

# Quality Assurance for Service-based Systems

From Software Engineering to  
Service Engineering

**Andreas Metzger**

Software Systems Engineering

Institute for Computer Science and  
Business Information Systems (ICB)

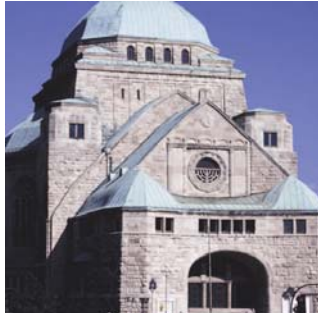
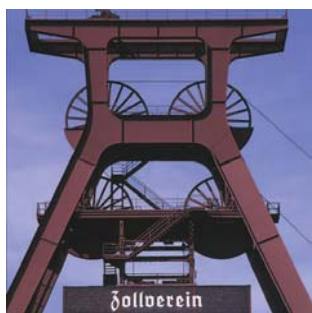
University of Duisburg-Essen, Germany

[www.sse.uni-due.de](http://www.sse.uni-due.de)

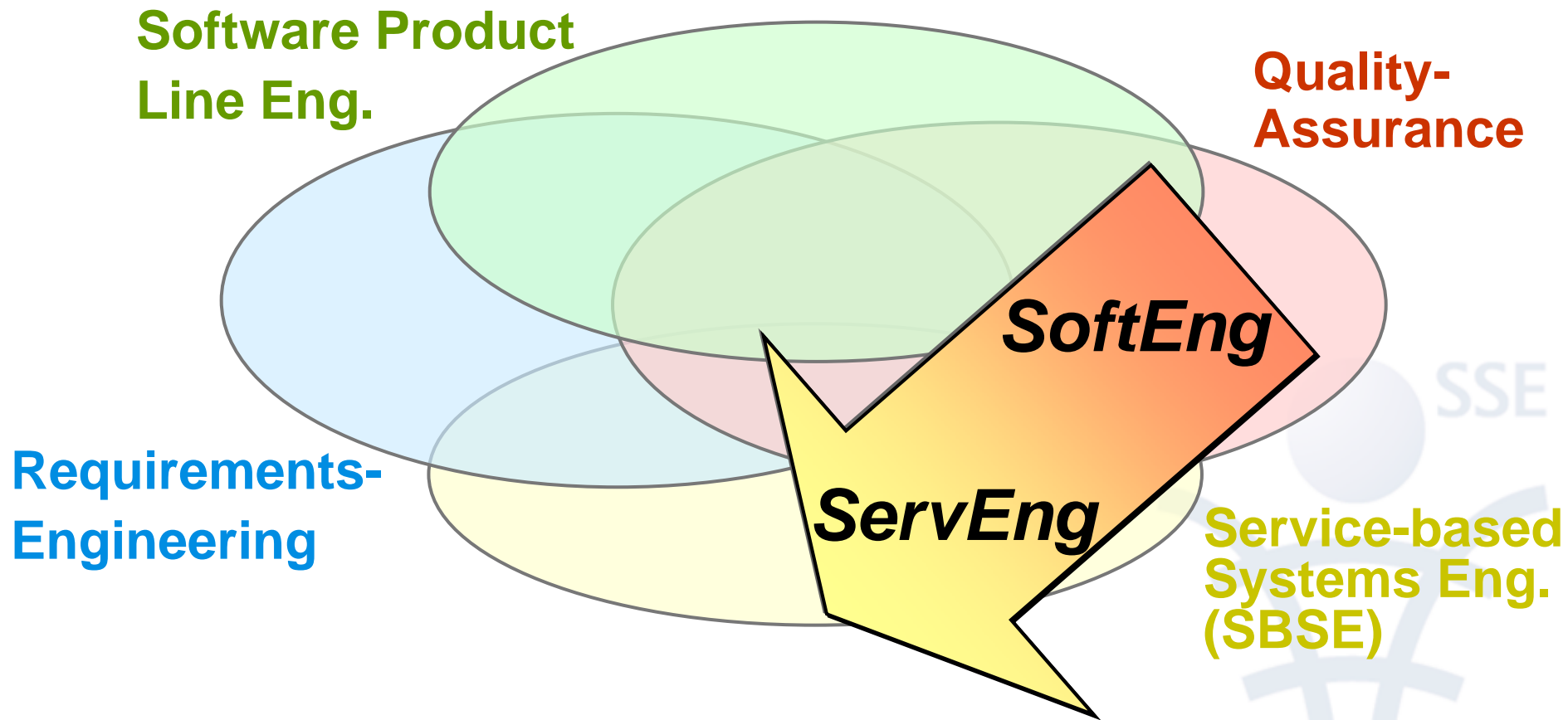


# Software Systems Engineering (SSE) The Research Group

- **Head:** Prof. Dr. Klaus Pohl
- Located at
  - University of **Duisburg-Essen**
  - **Essen** Campus
  - Part of **Institute for Computer Science and Business Information Systems (ICB)**



# Software Systems Engineering (SSE) Our Competencies and Experience



# Motivation

## Some recent software and service “bugs”

- **2006:** U.S. government student loan service erroneously **made public personal data of 21,000 borrowers**. [URL 1]
- **2006:** Software error resulted in **overbilling of several thousand dollars to each of 11,000 customers** of major telecom. company. [URL 1]
- **2007:** Problems in payroll system for a large urban school system lead to one third of employees receive incorrect pay checks, resulting in **overpayments of \$53 million, as well as underpayments**. [URL 1]
- **2009:** The social mini-blog platform „Twitter“ was haunted by a worm, that was able to **infiltrate the profiles of registered users, proliferate itself, and send messages of questionable content under different names**. The worm made use of a security loop-hole that made Twitter vulnerable for cross-site scripting attacks [URL 2]
- **2009:** A bug caused inconvenience to Google users by showing the message **‘This site may harm your computer’ along with each and every search result**. [URL 3]

# Aims of this Presentation

- **What kind of “bugs” are there?**
  - Fundamental concepts: failure, fault and error, ...
- **How to find “bugs”?**
  - Quality assurance techniques and synergies
- **When and why to look for “bugs”?**
  - Quality assurance processes and phases

# Agenda

- **Fundamentals**
- Quality Assurance Techniques
- Quality Assurance Processes
- Summary and Outlook



# Fundamentals - *SoftEng*

## Failure, Fault and Error

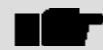


**Failure** = deviation of the functionality / quality of the software from its expected functionality / quality;

*e.g., accounting software returns wrong balance of account*

failure

can lead to



**Fault** = defect in a software artefact;

*e.g., wrong initialization of loop variable (e.g., 1 instead of 0).*

fault

leads to



**Error** = Mistake or misunderstanding of a developer when creating a software artefact;

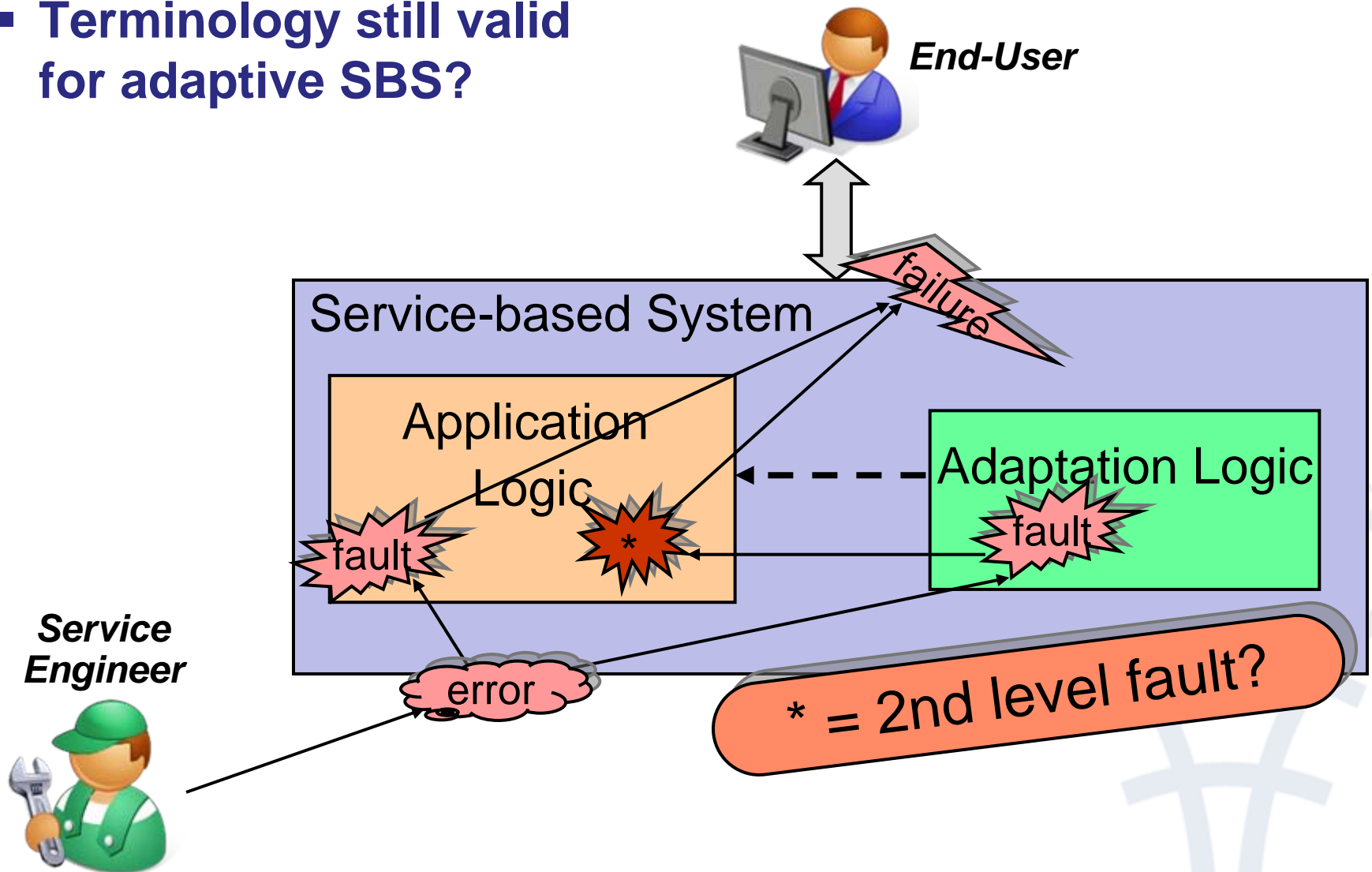
*e.g., overlooked that arrays in C++ start at index 0*

error

[Spillner, 2007]

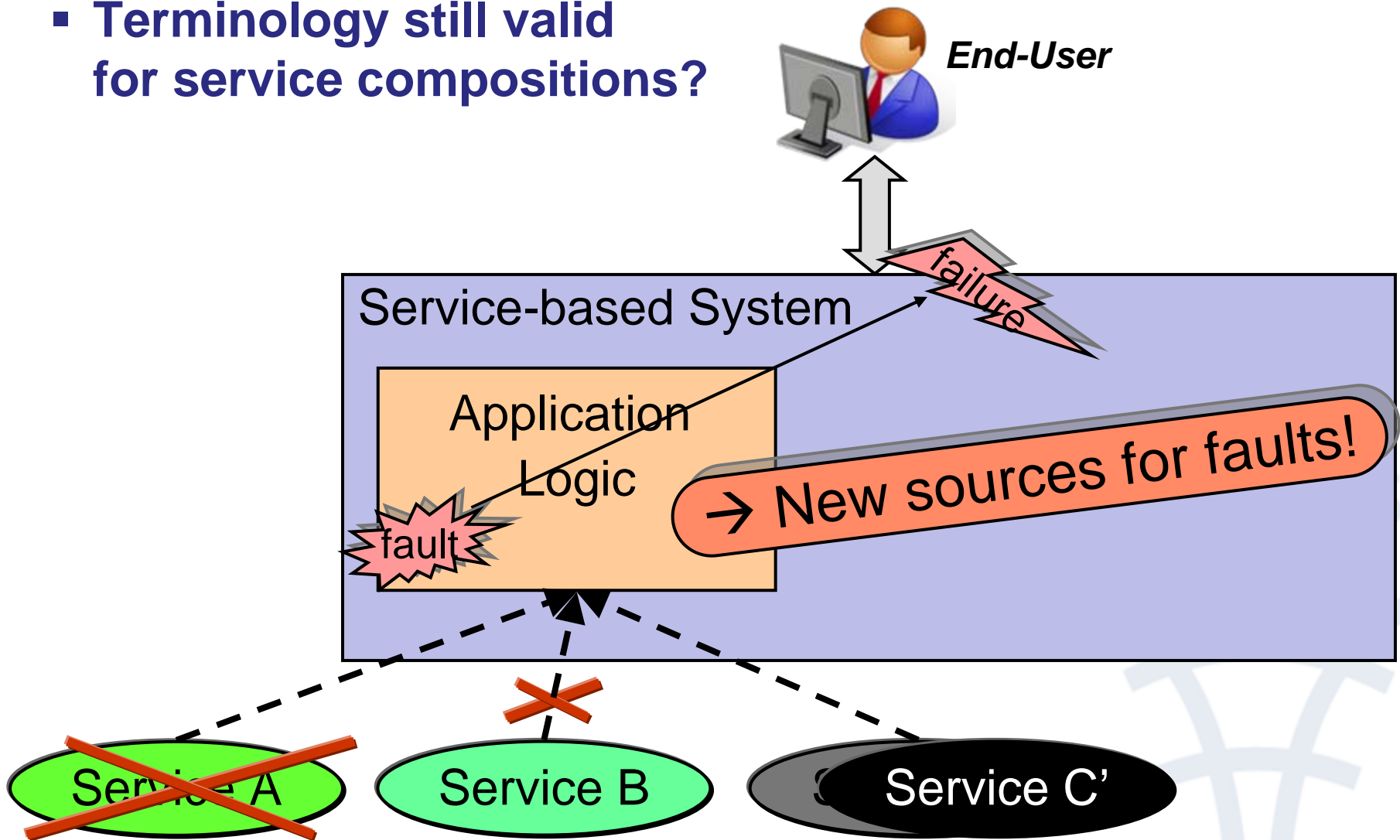
# Fundamentals - *ServEng* Failure, Fault and Error (1)

- Terminology still valid for adaptive SBS?



# Fundamentals - *ServEng* Failure, Fault and Error (2)

- Terminology still valid for service compositions?



# Fundamentals

## ServEng Research Challenges

- What types of service-specific “bugs” are possible?
  - **Adaptation bugs**
    - How can we define and refine an „adaptation fault taxonomy“ to support QA for adaptation?
    - Which novel QA techniques are needed for discovering adaptation „bugs“? [Kazhamiakin et al. 2008]
  - **Bugs due to external service invocation**
    - How to uncover faults (“debugging”) if those are implied by the use of 3rd party services?
- ...



# Agenda

- Fundamentals
- **Quality Assurance Techniques**
- Quality Assurance Processes
- Summary and Outlook



# Quality Assurance Techniques - *SoftEng*

## Overview: Classification

- **Dynamic Checks:** Examine individual executions of the software

*Examples:*

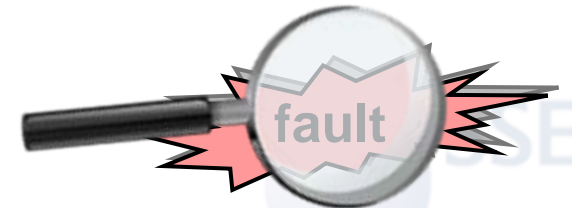
- *Testing*
- *Monitoring*
- *Profiling*



- **Static Analysis:** Examine software artefacts without execution

*Examples:*

- *Reviews, walkthroughs and inspections*
- *Correctness proofs*
- *Model checking*



- **Error Abstraction:** Understand the “patterns” of mistakes



(based on [Liggesmeyer, 2002])

# Quality Assurance Techniques - *SoftEng*

## Overview: Assessment

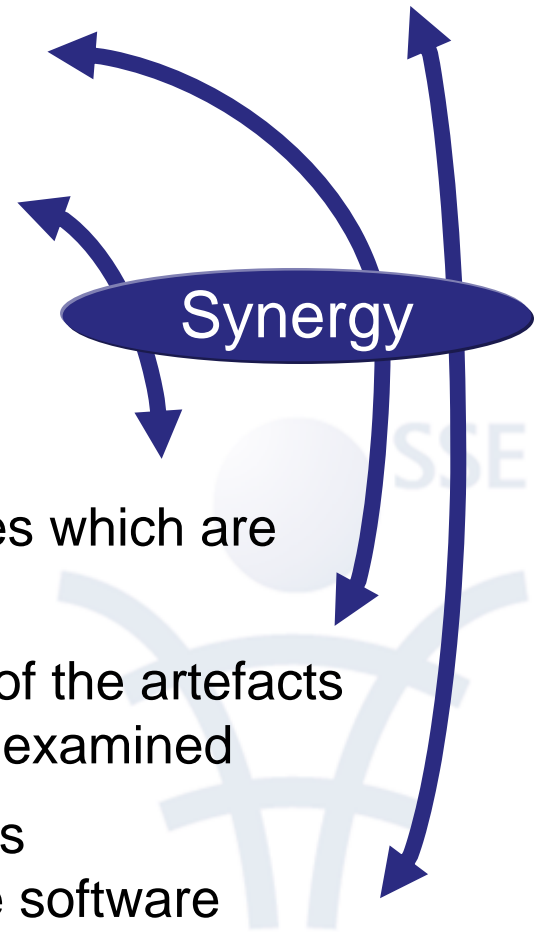
### ■ **Dynamic Checks**

(based on [Liggesmeyer, 2002])

- + software can be checked in real usage contexts and with real inputs
- dynamic checks cannot (except for trivial cases) check all potential execution traces
- executable software artefacts must be available
- needs “debugging” to determine the faults

### ■ **Static Analysis**

- + can be applied to all kinds of artefacts (even ones which are not executable)
- + more universal statements about the properties of the artefacts possible, because classes of executions can be examined
- relevant aspects might be overlooked as analysis typically is based on a model (abstraction) of the software



## Two Important Dynamic Checks: Testing vs. Monitoring

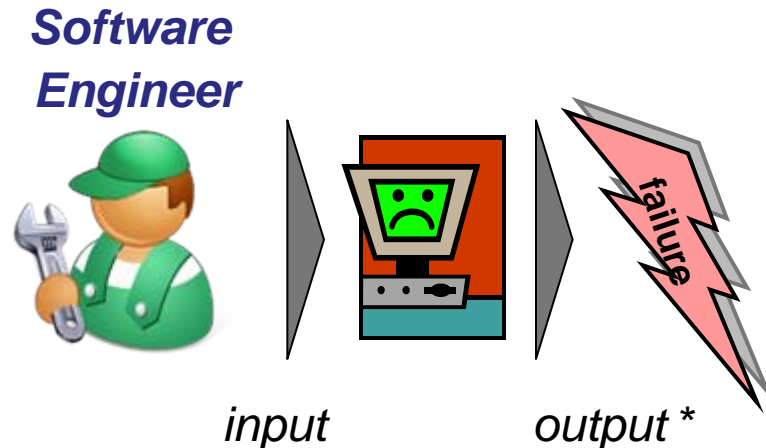


# Quality Assurance Techniques

## Two Important Dynamic Checks: Overview

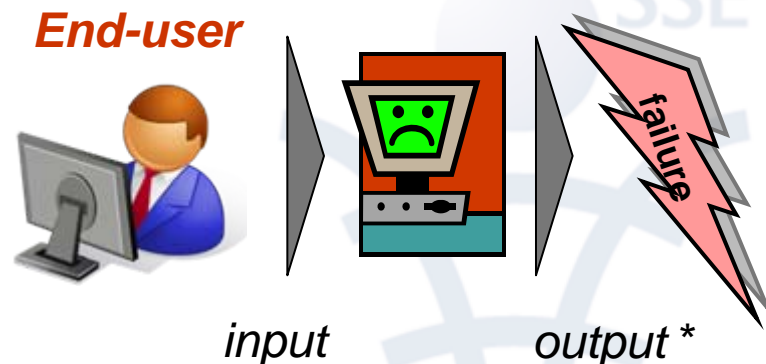
### ■ Testing (prominent in *SoftEng*)

- Systematically execute the software
  1. **Software is fed with concrete pre-determined inputs (test cases)**
  2. Produced outputs\* are observed
  3. Deviation = failure



### ■ Monitoring (prominent in *ServEng*)

- Observe the software during its current execution (i.e., actual use / operation)
  1. **End-user interacts with the system**
  2. Produced outputs\* are observed
  3. Deviation = failure



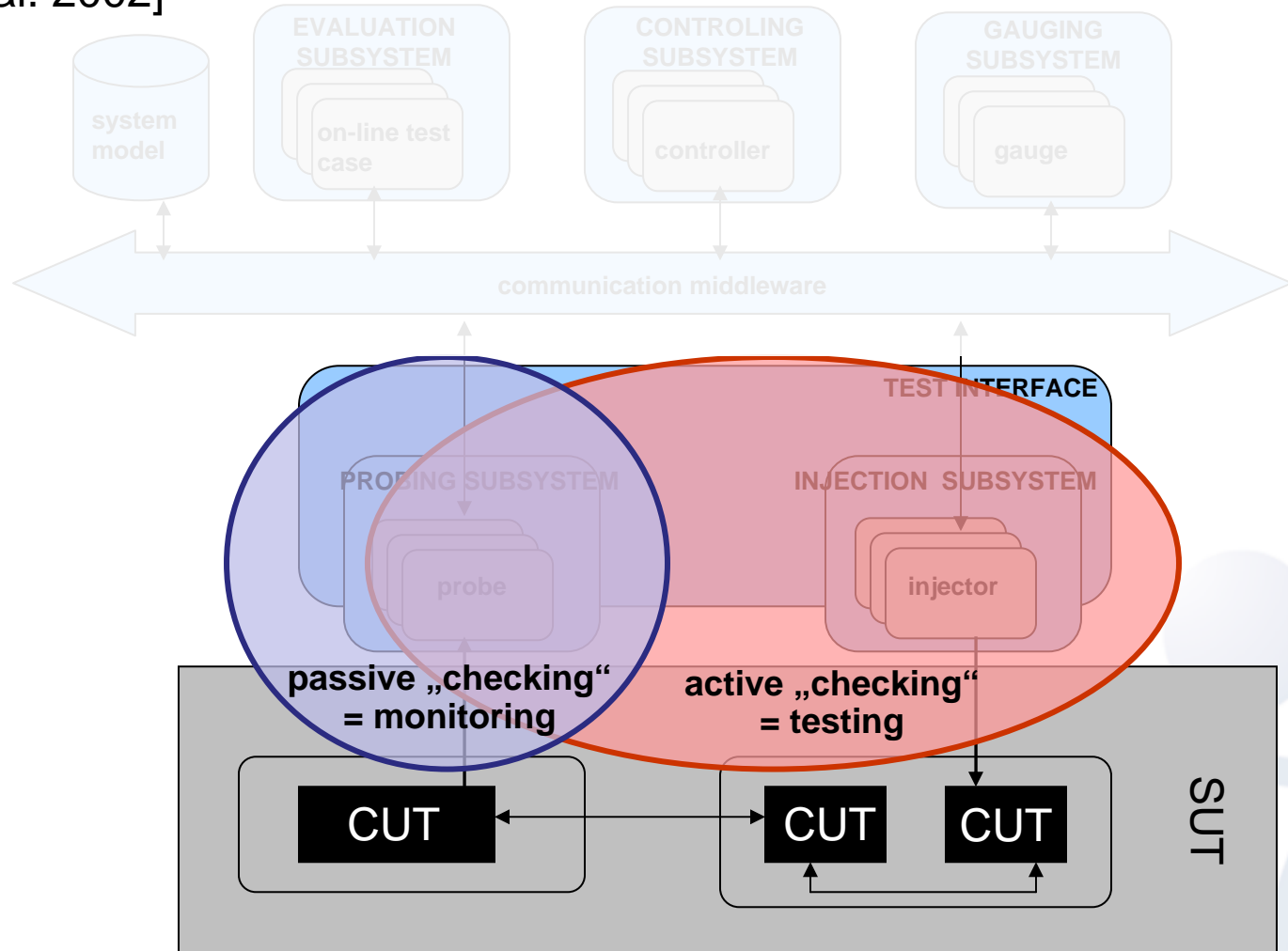
[PO-JRA-1.3.1; S-Cube KM]

\* incl. internal data collected for QA purposes

# Quality Assurance Techniques

## Two Important Dynamic Checks: “Architecture”

[Deussen et al. 2002]



SUT = system under “test”; CUT = “component” under “test”

# Quality Assurance Techniques

## Two Important Dynamic Checks: Assessment

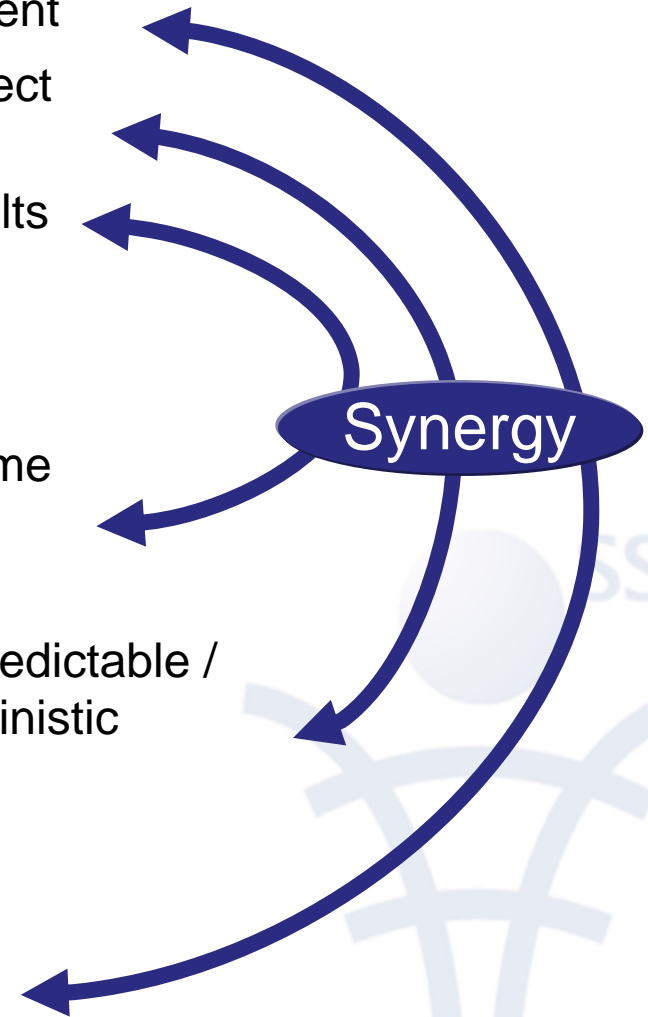
### ■ Testing

(based in part on [Canfora & Di Penta, 2006])

- + Testing can uncover failures before deployment
- + Testing can systematically cover the test object (e.g., branch coverage)
- Testing cannot guarantee the absence of faults
- Testing can have side effects (it is “active”)

### ■ Monitoring

- + Monitoring can uncover failures during run-time which have escaped testing and can trigger adaptations (recovery actions)
- Coverage of monitoring is not systematic / predictable / deterministic (because based on non-deterministic user inputs / interactions)
- Monitoring can have side effects (e.g., performance impacts of “probes”)
- Late (in the life-cycle) discovery of failures



# Quality Assurance Techniques

## ServEng Research Challenges

### ■ Testing Challenges

(based in part on [DiNitto et al. 2008])

- How to we cater for the fact that **services are under the control of different parties**?
  - How can we test a service which is under the control by an external service provider?
  - How can we extend service interfaces for test purposes while still maintaining key properties like isolation, information hiding and loose coupling?
- How can we address the fact that during design-time **not all services of the SBS are available** (i.e., design-time testing feasible)?
- How can we provide **adequate test coverage** if services do not allow access to “internals“ (i.e., no code-based coverage possible)?
- How to compensate for the fact that **services cannot be tested in all possible compositions** due to potentially unbound number of service compositions (i.e., difficulties for integration testing)?
- ...

# Quality Assurance Techniques

## ServEng Research Challenges

### ■ Further QA Challenges

(based in part on [DiNitto et al. 2008])

- How can we **exploit the synergies** between testing and monitoring?
  - What benefits does **online (run-time) testing** provide?
  - How to enable online (run-time) testing?
- How can we exploit other **QA techniques during run-time**?
  - How to devise run-time (model) analysis techniques?
  - How to keep models in sync with the system?
- How do we do **debugging / testing without side-effects**?
- Can we **“monitor” faults** (“preventive” monitoring)?
- ...



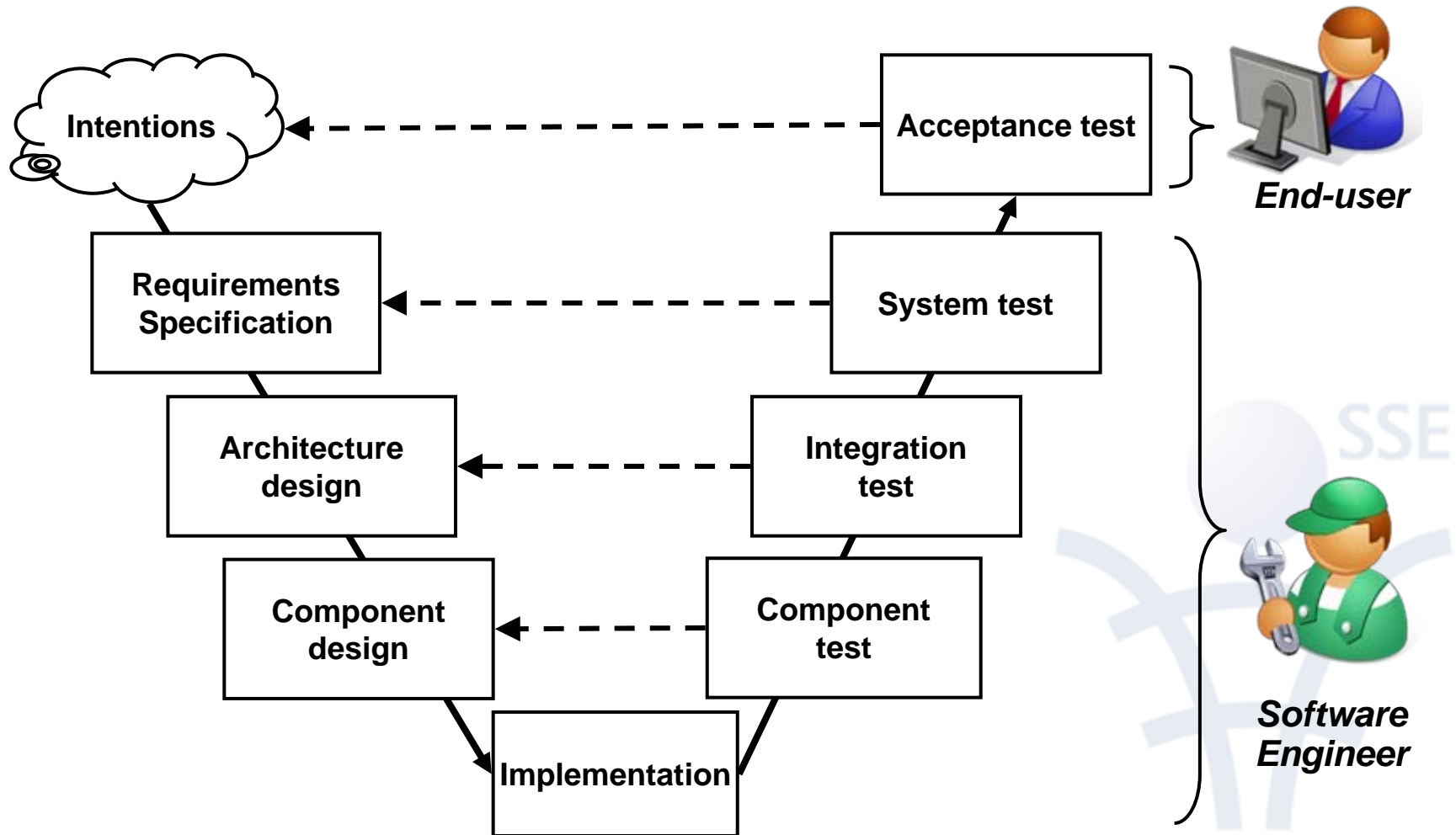
# Agenda

- **Fundamentals**
- **Quality Assurance Techniques**
- **Quality Assurance Processes**
- **Summary and Outlook**



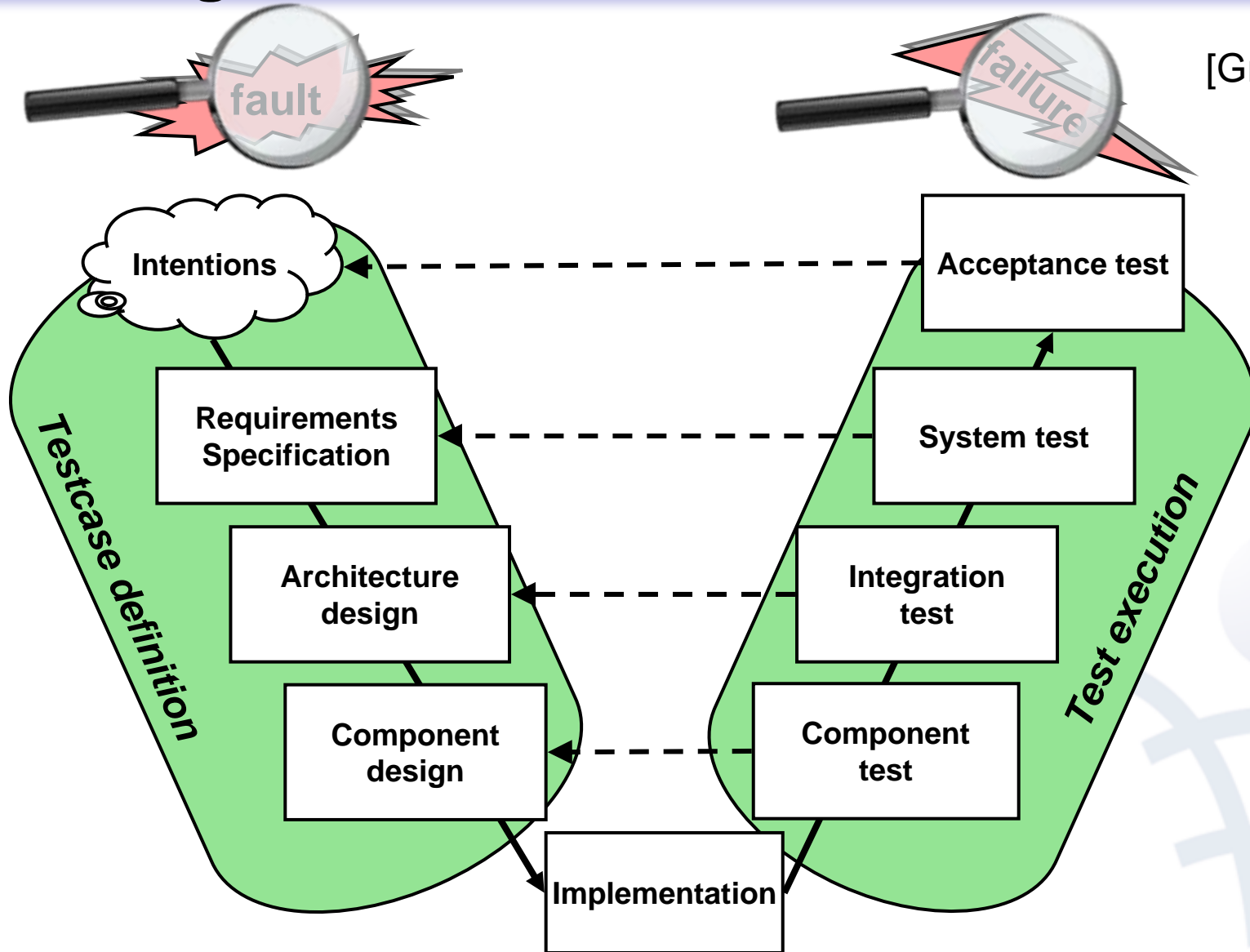
# Quality Assurance Processes - *SoftEng* Software Life-Cycle Model: V-Model

*Note: This model does not prescribe a „waterfall“ approach!*

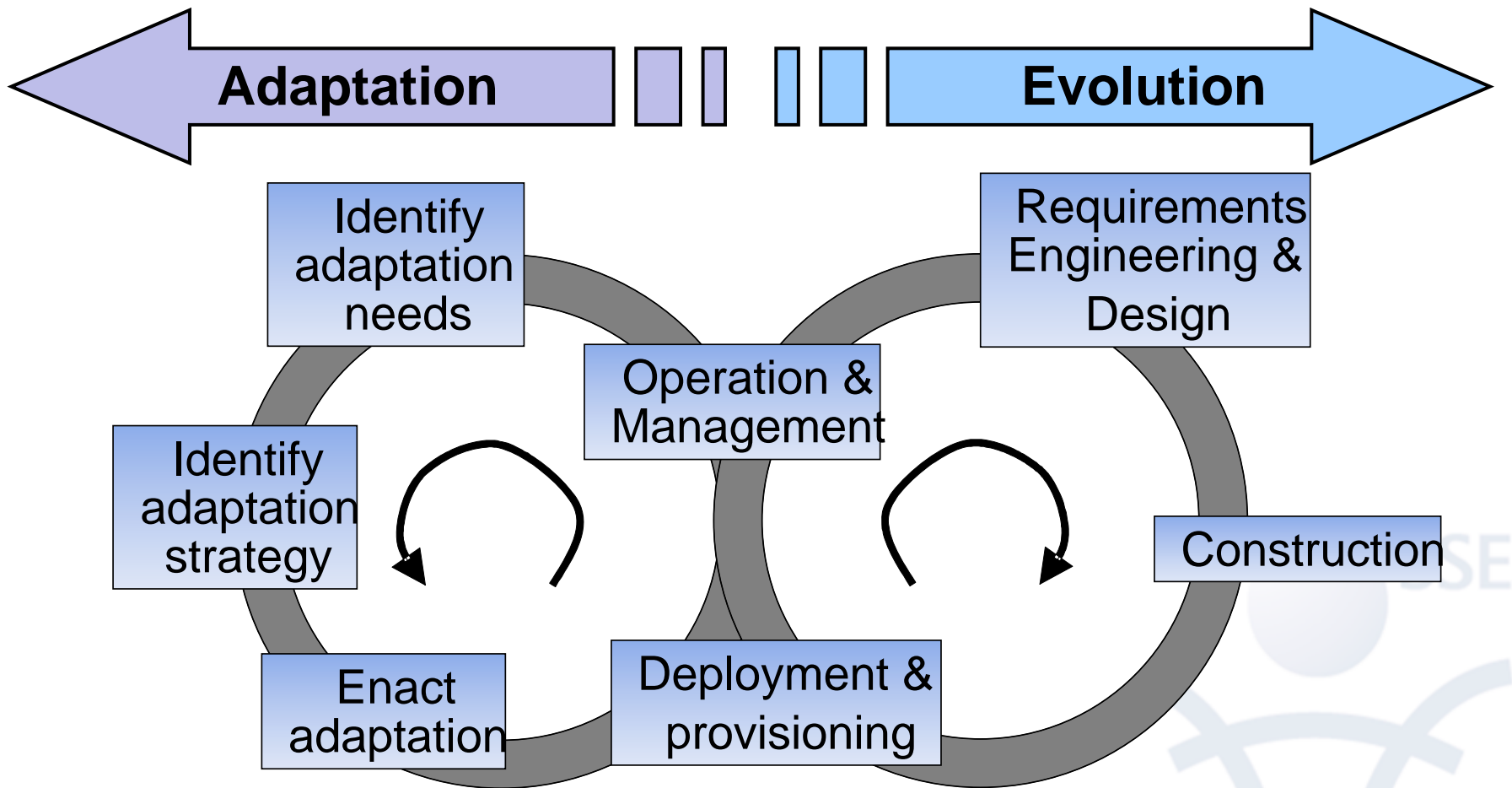


# Quality Assurance Processes - *SoftEng* Testing Activities in the V-Model

[Graham, 2002]



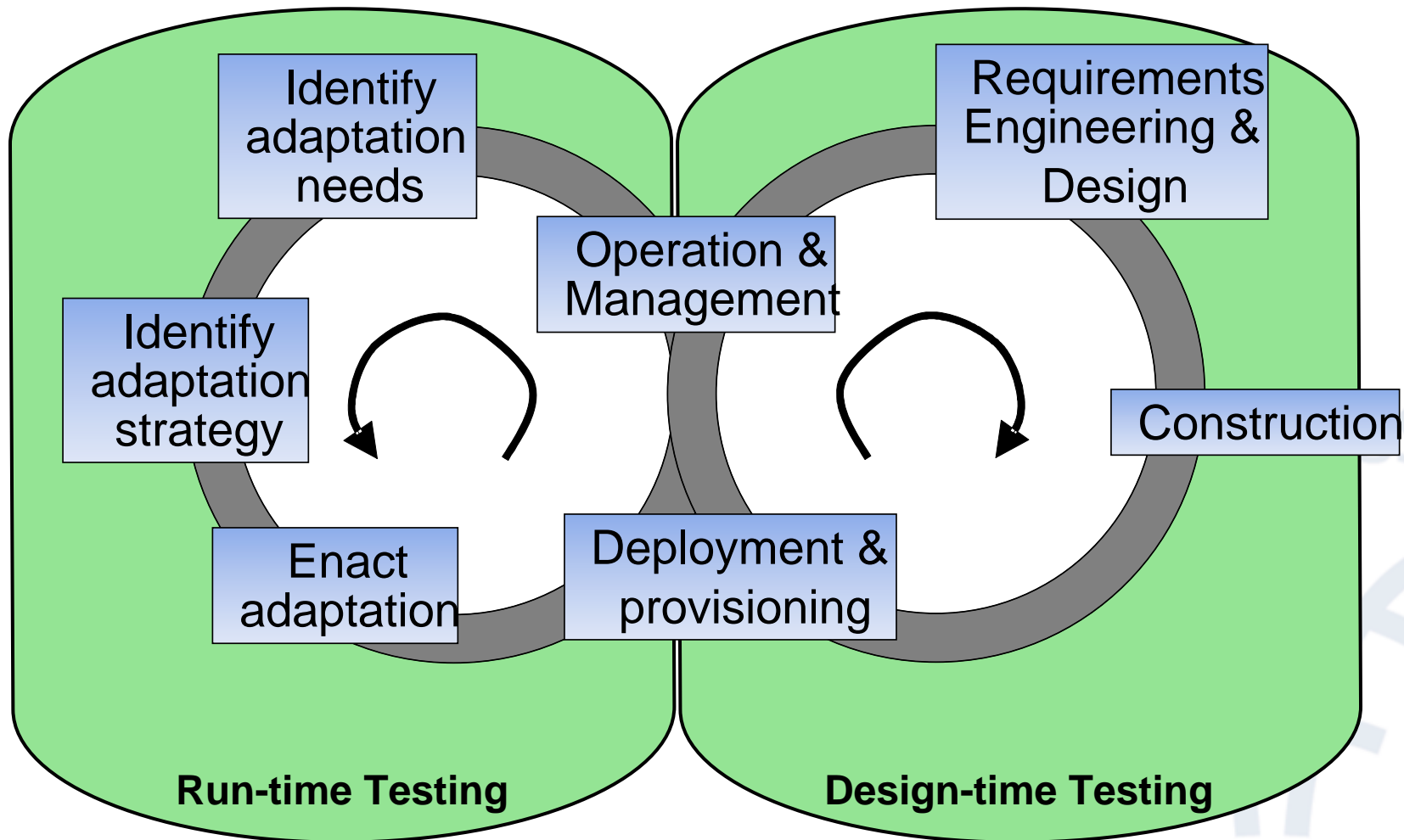
# Quality Assurance Processes - *ServEng* S-Cube Lifecycle Model



[PO-JRA-1.1.1]

# Quality Assurance Processes - *ServEng*

## Testing Activities in the S-Cube Lifecycle Model



# Quality Assurance Processes - *ServEng*

## Design-Time Testing Activities



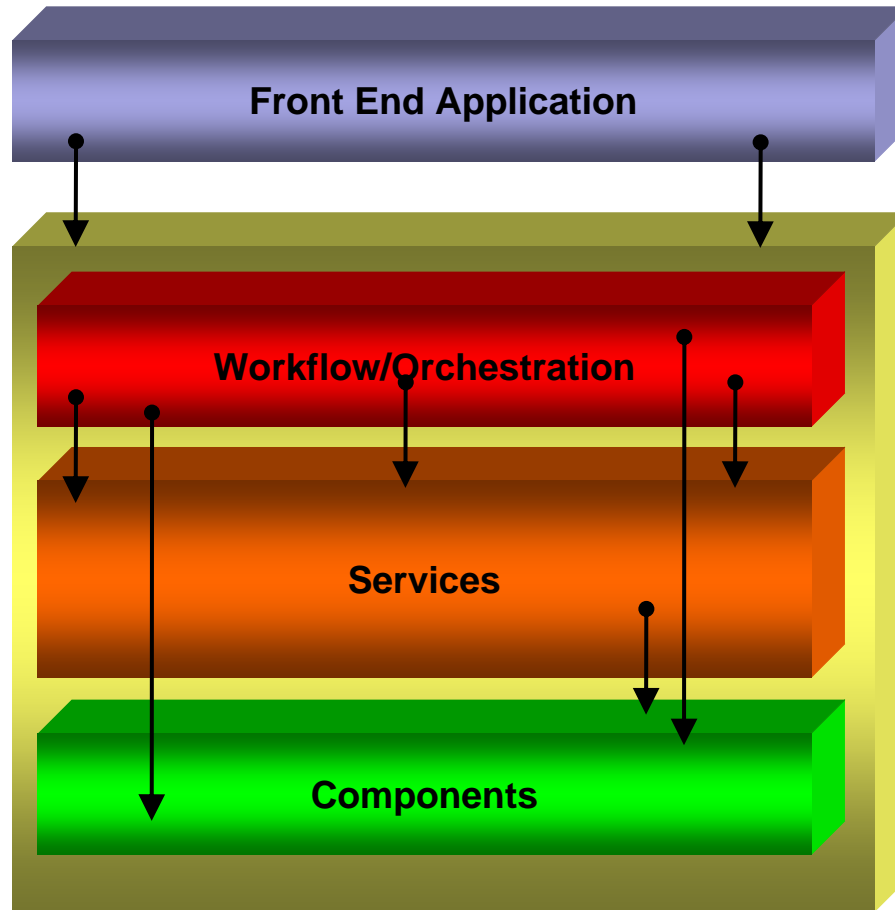
**End-user**



**Service Engineer  
@ Integrator**



**Service Engineer  
@ Provider**



**System test:** tests that the SOA solution has delivered the business requirements & acceptance criteria

**Workflow test:** ensures that services are operating collectively as specified

**Integration test:** determines if interface behaviour & information sharing between the services, are working as specified

**Service test:** ensures that the service is meeting project requirements + business & operational requirements of other services using that service

**Component test:** unit testing, e.g., basic functionality of components & functions within a service work as specified

[Linthicum, 2007]

# Quality Assurance Processes

## ServEng Research Challenges

- How to support **QA during run-time**?
  - How to integrate quality assurance in the service life-cycle?
  - Which run-time QA activities to provide and when?
  - How to enable the **prediction of quality for pro-active adaptation**?
  - What about regression testing for service evolution / continuous testing?
  - How and when to ensure the quality of adaptation?
- How to exploit **synergies between QA techniques** along the life-cycle?
  - What to do during design-time to be exploited during run-time?
- ...

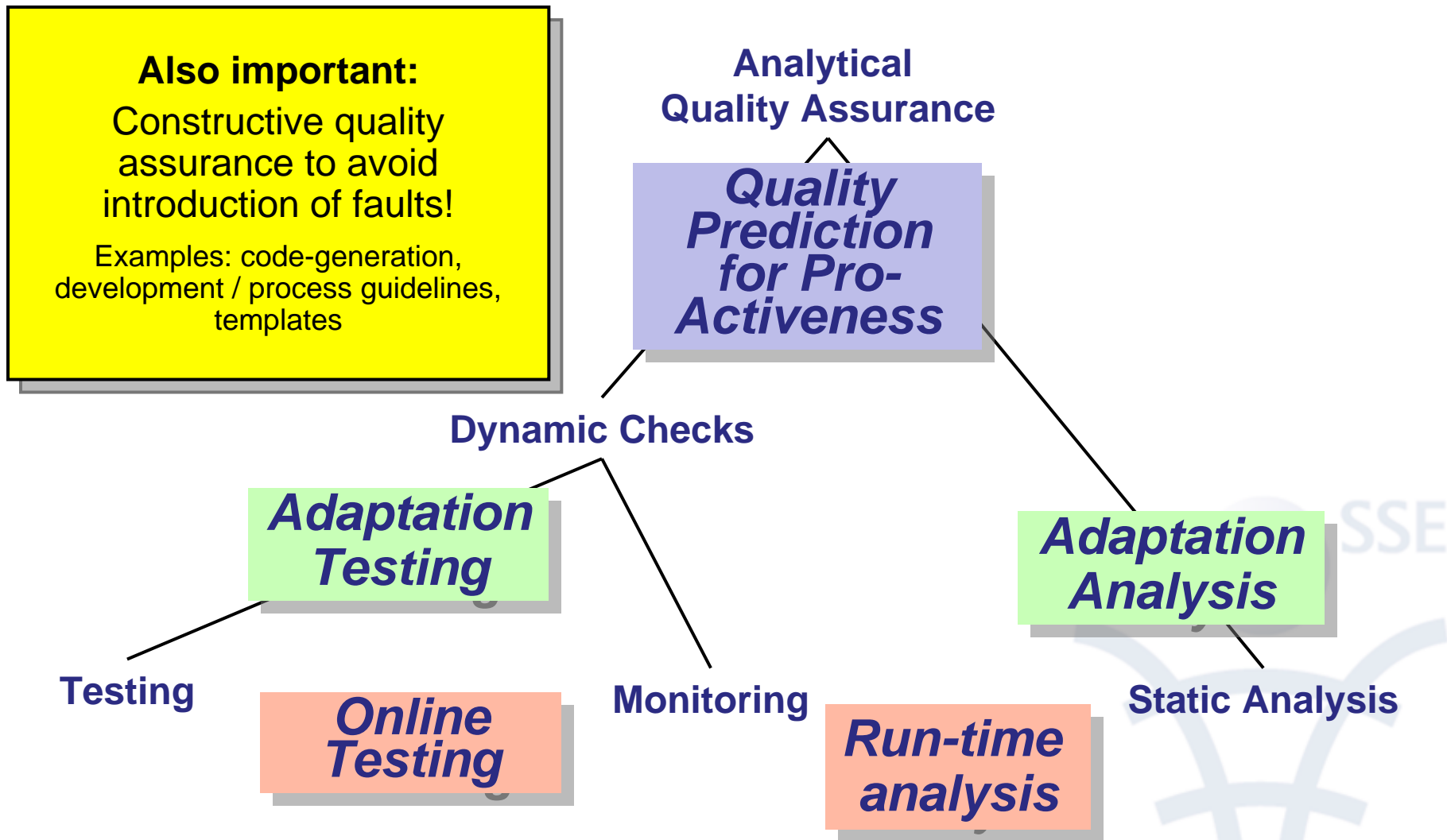
# Agenda

- **Fundamentals**
- **Quality Assurance Techniques**
- **Quality Assurance Processes**
- **Summary and Outlook**



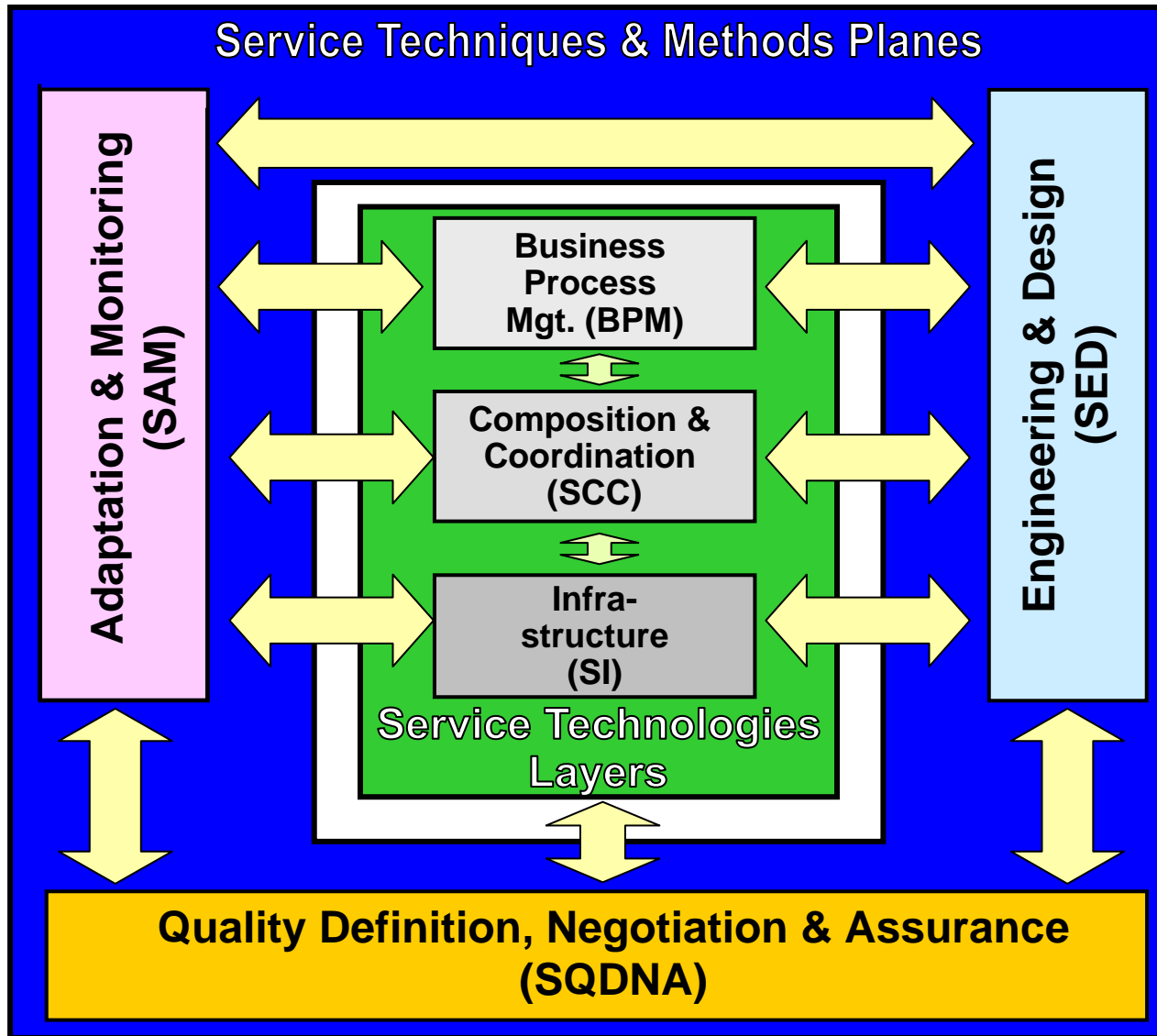
# Summary and Outlook

## QA Techniques and Challenges



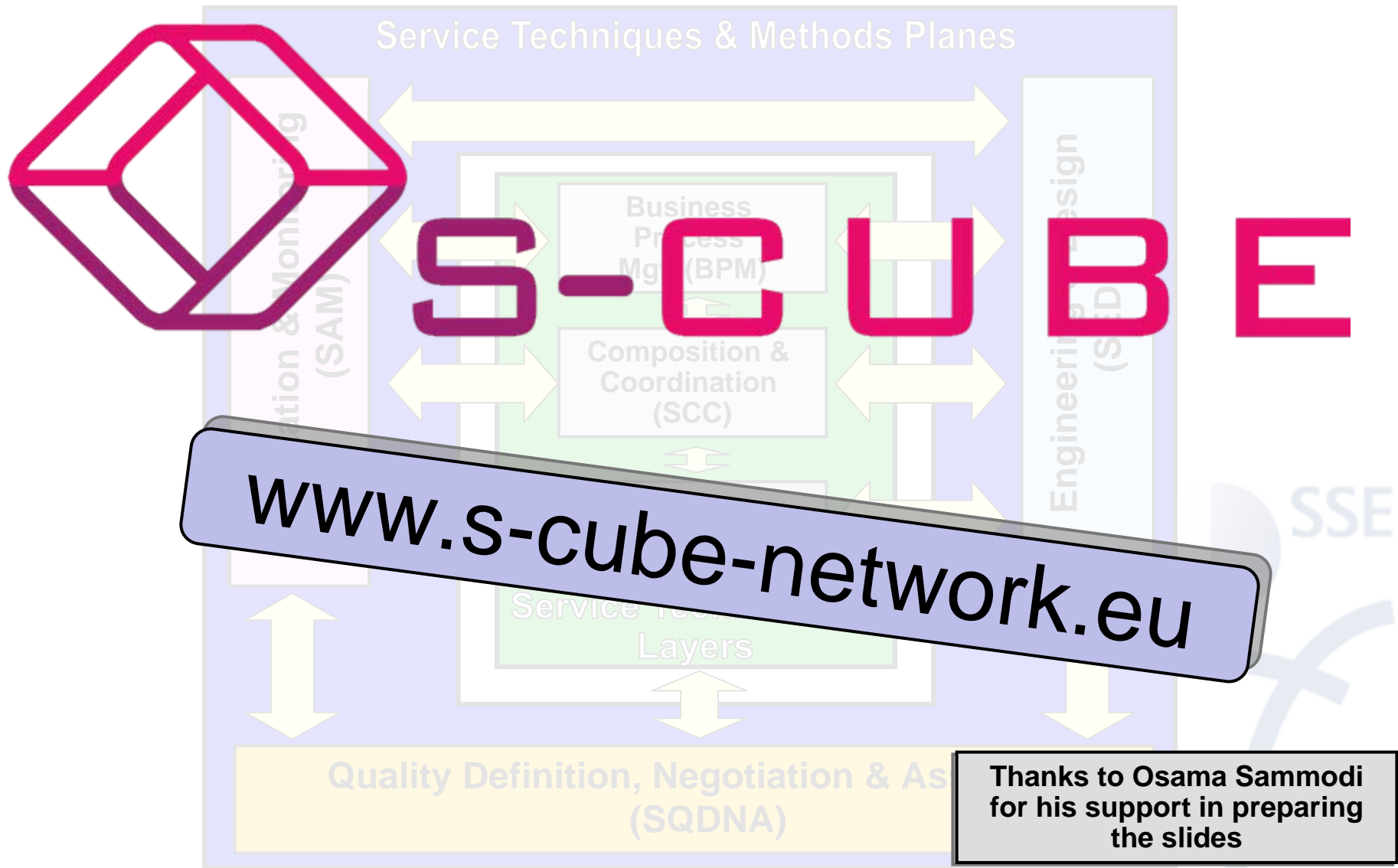
# Summary and Outlook

## Addressing the Challenges in S-Cube



# Summary and Outlook

## Addressing the Challenges in S-Cube



# References

- [**Canfora & Di Penta, 2006**] Canfora, G.; Di Penta, M.: SOA: Testing and Self-Checking. International Workshop on Web Services Modeling and Testing (WS-MaTe 2006).
- [**Deussen et al. 2002**] Deussen, P.H.; Din, G.; Schieferdecker, I.: *An on-line test platform for component-based systems*, Proceedings. 27th Annual NASA Goddard/IEEE Software Engineering Workshop, 2002.
- [**Di Nitto et al. 2008**] Di Nitto, E.; Ghezzi, C.; Metzger, A.; Papazoglou, M.; Pohl, K.: A Journey to Highly Dynamic, Self-adaptive Service-based Applications. Automated Software Engineering (2008).
- [**Graham, 2002**] Graham, D.: Requirements and Testing – Seven Missing-Link Myths. IEEE Software, Vol. 19, Nr. 5, IEEE Press, Los Alamitos, 2002, S. 15–17.
- [**Kazhamiakin et al. 2008**] Kazhamiakin, R.; Metzger, A.; Pistore, M.: Towards Correctness Assurance in Adaptive Service-Based Applications. In ServiceWave 2008, Nr. (5377), Springer, 10-13 December 2008.
- [**Liggesmeyer, 2002**] Liggesmeyer, P.: Software-Qualität – Testen, Analysieren und Verifizieren von Software, Spektrum Verlag, 2002.
- [**Linthicum, 2007**] SOA Meta Model, The Linthicum Group, 2007.
- [**PO-JRA-1.1.1**] S-Cube deliverable # PO-JRA-1.1.1: State of the art report on software engineering design knowledge and Survey of HCI and contextual knowledge; <http://www.s-cube-network.eu/results/deliverables/wp-jra-1.1>.
- [**PO-JRA-1.3.1**] S-Cube deliverable # PO-JRA-1.3.1: Survey of Quality Related Aspects Relevant for Service-based Applications; <http://www.s-cube-network.eu/results/deliverables/wp-jra-1.3>.
- [**S-Cube KM**] S-Cube Knowledge Model: <http://www.s-cube-network.eu/knowledge-model>
- [**Spillner, 2007**] Andreas Spillner, Tilo Linz, Hans Schäfer. Software Testing Foundations: A Study Guide for the Certified Tester Exam. 2nd edition, Rocky Nook, 2007
- [**URL 1**] <http://www.softwareqatest.com/qatfaq1.html>
- [**URL 2**] [http://www.pcworld.com/article/163054/twitter\\_worm\\_a\\_closer\\_look\\_at\\_what\\_happened.html](http://www.pcworld.com/article/163054/twitter_worm_a_closer_look_at_what_happened.html)
- [**URL 3**] [http://googleblog.blogspot.com/2009\\_01\\_01\\_googleblog\\_archive.html](http://googleblog.blogspot.com/2009_01_01_googleblog_archive.html)