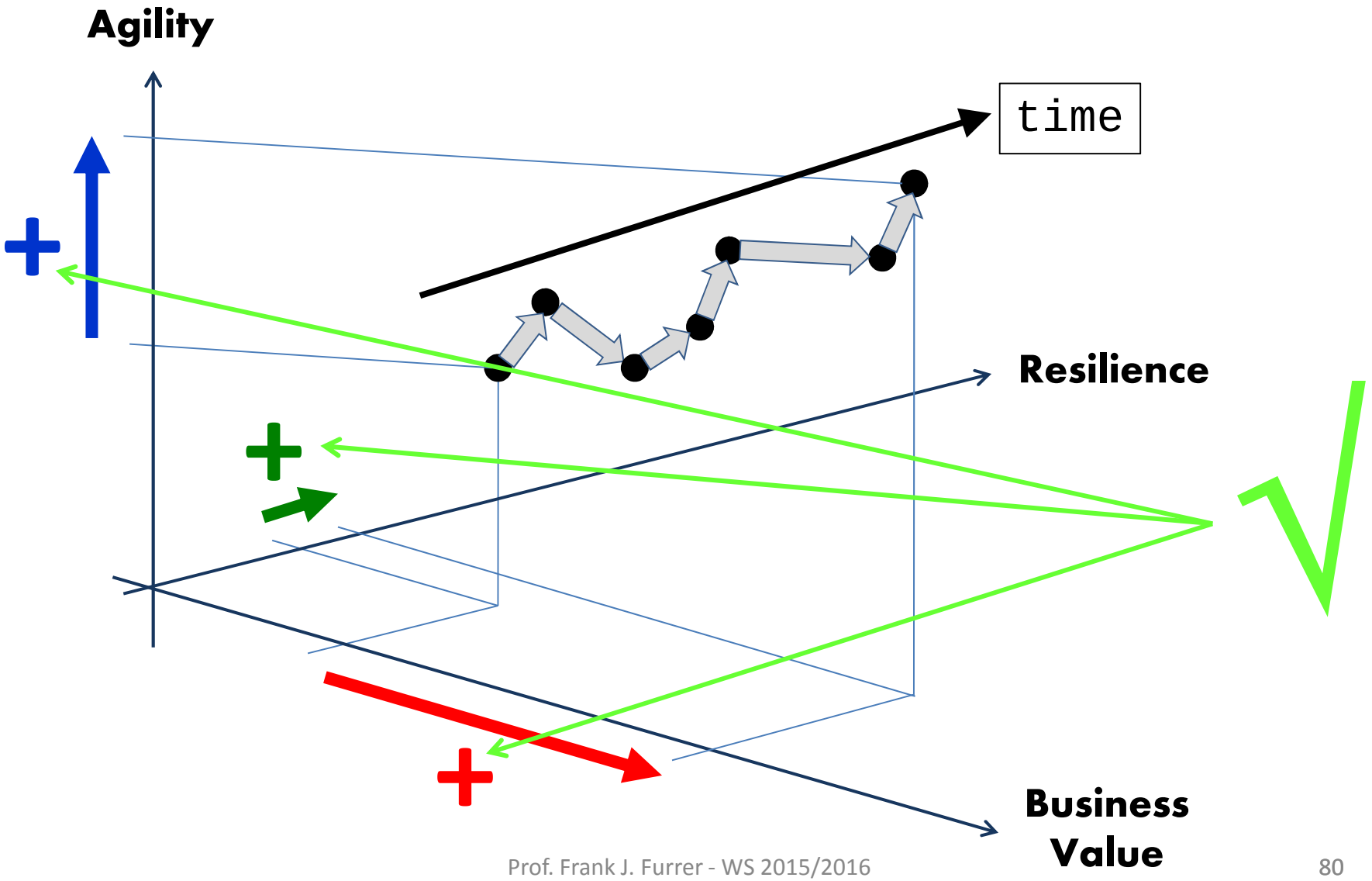
Unit: #UCP²/(days*k€)

Resilience Metric:

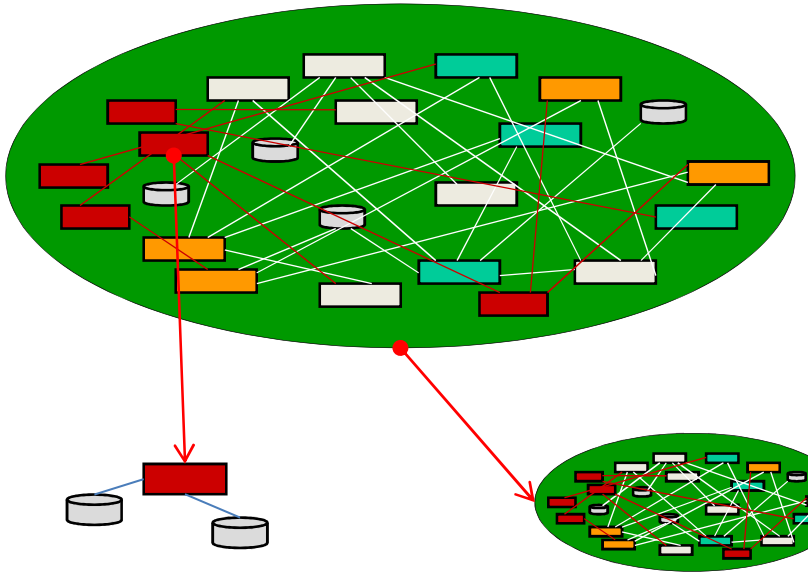
⇒ **Damage potential/impact assessment**

… & other quality metrics

Future-Proof Software-Systems: **Managed Evolution Trajectory**



Assessment: Future-Proof Software-System?



Applications Landscape

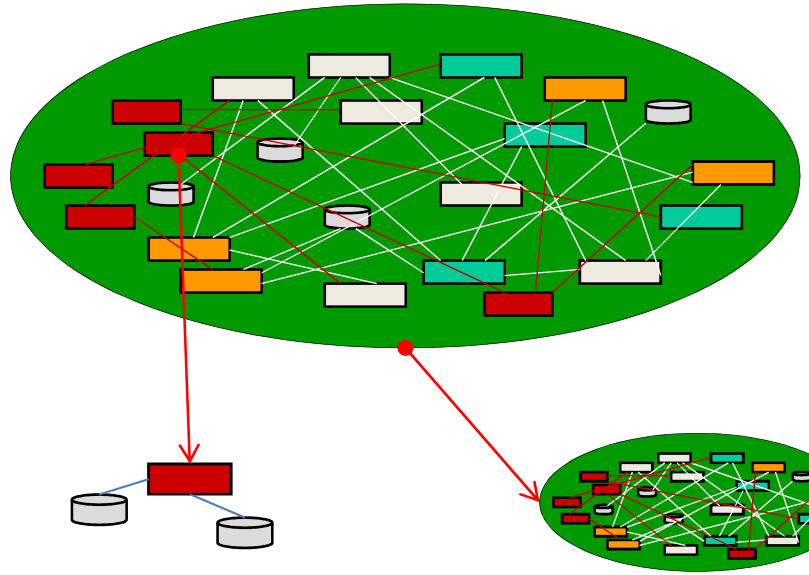
Single application

Application Landscape

Application #ABC	
Architecture Principle	***Compliance Metric***
A1	0.71
A2	0.32
A3	0.89
etc.	-

aggregate & average

Assessment: Future-Proof Software-System?



Applications Landscape

Single application | Application Landscape

√ Architecture Quality

Functional (Business) „Fit-for-Future“)

Development

Assessment & Rectification

Technical Debt

http://www.telegraph.co.uk

... find and eliminate technical debt

... avoid technical debt

… we are never really there

http://fleurdelyz.com

References:

References	
Fairbanks10	George Fairbanks: **Just Enough Software Architecture – *A Risk-Driven Approach*** Marshall & Brainerd, Boulder CO, USA, 2010. ISBN 978-0-9846181-0-1
Clements02a	Paul Clements, Rick Kazman, Mark Klein: **Evaluating Software Architectures – *Methods and Case Studies*** Addison-Wesley, Boston, USA, 2002. ISBN 0-201-70482-X
Anderson12	Stuart Anderson, Massimo Felice: **Emerging Technological Risk – Underpinning the Risk of Technology Innovation** Springer-Verlag, London, UK, 2012. ISBN 978-1-4471-2142-8
Apel13	Sven Apel, Don Batory, Christian Kästner, Gunter Saake: **Feature-Oriented Software Product Lines – *Concepts and Implementation*** Springer-Verlag, New York, USA, 2013. ISBN 978-3-642-37520-0
Clements10	Paul Clements, Felix Bachmann, Len Bass, David Garlan, James Ivers, Reed Little, Paulo Merson, Robert L. Nord: **Documenting Software Architectures: Views and Beyond** SEI Series in Software Engineering. Addison Wesley, MA, USA, 2nd revised edition, 2010. ISBN 978-0-321-55268-6
Cusumano04	Michael A. Cusumano: **The Business of Software** Free Press, New York, N.Y., USA, 2004. ISBN 978-0-7432-1580-0
Godinez10	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
Hohmann03	Luke Hohmann: **Beyond Software Architecture – *Creating and Sustaining Winning Solutions*** Pearson Education, Addison-Wesley, Boston, USA, 2003. ISBN 978-0-201-77594-8

Future-Proof Software-Systems:

The Engineer of 2020

2014

2020

1 – Globalization

2 – Technology Progress

3 – Systems-of-Systems (SoS‘s) and
Cyber-Physical Systems (CPS‘s)

4 – Systems Assembly, Contract-based Engineering

5 – Worldwide Engineering/Development Teams

6 – Threats and Risks

7 – Economy and Social Requirements

8 – Regulations & Liability

References:

References	
NAE04	U.S. National Academy of Engineering: **Engineer of 2020: Visions of Engineering in the New Century** National Academy Press, Washington D.C., USA, 2004. ISBN 978-0-309-09162-4
NAE05	U.S. National Academy of Engineering: **Educating the Engineer of 2020: Adapting Engineering Education to the New Century (Phase II)** National Academy Press, Washington D.C., USA, 2004. ISBN 978-0-309-09649-9
Cusumano04	Michael A. Cusumano: **The Business of Software** Free Press, New York, N.Y., USA, 2004. ISBN 978-0-7432-1580-0
Nanz11	Sebastian Nanz (Editor): **The Future of Software-Engineering** Springer-Verlag, Heidelberg, 2011. ISBN 978-3-642-15186-6
Daylight11	Edgar G. Daylight, Sebastian Nanz (Editors): **Conversations: The Future of Software Engineering – *Panel Discussions.*** 22-23 November 2010, ETH Zurich. Lonely Scholar bvba, Heverlee, Belgium, 2011. ISBN 978-94-91386-01-5
PCAST14	President's Council of Advisors on Science and Technology (PCAST): Designing a Digital Futre: Federally Funded Research and Development in Networking and Information Technology. January 2013. Downloadable from: http://www.whitehouse.gov/sites/default/files/microsites/ostp/pcast-nitrd2013.pdf [last accessed: 24.01.2014]
Anderson12	Stuart Anderson, Massimo Felice: **Emerging Technological Risk – Underpinning the Risk of Technology Innovation** Springer-Verlag, London, UK, 2012. ISBN 978-1-4471-2142-8

Future-Proof Software-Systems

Parting Notes

http://www.mikmak.co.il/gallery?id=95608&ctgId=1

Future-Proof Software-Systems: Parting Notes

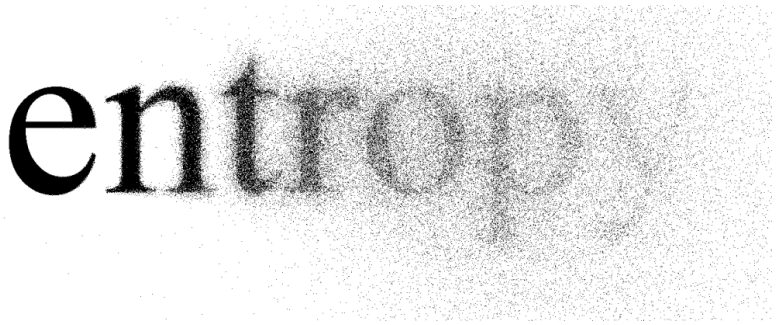
www.fotolia.com

„Usually, any *reasonable* architecture will support the desired functionality, but only a *carefully chosen* architecture will enable the desired quality properties"

George Fairbanks: **Just Enough Software Architecture**, 2010, ISBN 978-0-9846181-0-1

„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



http://larvalsubjects.wordpress.com/2013/02/18/against-holism-the-argument-from-entropy/

Future-Proof Software-Systems: Parting Notes

http://www.123rf.com

Essential Reading:

References	
Hendricksen12	Dave Hendricksen: **12 Essential Skills for Software Architects** Pearson Education, Addison-Wesley, J.J., USA, 2012. ISBN 978-0-321-71729-0
NAE04	U.S. National Academy of Engineering: **The Engineer of 2020 – *Visions of Engineering in the New Century*** National Academy Press, Washington D.C., USA, 2004. ISBN 978-0-309-09162-4. Downloadable from: http://www.nap.edu/download.php?record_id=10999 [last accessed 11.09.2013]
DeWeck11	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

Contact Details:

frank.j.furrer@bluewin.ch
Mobile: +41 (0)79 401 48 60
Phone: +41 (0)52 740 32 28

Postal Address:
Dr. Frank J. Furrer
Guldifuess 3
CH-8260 Stein am Rhein
Schweiz


The
End