Driving triumvirate
for testing & quality engineering

SIEMENS

Process quality, certification and audits, regulations and norms

across lifecycles and versions

Quality Manager

Process

Quality goals
Productivity

Management

Test Manager

Budget, people, plans, logistics

for the test organization

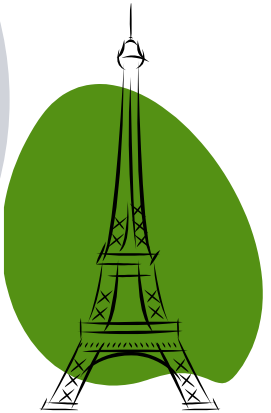
Strategy

Technologies

Test Architect

Cost efficient, accelerated, innovative, sustainable testing solutions

for test methodology and test infrastructure



As a Test Architect
You have one Role – but you are wearing two Hats

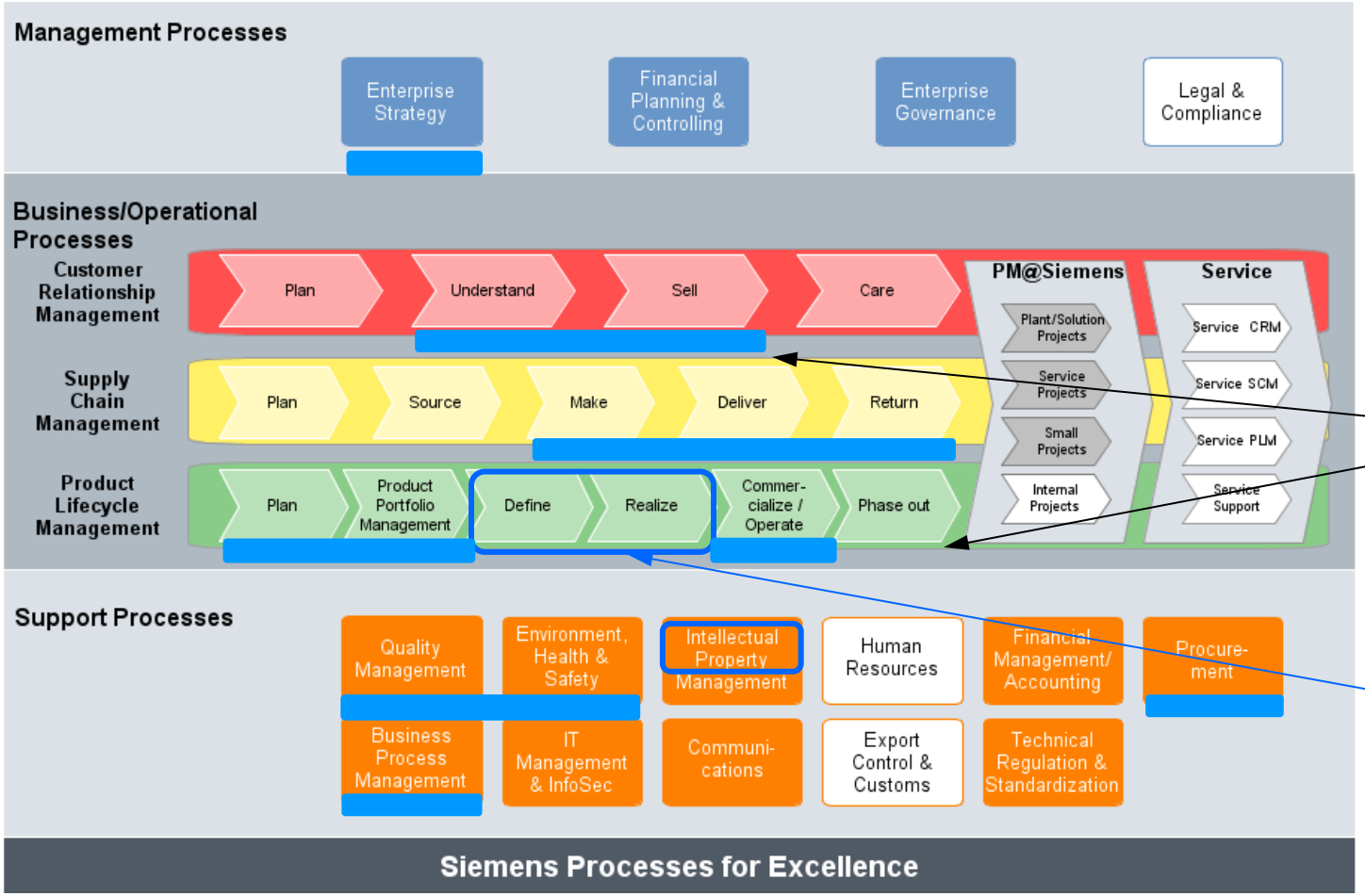
Test Expert
for the system under test (SUT)

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

Software / System Architect
for the test system

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

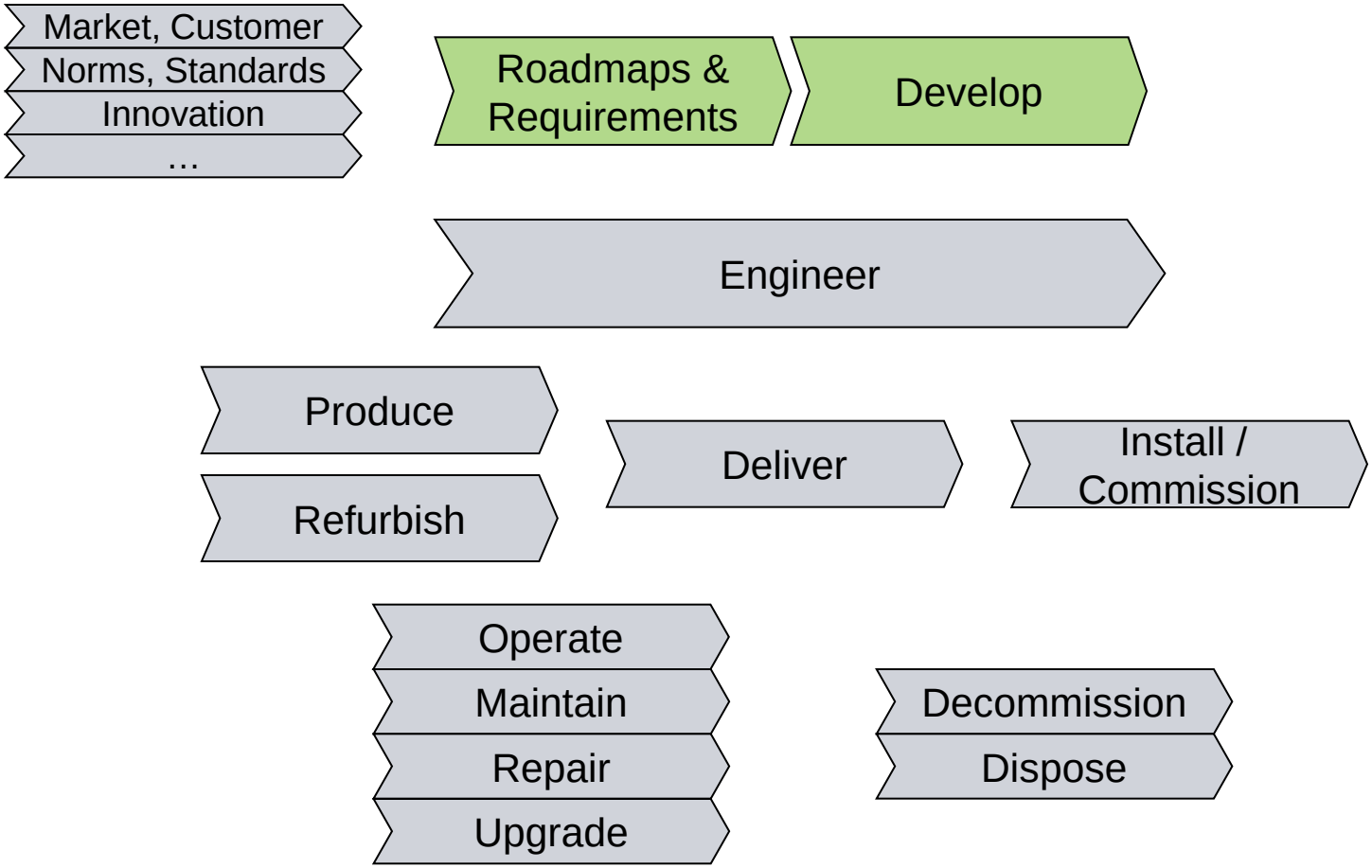
Test Architect's scope in SIPEX



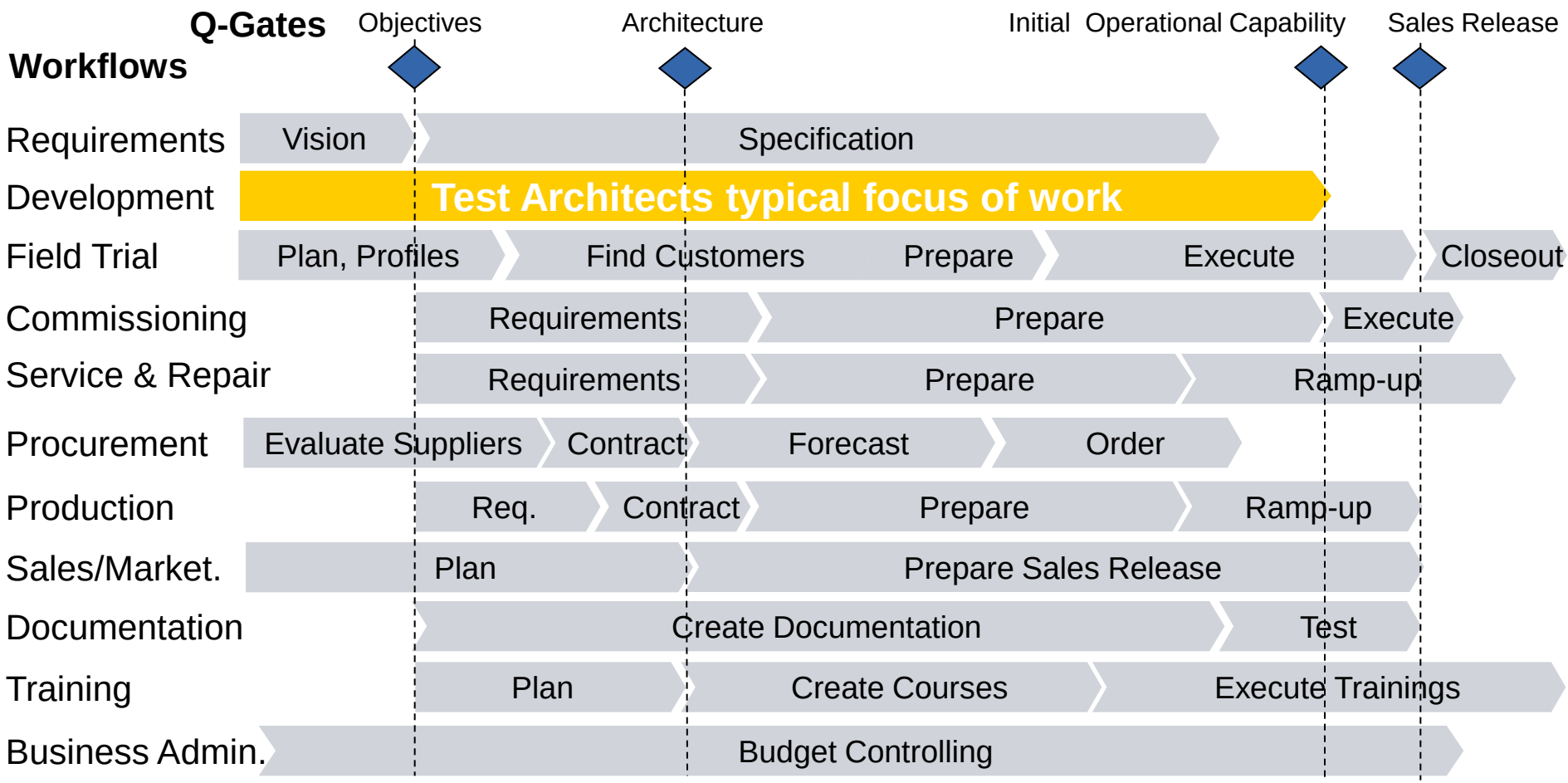
Interfaces, Support

Main focus of responsibility

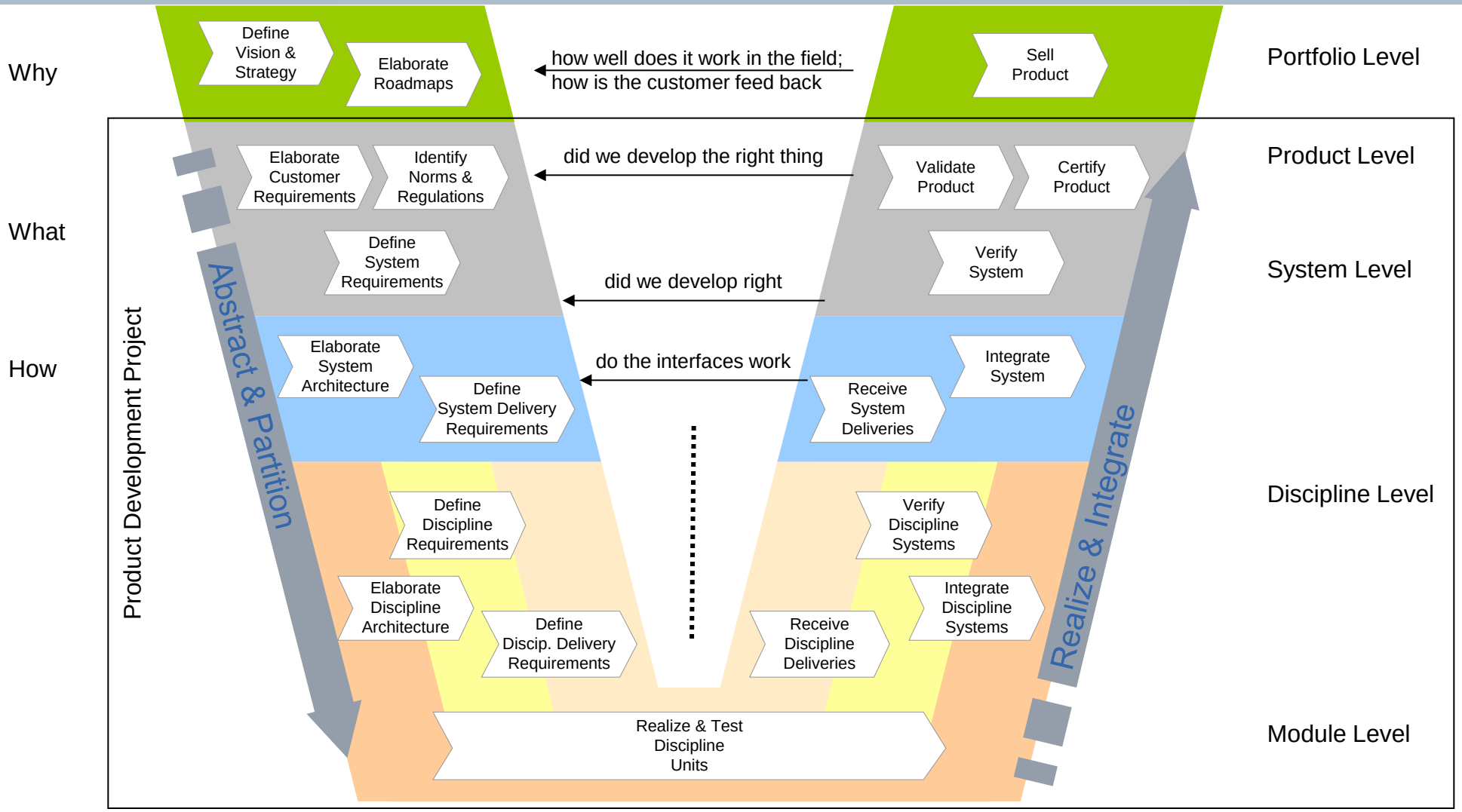
Major system lifecycle elements that influence a system architecture



Parallelism of Processes

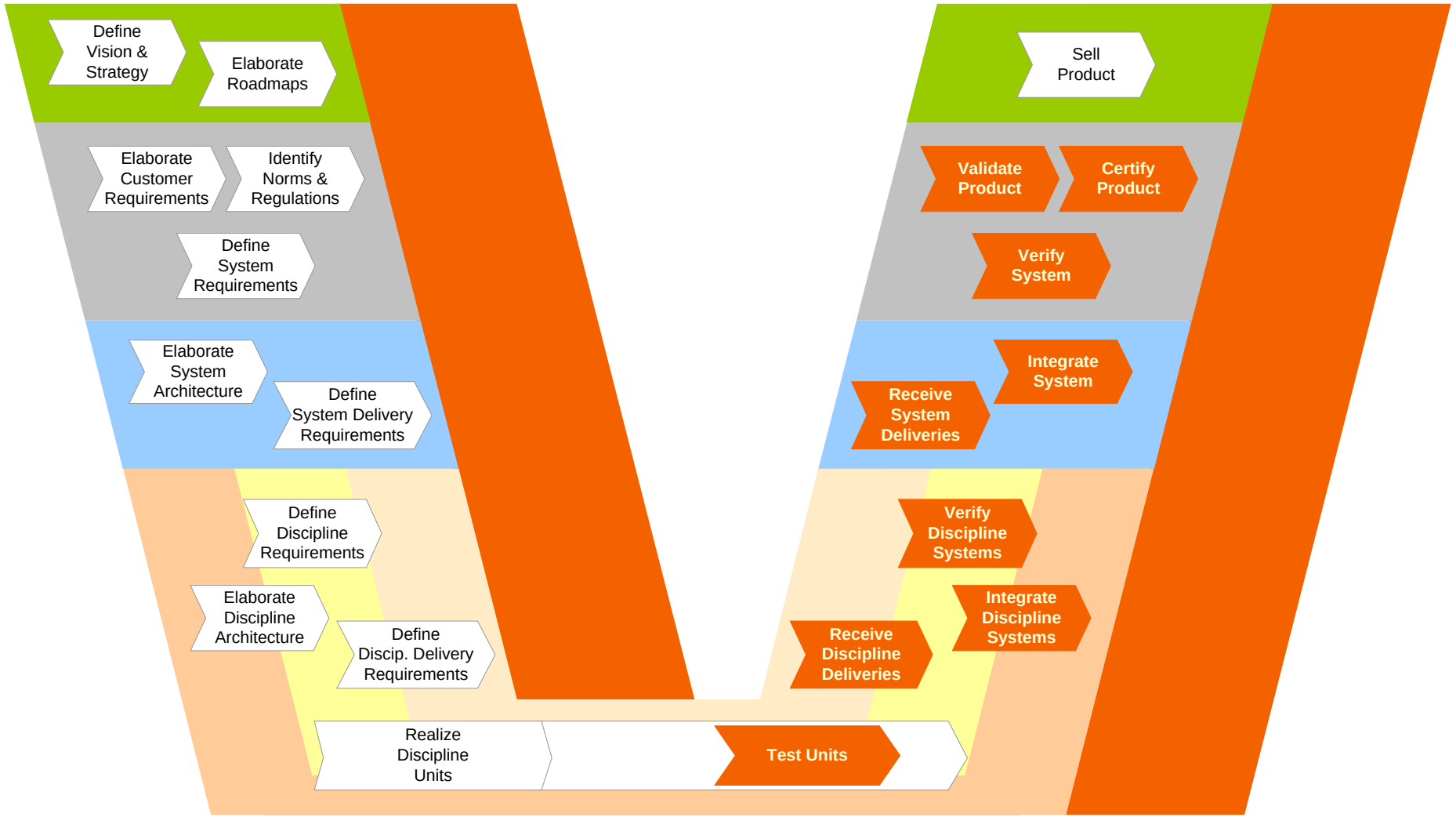


System Development Process Model
basic V-Model



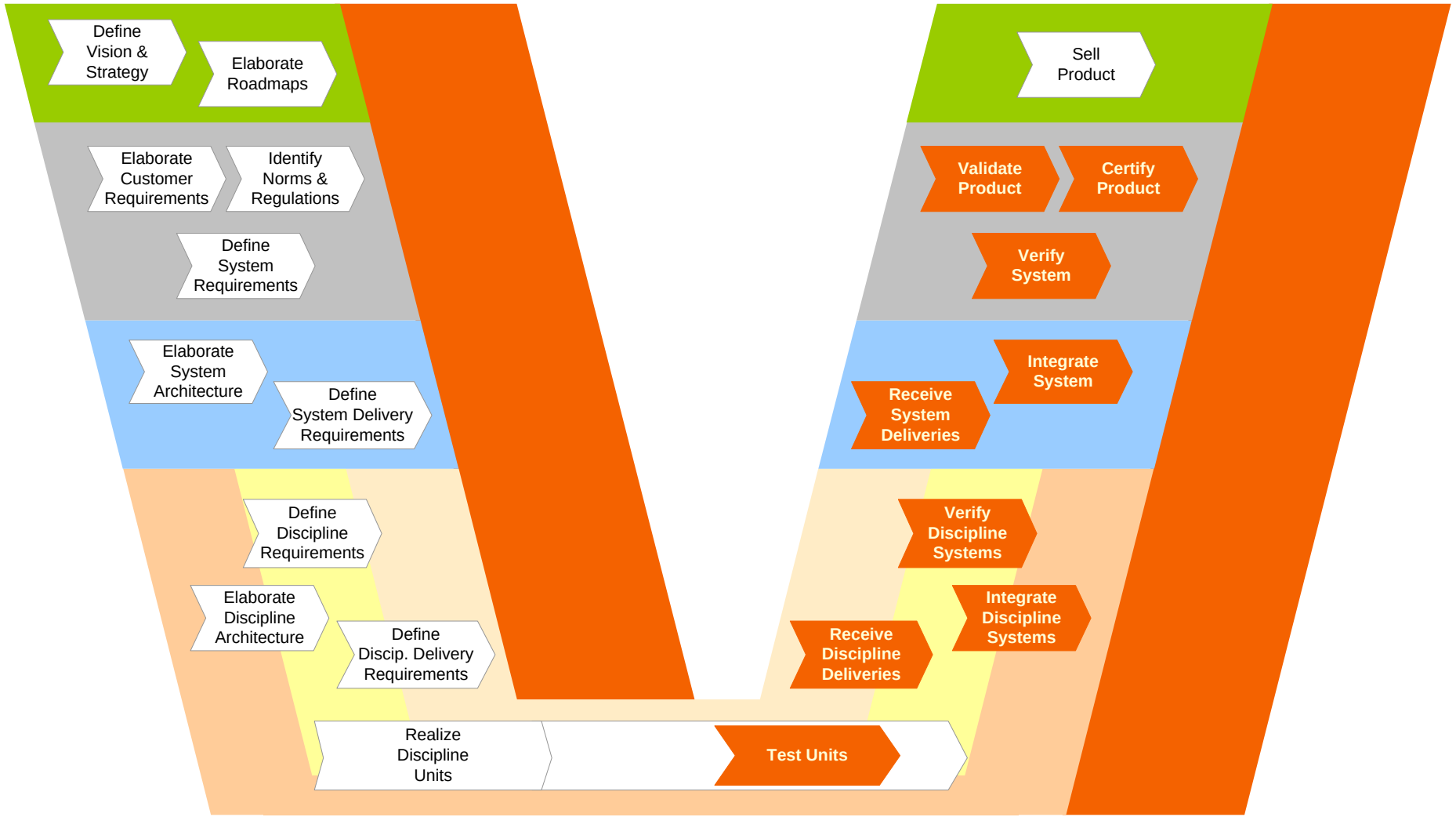
Based on the V-Model develop a Test-𝕎-Model

SIEMENS



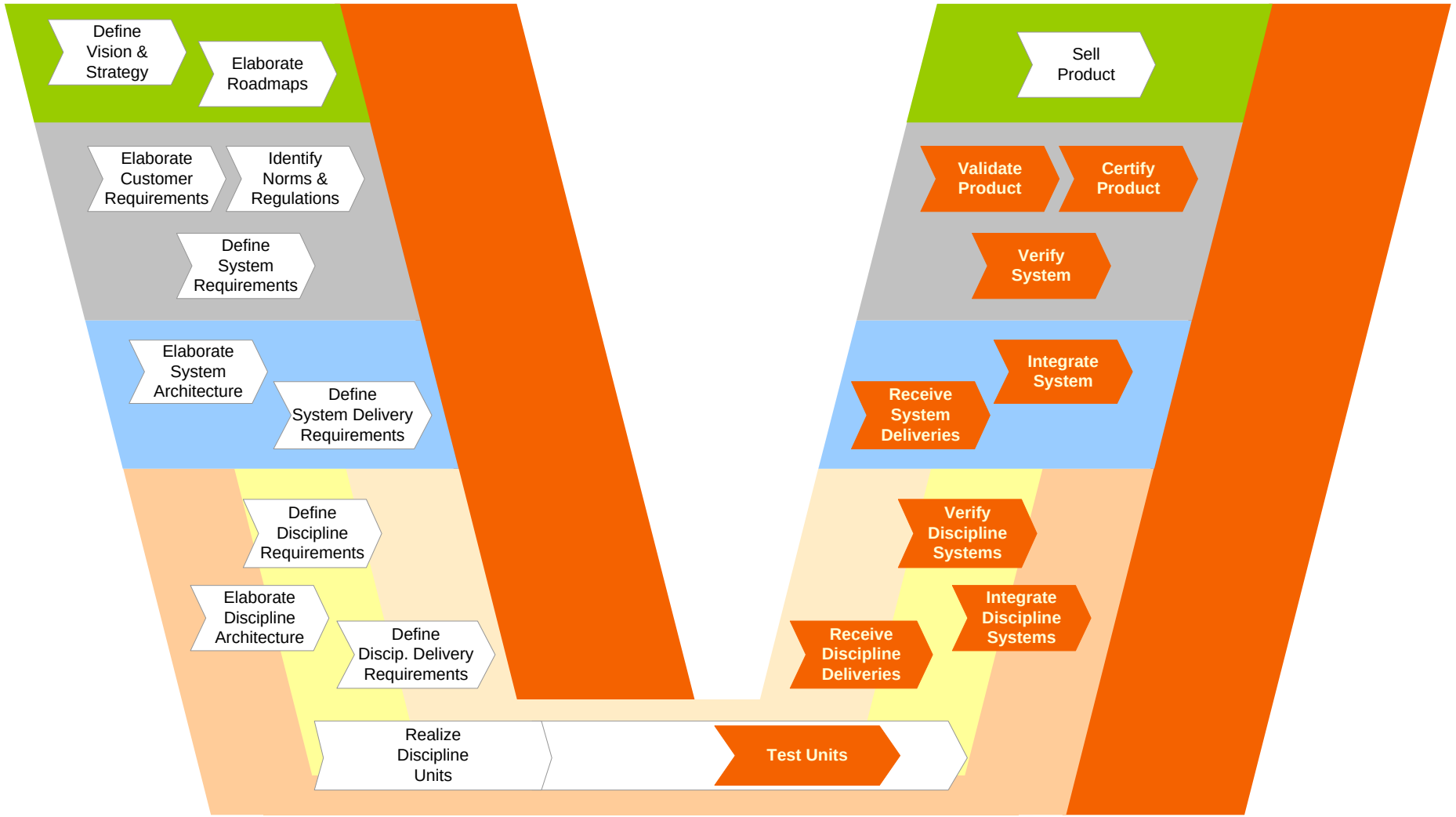
Based on the V-Model develop a Test-𝕎-Model

SIEMENS



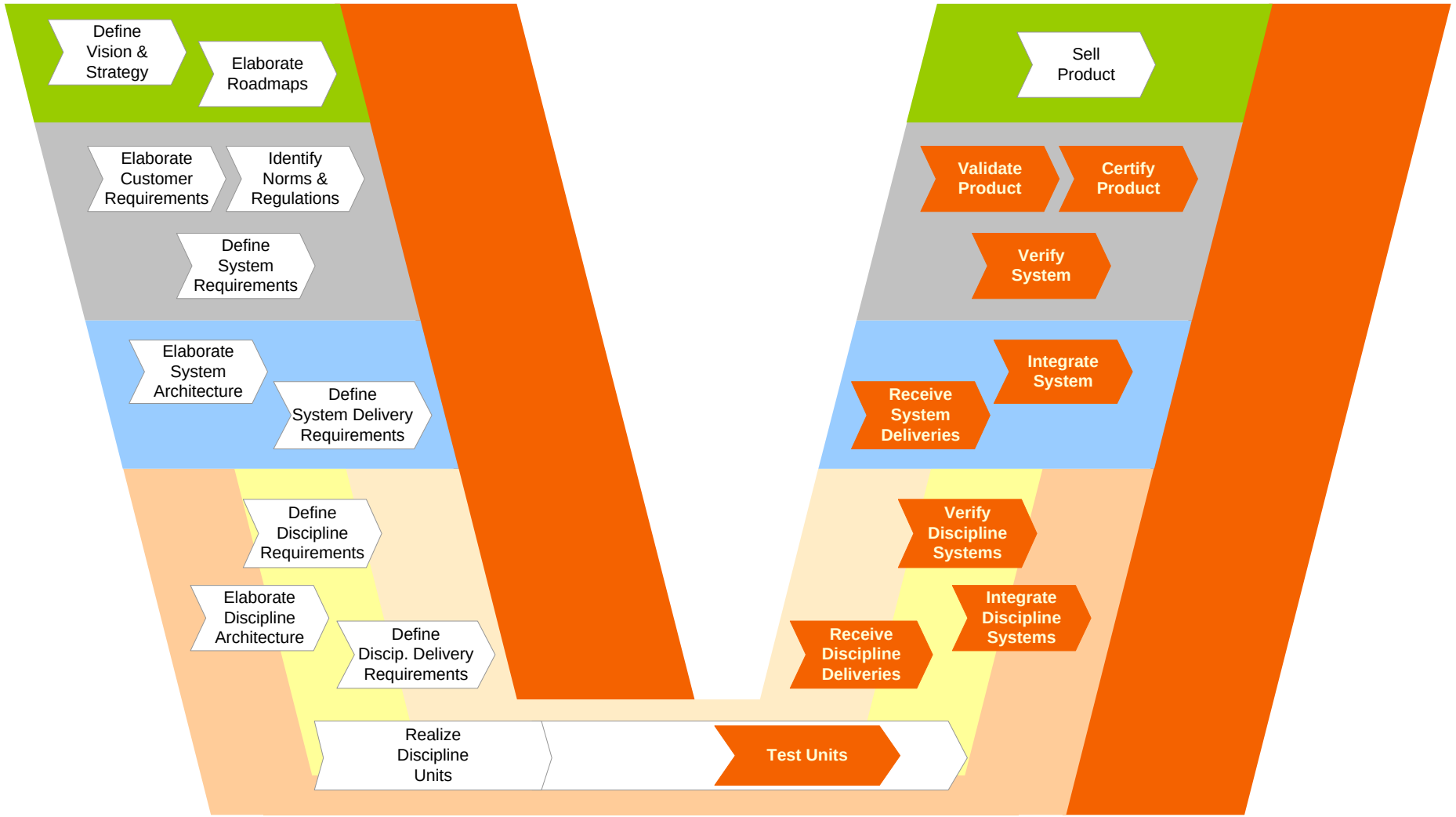
Based on the V-Model develop a Test-𝕎-Model

SIEMENS



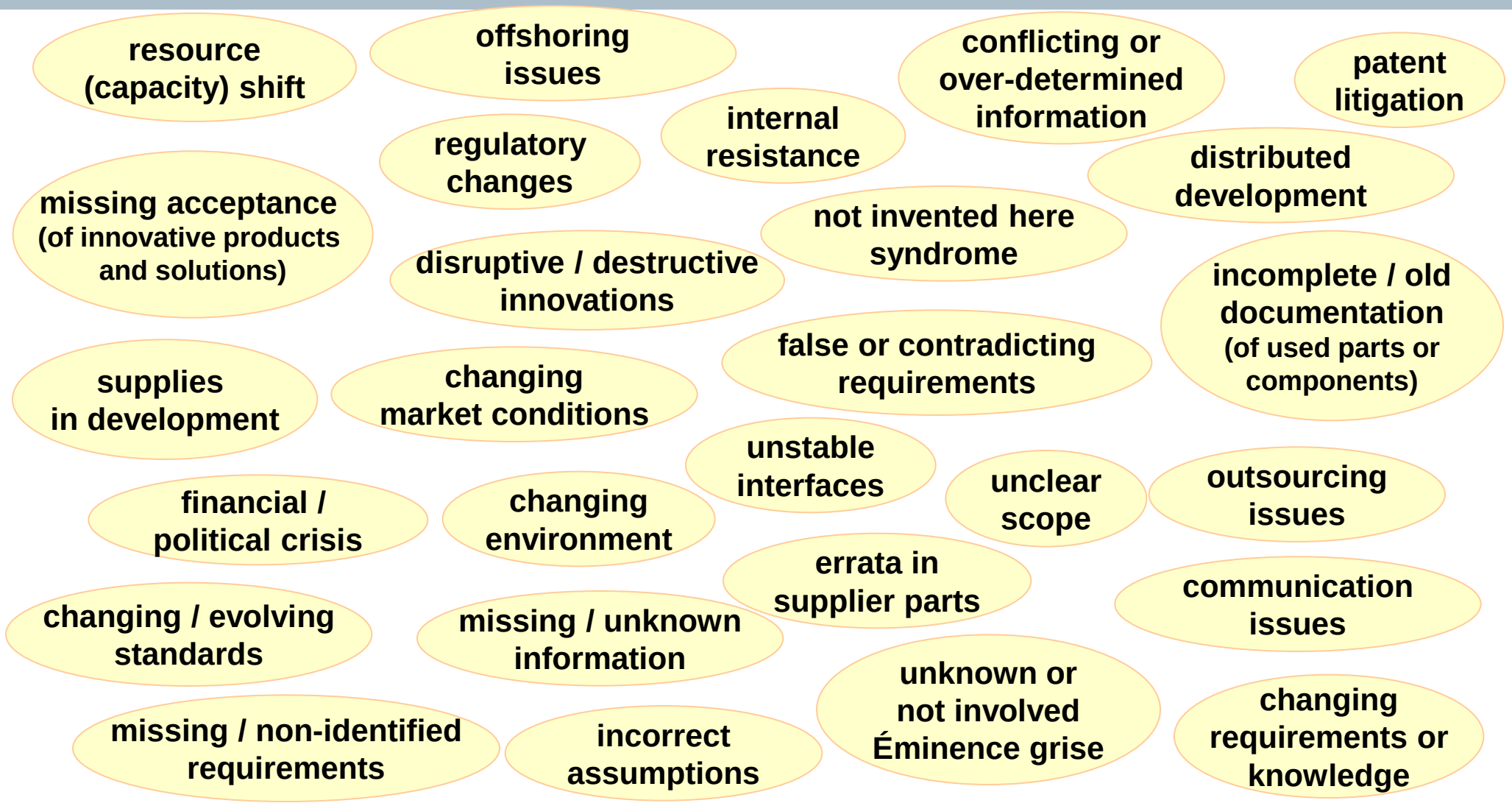
Based on the V-Model develop a Test-𝕎-Model

SIEMENS



SIEMENS

Uncertainty Sources



SIEMENS

What do you need to know to build a system?

Stakeholder

System Vision

System Boundaries

Customer Needs

Purpose of RE

Common Understanding

Agreement

Non Functional Requirements

Baselining and Tracing

Artifacts

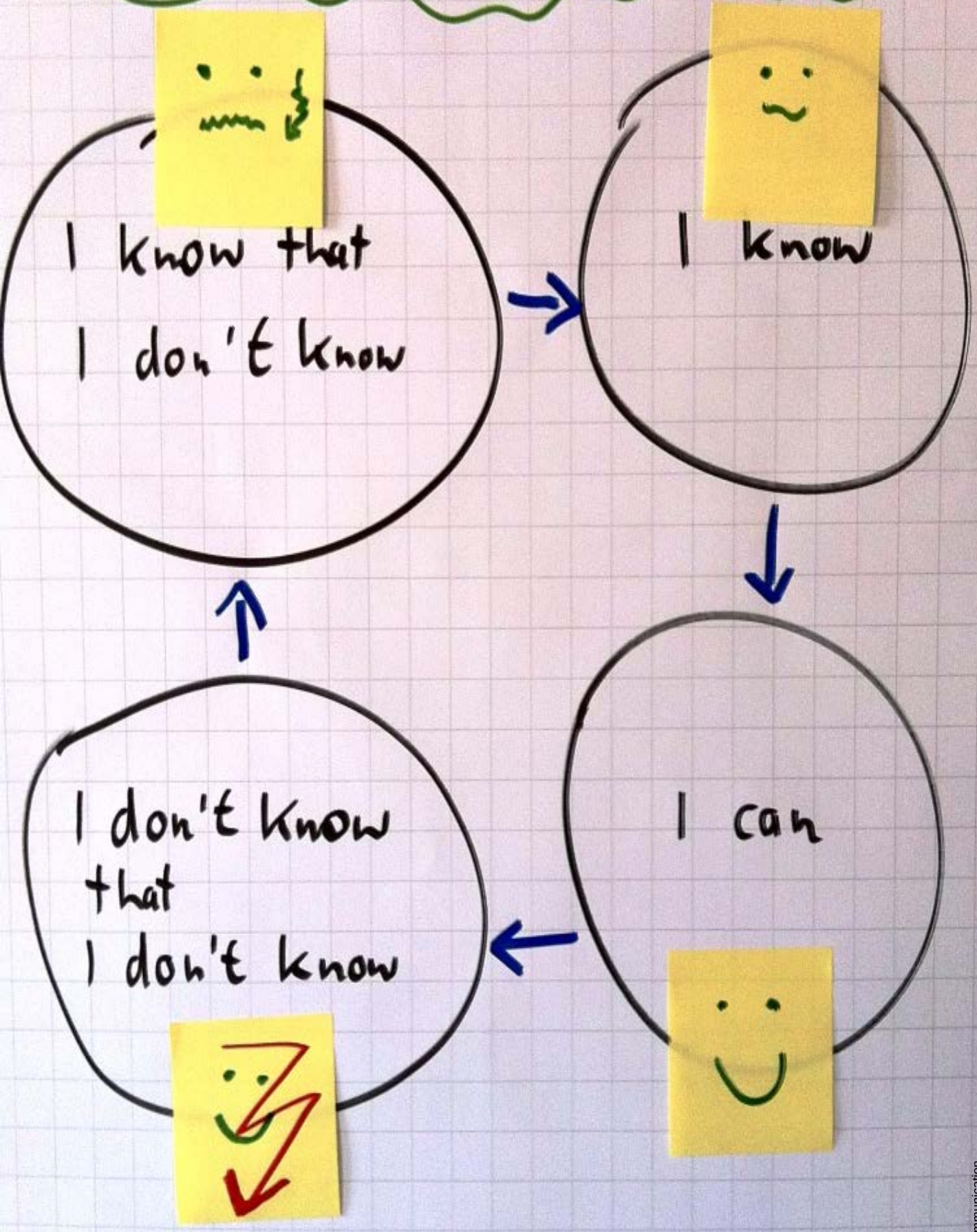
Prioritization

Mapping of IEEE 29148 characteristics to Requirement Pyramid

SIEMENS

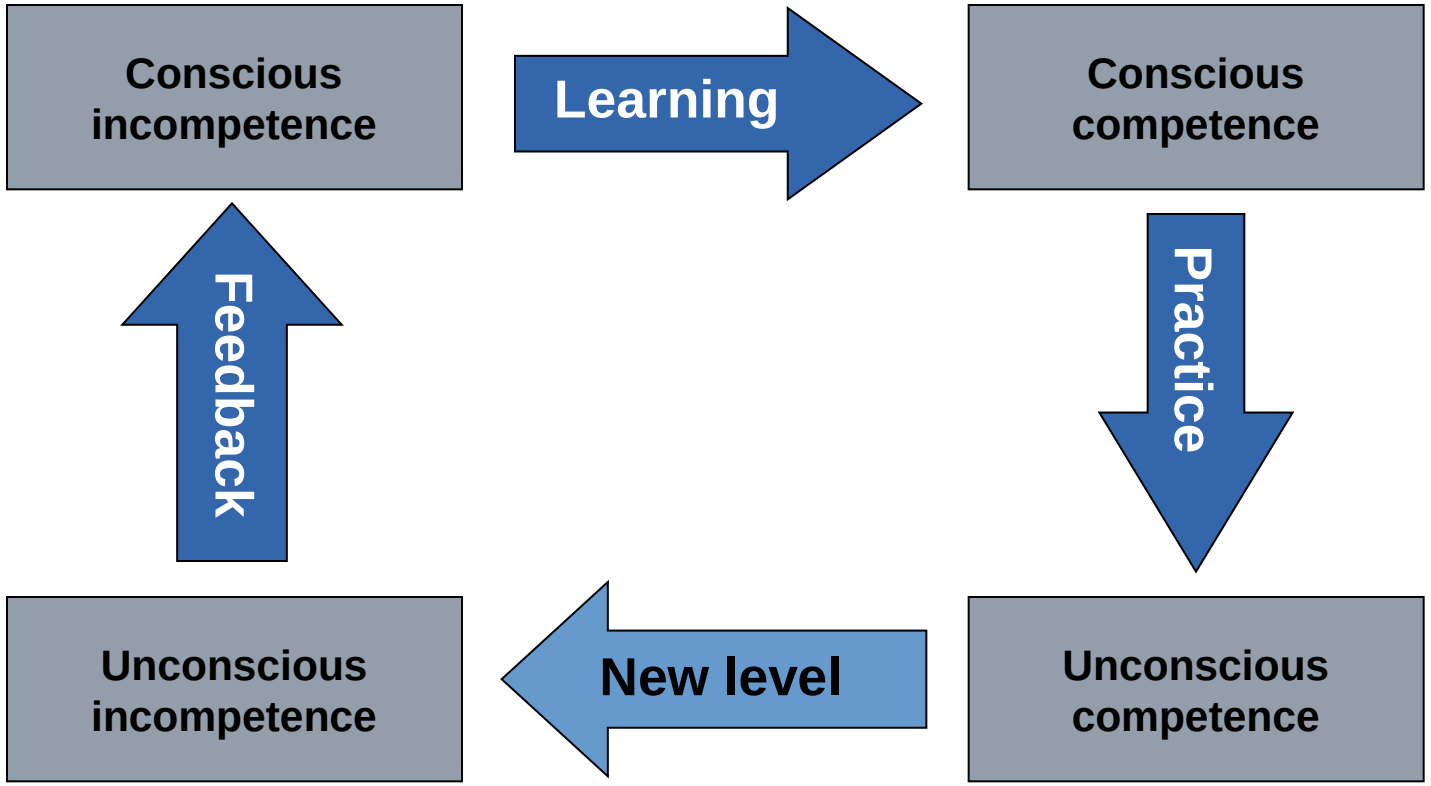
Quality Criteria	Business Goals	Customer Requirements	System Requirements
Individual Requirements			
Necessary	Mandatory	Mandatory	Mandatory
Implementation free	Mandatory	Mandatory	Mandatory
Unambiguous	Optional	Mandatory	Mandatory
Consistent	Mandatory	Mandatory	Mandatory
Complete	Optional	Optional	Mandatory
Singular	Mandatory	Optional	Mandatory
Feasible	Optional	Optional	Mandatory
Verifiable	Optional	Mandatory	Mandatory
Traceable	Mandatory	Mandatory	Mandatory
Set of Requirements			
Complete	Optional	Optional	Mandatory
Consistent	Mandatory	Mandatory	Mandatory
Affordable	Optional	Optional	Mandatory
Bounded	Optional	Optional	Mandatory

Learning Model

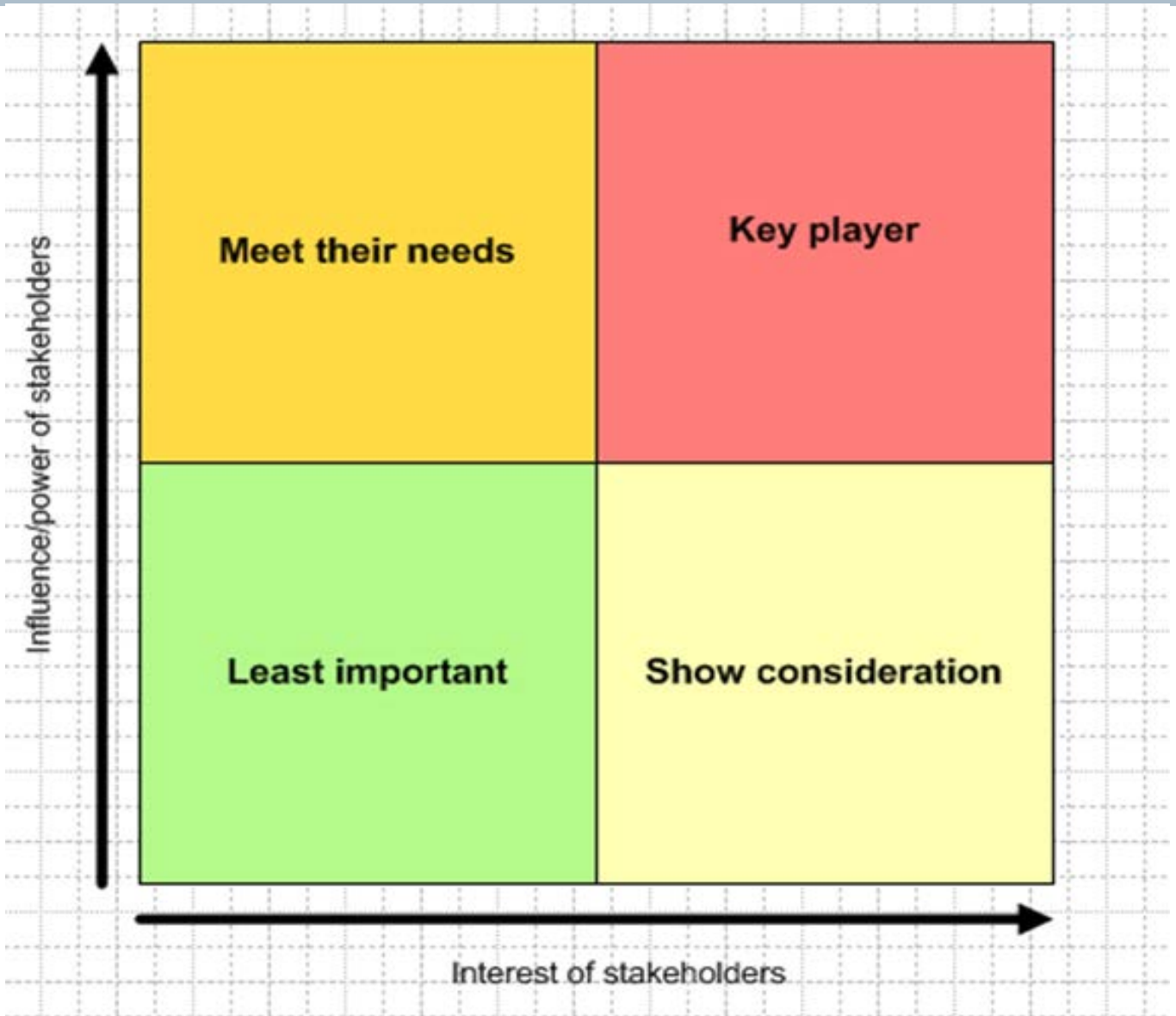


From good to great - through reflection and practice

Learning model



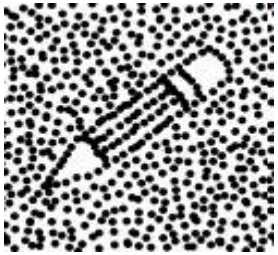
If you want success – know your stakeholders!



Active Listening – first understand than be understood

Show interest

Paraphrase



Reflect emotions

Body language

„Hmhm ... hmhm ... yes ...“,
nodding,
eye-contact and
nose-navel-axis.

Are your interested?
Show that your are listening.

Facts

„Did I understand you correctly,
you think that ... ?“

You repeat the core statements in own words and ask for affirmation.

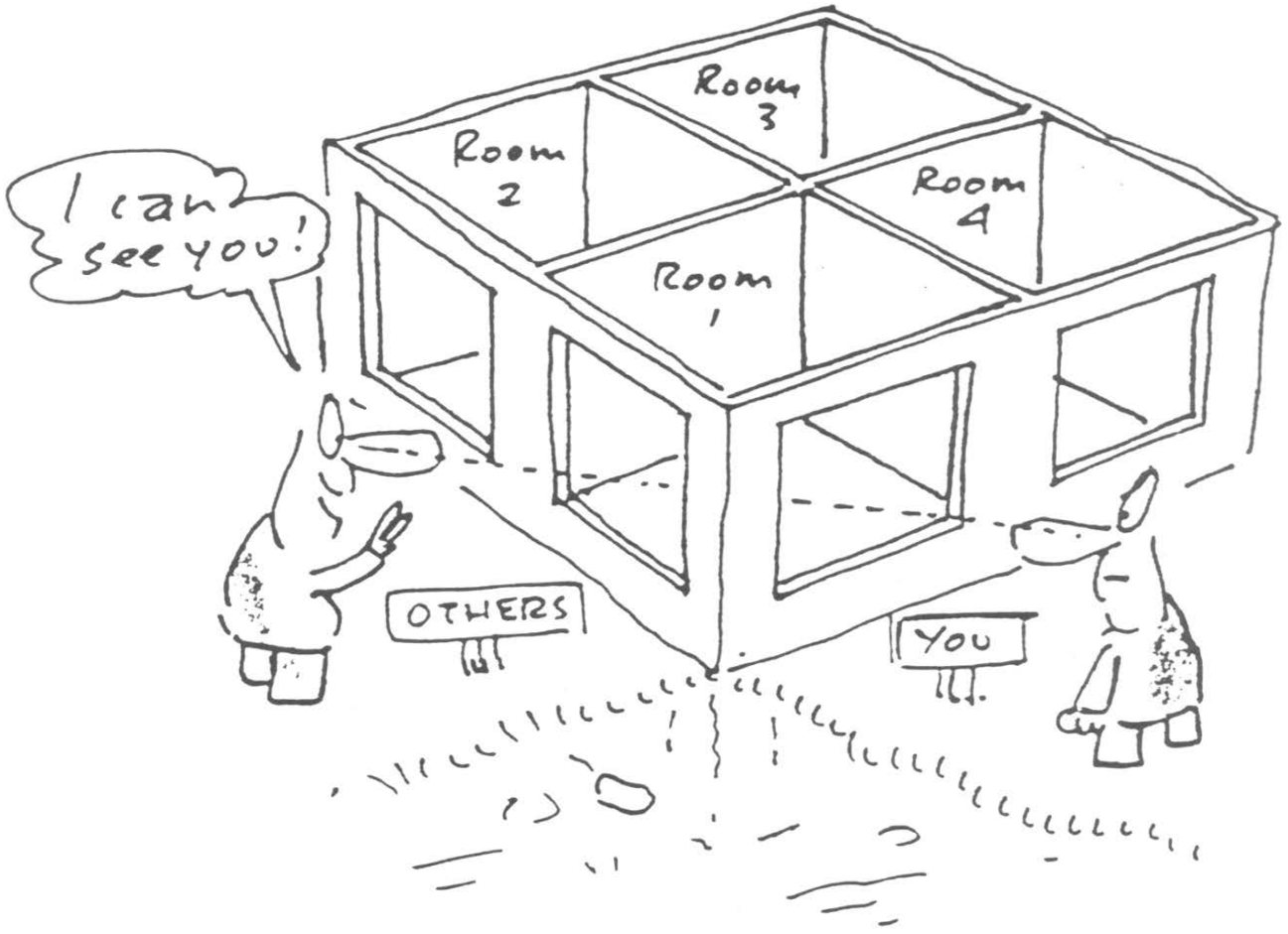
Feelings

„ ... and that made you happy.“

„ ... and that‘s why you’re angry.“

You say how you think your dialogue partner feels. Than see if he/she agrees.

The Johari window is a useful model for communication and cooperation



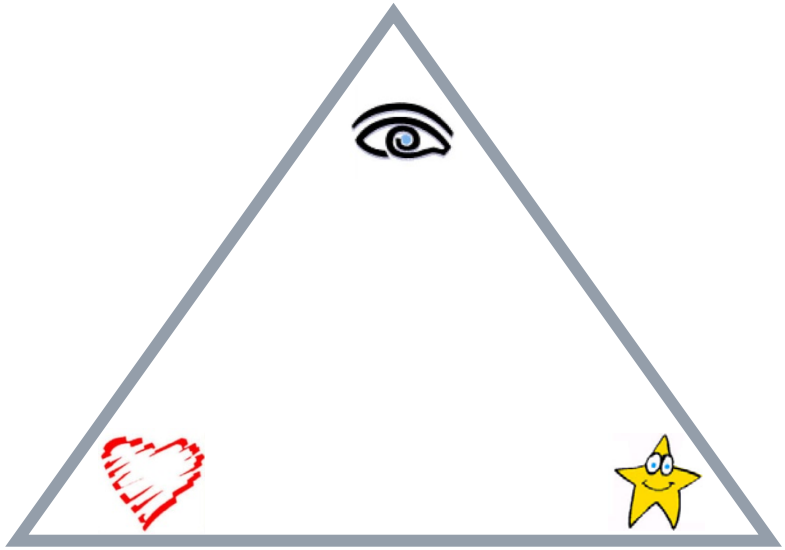
Providing WWW Feedback is a chance to grow

W hat did I see?
W hat did I feel?
W hat do I wish?

In German:
Wahrnehmung
Wirkung
Wunsch

Perception (objective)

Describe a concrete perception of a behavior



Effect (subjective)

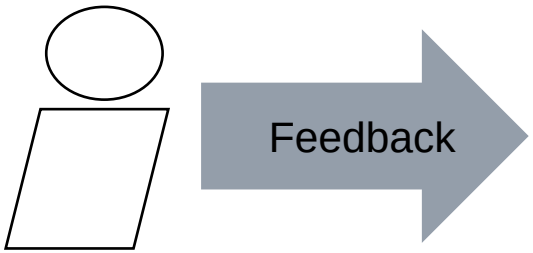
Describe the effect of the observed behavior

Wish (acceptable)

Formulate a wish and give opportunity to clarify

Providing WWW Feedback is a chance to grow

When you give feedback …



Perception What did you see and hear?

Effect What effect did it have on you? How did you feel?

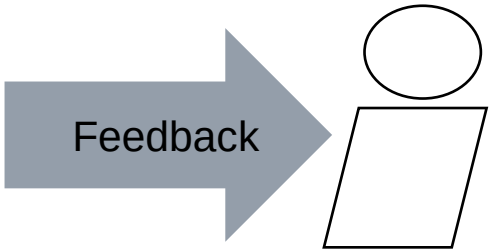
Wish What kind of behavior would you like to see?

… don't forget:

- ✓ Use **"I"** instead of "one" or other general terms
- ✓ Speak directly to him / her (not about him / her)
- ✓ One positive comment is often more effective than ten negative remarks

Providing WWW Feedback is a chance to grow

When receiving feedback …



Perception If not given – ask for a specific situation (example)?

Effect Try to understand how the feedback partner felt?

Wish What kind of behavior would he/she like to see?

... don't forget:

- ✓ See receiving feedback as an opportunity
- ✓ Listen carefully
- ✓ If you are not able to listen now – say when
- ✓ Never defend, explain or justify !!!

SIEMENS

Why do we test? Dimensions of testing

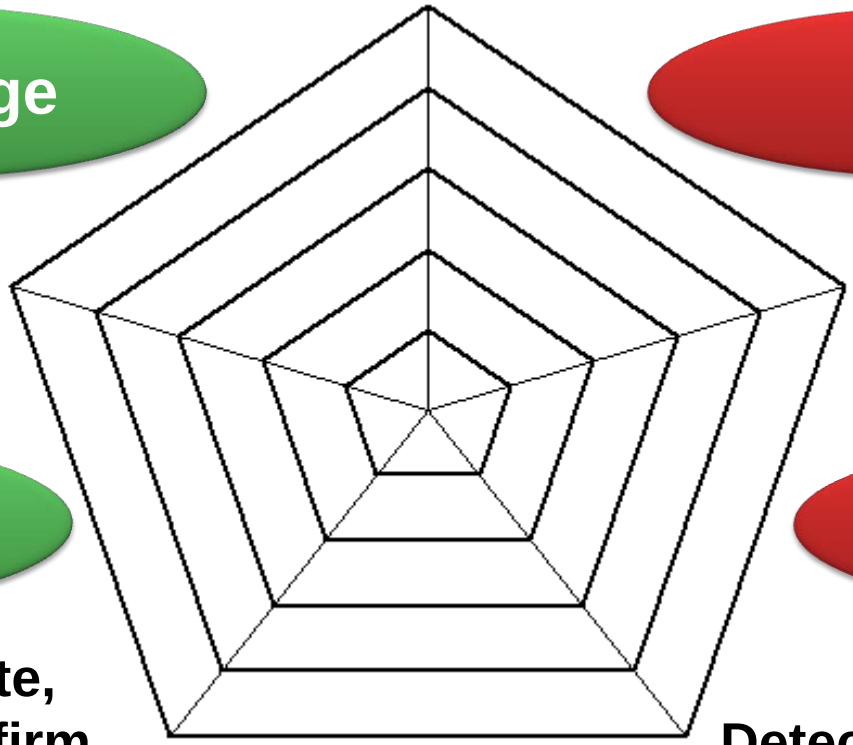
Prevent, protect, respond, control, influence, enable, and drive quality, support, drive, and speed up development

Coverage

Measure, assess, evaluate, predict

Risks

Mitigate, reduce risks, investigate, explore



Goals

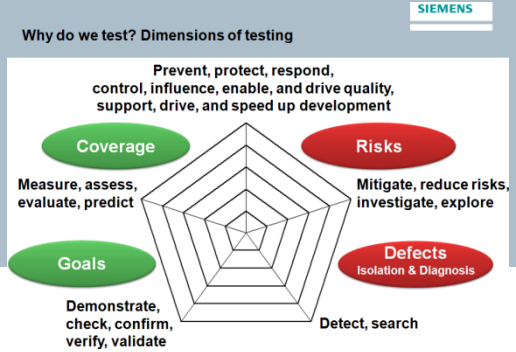
Demonstrate, check, confirm, verify, validate

Defects
Isolation & Diagnosis

Detect, search

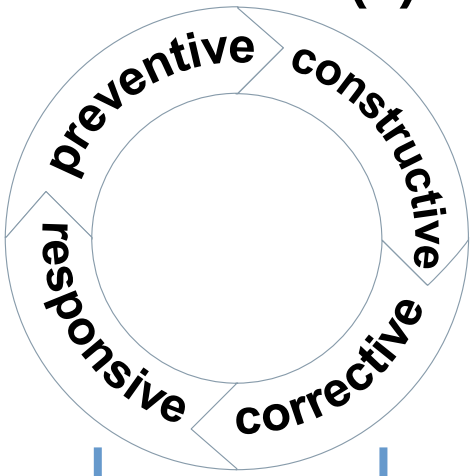
Reference: Peter Zimmerer, "The Value of Testing in 5 Dimensions", *Testing Experience*, September 2010

Why do we test?



SIEMENS

Historical and future (?) view



History consists of a series of accumulated imaginative inventions.

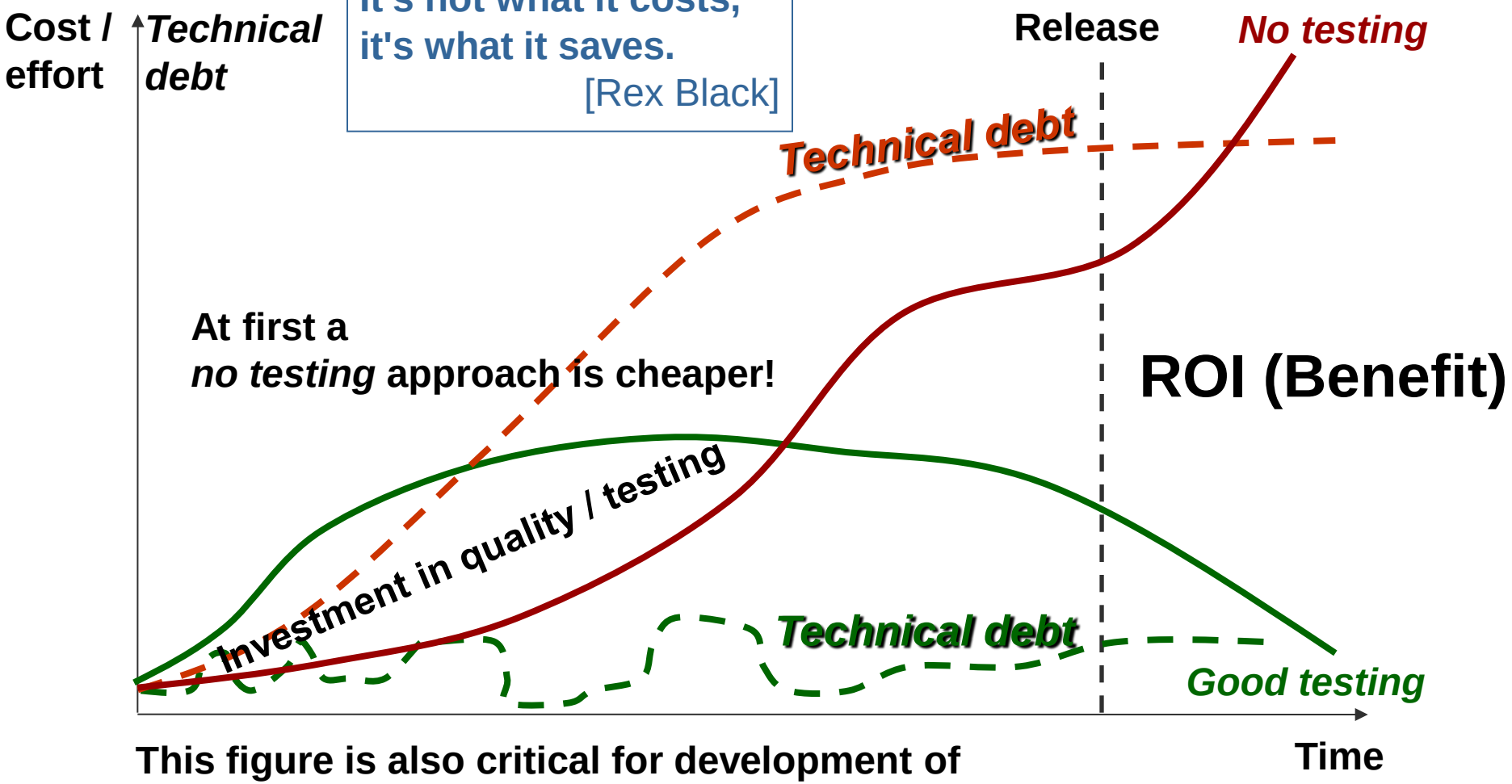
Voltaire, 1694–1778

Show it works	Find defects	Assess quality Evaluation Measure quality	Influence quality Control quality	Investigate Explore Learn	Speed up Protect Respond
Demonstration	***Detection***	***Prediction***	***Prevention***	***Exploration***	***Productivity***
1950s view	**1975s view**	**1980s view**	**1990s view**	**2000s view**	**2010s view**

SIEMENS

Overall cost & effort – Technical debt

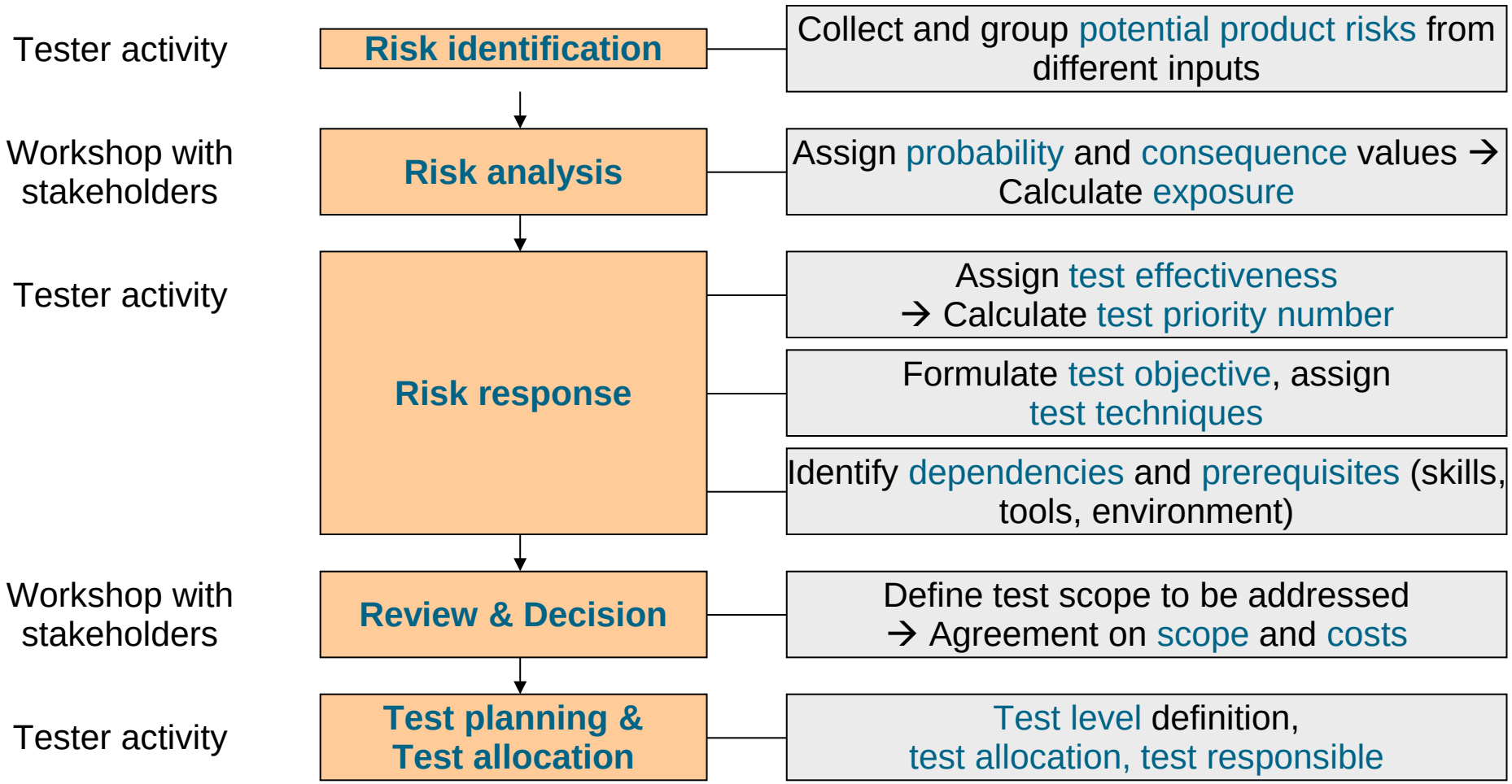
It's not what it costs,
it's what it saves.
[Rex Black]



This figure is also critical for development of frameworks, platforms, product lines, ecosystems!

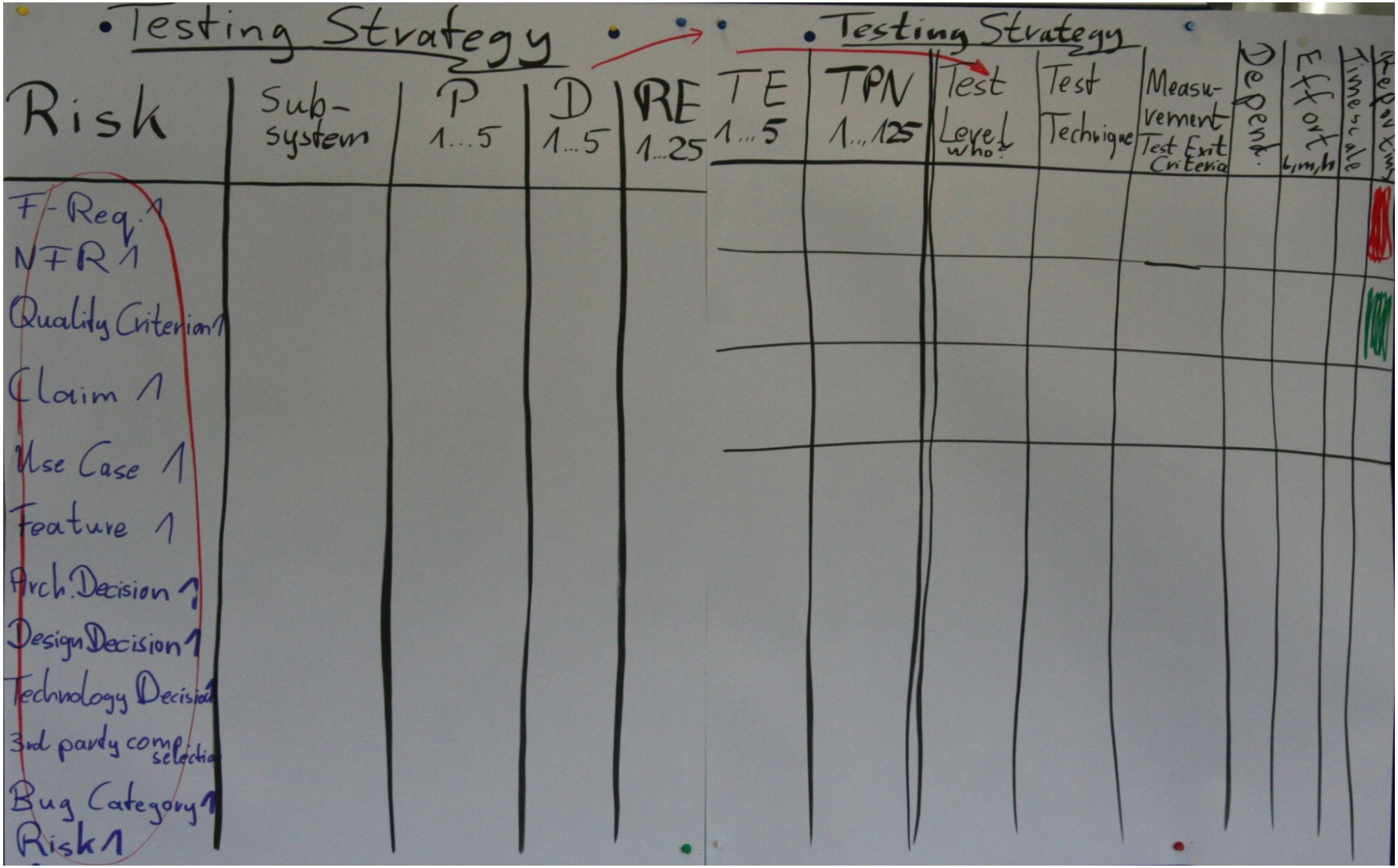
SIEMENS

Risk-based test strategy – Steps



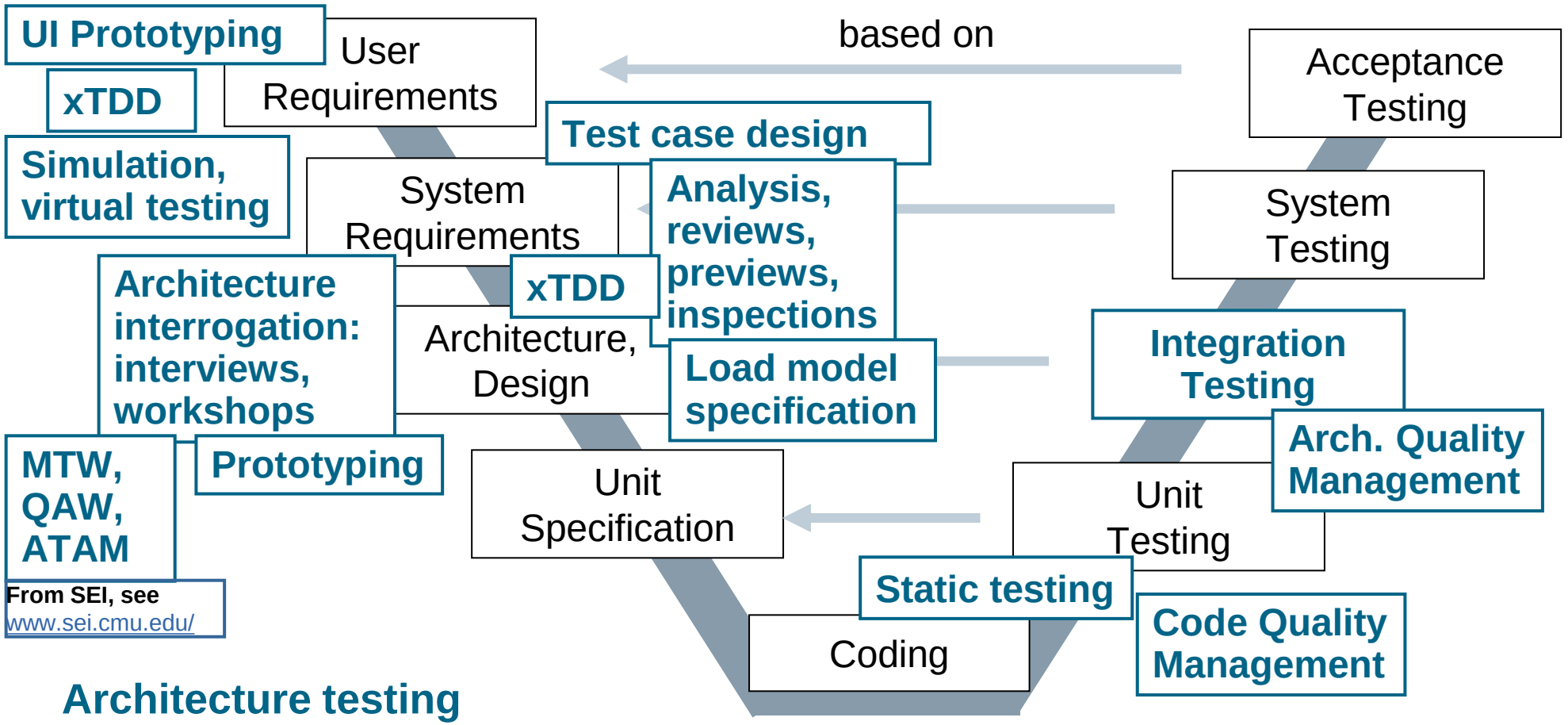
Adapted from: Paul Gerrard, Neil Thompson: Risk-Based E-Business Testing, 2002

Risk-based test strategy – On one page …



Test levels – Example V model with architecture testing

SIEMENS



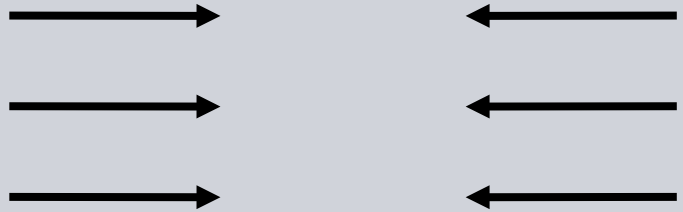
Architecture testing

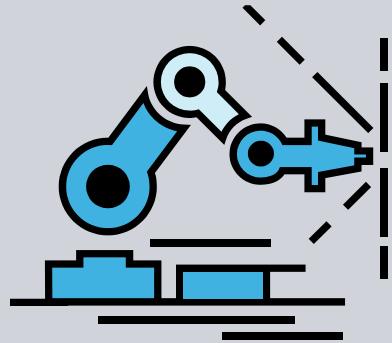
is any testing of architecture and architectural artifacts

→ mapping of architectural risks to test levels

SIEMENS

Example: Integration Testing





Control system
- Set operation mode
- Control robot actions
- Visualize status

Requests
- Init/Shutdown
- Move to position
- Stop moving
- Get operation status

Replies
- ACK/NACK
- Return state

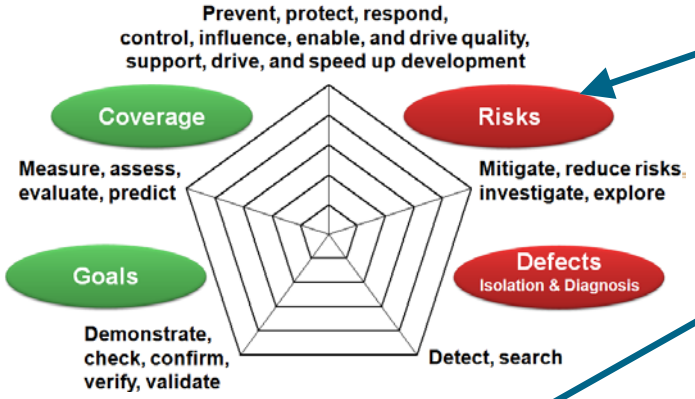
Robot system
- Operational states
- Error states
- Real time constraints
- Safety constraints

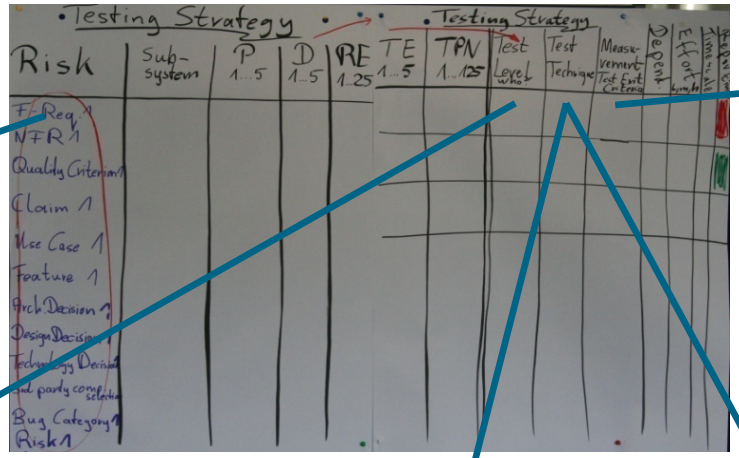
- What aspects of this system would you test on **Integration Test** level?
- What kind of input would you need for these tests?
- Which stakeholder(s) can provide these inputs?
- What do you – as **Test Engineer / Test Architect** – have to provide?

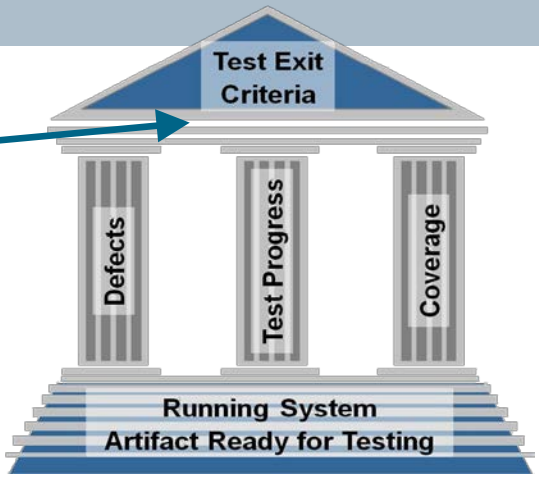
Summary: Building blocks of a *Risk-based Test Strategy (RBT)*

SIEMENS

Why do we test? Dimensions of testing

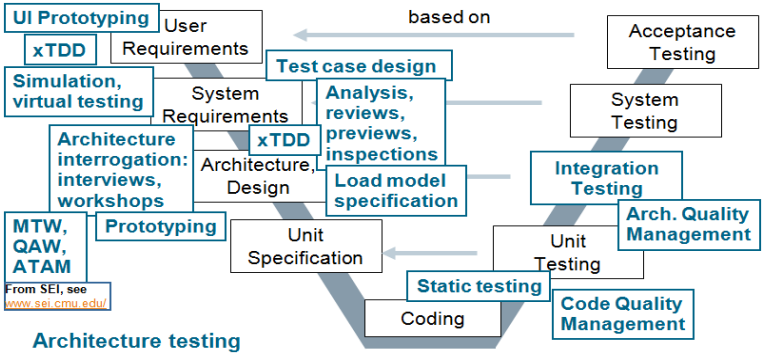






SIEMENS

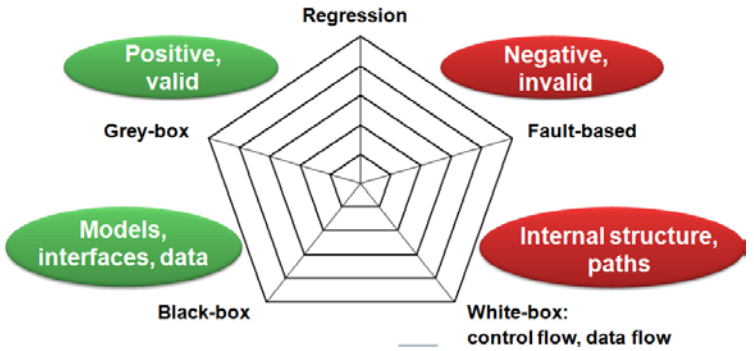
Test levels – Example V model with architecture testing

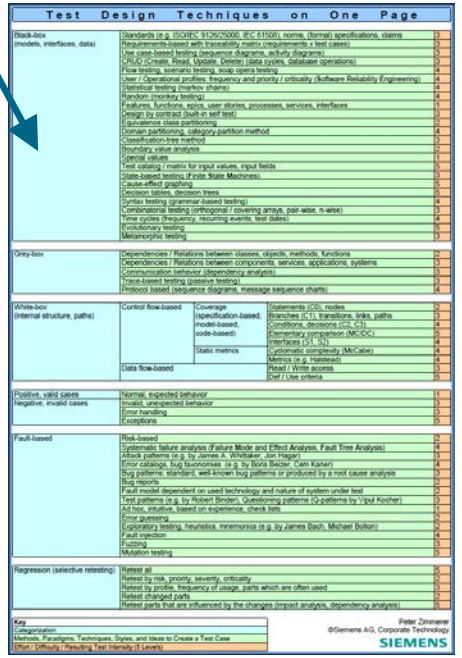


Architecture testing
is any testing of architecture and architectural artifacts
→ mapping of architectural risks to test levels

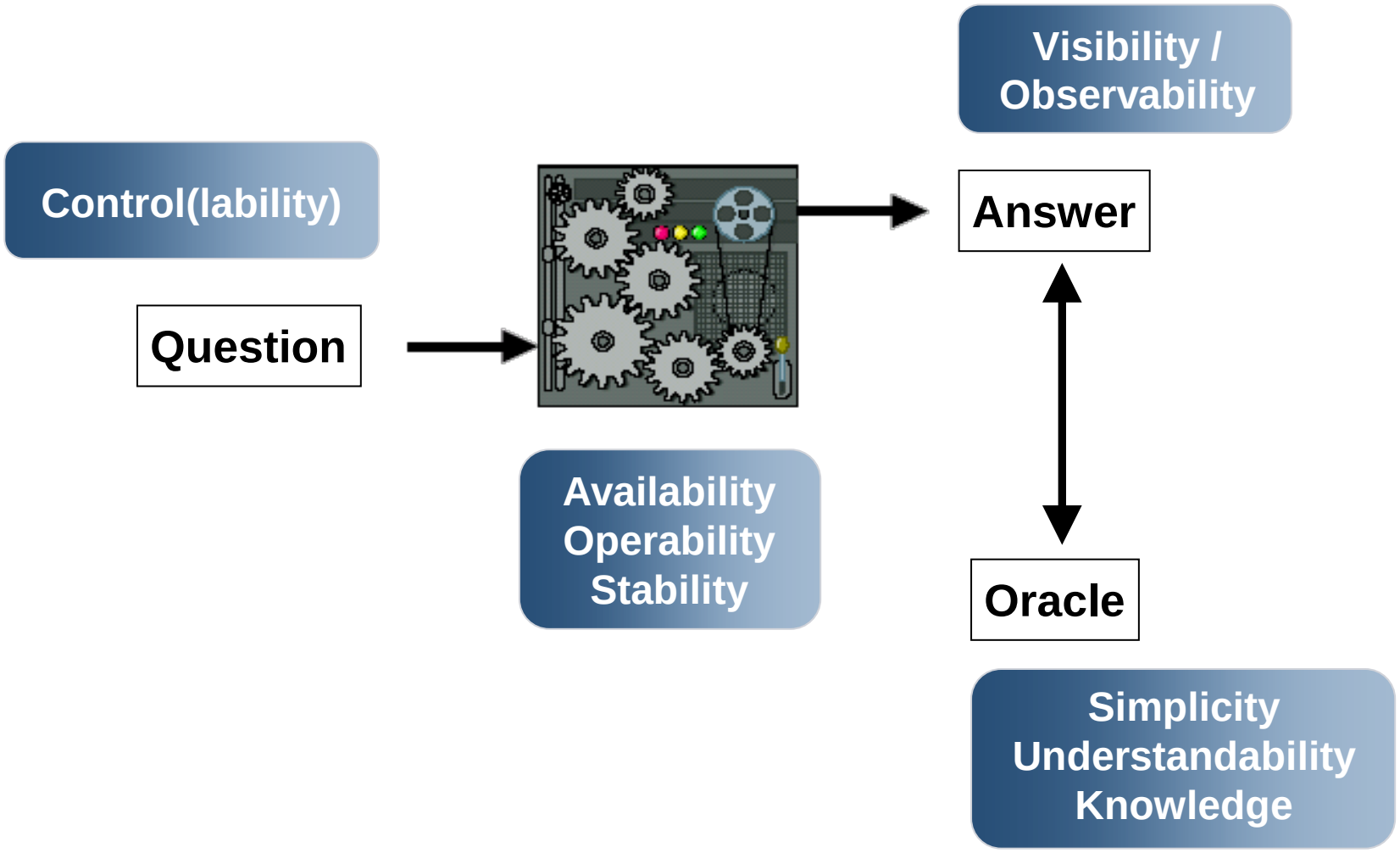
SIEMENS

Categories of test design techniques

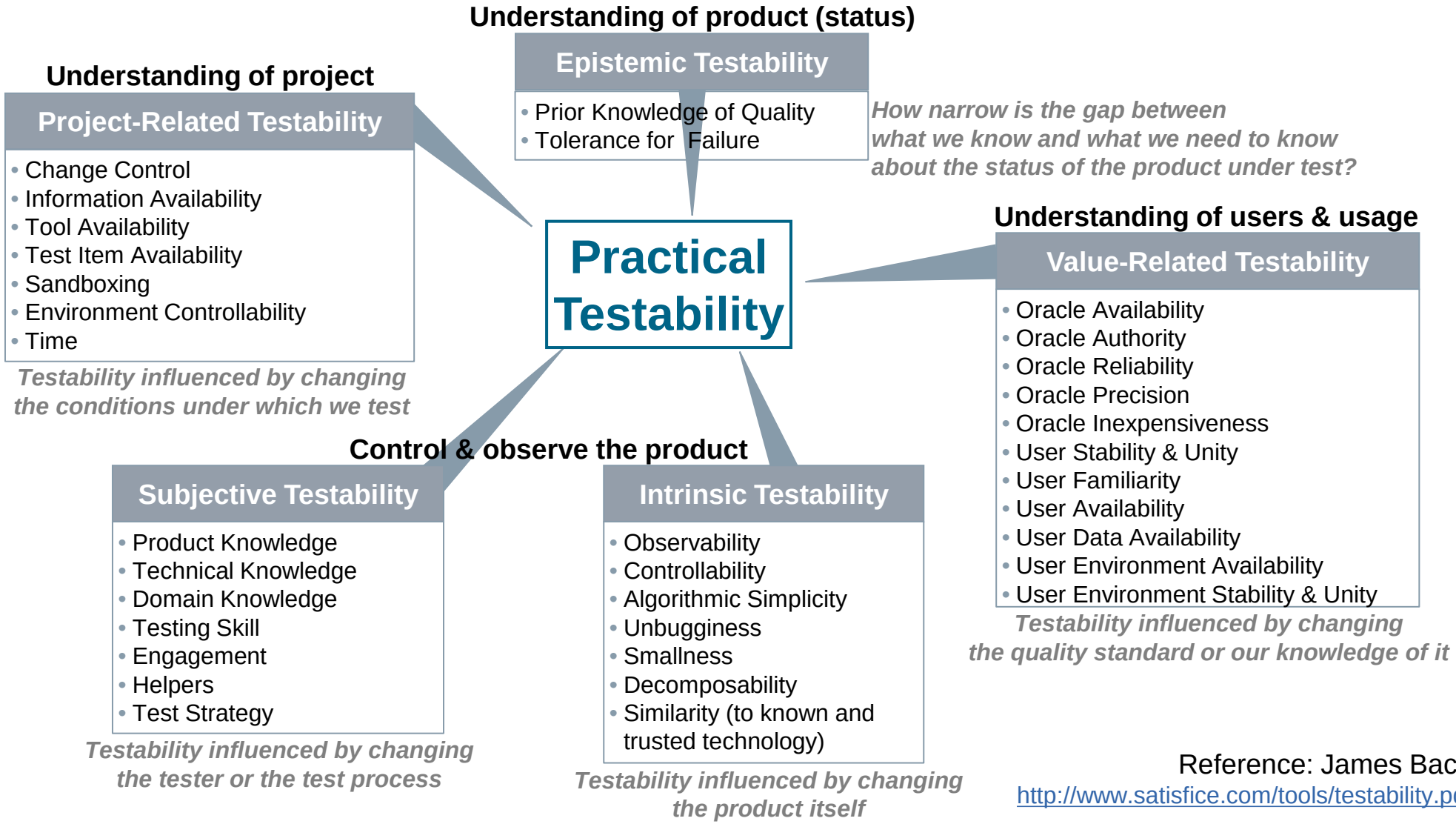




Testability – Factors (3)



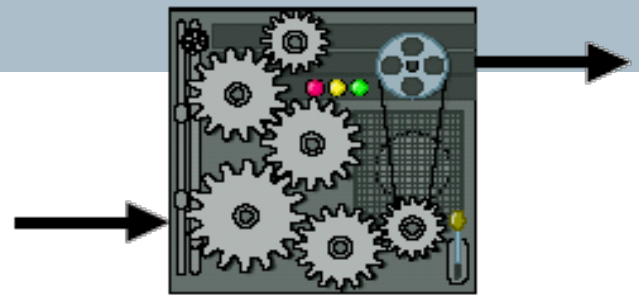
Heuristics of Testability



Reference: James Bach
http://www.satisfice.com/tools/testability.pdf

SIEMENS

Design for Testability (DfT) – How?



Educate stakeholders on benefit of DfT

Suitable test architecture, good design principles
Choice of technologies, libraries, frameworks, services

Interaction with the system under test through well-defined control points and observation points

Additional (scriptable) interfaces, ports, hooks, mocks, interceptors for testing purposes (setup, configuration, simulation, recovery)

Coding guidelines, naming conventions
Internal software quality (architecture, code)

Control(lability)

Visibility / observability

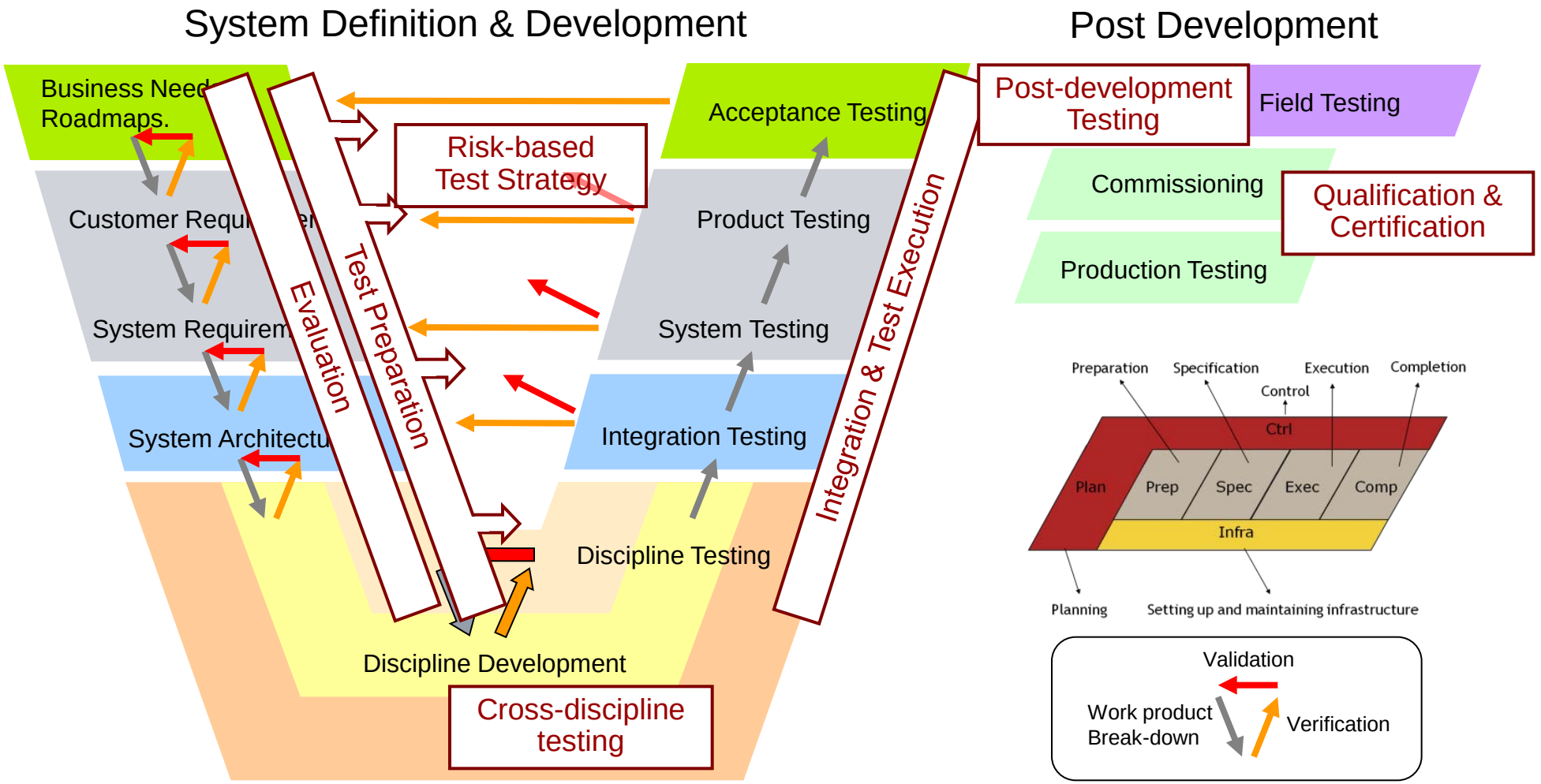
Built-in self-test (BIST), built-in test (BIT)
Consistency checks (assertions, design by contract, deviations)

Logging and tracing (AOP, counters, monitoring, probing, profiling)

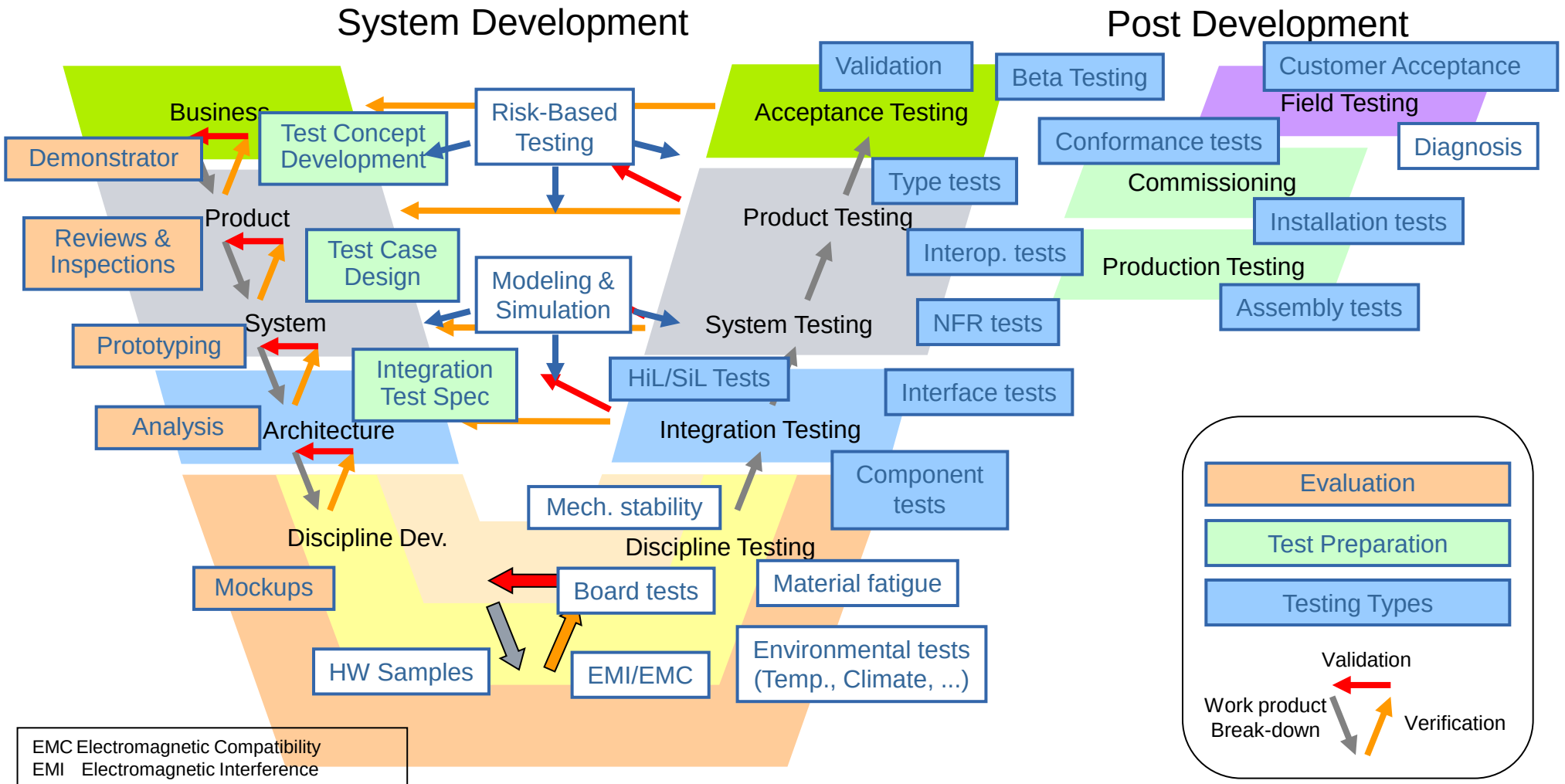
Diagnosis and dump utilities, black box (internal states, resource utilization, anomalies at runtime, post-mortem failure analysis)

Think test-first (xTDD): how can I test this?

Testing and evaluation throughout the lifecycle



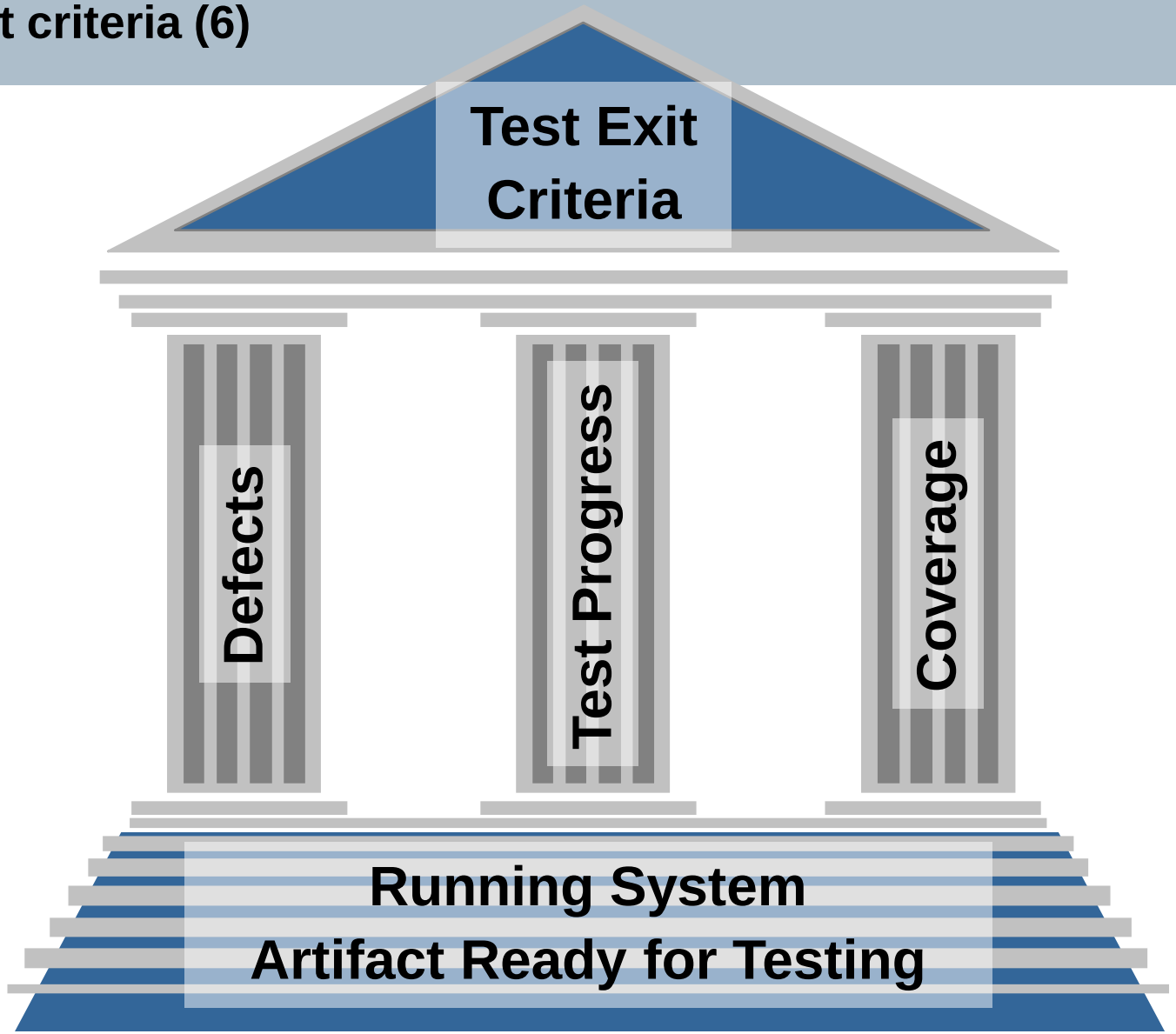
Example: Evaluation methods and testing types



EMC Electromagnetic Compatibility
EMI Electromagnetic Interference
HiL HW in the loop
HW Hardware (Electronic, Mechanic, …)
SiL Software in the loop

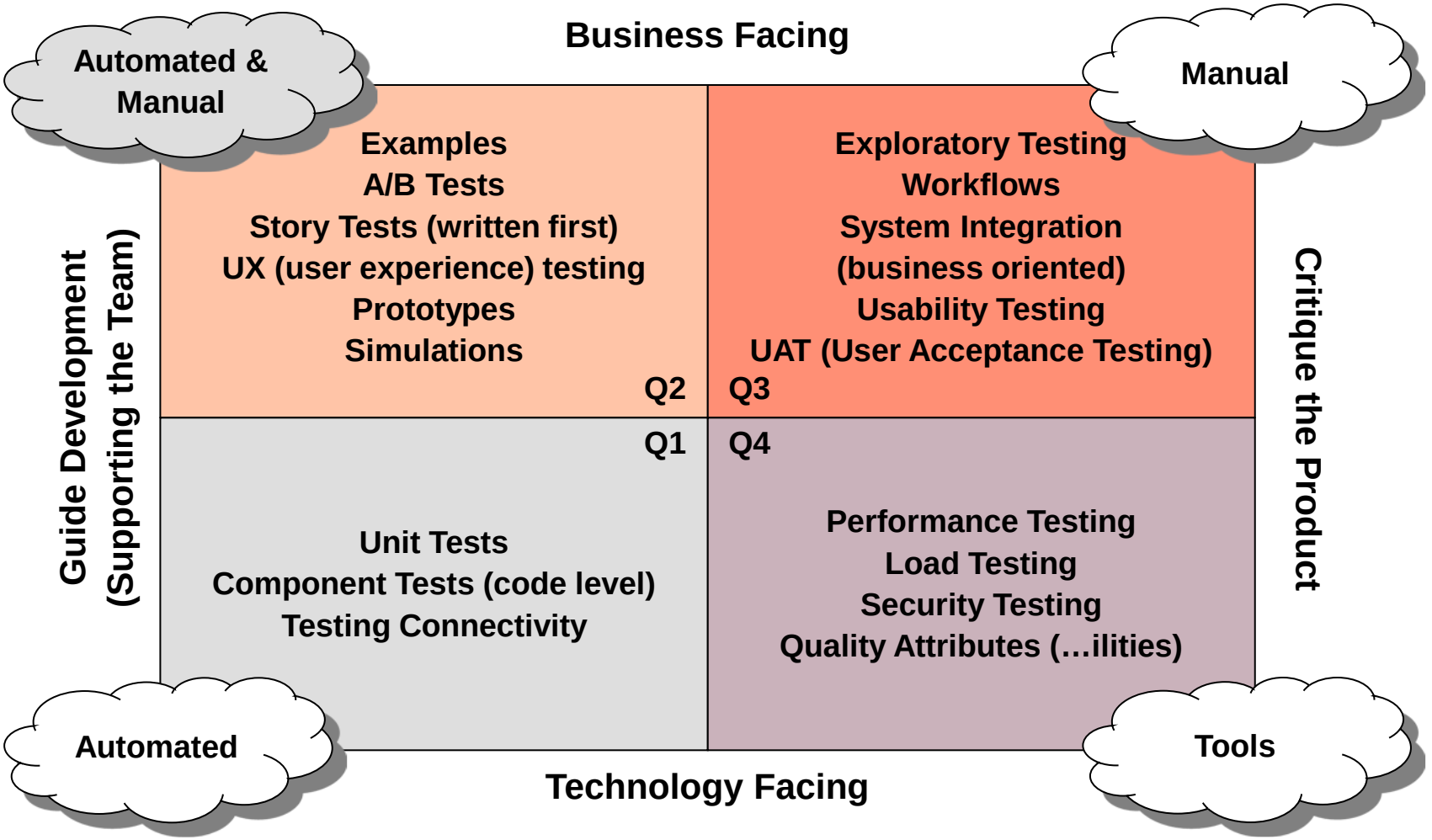
SIEMENS

Test exit criteria (6)



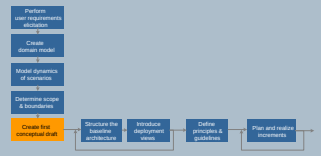
SIEMENS

Agile testing quadrants



Reference: Brian Marick, Lisa Crispin, Janet Gregory

Domain model is fundamental!



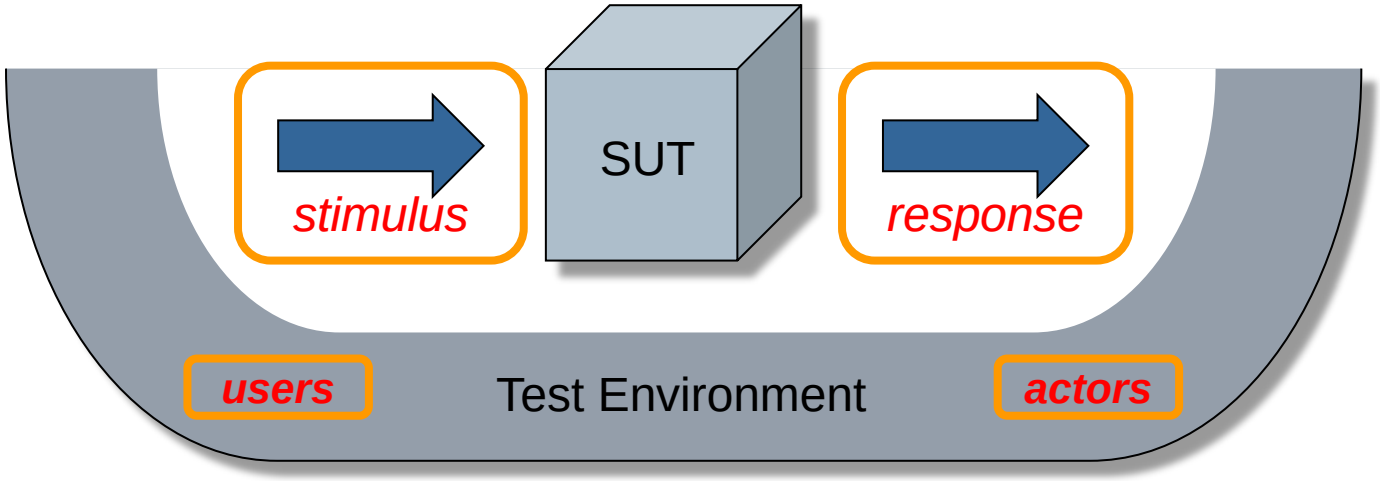
For the product / system it clearly defines the **boundaries**

what is **IN** and what is **OUT** **of scope**

Your test system is a companion, you have to cover all the **other** elements,

e.g.

- Human user interaction (simulated or real)
- Communication with other actors (external systems, …)
- Identify capabilities of these elements and design test cases accordingly



Architecture in a nutshell

"Architecture is about everything costly to change"

Main activities

- Taking decisions, based on **architecture significant requirements (ASR)**, aka. **architectural drivers**
- **Structuring** and **partitioning** the **technical realization**, based on domain knowledge
- Defining **interfaces** and **functional responsibilities**
- Assigning **functional components** to **hardware deployments**
- Documenting and disseminating **architecture descriptions**

Architecture governance

- Make sure that the **implementation follows the architecture**
- **Supporting** and **coaching** other **stakeholders**

➔ TeA works in close collaboration with other architects!

Why creating an architecture for a test system?

Test architect has responsibility to create a test environment that **other stakeholders** can use

- Unit test environment: developer creates test cases
- Test automation

Your test environment may become a **product**!

Integrators, factory, CS, key customers, … can use your test environment

- Test environment requires product quality
- Requirement to create copies of the test environment in an efficient way
- Test the test environment for effectiveness

Test architect is the owner of vital **properties of test environment**

- Functional correctness of test execution
- Determinism (repeatability, dependability), extensibility, adaptability, cost efficiency, performance, configurability, availability

➔ Follow a systematic method to satisfy these qualities!