

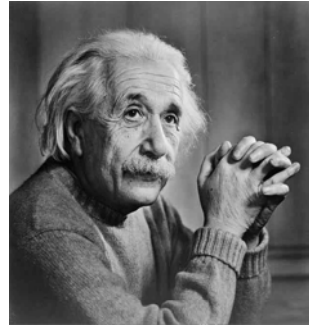


SIEMENS

Architects should be geniuses

**Experts solve
problems, geniuses
avoid them**

[Albert Einstein]



Page 3

SIEMENS

Let's enter the Road to Agile Architecture ...







Page 4

One Side of the Coin: Architecture & Design

You could design the best architecture if you knew **everything** in advance

In that case, the Waterfall model would be a perfect fit

Unfortunately, the real world is not perfect



Page 5

The Other Side - Agile Manifesto

Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan

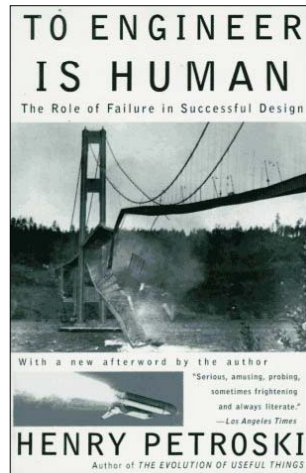
In Software Architecture embracing change is important

However, change should be planned.

Page 6

SIEMENS

Learning from failure



Failure and understanding failure is a key factor for successful design!

[Henry Petroski]

Agile Architecture requires us to detect failures early and draw the right conclusions

Page 7

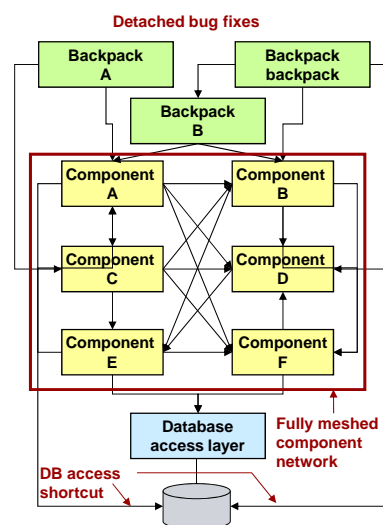
SIEMENS

War Story I: Architecture Quality is essential

A telecommunications project where the architects tried to forecast the future,

... but failed causing several ad-hoc design extensions

Ad-hoc changes inevitably cause design erosion!



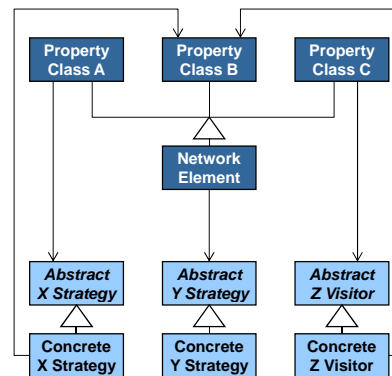
Page 8

War Story II: Change must not be unlimited or uncontrolled

SIEMENS

The architecture that was
addicted to changes,
... but suffered from lack of
qualities!

Agility ain't anarchy!
It is about KiSS



Page 9

Learning from Failure: Extreme approaches do seldomly work

SIEMENS

Purely Change-Driven Architecture
Design



Fragile architecture!

BDUF Architecture (Analysis
Paralysis)




Either unsuitable or
overengineered architecture!

Page 10

SIEMENS


Can we combine both architecture and agility?



Page 11

SIEMENS

Understanding Agile Development – Iterative-Incremental Architecture



Iterative means re-do:
a rework scheduling strategy which helps to improve the (quality of the) product

Incremental means add onto:
a staging and scheduling strategy which helps to improve the process and feature set by avoiding a big-bang Integration

Software architecture design requires continuous rework and quality improvement

Software architecture is grown piece by piece in an evolutionary approach with product quality after each end-to-end slice

Page 12

Agility == Iterative-Incremental ++

An agile process is an iterative-incremental process with additional characteristics:

- requirements are not frozen upfront,

- iterations are time-boxed,

- In addition, set of principles, values and development methods for all stakeholders.

From centralization to swarm intelligence



Page 13

How does this relate to Software Architecting

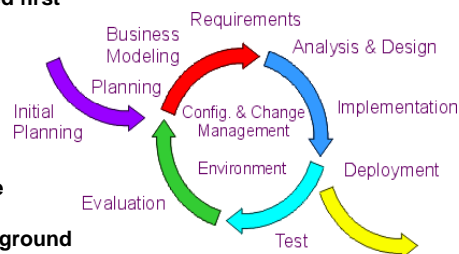
Agile principles help increase architectural quality:

An **iterative, time-boxed** approach provides continuous feedback

Risk- and requirements orientation ensures that the most important aspects of the system's realization are addressed first

A **test-driven** approach provides concrete feedback on the quality of the architecture and its realization

The goal of each iteration is to produce **product quality and less risk**, so that the next iteration can be taken on safe ground



Page 14

According to Frederik P. Brooks iterative evolutionary design is essential

SIEMENS

In the first phase requirements engineering and architecture design should go hand in hand, because

While architects don't fully understand the requirements,
Other stakeholders don't fully understand the design.

It is important to write down all assumptions about users and their uses in the beginning.

And always learn from your predecessors.



Page 15

Why should we care about Architecture, anyway

SIEMENS



**If you think good
architecture is expensive,
try bad architecture**

[Brian Foote and Joseph Yoder]

Page 16

Architecture and Design (Len Bass)

Design is a continuous activity of making decisions

beginning with a collection of decisions that have broad system wide scope

and moving to a collection of decisions that have very narrow scope



A decision is **architectural** if it has one or more of the following properties:

it has **system wide impact**

it affects the achievement of a quality attribute important to the system

- „Architecture is about the important things“ [Martin Fowler]
- „Architecture is about everything costly to change“ [Grady Booch]

Page 17

Strategic and Tactical Design

Strategic design focuses on **global system scope**

At the beginning consider only **strategic requirements**, i.e., requirements with systemic and strategic impact:

All functional requirements

All operational and infrastructural requirements

Tactical Design encompasses all local design decisions with non-systemic impact

Tactical requirements are requirements with local scope such as developmental requirements (e.g., modifiability)



Page 18

The Art of Architecture



There are two ways of constructing a software design: One way is to make it so simple that there are obviously no deficiencies, and the other way is to make it so complicated that there are no obvious deficiencies. The first method is far more difficult

[C.A.R Hoare]

Page 19

Some Observations and Experiences

The best designs are not invented by committees but by (a small number of) individuals

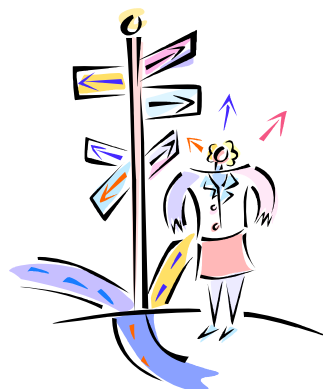
=> Small team of architects with architecture owner

Architecture is about communication

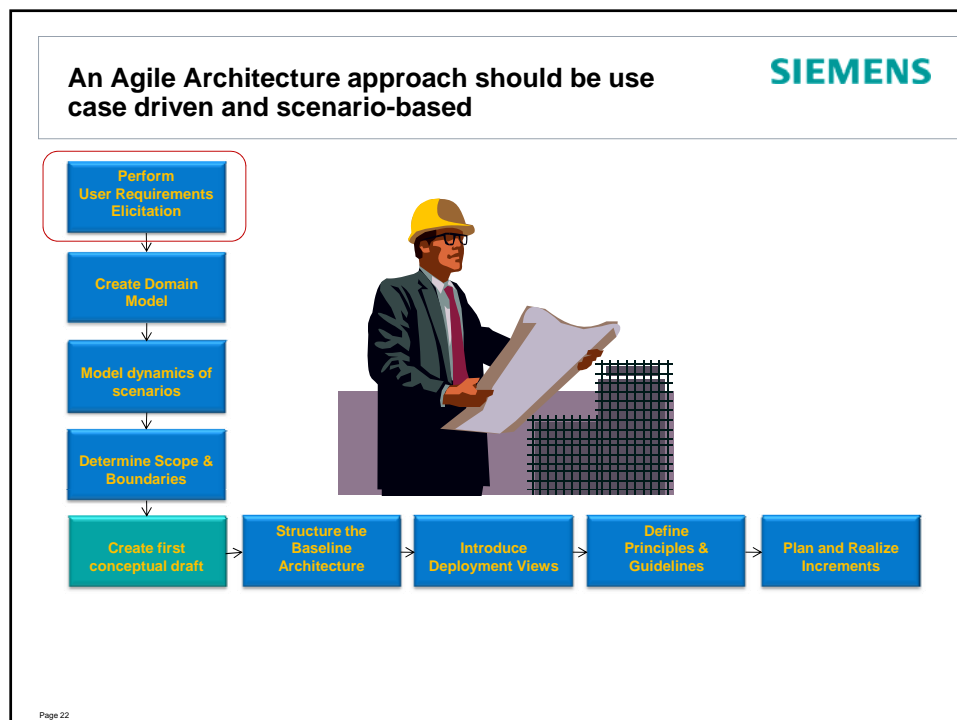
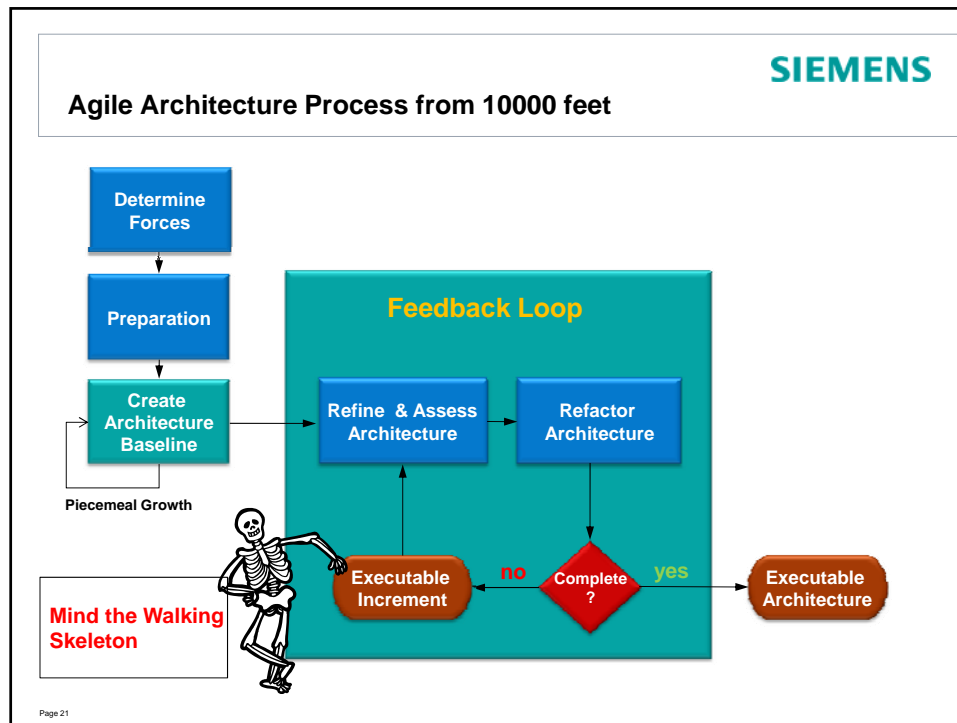
=> Architecture and design require collaboration of stakeholders

Change is the rule not the exception

=> Architecture must embrace change in a planned way



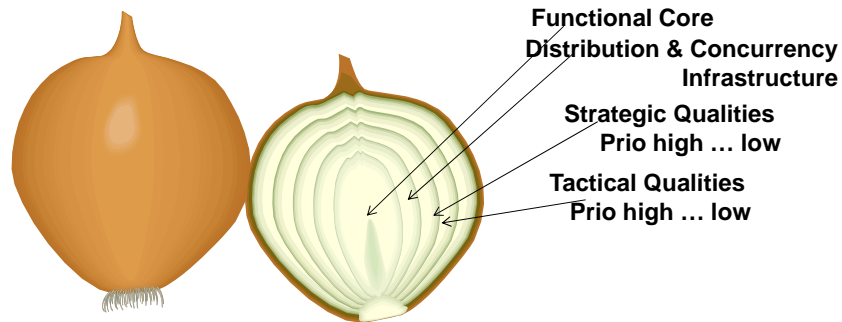
Page 20



Another Perspective - Onion Model

Architecture design follows the structure of an onion:

- start with the inner core
- incrementally continue with outer layers



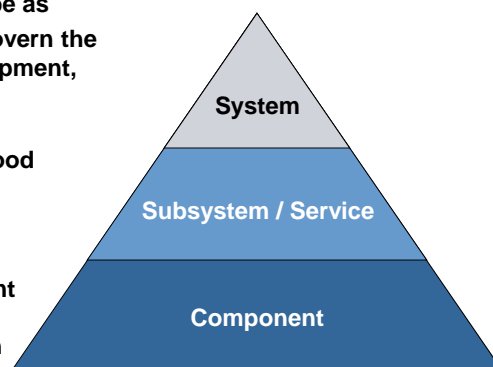
Page 23

How Deep do we go? - The Pyramid Model

The baseline architecture must be as complete as necessary to govern the subsequent software development, and, as simple as possible to be communicated und understood

Three levels of detail to limit depth

A focus on architecturally significant requirements and corresponding architecture views to limit breadth



Page 24

SIEMENS

The Not Invented Here Syndrome

Human beings, who are almost unique in having the ability to learn from the experience of others, are also remarkable for their apparent disinclination to do so.



[Douglas Adams. 1952-2001.
Last Chance to See]

Page 25

SIEMENS

One further Ingredient: Re-Use

Re-use helps increase quality and abides to the KiSS principle

Efficiency and effectiveness are more important than originality

Proven expertise instead of inventing new solutions to avoid accidental complexity

Mind the different levels of re-use:

Components, Patterns, Reference Architectures, ...



Page 26

Yet Another Ingredient: Domain Model as Common Language

SIEMENS

Goal:

Provide knowledge about problem domain and proven solutions

Benefits:

Provides common language for all stakeholders

Serves as a base for understanding functional requirements

May even evolve into a DSL supporting MDSD



Page 27

Panta rhei - Evolutionary Design embraces Systematic Change

SIEMENS



There is nothing permanent except change



[Heraclitus, 535–475 BC]

Page 28

SIEMENS

Design erosion is the root of evil

In the lifecycle of a software system changes are the rule and not the exception => all projects become brown field projects

Unsystematic approaches ("workarounds") cure the symptom but not the problem

After applying several workarounds, software systems often suffer from design erosion

Such systems are doomed to fail (negative impact on operational & developmental properties)

Page 29

SIEMENS

How we know we must improve

Lack of Internal or External Quality

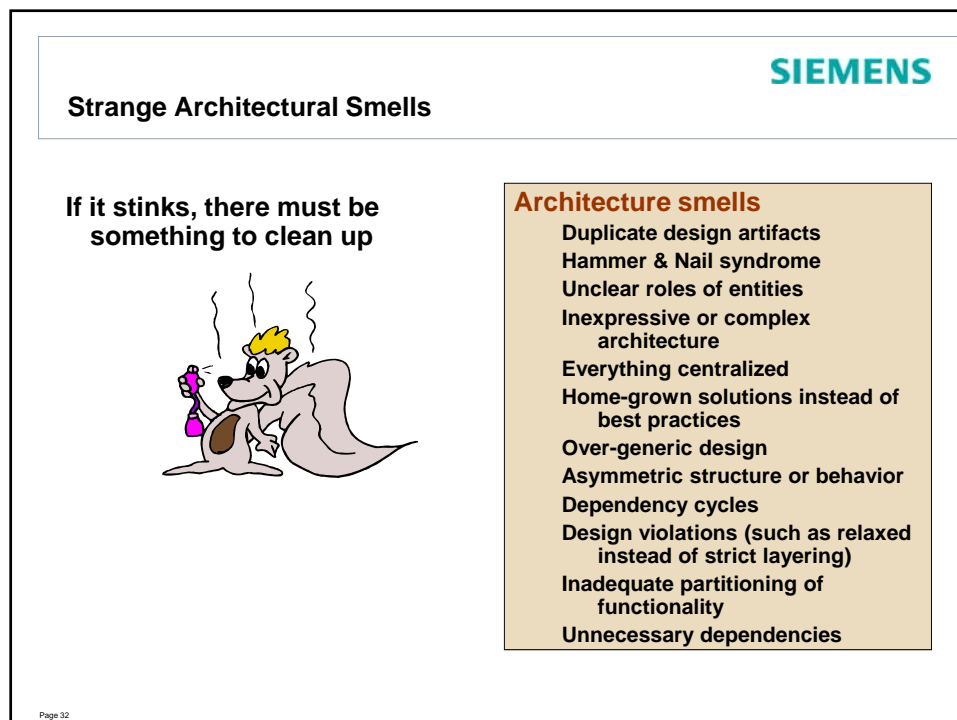
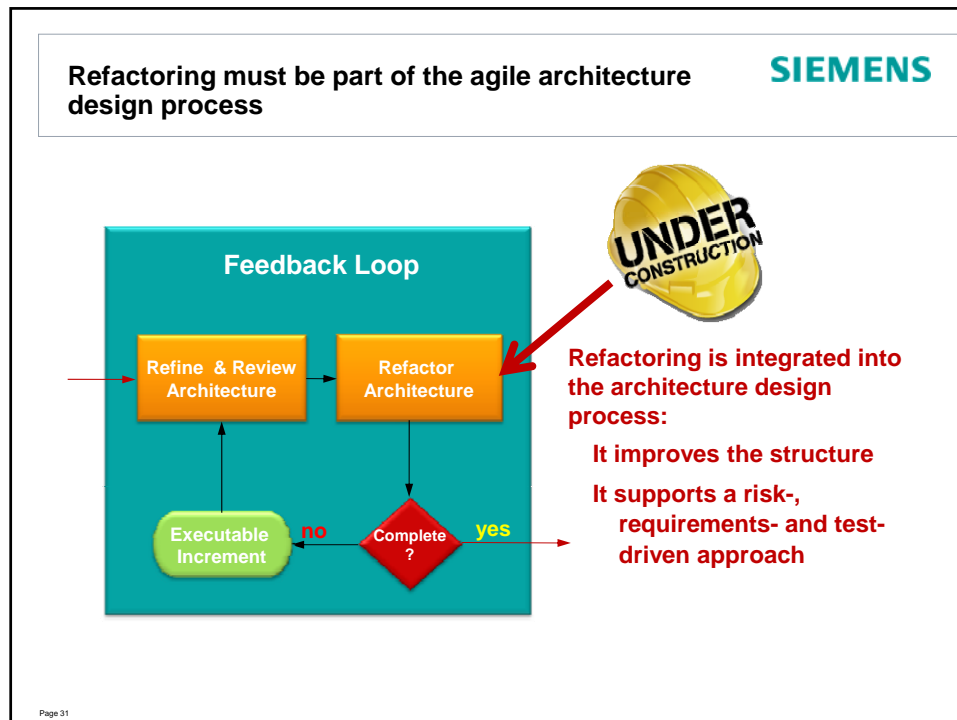
Quality Attributes, and,

Structural quality indicators which include

- Economy**
- Visibility**
- Spacing**
- Symmetry**
- Emergence**

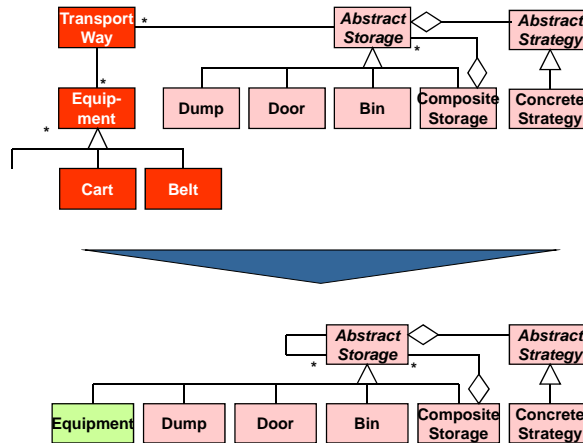
Consequently, the goal of agile architecture evolution and improvement is to achieve or meet such qualities

Page 30



Example of Architecture Refactoring Problem

A true story: In this example architects introduced Transport Way as an additional abstraction. But can't we consider transport ways as just as another kind of storage? As a consequence the unnecessary abstraction was removed, leading to a simpler and cleaner design.



Page 33

Dealing with Technical Debt

Technical debt is a state of lack of quality

Sometimes, we need to temporarily accept it, e.g., refactoring shortly before next release would be too risky

However, technical debt in strategic parts might be critical



Page 34

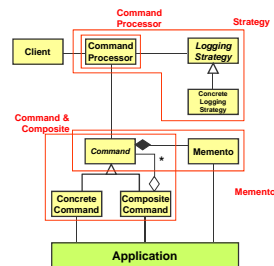
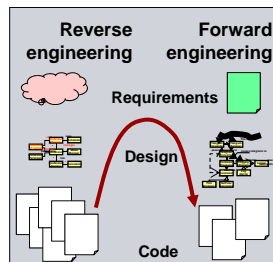
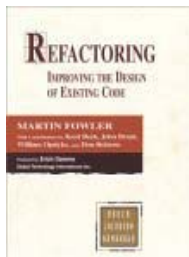
Refactoring, reengineering, and rewriting

Refactoring, reengineering, and rewriting are complementary approaches to sustain architecture and code quality

Start with refactoring - it is cheap and (mostly) under the radar

Consider reengineering when refactoring does not help - but it is expensive

Consider rewriting when reengineering does not help - but it is expensive and often risky



Page 35Page 35

Software Architect's Dilemma

Life must be understood backwards; but ... it must be lived forward

[Søren Aabye Kierkegaard, Danish philosopher and theologian, 1813-1855]



Page 36

Architecture Assessments help finding the Bad Smells

SIEMENS

Quantitative Architecture Reviews

- Code quality assessment
- Simulations
- Prototypes



Qualitative Architecture Reviews

- Scenario-based approaches
- Experience-based approaches



*An Architecture Assessment or Review should **not** be considered an afterthought.*

It is a means to check a system regularly and find problems early

Page 37

Quantitative review

SIEMENS

Benefits

- Yields "hard" results
- Quantifiable, objective means for selecting alternatives
- Experiments by altering the parameters relatively easy

Liabilities

- Focus on only a couple of concerns or system parts
- Works only if data is interpreted correctly
- Effect on quality attributes other than the focus is unknown
- Probably costly



Page 38

Qualitative review

Benefits

- Involves all relevant stakeholders
- Overview of the whole system
- Improve understanding for all participants
- Relatively cheap to execute
- Can be conducted as soon as high level architecture design is available



Liabilities

- Relies mainly on documents and statements from personally involved stakeholders
- Experienced reviewers required
- No "hard facts" (unless supported by quantitative assessments)

Page 39

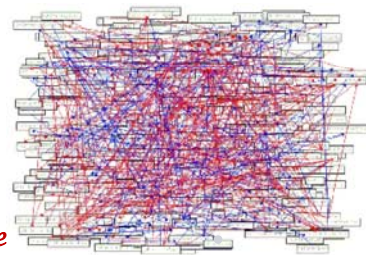
Visualization Tools help keeping the system in good Shape

In many projects the responsibility for internal code and design quality is not well defined

The software architect has to ensure that the required CQM activities are established

The software architect should be the protector of the quality of the software system!

Use Visualization Tools at least in larger code bases



By the way:
this is a real
system

Page 40

Architects & Requirements – Problem: Do we know the *right* requirements?

SIEMENS

A program which perfectly meets a lousy specification is a lousy program

[Cem Kaner, Software Engineering Professor and Consumer Advocate]



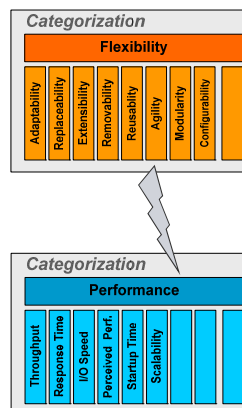
Page 41

=> Requirements must have high Quality

SIEMENS

Quality of Requirements determines Quality of Software Architecture

Cohesive
Complete
Consistent
Correct
Current
Externally Observable
Feasible
Unambiguous
Mandatory
Verifiable



Page 42

SIEMENS

Knowing the expectations is essential

At least at project begin,

- Architects don't understand requirements very well
- Customers tell what they want, not what they need
- Architects may even not know the implicit requirements

Hence,

- Keep in touch with Customers
- Apply KANO Analysis
- Understand Business Goals

In first phase, develop Design and Requirements in parallel

Page 43

SIEMENS

Requirements Traceability is essential for planned change

Traceability

is detailed by

can be handled by

Software Architecture

is designed with

Performance

SIEMENS

Testing as a never ending story

**Testing is an infinite process of
comparing the invisible to the
ambiguous in order to avoid
the unthinkable happening to
the anonymous**

[James Bach, Test Guru]



Page 45

SIEMENS

Architecture Testing should be part of the Agile Mindset

Test-Driven Design for Agile
Architecture Development
comprises

mainly Integration Testing,
System Testing, Acceptance
Testing, Unit Testing,
Architecture, Design, and Code
Reviews,

Test-First Design,
Design for Testability,

...



Source: Wikipedia

Page 46

Mind all Risks and conduct a Risk Analysis early

Approach for risk analysis according to Christine Hofmeister ("Applied Software Architecture"):

Description of risk: e.g., dependence on persistence layer

Influential factors that lead to this risk: e.g., requirement to decouple business from persistence layer, not enough technology skills in team

Solution approach: e.g., introduce data access layer

Possible strategies: e.g., give subproject to external company, use open source solution, use platform-specific solution

Related topics and strategies: e.g., decoupling business logic from other backend layers



Page 47

Risk Based Test Strategy

- Evaluate risks:
 - What is the risk
 - Which part of the system does it affect
 - How likely is the risk
 - How big is the possible damage
 - What priority does the risk have
 - Can it be tested? If yes, when and using what method
 - Can the test be automated
 - Which resources (budget, time, - --) are required



Page 48

Communication is essential

Software Development is a collaborative game

[Alistair Cockburn]

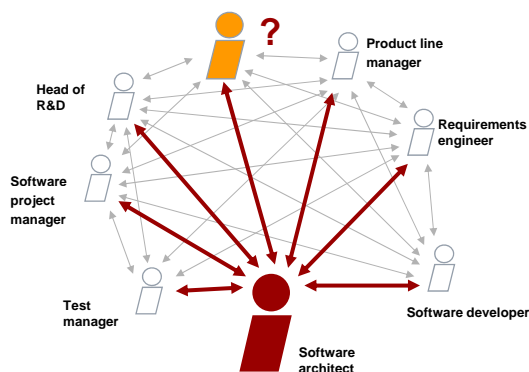


Page 49

Frequent change requires agile communication

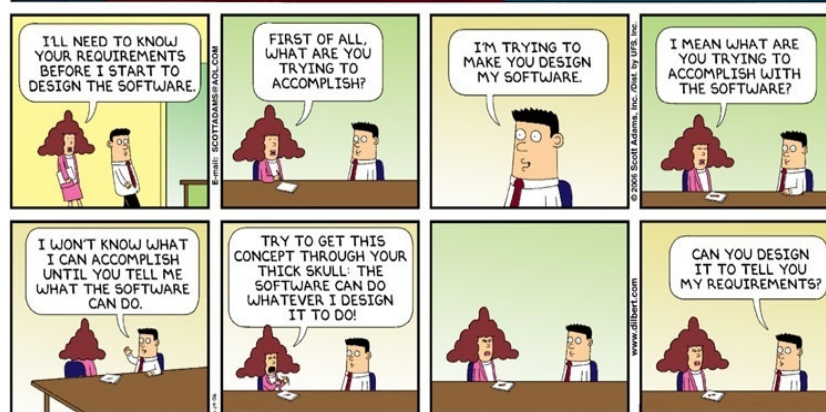
Agile communication and interaction with other roles are essential in an evolutionary ecosystem

- Enforcing the architecture by coaching and mentoring developers
- Getting buy-in from management
- Obtaining, clarifying, prioritizing requirements with product line managers or customers (incl. iteration planning)
- Ensuring quality control with test managers
- Obtaining (early) feedback from customers
- Planning required resources with project managers



Page 50

Communication with stakeholders can be surprisingly challenging



© Scott Adams, Inc./Dist. by UFS, Inc.

Page 51

Conclusions I: Agile Manifesto and Architecture – Mission Accomplished

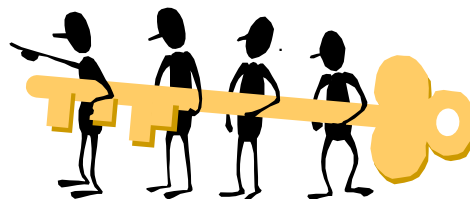
SIEMENS

Individuals and interactions over processes and tools ✓

Working software over comprehensive documentation ✓

Customer collaboration over contract negotiation ✓

Responding to change over following a plan ✓



Page 52

Conclusions II: Agility and Architecture

Are no contradiction, but two sides of the same coin

Agile design is about embracing change

Agile architecture is about planned change

Thus, a sustainable architecture baseline is the precondition for successful agile development



Page 53

Conclusions III: Architect's Framework for Agile Architecture

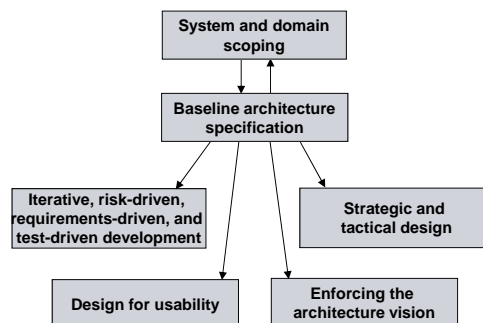
The mindset, activities, practices, methods, and technologies for defining and realizing software architectures form a best practice framework for software architects to ...

Specify and implement a software architecture systematically and in a timely fashion

Check and ensure the appropriate architectural quality

Respond to changes of all kinds, such as changing requirements and priorities

Deal with problems that arise during the definition and realization of the software architecture



As taught in the
Siemens Senior Software Architect Education Program

Page 54

SIEMENS

A departing thought

**Each problem that I solved
became a rule which served
afterwards to solve other
problems.**

[René Descartes, 1596–1650, in "Discours
de la Methode"]



**Remember Einstein's Quote: Geniuses
avoid problems. So, be a genius!**

Page 55

SIEMENS

Leaving the Road to Agile Architecture



Page 56