Every conflict contains an inherent opportunity for positive change

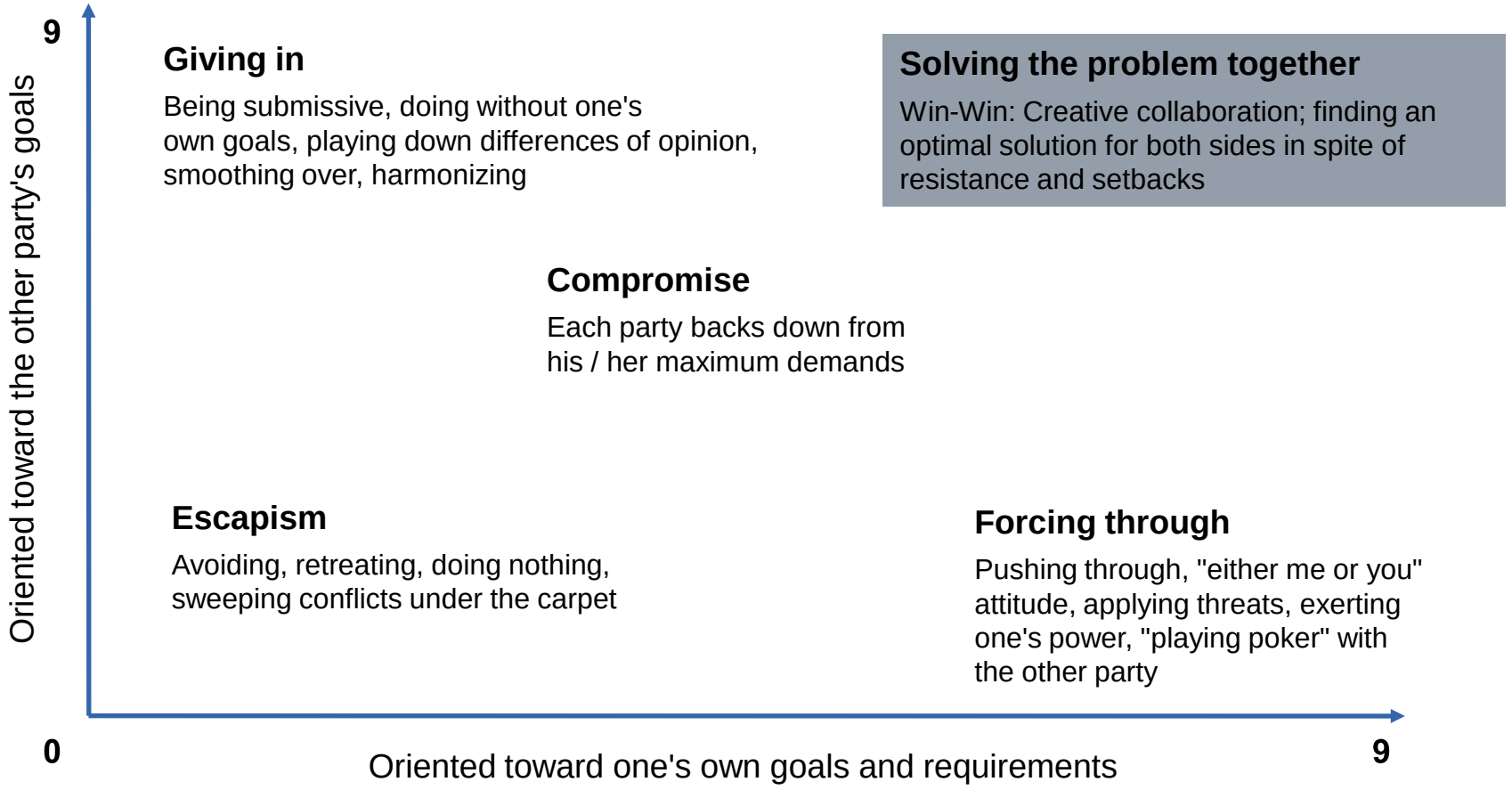
SIEMENS

Conflict:

A conflict is a clash between opposing forces.

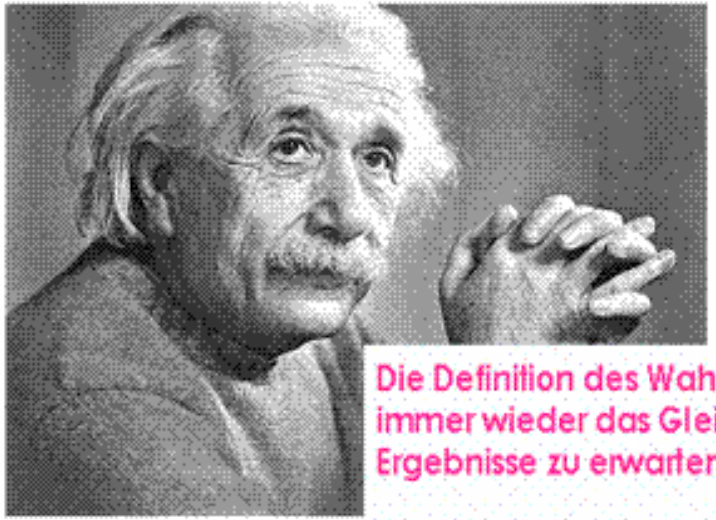
FRIEDRICH GLASL: "With conflicts, the real problem is not the existence of differences but rather the way in which these differences are dealt with."

In the long run the win-win strategy is the most promising one



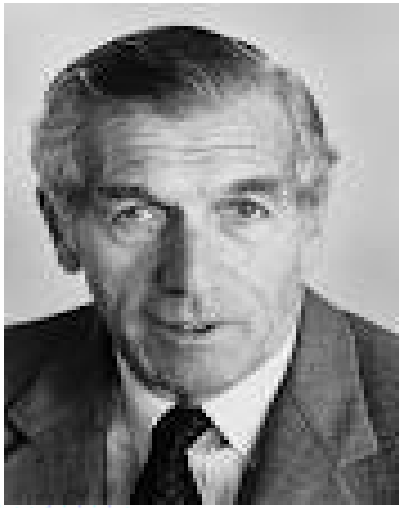
When the Solution becomes the Problem … try something else!

SIEMENS



Die Definition des Wahnsinns ist, immer wieder das Gleiche zu tun und andere Ergebnisse zu erwarten."

„The definition of madness is working in uniformed ways and expecting different results."
ALBERT EINSTEIN



If something doesn't work …

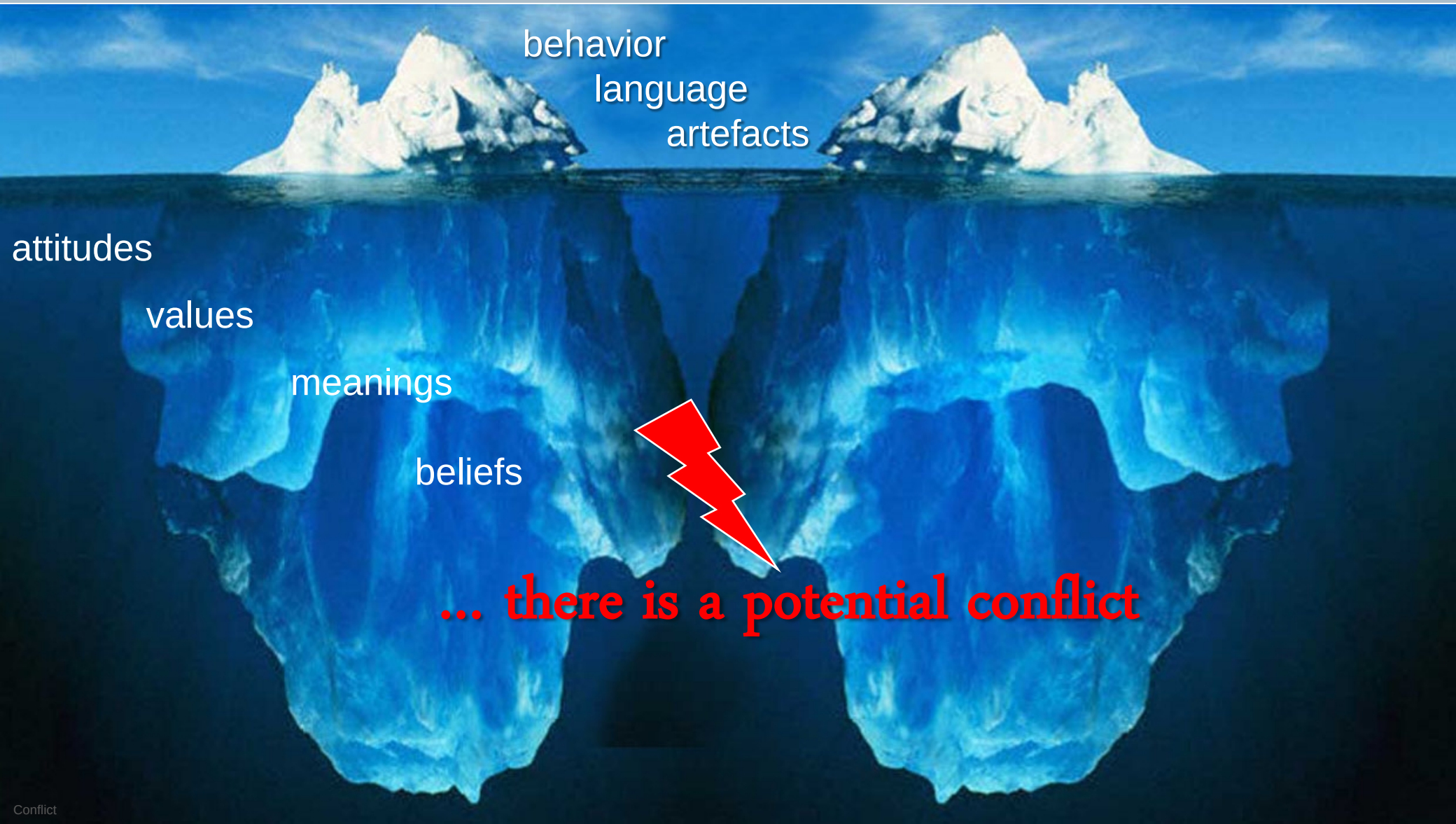
do *not* more of the same,

do something else.

PAUL WATZLAWICK

SIEMENS

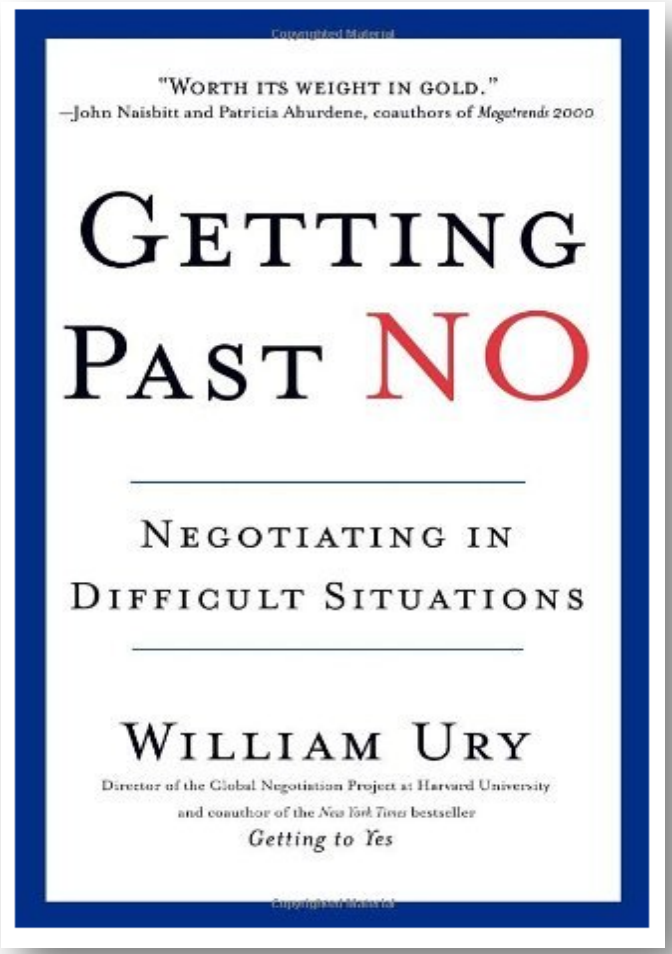
When two cultures meet …



Getting past “No”:
Five steps to master difficult conversations

SIEMENS

1. Go to the balcony
2. Step to the other side
3. Reframe the game
4. Build the golden bridge
5. Make it hard to say "no"



1 Avoid spontaneous "natural" reactions – go to the balcony and analyze the game

SIEMENS

1. **Go to the balcony**
2. **Step to their side**
3. **Reframe the game**
4. **Build them a golden bridge**
5. **Use power to educate**

- Spontaneous "natural" reactions can be dangerous
- View the situation as "from the balcony"
- Be clear about your interests and your fallback solution
- Analyze the game
- Take time to think

Source: W. Ury, SLE

2 Create a favorable climate by stepping to their side

SIEMENS

1. Go to the balcony
2. **Step to their side**
3. Reframe the game
4. Build them a golden bridge
5. Use power to educate

- Listen actively: release your agenda
- Acknowledge their points
- Agree wherever you can
- Acknowledge the person
- Express your views in an assertive way
- Create a favorable climate

Source: W. Ury, SLE

3 Use probing to reframe the situation

1. Go to the balcony
2. Step to their side
3. **Reframe the game**
4. Build them a golden bridge
5. Use power to educate

- Change the frame
- Ask problem solving questions (probing)
  - Why / why not?
  - What if?
  - What is your advice?
- Reframe tactics
  - Go around "stone walls"
  - Deflect attacks
  - Expose tricks
- Negotiate about the rules of the game

Source: W. Ury, SLE

4 Start from their point of view in order to guide him / her towards agreement

1. Go to the balcony
2. Step to their side
3. Reframe the game
4. **Build them a golden bridge**
5. Use power to educate

- Explore possible obstacles to agreement (fear of losing face, …)
- Start from their point of view in order to guide them towards agreement
- Involve them to craft an agreement together
- Look for unmet interests and try to satisfy them
- Go slow to go fast

Source: W. Ury, SLE

5 Probe for the consequences of failing to reach agreement

SIEMENS

1. Go to the balcony
2. Step to their side
3. Reframe the game
4. Build them a golden bridge
5. **Use power to educate**

- Make sure that they see the full reality
- Probe for the consequences of failing to reach agreement
  ("What do you think will happen if we don't agree?")
- Use your power, but defuse the reaction
  (Use third parties, build coalitions, ...)
- Show them the way out: the golden bridge
- Aim for mutual satisfaction, not your victory

Source: W. Ury, SLE

Architectural views & documentation – “Stakeholder Stories”

User’s Perspective

Requirements Management

- Existing Drawbacks
- Application Framework
- Root Causes
- Behavior Requirements
- Structure Requirements
- Implementation Framework
- Non-functional Requirements
- …

Conceptual and Context views
Operation and Behavior
Maintenance
Support Systems

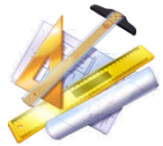
Designer’s Perspective

Systems Architecting / Design

- Solution Model
  - Structure
  - Behavior
  - Interfaces
- Deployment Case Model
- Solution understanding (simulation,..)
- Test Strategy
- Procedural Model
- Risk Assessment
- …

Logical Views
Architecture View
Scope
Function

Builder’s Perspective



Project Management

- Time Framework
- “Quality” Deliverables
- Granted Resources
- Accepted Risk level
- Implementation Plan(s)
- Organization Structure
- …

Physical views
Implementation
Manufacturing
Deployment

SIEMENS

The 4+1 View explained

Note: A view is more than a set of diagrams.
It needs graphical representations and textual explanations

User view

- The ***Use Case View*** explains all possible scenarios users expect from the system

Functional aspects:

- The ***Logical View*** shows how the functionality defined in the use cases is modeled, while the
- ***Development / Implementation View*** shows how the functionality is implemented (using source code, libraries, executables, documents, ...)

Non-functional aspects:

- The ***Process View*** illustrates how artifacts will be executed in terms of concurrency, scalability, synchronization
- The ***Deployment View*** maps these software artifacts to concrete hardware entities and shows the distribution of functionality

Each view can be modeled using diagrams (e.g., UML), but it is more important what is in the view (semantics) than how to express it syntactically

Likewise, it is important that the stakeholders targeted have knowledge of what the views mean

SIEMENS

Test architect's stakes in architecture documentation

TeA is an important stakeholder in the software / system architecture documentation
You need to ***read it***
You need to ***understand it***
You need to ***review it***
Most decisions of the test architecture are rooted in the software / system architecture

Test system requires product quality: ***you need to document***
Architecturally significant requirements
Decisions and **rationale**
Architecture of the test system, i.e. the **test architecture**

Other methods used for documentation, e.g.
Additional views beyond *Kruchten 4+1*
Modeling: UML, SysML, …

SIEMENS

Exercise

Sketch the system architecture,
i.e. the architecture of your system under test

- Split up in pairs (***A***, ***B***) of participants with similar context:
  - Type system: Software only / System / Solution
  - Domain: Industry, Energy, Healthcare,
  - ...
- ***A***: Explain to your partner ***B*** the most important aspects of your system architecture
- ***B***: As a test architect for this system, ensure that **A** covers all information that is important for your test system
- ***Both***: document in an appropriate way on flipchart
  - How do you document these aspects? Why?
  - Which views do you select? Why?
  - Which diagrams are most important? Why?

Exercise:
System Architecture
Pairs of participants, 20min

ISO/IEC/IEEE 29119-3: Test documentation Overview

SIEMENS

Organizational test process

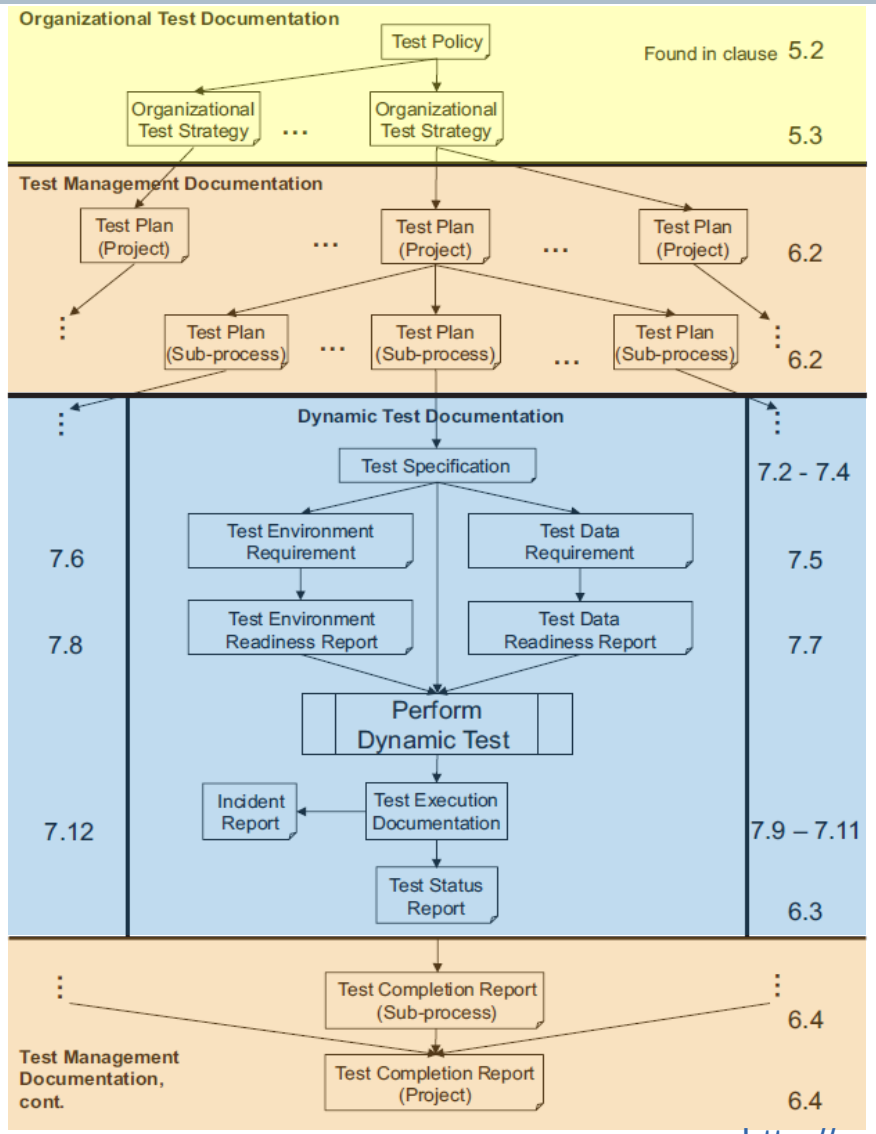
- Test policy
- Organizational test strategy

Test management processes

- Test plan
- Test completion report

Dynamic test processes

- Test plan
- Test specification
- Test data requirements
- Test data readiness report
- Test environment requirements
- Test environment readiness report
- Test execution log
- Incident report
- Test status report

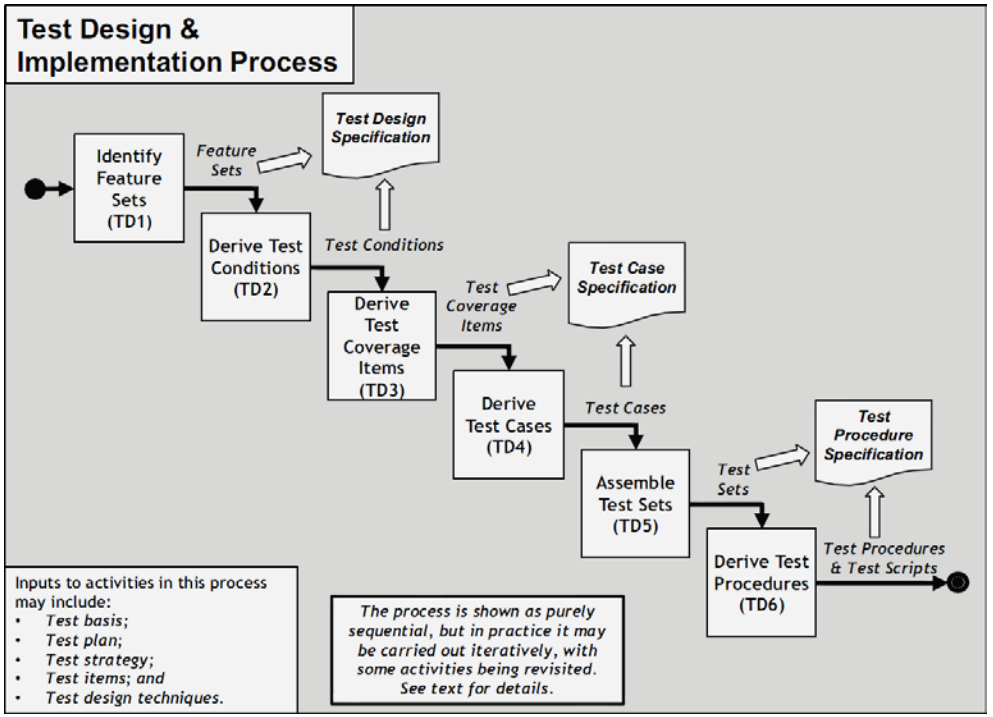


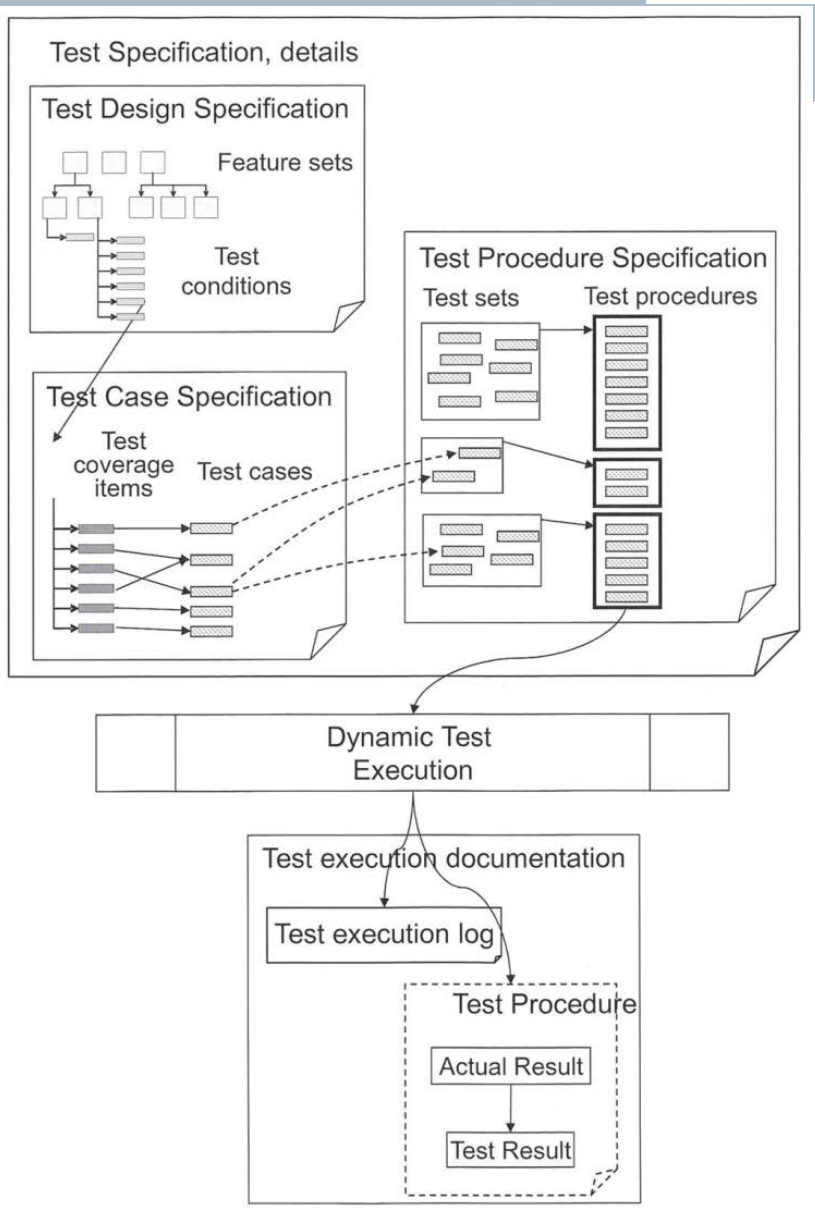
http://www.iso.org/

ISO/IEC/IEEE 29119-3: Test documentation Overview

SIEMENS

The hierarchy between the documents produced in completing the ***Test Design and Implementation Process*** outlined in ISO/IEC/IEEE 29119-2





http://www.iso.org/

Scope of *Internal Quality*

As a Test Architect

You have one Role – but you are wearing two Hats

Remember WS1

SIEMENS

Test Expert

for the system under test (SUT)

- Design the test approach
- Apply innovative test technologies
- Drive the quality of the SUT

Internal Quality of the SUT (production code)

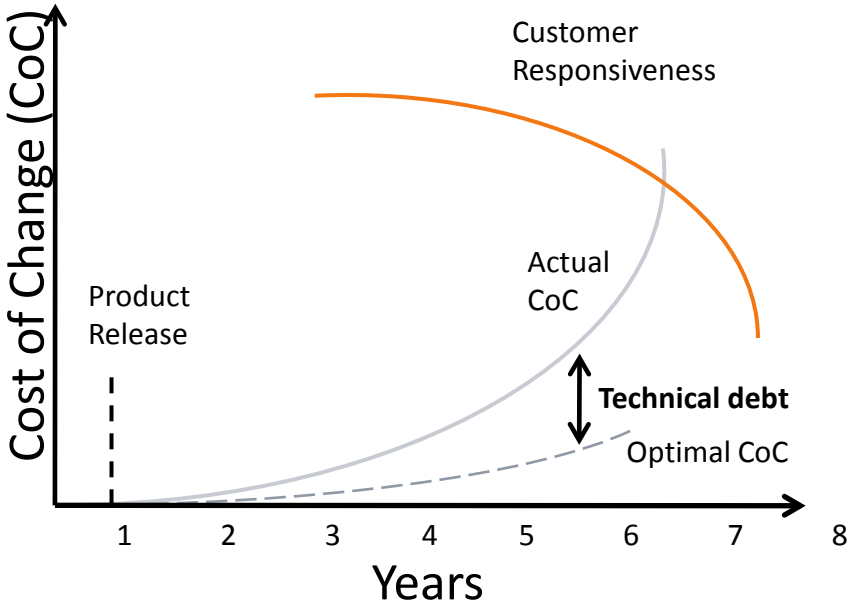
Software / System Architect

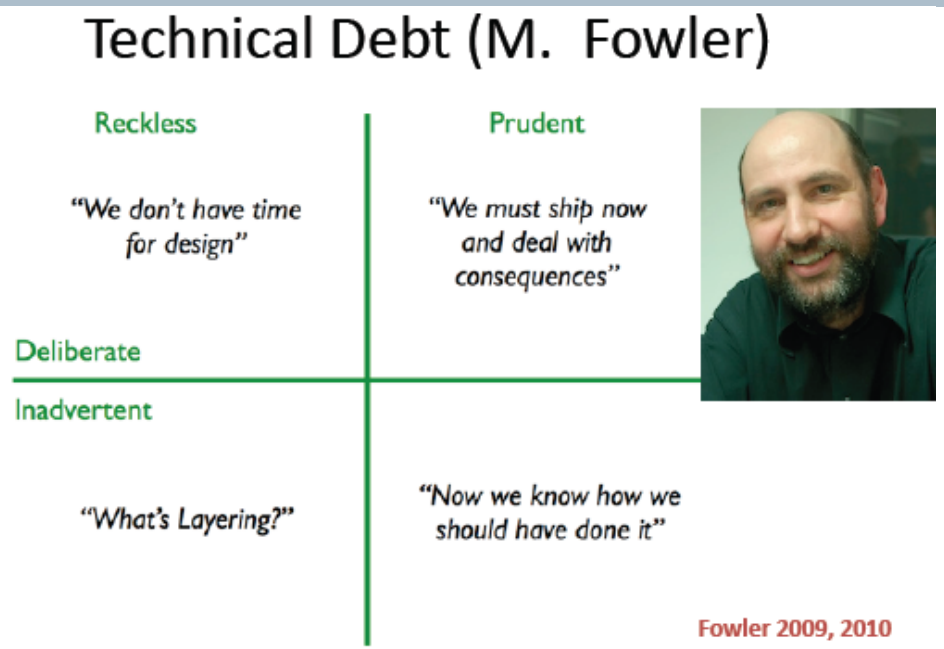
for the test system

- Design and realize the test architecture
- Apply innovative software technologies
- Drive the quality of the test system

Internal Quality of the test system (test code)

What is technical debt?



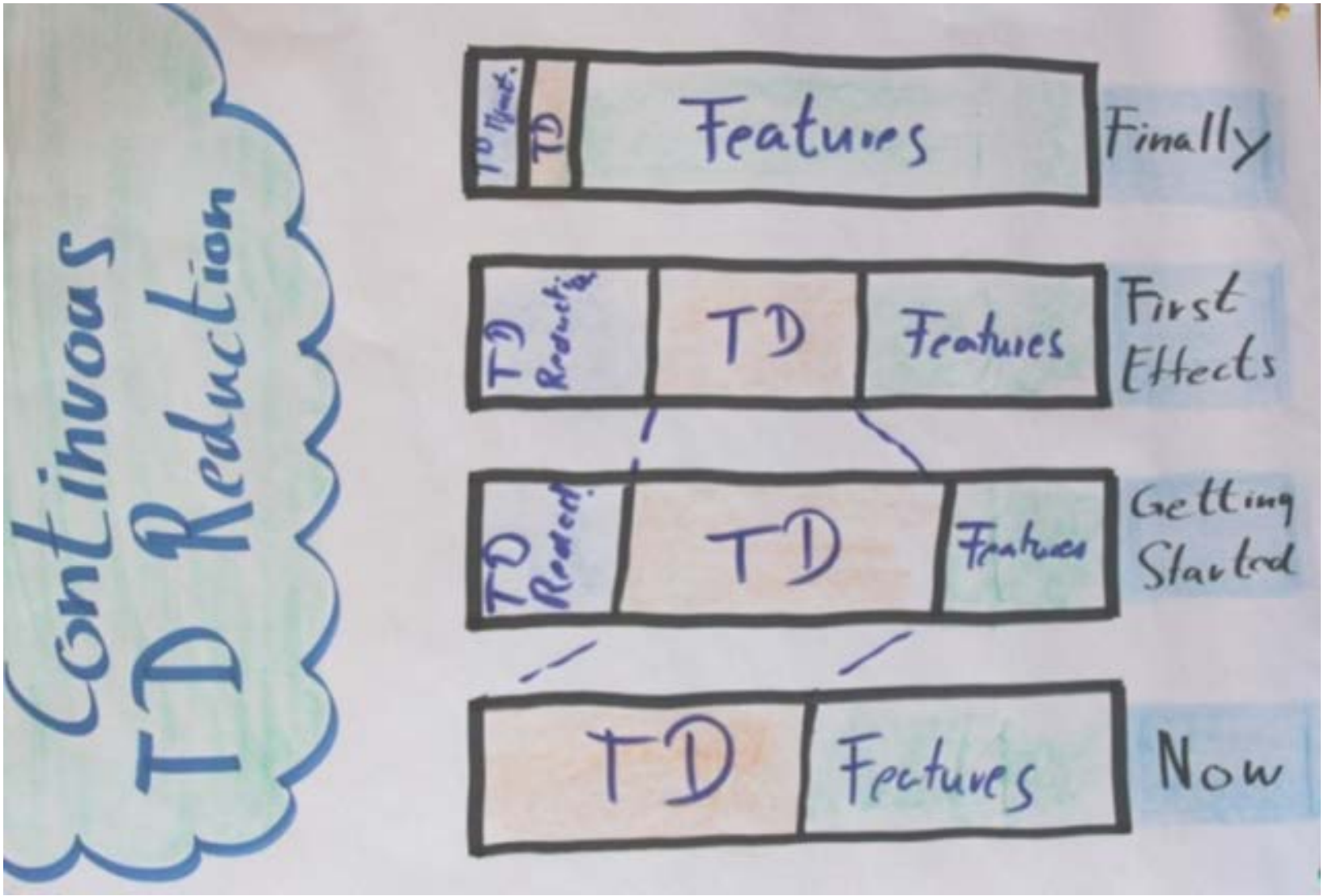


Technical debt (TD) accumulates for several reasons:

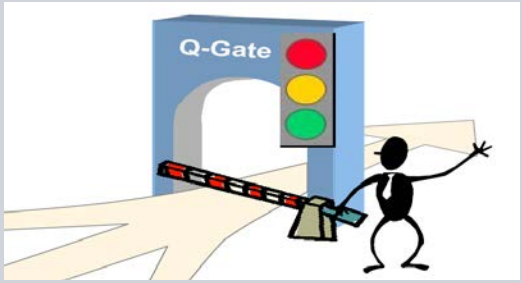
- Daily tradeoffs between quick value, quality and project constraints (cost, schedule)
- Maintenance postponed in favor of important „business value add“ projects
- Deliberate or inadvertent ignorance of (internal) quality
- Lack of maintenance for additional functionality and complexity
- Aging effects of continuously growing or changing engineering artifacts
- Postponed upgrades of the underlying platform infrastructure

SIEMENS

Continuous technical debt reduction

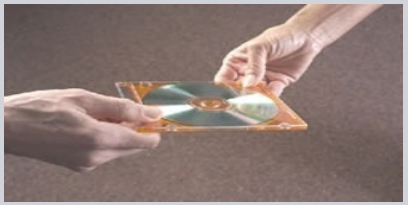


(Test) Code quality assessments help to make the internal quality transparent



Go / no-go decisions
at important milestones

Strategic decisions
like
platform selection

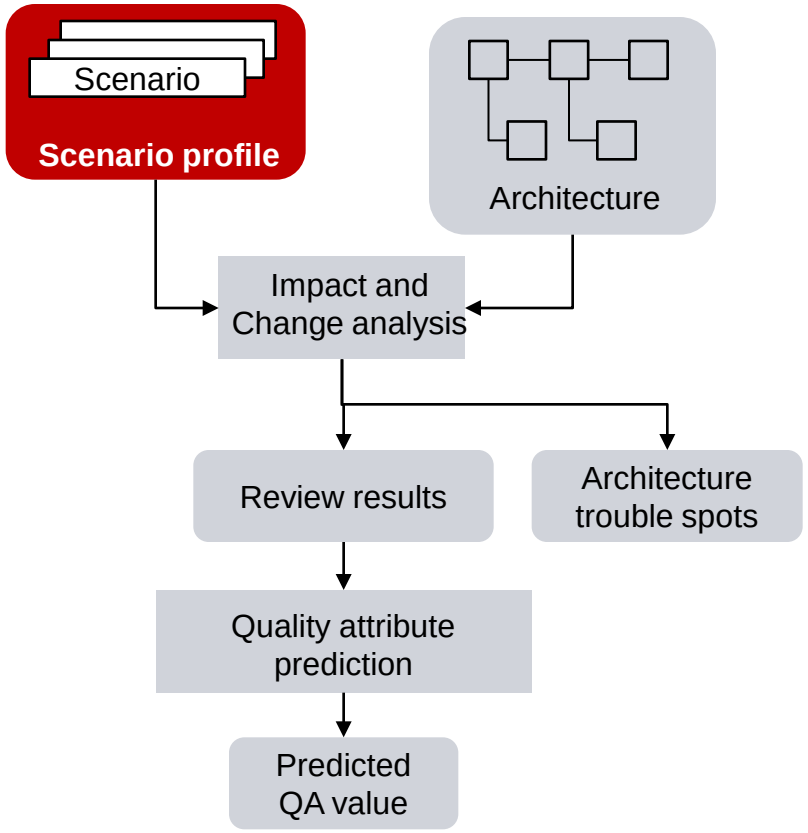


Making
software aging
visible

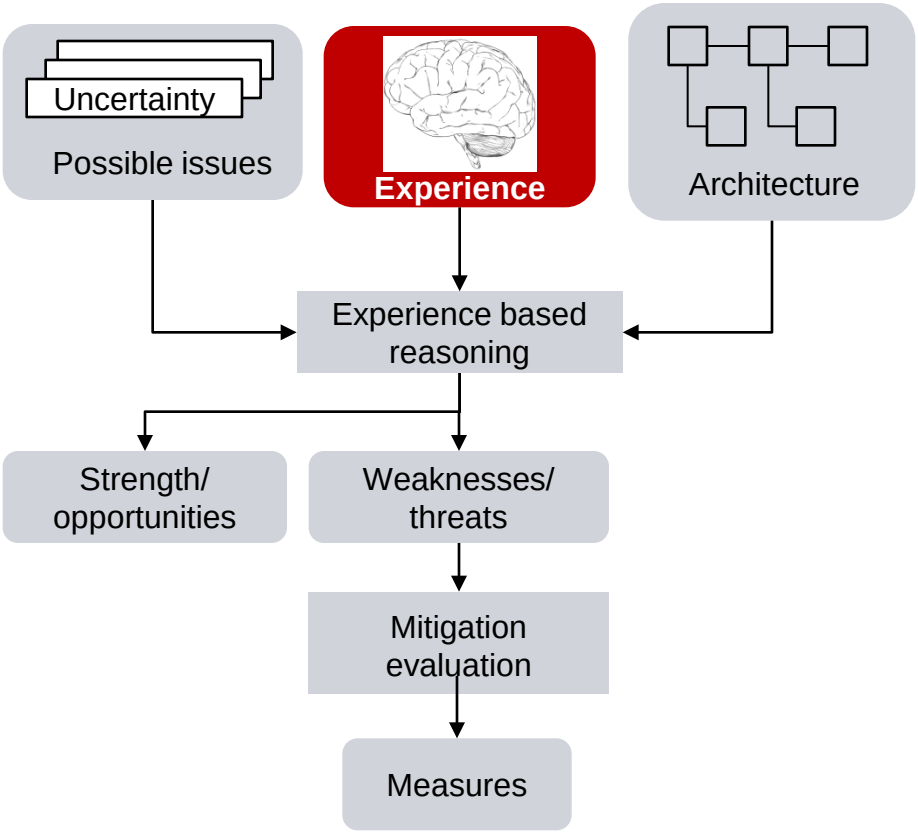
Identifying
"servicing" needs and
support maintenance

Types of qualitative reviews

Scenario-based



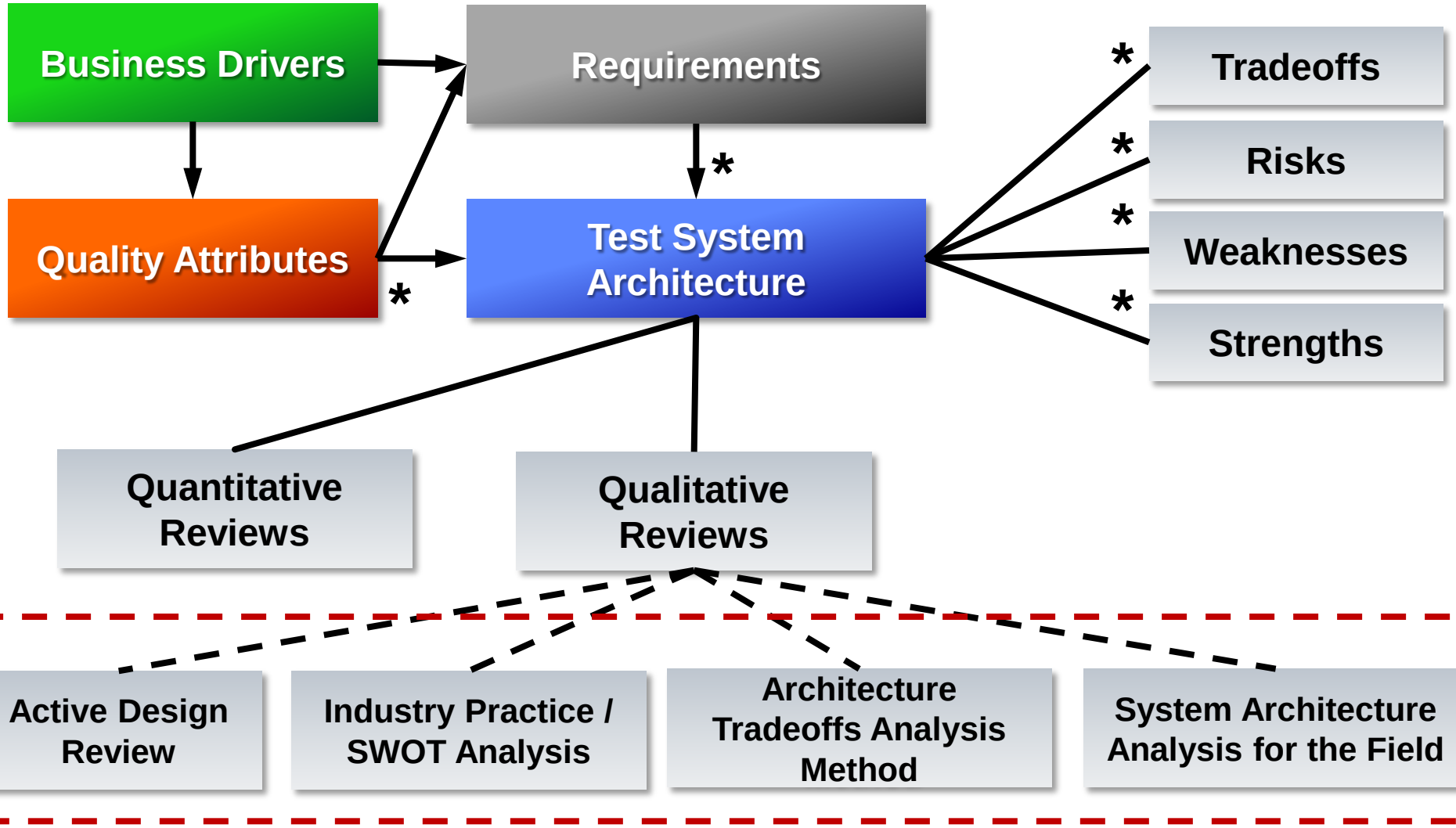
Experience-based



Bosch, 2000

SIEMENS

Qualitative Review Toolbox



SIEMENS

Typical phases of reviews

1. ***Preparation*** – reviewer and customer – discussion
   - Clarification of concrete, project-specific review goals and questions
2. ***Collection*** – reviewer and interviewees – interview
   - Interviews with architects, developers and stakeholders
   - Analysis of documents and source code
   - Demonstration of the running software, …
3. ***Elaboration*** – reviewer – documentation and analysis
   - SWOT analysis of the software architecture: Strengths, weaknesses, opportunities, threats
   - Measures for dealing with weaknesses and threats
4. ***Consolidation*** – reviewer and customer – clarification
   - Clarification and consolidation of the final report with key stakeholders
5. ***Presentation*** – reviewer – presentation
   - Presentation of review results to stakeholders;
     constructive view on the potential for improvement
6. ***Optional: workshop*** – all stakeholders – discussion
   - Joint discussion of results and measures with stakeholders;
     development of concrete improvement scenarios

Industry Practice review method

Purpose: Confirm strength, find challenges and identify measures

Reviewers are experienced architects
System description by project externals

- Elaboration of the key requirements
- Elaboration of the key design elements

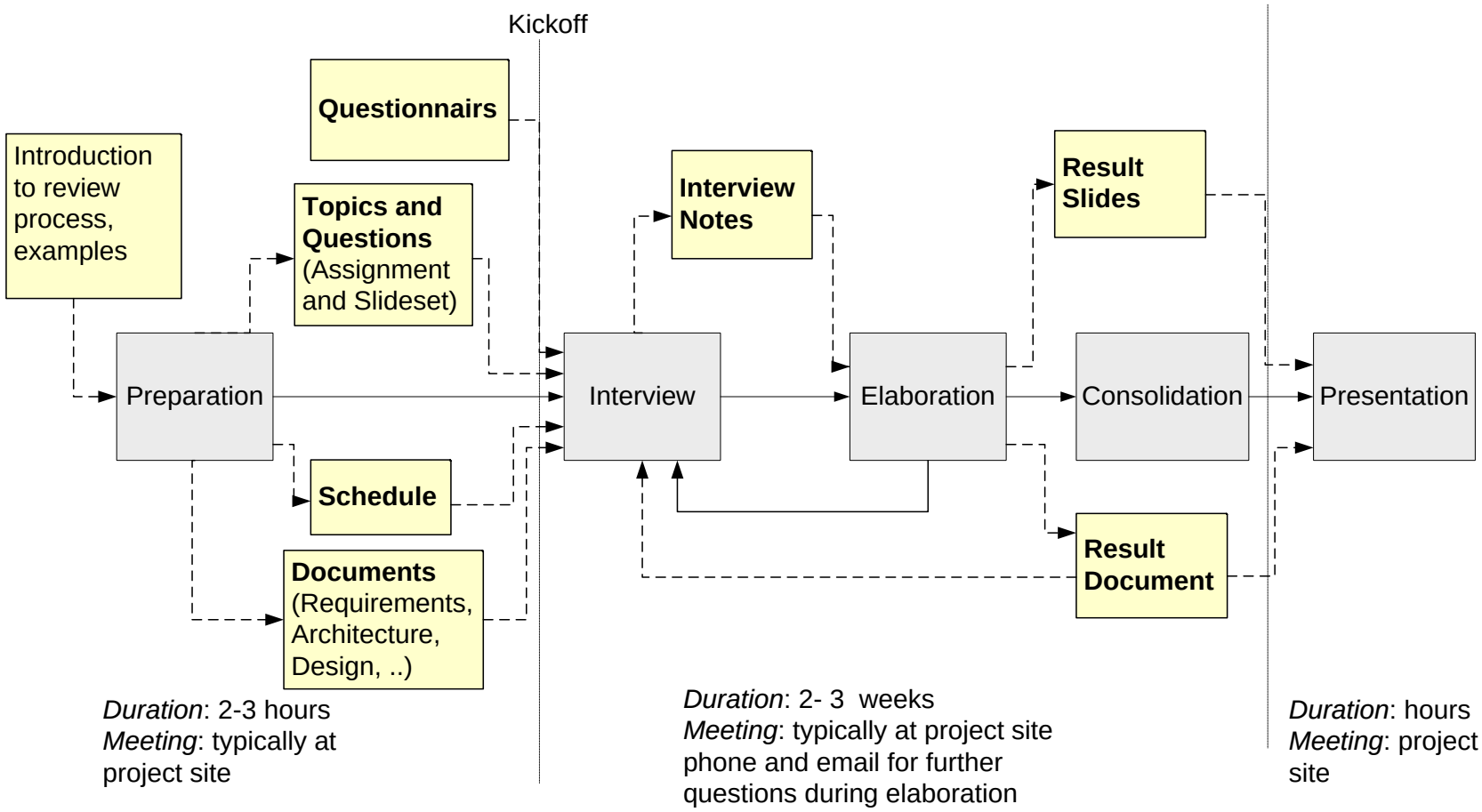
Analysis and documentation of strengths, weaknesses, opportunities, and threats

Effort: Regular review: Reviewer team 20–60 person days, project team 8–16 p.-days
Flash review: Review team 2–3 days, project team 2–3 hours

Results: Detailed report including architecture description, SWOT analysis, measures

Benefits: Rating of a (test) system architecture regarding compliance to its requirements,
dedicated measures; minimal effort for project team

Review process for industry practice reviews

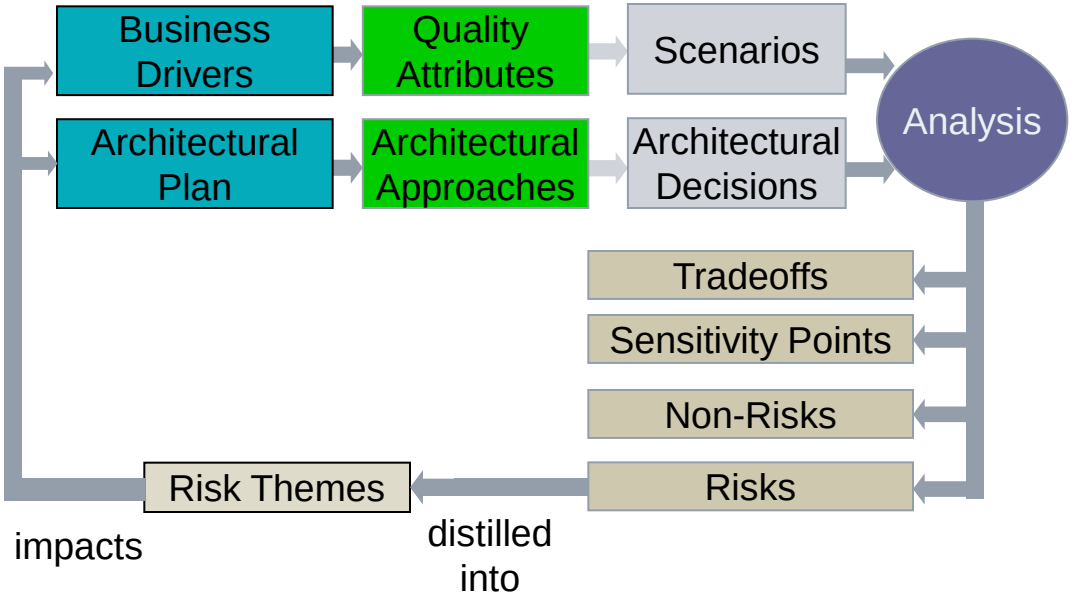


Architecture Tradeoff Analysis Method (ATAM)*

Purpose:
Identify risks, sensitivity points and tradeoffs.

Workshop steps:

Present the ATAM.
Present business drivers
Present architecture
Identify architectural approaches.
Generate quality attribute utility tree
Analyze architectural approaches
Brainstorm and prioritize scenarios
Analyze architectural approaches.
Present results



Effort: 3–4 day workshops, evaluation team 30–40 person days, project team 30–40 person-days (In practice, a lot less, because of previous experiences and result reuse)

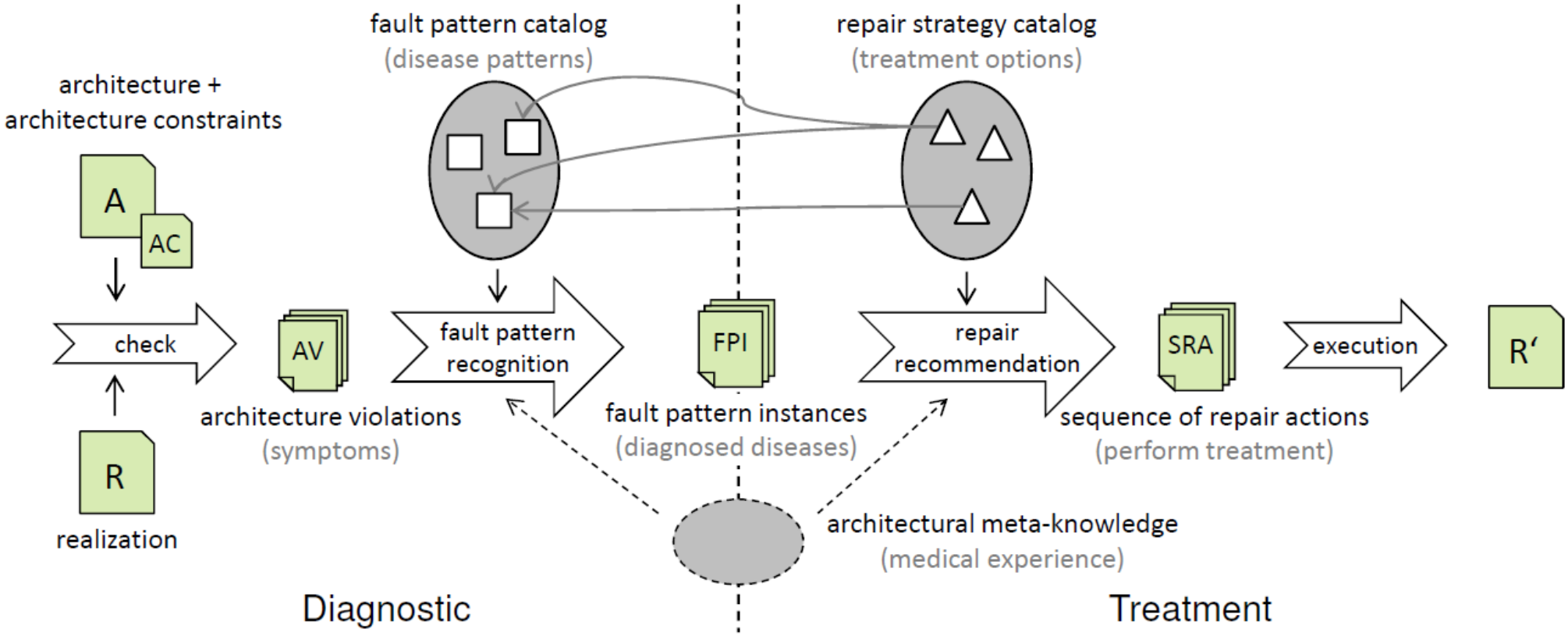
Results: Prioritized list of scenarios regarding business drivers, risks and tradeoff points related to architectural decisions

Benefits: Identified risk, documented basis for architectural decisions

*Source: Software Engineering Institute, Carnegie Mellon University

SIEMENS

Architecture Erosion: Treating the Patient



Source: Mair et. al. *Towards Flexible Automated Software Architecture Erosion Diagnosis and Treatment*

SIEMENS

Architecture refactoring - Definition

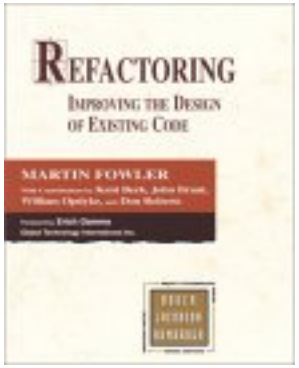
- Architecture refactoring is about the **semantic-preserving transformation** of a software **design**
- It **changes structure** but not behavior
- It applies to **architecture-relevant** design artifacts such as UML diagrams, models, DSL expressions, aspects
- Its goal is to **improve architecture and design quality**.
- You got an "architectural smell"? Use an architecture refactoring pattern to solve it!

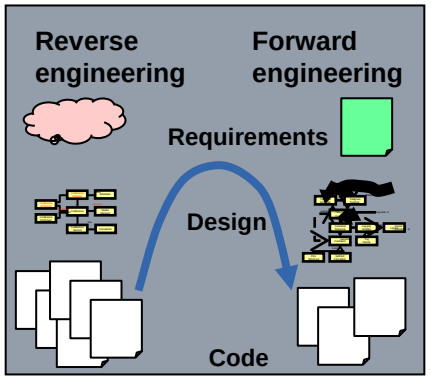
Note:
A smell is only an ***indicator*** of a possible problem, not a proof

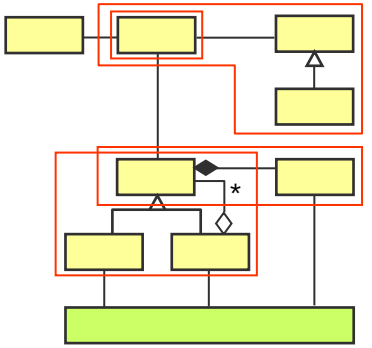
Refactoring, reengineering, and rewriting comparison (1)

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**







SIEMENS

Refactoring, reengineering, and rewriting comparison (2)

	Refactoring	Reengineering	Rewriting
Scope	▪ Many local effects	▪ Systemic effect	▪ Systemic or local effect
Process	▪ Structure transforming ▪ Behavior / semantics preserving	▪ Disassembly / reassembly	▪ Replacement
Results	▪ Improved structure ▪ Identical behavior	▪ New system	▪ New system or new component
Improved qualities	▪ Developmental ▪ Operational	▪ Functional ▪ Operational ▪ Developmental	▪ Functional ▪ Operational ▪ Developmental
Drivers	▪ Complicated design / code evolution ▪ When fixing bugs ▪ When design and code smell bad	▪ Refactoring is insufficient ▪ Bug fixes cause rippling effect ▪ New functional and operational requirements ▪ Changed business case	▪ Refactoring and reengineering are insufficient or inappropriate ▪ Unstable code and design ▪ New functional and operational requirements ▪ Changed business case
When	▪ Part of daily work ▪ At the end of each iteration ▪ Dedicated refactoring iterations in response to reviews ▪ It is the 3rd step of TDD	▪ Requires a dedicated project	▪ Requires dedicated effort or a dedicated project, depending on scope