Fraunhofer IESE

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

SAA 06 – Architecture Evaluation

Dominik Rost
dominik.rost@iese.fraunhofer.de

TU Kaiserslautern, SS2018
Lecture "Software and System Architecture (SSA)"

Discussion

- **RECAP LAST LECTURE**

- Explain the contents of the last lecture
  - What were the topics?
  - Why do we need it?
  - How does it work?
  - How is it created, used, and/or evolved?

Example Architecture Evaluation

Architecture Evaluation Example

- The situation before the audit
  - Customer contracted a solution provider to deliver a new business information system to enter new markets with improved qualities and unique features
  - The solution provider designed an architecture of the business information system but outsourced implementation and testing

- After approx. three-years of development
  - Customer made high investments into new system
  - First prototypes of the final product were assessed as not satisfying
  - Solution provider already delivered behind schedule
  - Discussion with the solution provider seemed to be fruitful
  - The solution provider submitted an offer to finalize the system with new promises

Architecture Evaluation Example

- The questions – trigger of the audit
  - Is it worth to continue investing in this situation?
  - Can we trust the promises?
  - Will the prototypes mature over time?
  - Can our necessities be met with respect to functionality and quality?

→ Architecture evaluation audit

Architecture Evaluation Example – Actions Taken

- After the audit: customer **rejected the new offer** made by the solution provider
- Project was canceled due to architecture evaluation results (the previous example was just one of many cases)
  - Confidence in architecture was **too low**
  - Distance investigation revealed **gap** between the realized system level and the intended solution on architecture level
  - Correction effort was estimated higher than the new offer
- Impact
  - Product could not be delivered
  - High investments were made in vain
  - Costs for audit less than 1% of project budget

Architecture Evaluation Example – Synopsis

- Architecture is the conceptual tool to cope with complex systems
- Architecture evaluation
  - Provides valuable input to decision making
  - Can be applied with limited effort
  - Can produce results quickly
- In this case, architecture evaluation…
  - … should have been applied earlier
  - … might have saved the investment
  - … should have been repeated regularly
  - … might have lead to an improved architecture and compliant implementation

Typical Problems found in Architecture Evaluations

Architectures not adequate for requirements (any more)

Mismatch of architectures of systems to be integrated

No connection between architecture and implementation

Mismatch between architecture and organization / processes

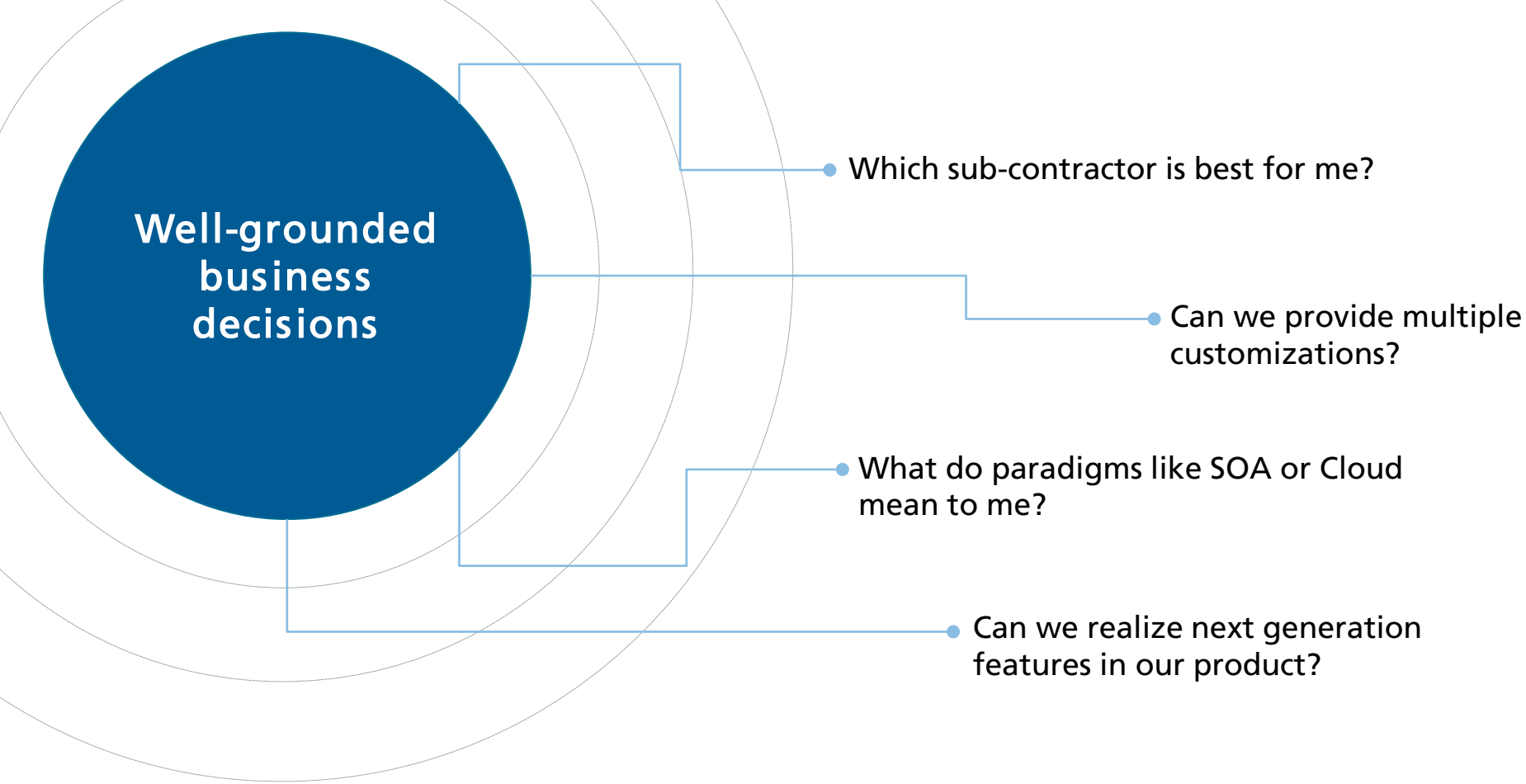
Architecture Evaluation Goals

Mission of Architecture Evaluation

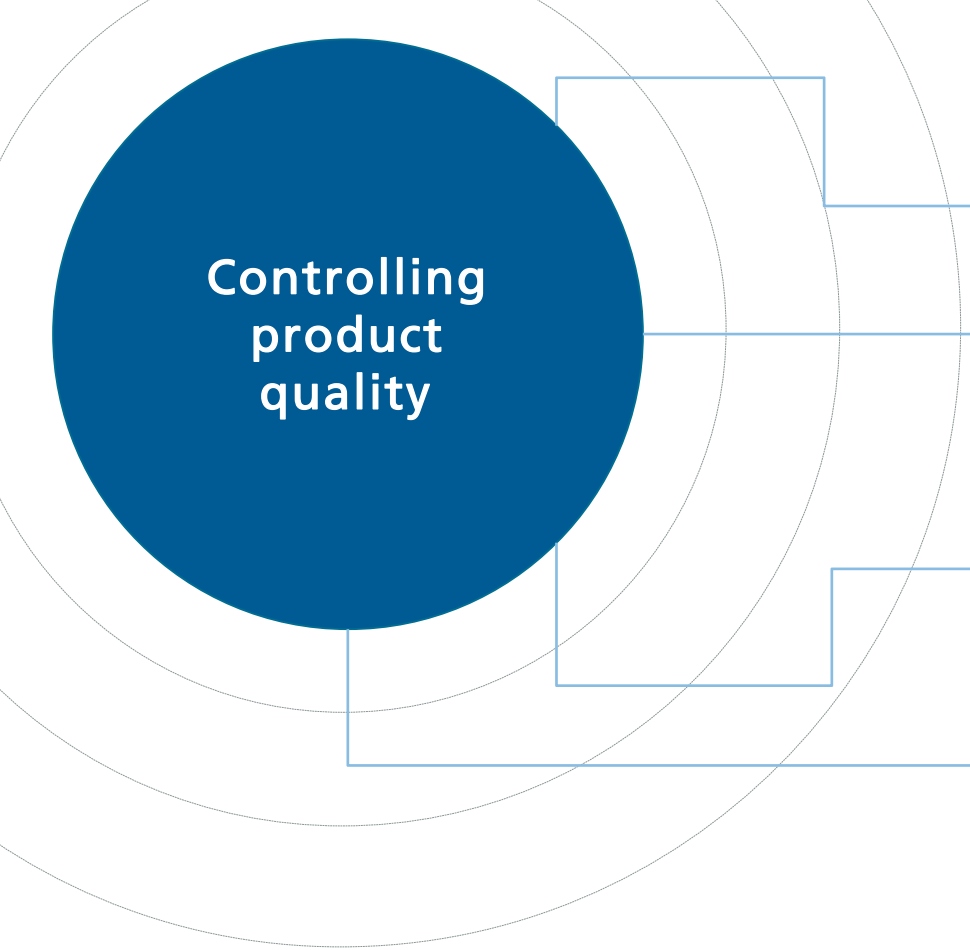
- **The Mission is Mitigating (Technical) Risks**
  - It is not about good or bad, it's about adequate or not
  - It is not about "state-of-the-art" or not, it's about adequate or not
  - Adequacy is always checked relative to concrete concerns and (future) requirements
- **The Mission is to Determine the Quality of**
  - **The system** (system in use, operation, change)
  - **(Auxiliary) artifacts** created in engineering (documentation, models, code, …)

- Evaluation of software architecture aims at determining
  - How **well-known** are the stakeholder concerns
  - How **well-suited** for the purpose is the architecture of a system
  - How **well-documented** is the architecture solution
  - How **well-realized** the architectural solution by the implementation
  - How **well-written** is the implementation

  - With respect to **concrete stakeholders concerns**

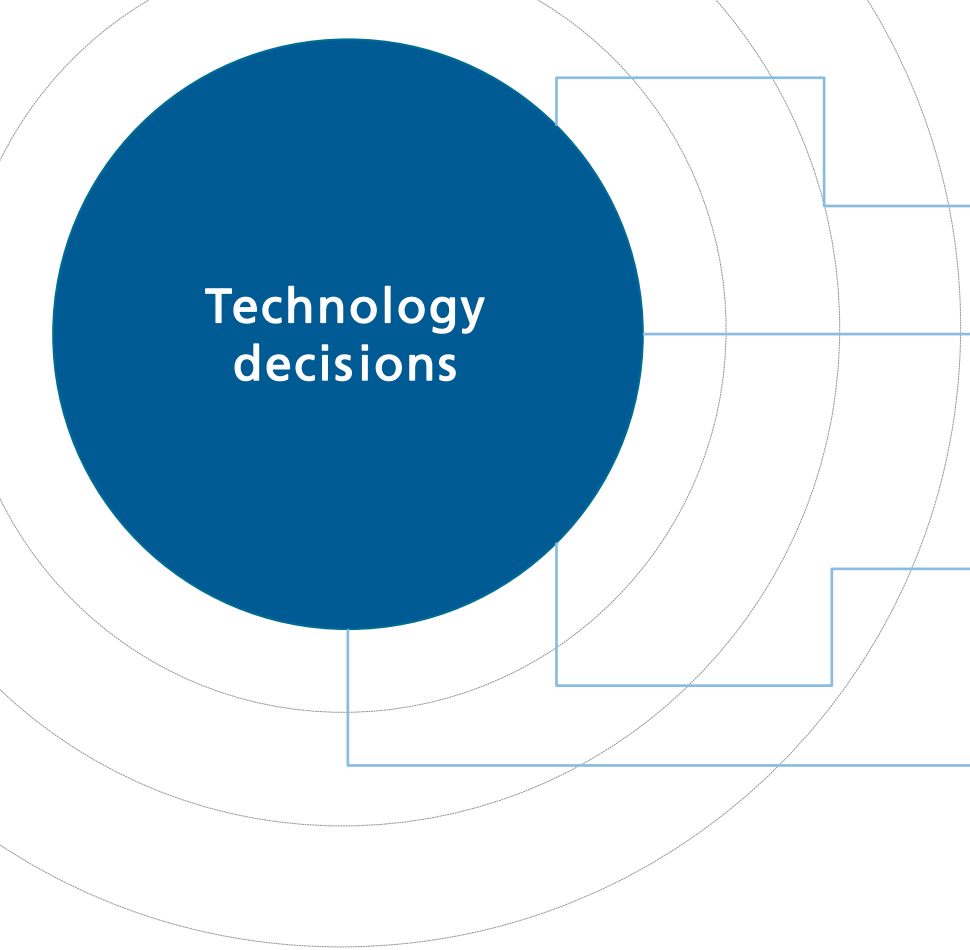


Fraunhofer IESE



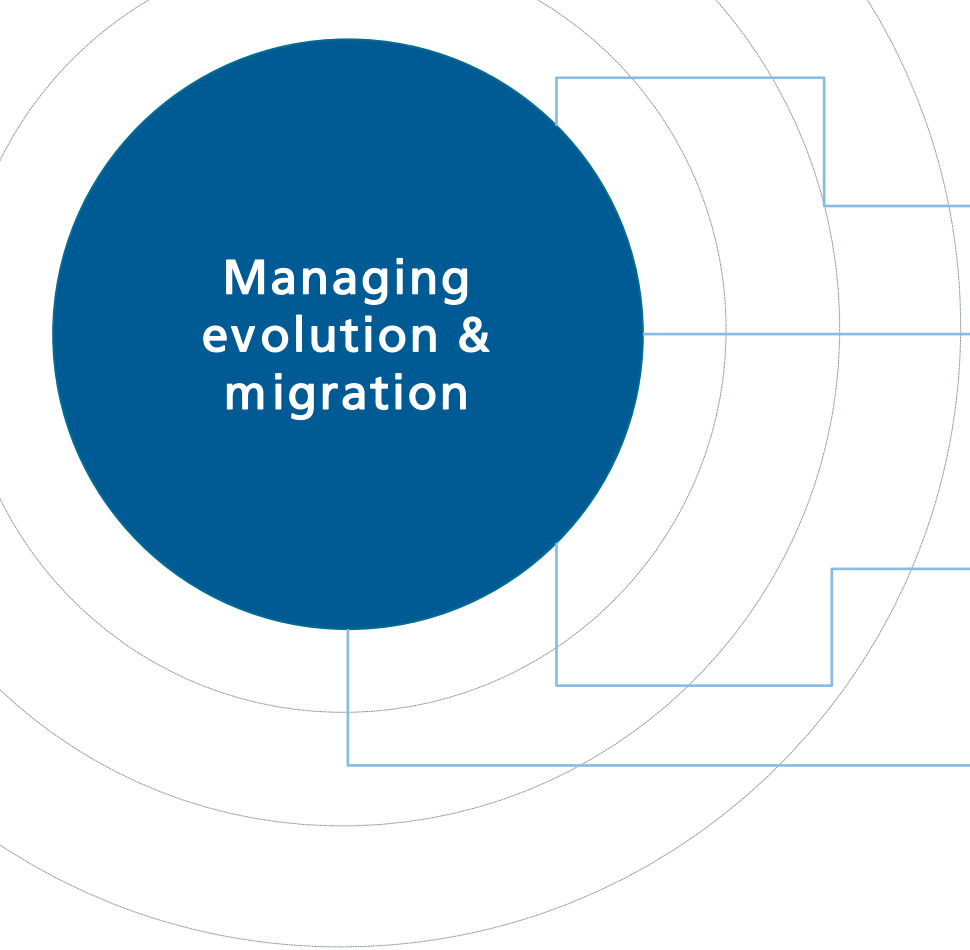
- Does my system fulfill the performance requirements?
- Is my system flexible enough for future changes?
- How well does my system support parallel development?
- What are the operation costs of my system?

Fraunhofer IESE



- Which technology fulfills my needs best?
- What is the impact of the adoption of this technology?
- What benefits can a particular technology offer?
- How can legacy technologies be replaced?

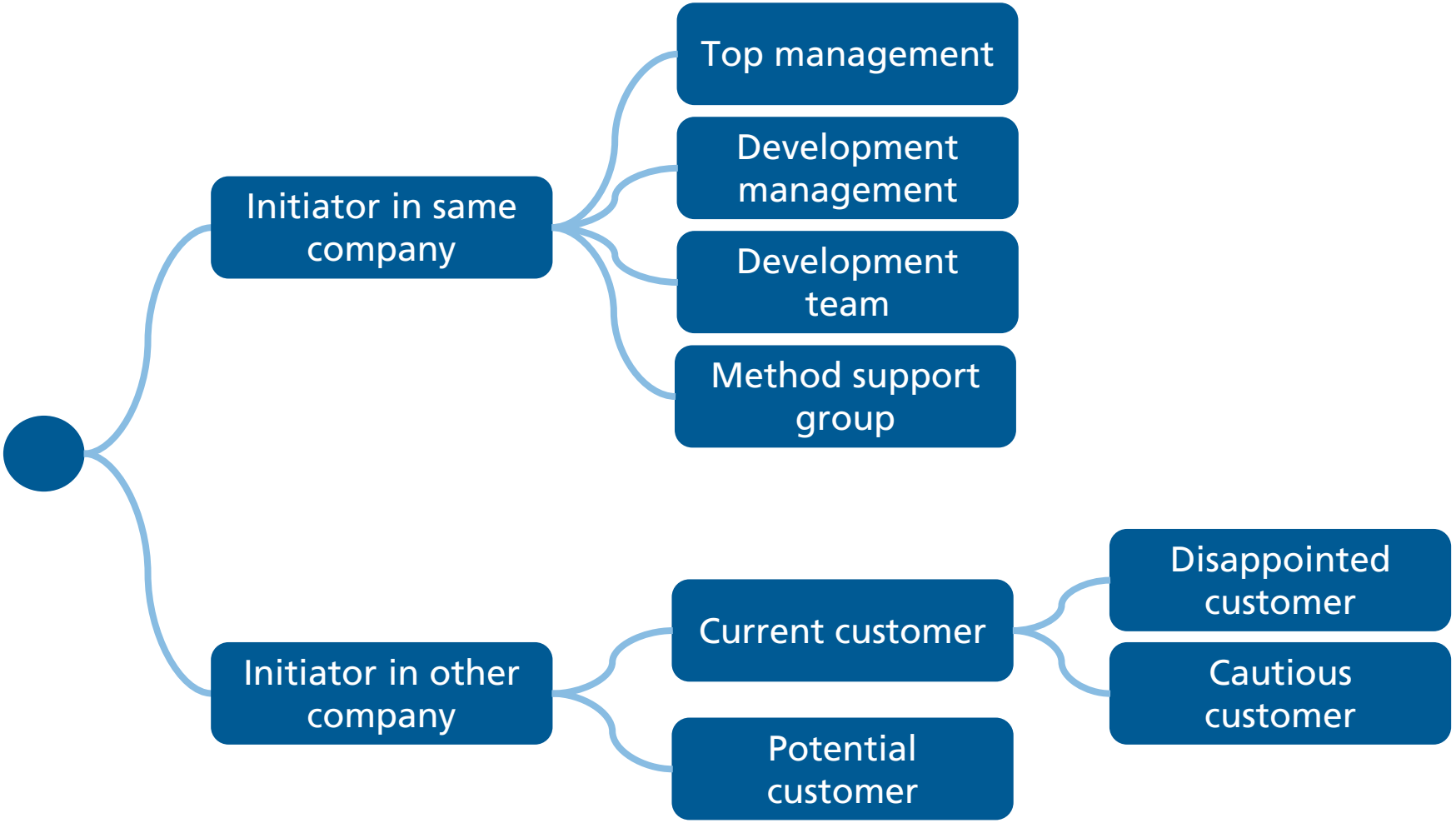
Fraunhofer IESE



- How can I deliver constantly during evolution?
- How to maintain my existing products?
- How to migrate to a new technology platform?
- How do I develop my next generation platform?

Fraunhofer IESE

Initiators of Architecture Evaluation



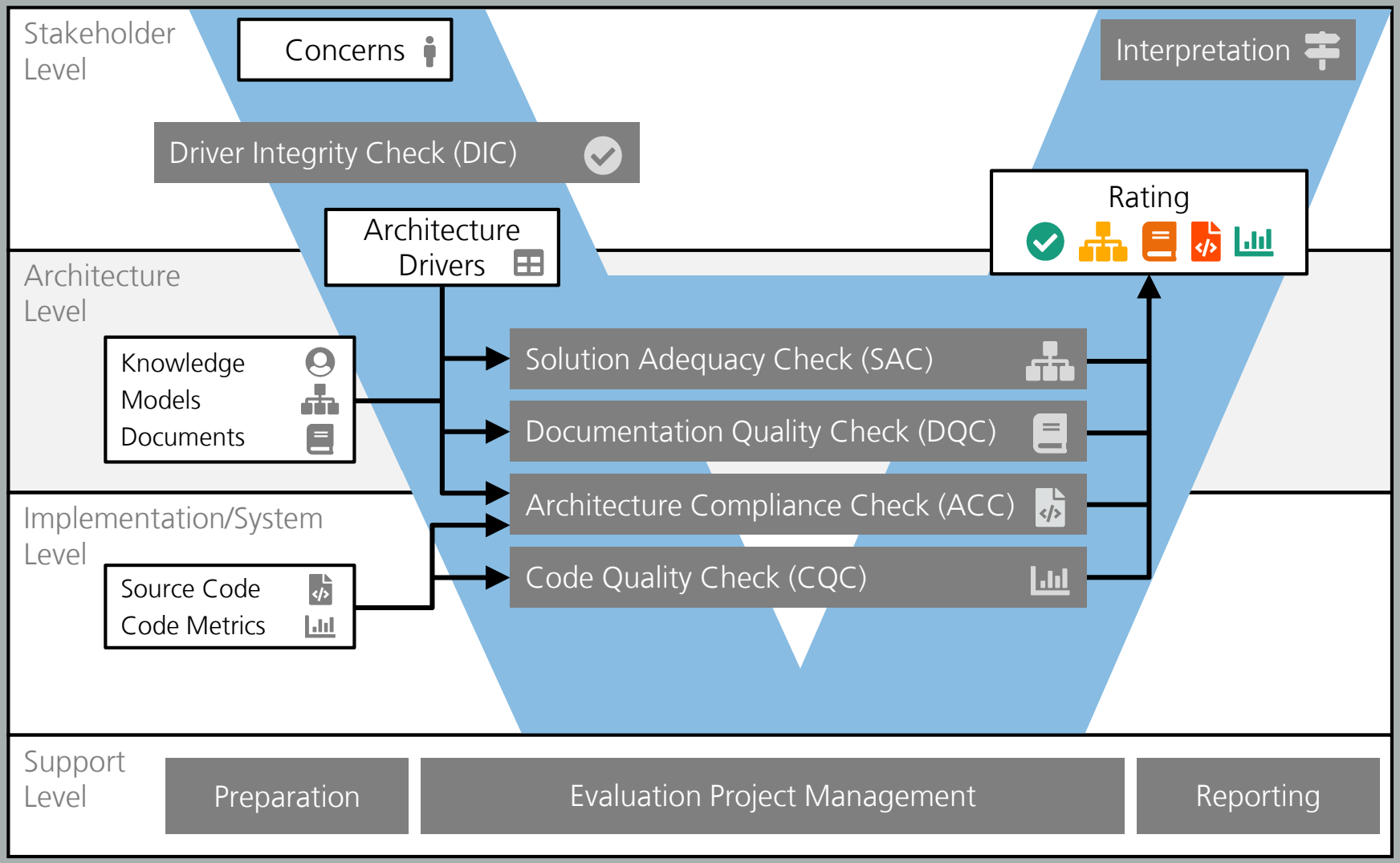
Fraunhofer IESE

Benefits of Architecture Evaluation

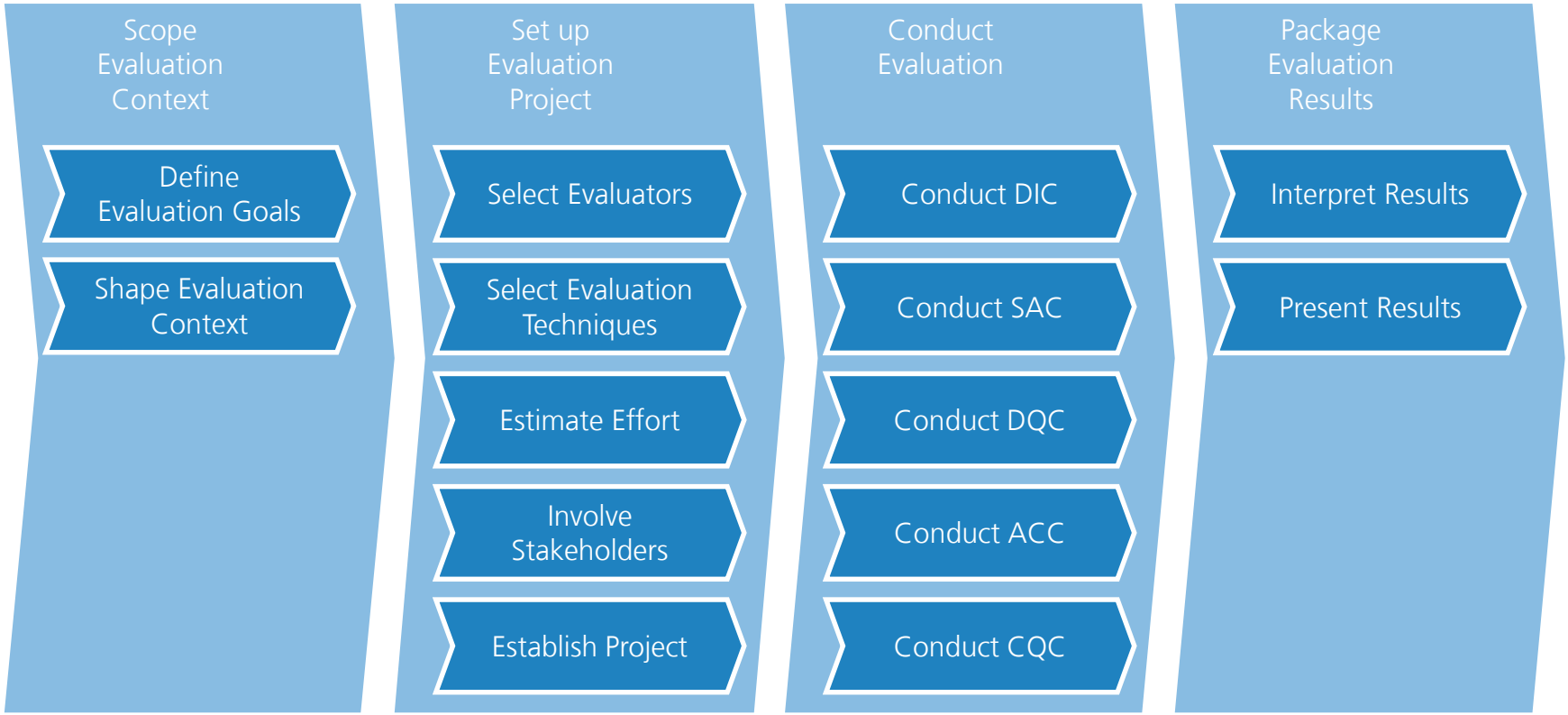
- **Improvements**
  - Improved software architectures
  - Improved architecture documentation
  - Improved implementations of architectural solutions
- **Risk mitigations**
  - Early detection of problems
  - Clarified quality attribute requirements
- **Communication**
  - Improved understanding of design decisions
  - Higher architecture awareness in the organization
- **Sustainability**
  - Traceability of architecture solutions over time
  - Higher or full compliance in implementations
- Competence
  - Improved architecture competence of involved stakeholders

Architecture Evaluation

Architecture Evaluation with Fraunhofer RATE



Approach of Evaluation Projects



Fraunhofer IESE

Levels of Confidence

Artifact Quality	Believed	Inspected	Measured	
System Quality	Believed	Predicted	Probed	Tested

- The higher the confidence, …
  - … the lower the risk of having made a wrong design decision
  - … the higher the effort to invest
  - … the lower the number of concerns that can be checked

Interpretation of Evaluation Results

Rating		Severity of findings				Legend
		Critical	Harmful	Minor	Harmless / Advantageous	N/A
Balance of findings	Mainly negative findings					NO
	Negative findings predominate					PARTIAL
	Positive findings predominate					LARGE
	Mainly positive findings					FULL

Fraunhofer IESE

		Evaluated System
Architecture Requirements	DIC	Are the architecture requirements clear?
Architecture	SAC	Is the architecture adequate for the requirements?
Architecture Documentation	DQC	Is the architecture documentation adequate?
Architecture Compliance	ACC	Is the code consistent with the architecture as it was planned?
Code Quality	CQC	Does the code have good overall quality?

Architecture Evaluation Limitations

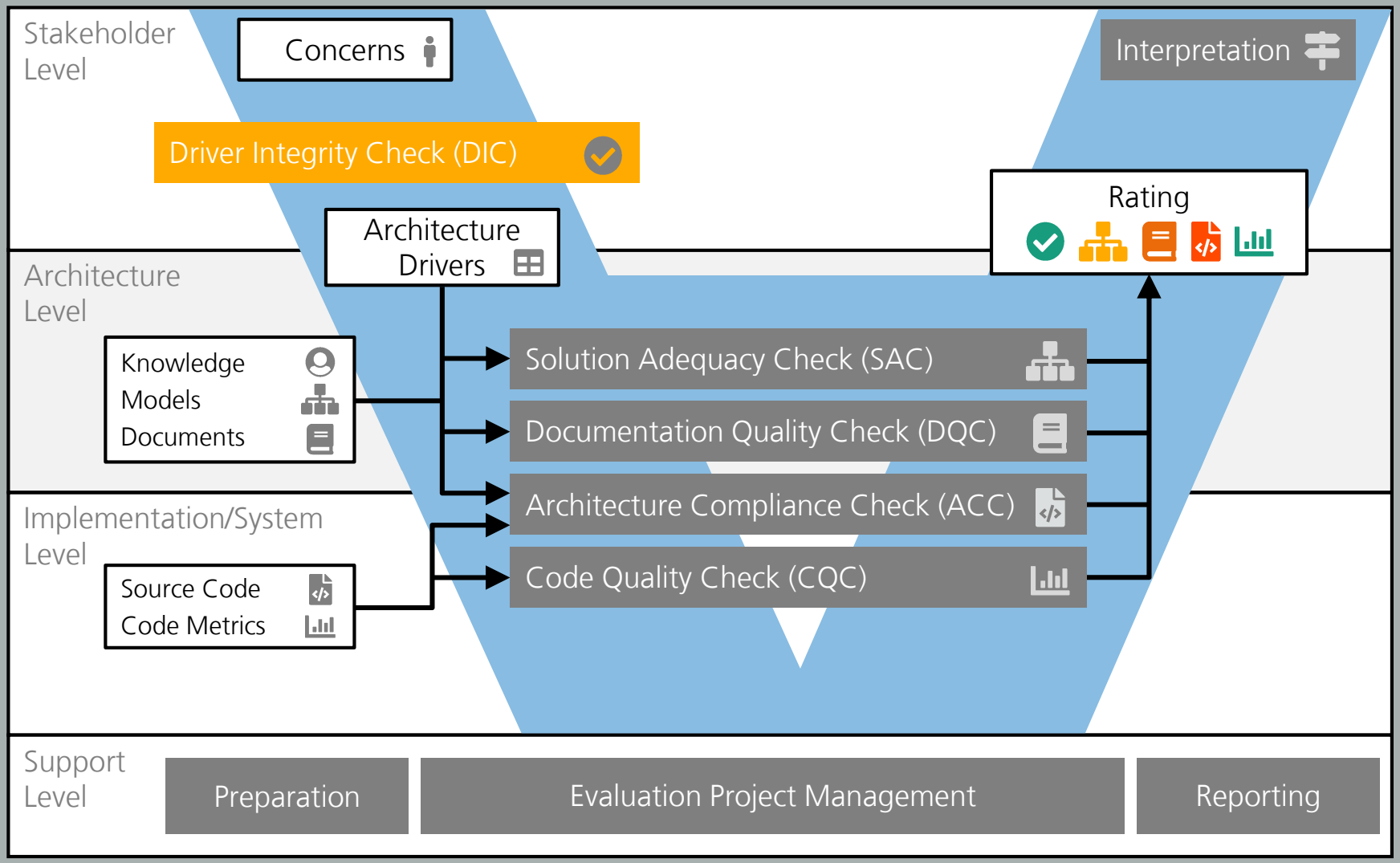
- **Architecture can only be evaluated indirectly**
  - Based on the input of stakeholders
  - Based on available architecture documentation
- Architecture evaluation requires **cooperation**
  - Open and cooperative climate for audits
  - Common goal to improve
- **Absolute architecture evaluation typically not possible**
  - Exact measurement is not always possible
  - Trade offs between competing qualities avoid unique, objective winners

Architecture Evaluation Limitations

- Architecture evaluation
  - Cannot guarantee quality
  - Component design and implementation also impact system qualities
- Examples for negative impact on quality at implementation level
  - Performance: inadequate algorithms
  - Maintainability: low code quality, unreadable code, …

RATE: Driver Integrity Check

Architecture Evaluation with Fraunhofer RATE



Driver Integrity Check (DIC)

It serves to check the integrity of architecture drivers manifesting the stakeholders' concerns.

Input

- Requirements documentation
- Architecture documents (if available)

Involved Stakeholders

- All stakeholders of the system
- Architects of system under evaluation (optional)

Execution

- Identify and involve relevant stakeholders
- Elicit and consolidate stakeholder concerns
- Find areas of interests, recurring items, hot spots, disagreements, and potential conflicts
- Merge, unify and align terminology used
- Document all architecture drivers
- Check for common agreement and approval
- Rate the integrity of the concerns
- Package the results

Output

Architecture Drivers

Findings (deviations, inconsistencies, ambiguities) in and consolidation of architecture drivers (business goals, constraints, quality attributes, key functional requirements)

Rating

Severity and balance of findings

Evaluators

- Architect
- Peers
- External auditor

Tools

- Documentation tools

Confidence Levels

- Predicted
- Probed

Fraunhofer IESE

Architecture Driver Template

Categorization	
Driver Name	
Driver ID	
Status	
Priority	

Responsibilities	
Supporter	
Sponsor	
Author	
Inspector	

Description		
Environment		
Stimulus		
Response		

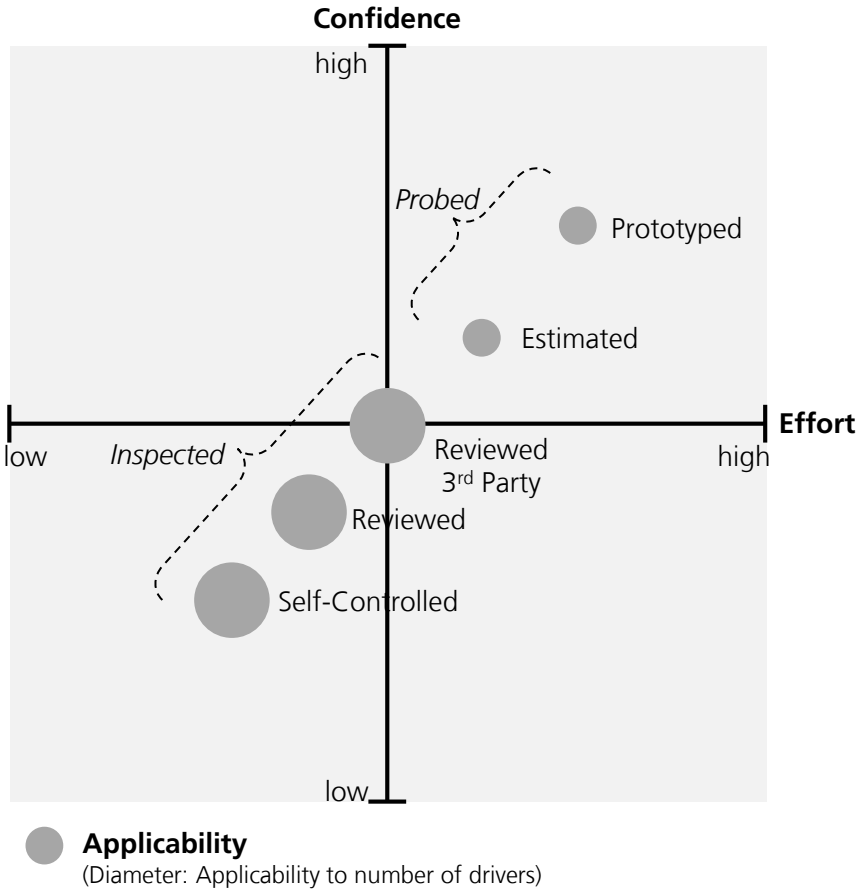
Use to document drivers elicited during Driver Integrity Check

Fraunhofer IESE

Rating of Driver Integrity

- **N/A** means that the driver integrity of the architecture driver has not (yet) been checked.
- **NO Driver Integrity** means there is strong disagreement among the stakeholders (conflicting concerns or priorities), or between stakeholders' concerns and the architecture driver specified by the assessor.
- **PARTIAL Driver Integrity** means that the architecture driver consolidates the stakeholders' concerns to some extent, but that parts of the driver need further elaboration before getting approval from the stakeholders.
- **LARGE Driver Integrity** means that the stakeholders have no major objections and approve the architecture driver in principle; some details may require further refinement or elaboration.
- **FULL Driver Integrity** means there is shared agreement among stakeholders and assessors about the architecture driver and the driver has been approved by the stakeholders.

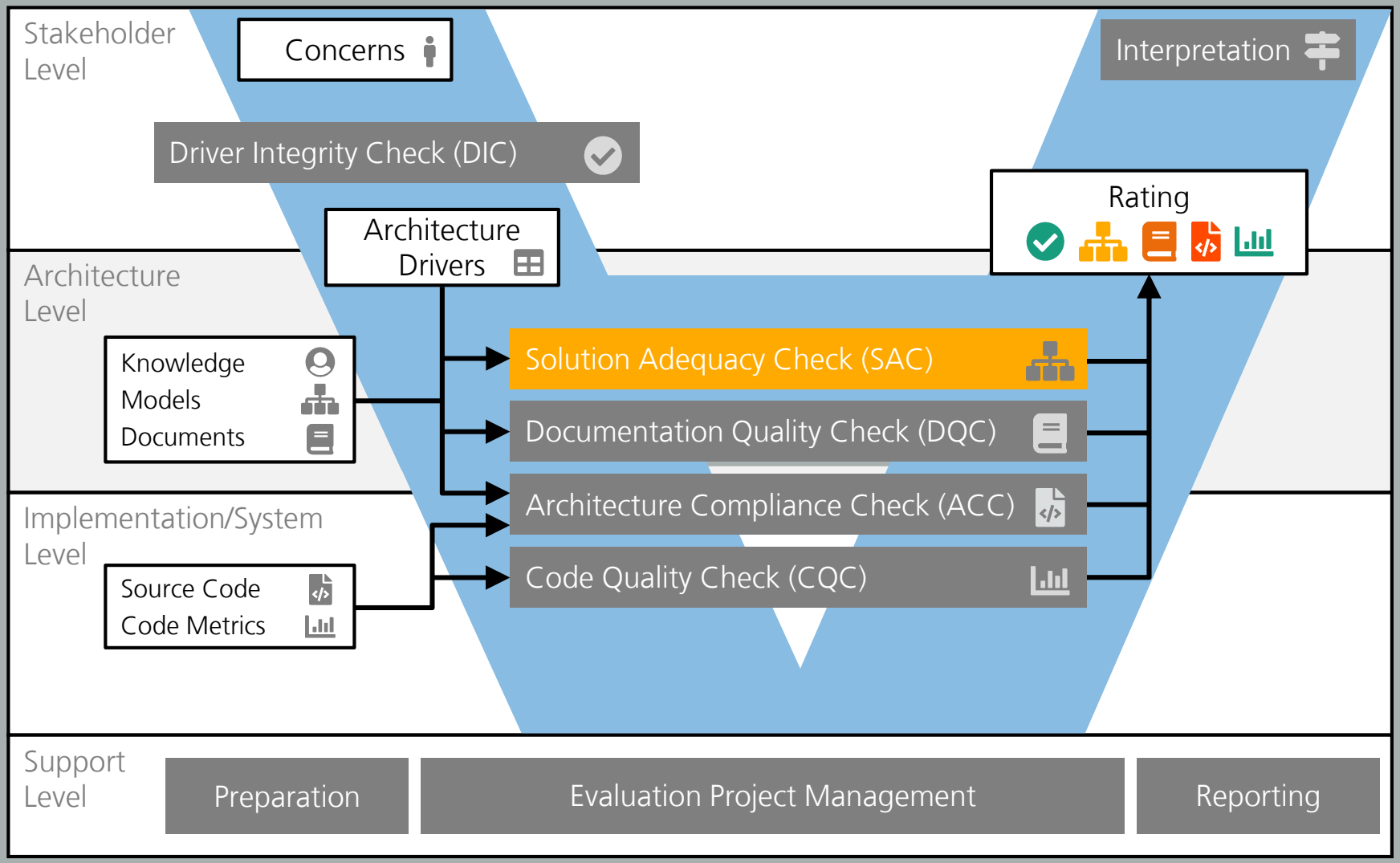
Integrity Levels of Driver Integrity Check



Fraunhofer IESE

RATE: Solution Adequacy Check

Architecture Evaluation with Fraunhofer RATE



Solution Adequacy Check (SAC)

It serves to check whether the architecture drivers of a system are adequately addressed in its architecture.

Involved Stakeholders

- Architects of system under evaluation
- Further stakeholders of system (optional)

Rating

Severity and balance of findings

Input

- Architecture drivers
- Architecture documentation

Execution

- Overview explanation of the architecture
- For each architecture driver
  - Reconstruct and discuss detailed solution
  - Document design decisions, risks, tradeoffs
  - Rate adequacy of the solutions
  - If necessary, increase confidence with other analyses
- Guidelines
  - Challenge the architect: ask for details
  - Ask about the "why?"
  - Use your experiences from other systems
  - Explore boundary areas

Output

Architecture decisions
Architecture driver solutions
Architecture diagrams

Findings on adequacy of architecture decisions to fulfill the architecture drivers (explicit rationales, risks, tradeoffs, assumptions)

Evaluators

- Architect
- Peers
- External auditor

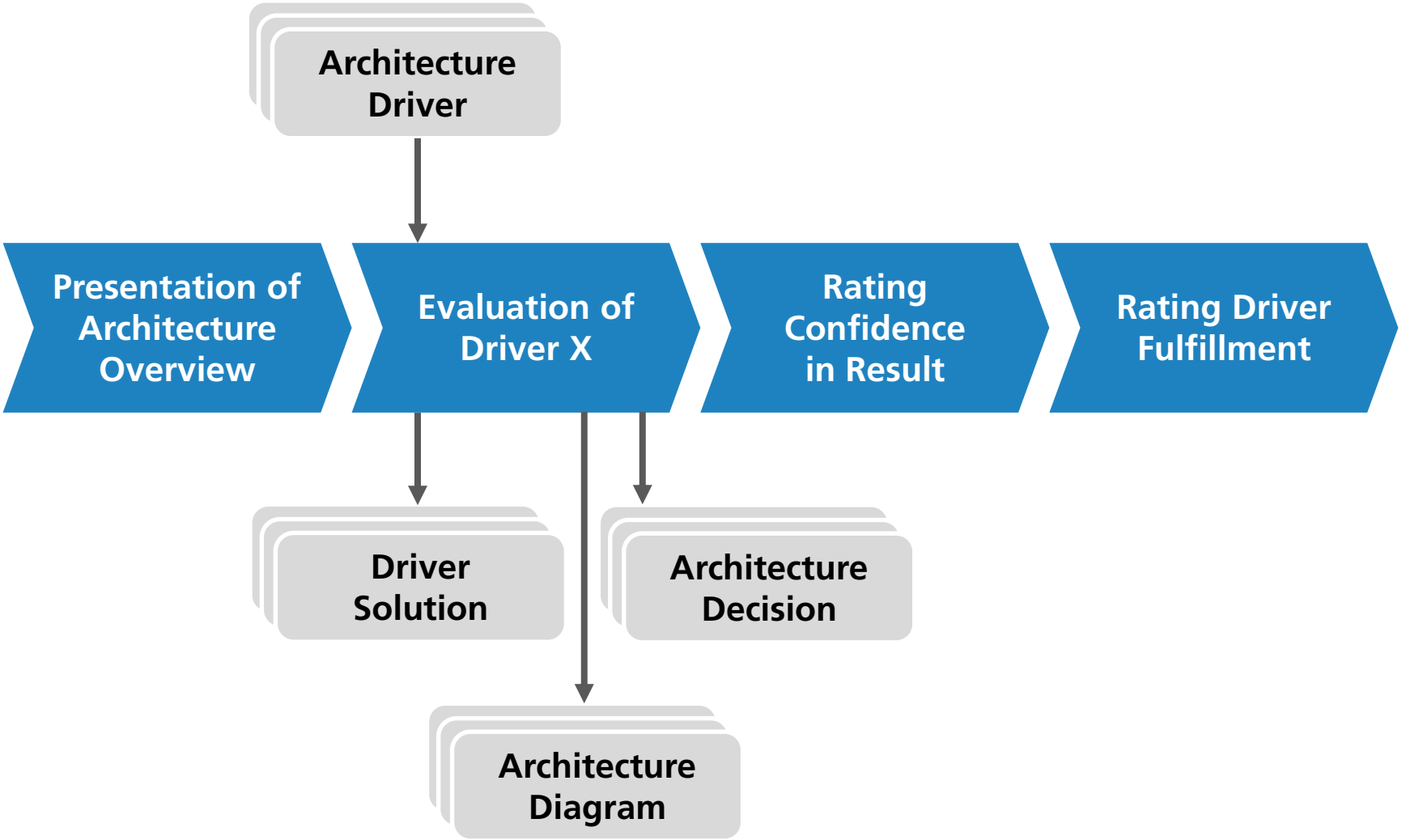
Tools

- Simulation tools
- Documentation tools

Confidence Levels

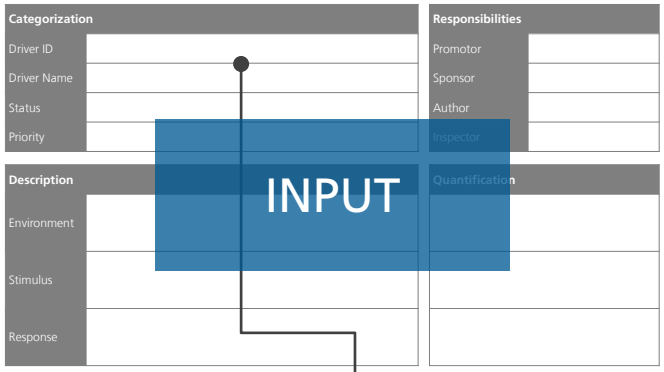
- Predicted
- Probed
- Tested

Approach for Solution Adequacy Check

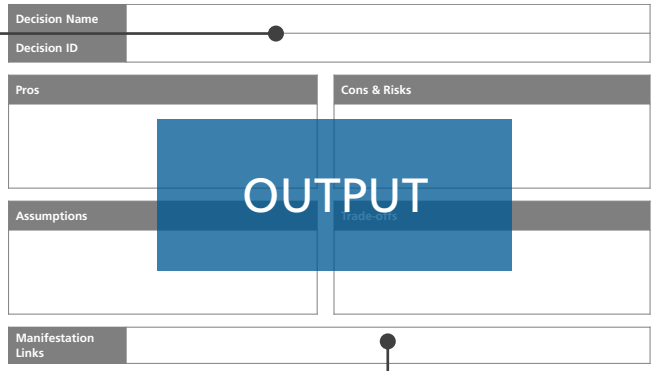


Fraunhofer IESE

Architecture Drivers (Input)



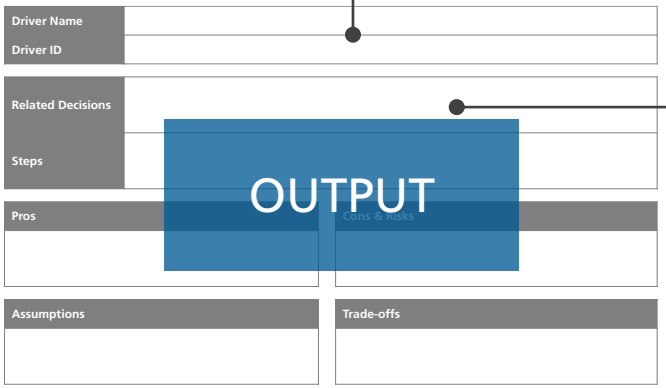
Design Decisions (Output)



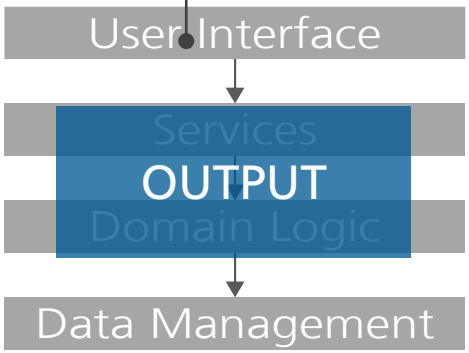
n:m

1:1

n:m



Driver Solutions (Output)



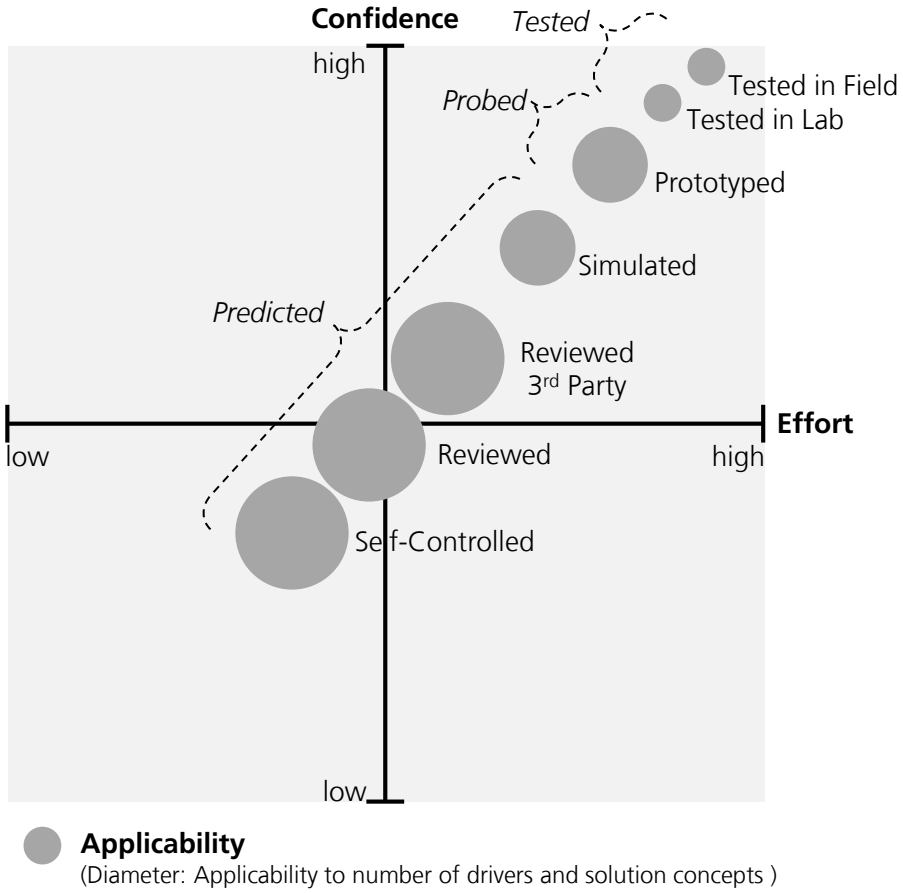
Architecture Diagrams (Output)

Fraunhofer IESE

Questioning Guidelines for Discussing Adequacy of Solution Concepts

- Ask for the solution concepts addressing the architecture driver
- Challenge the architects
- Consider all aspects covered in the ADF
- Use your experience from previous systems
- Identify risks: information that is not available
- Explore the boundary values of the architecture driver and solution concepts (limitations & assumptions)
- Explore potential tradeoffs with other quality attributes/architecture drivers

Confidence Levels of Solution Adequacy Check



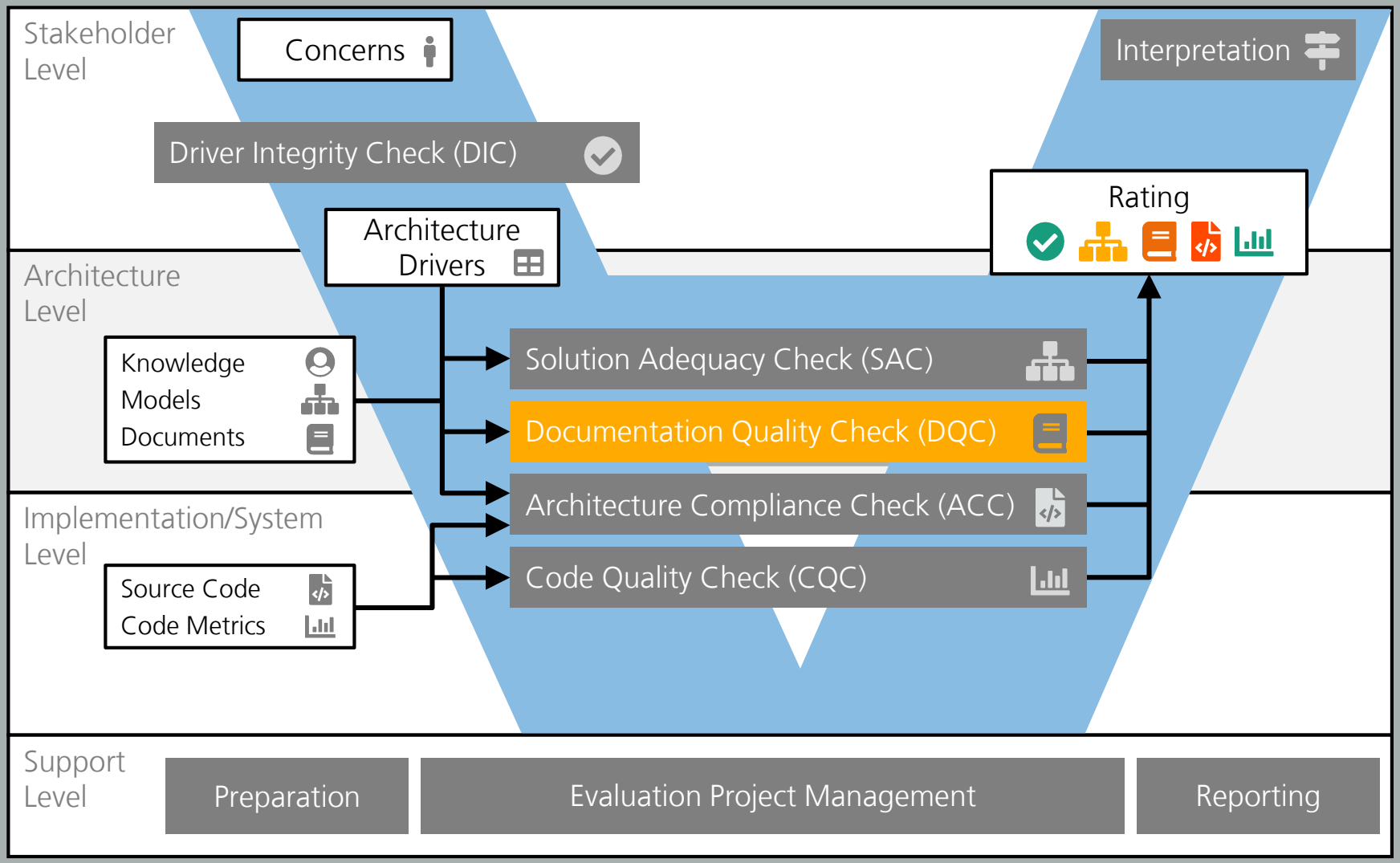
Fraunhofer IESE

Rating of Solution Adequacy

- **N/A** means that the solution of the architecture driver has not (yet) been checked. It can also mean that the check was not possible as the architecture driver was stated but not agreed upon.
- **NO Solution Adequacy** means there are major weaknesses in the solution or no solution may even be provided for the architecture driver.
- **PARTIAL Solution Adequacy** means that the architecture driver is addressed but there are still weaknesses and risks that require further clarification or architectural rework.
- **LARGE Solution Adequacy** means that the architecture driver is generally well addressed but with minor weaknesses or risks.
- **FULL Solution Adequacy** means there is confidence that the architecture driver is well addressed by the architecture decisions.

RATE: Documentation Quality Check

Architecture Evaluation with Fraunhofer RATE



Documentation Quality Check

Documentation Quality Check (DQC)

Serves to check the documentation of solution concepts and the adherence to documentation best practices.

Input
- Documentation purposes
- Architecture documents, models, wikis, sketches, API documentation
- Audience

Involved Stakeholders
- (Audience of documentation)

Execution
- Manual inspections
- Walkthroughs
- Tool-based measurement

Evaluators
- Architect
- Peers
- External auditor

Tools
- Best practice and style checkers

Output

Findings on adequacy of documentation and adherence to best practices

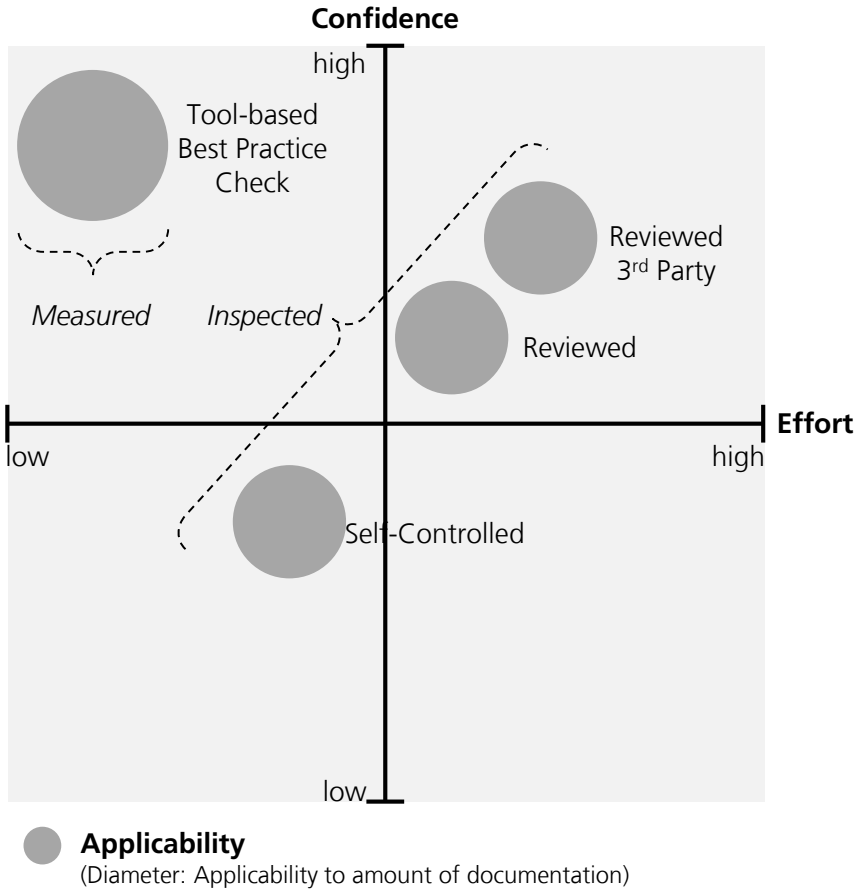
Rating

Severity and balance of findings

Confidence Levels
- Inspected
- Measured

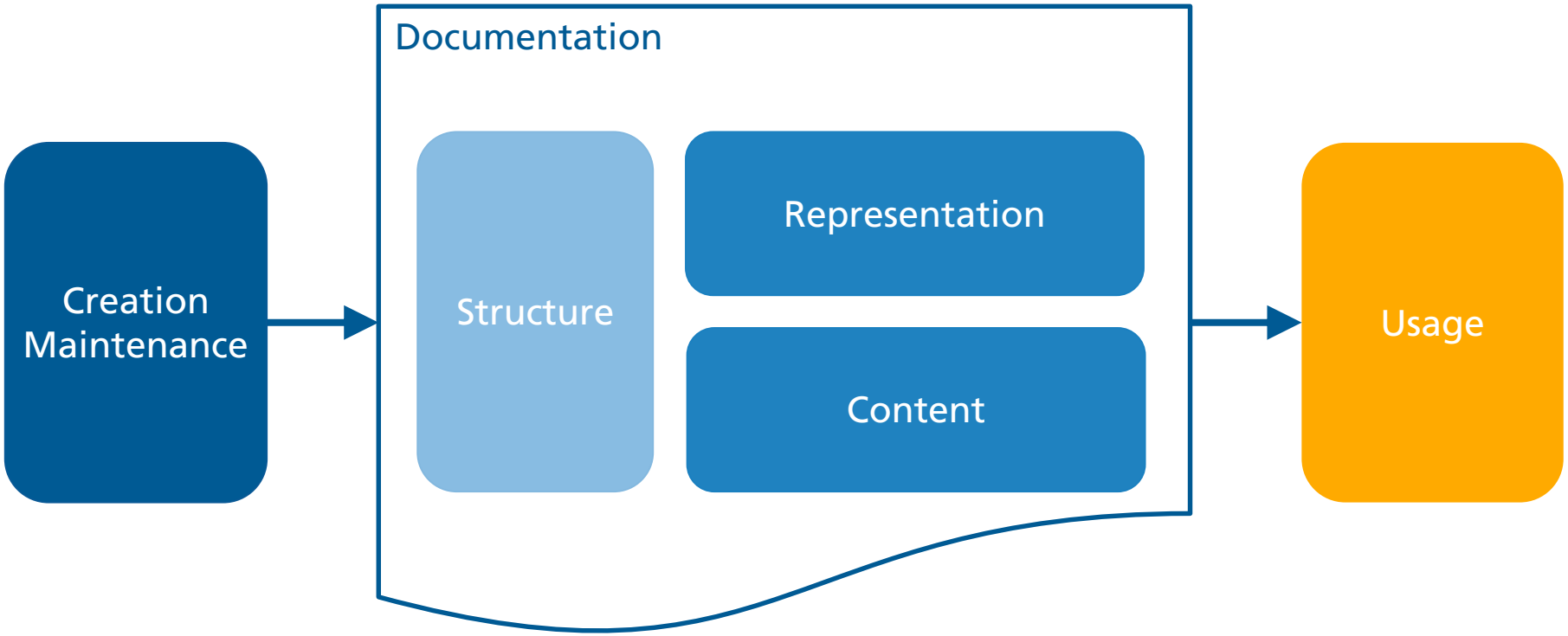
Fraunhofer IESE

Confidence Levels of Documentation Quality Check



Fraunhofer IESE

Architecture Documentation



General Properties of

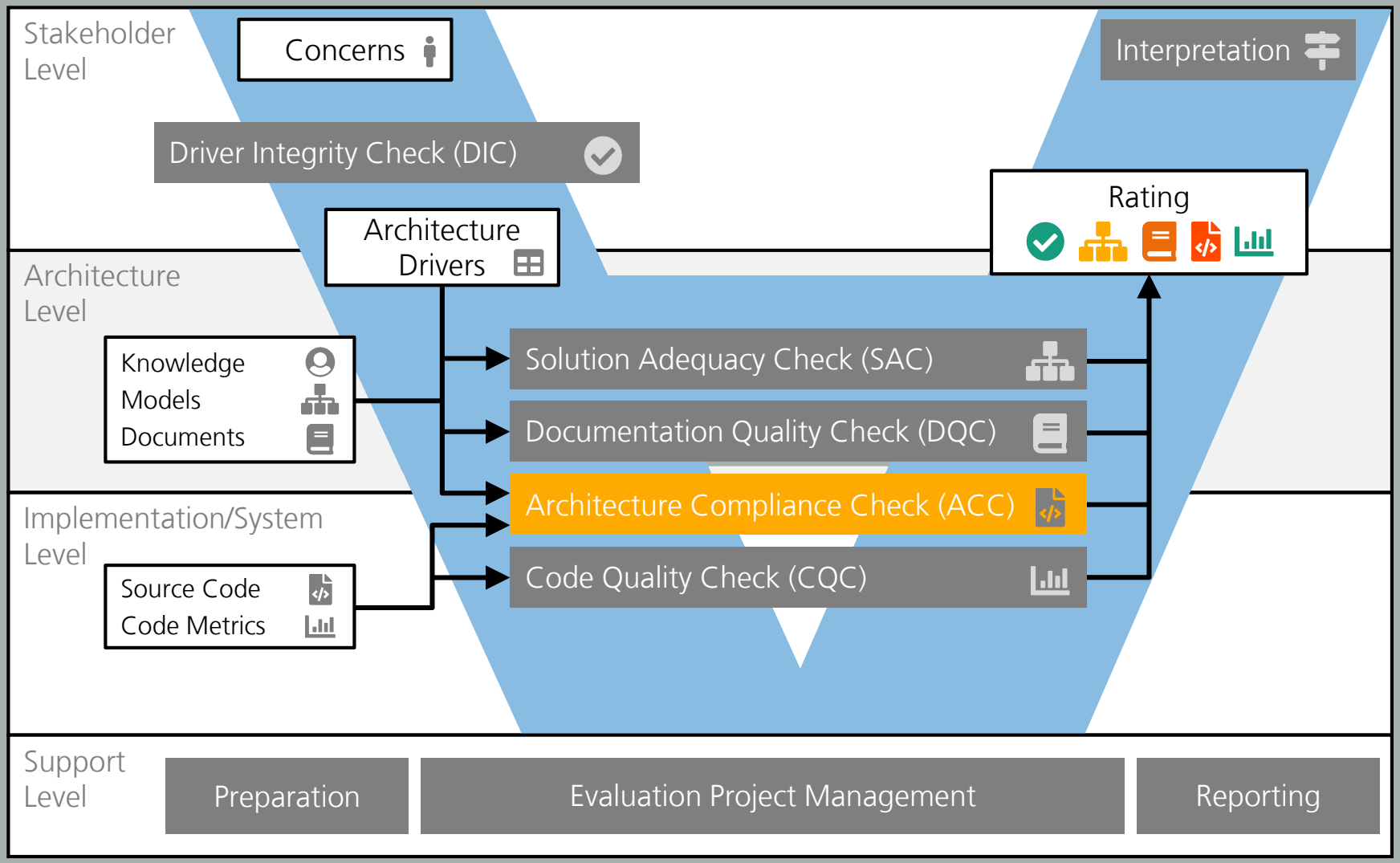
- Representation
  - Readability
  - Understandability
  - Memorability
  - Uniformity
  - Consistency (Internal and External with other Documents)
  - Compactness
  - Completeness
  - Correctness
  - Suitability for reader
  - Look and Feel (Usability)
  - …
- Structure
  - Structuredness
  - Simplicity
  - Navigation
  - Consistency
  - Redundancy-freeness
  - Retrievability
  - Traceability
  - Suitability for reader
  - …

Rating of Documentation Quality

- **N/A** means that the documentation quality for a criterion has not (yet) been checked.
- **NO Documentation Quality** indicates that major problems with the architecture documentation have been found. Significant amounts of effort and strong rework of the documentation concept are necessary.
- **PARTIAL Documentation Quality** means that a substantial number of deficiencies has been found in the documentation. These deficiencies endanger the usefulness of the documentation and require significant improvement.
- **LARGE Documentation Quality** means that only manageable deficiencies have been identified. The existing anomalies should be addressed explicitly and the estimated effort for fixing these fits into the next evolution cycle.
- **FULL Documentation Quality** means no or only few weaknesses were found in the documentation. Overall, the documentation is well suited for its purposes and follows documentation best practices.

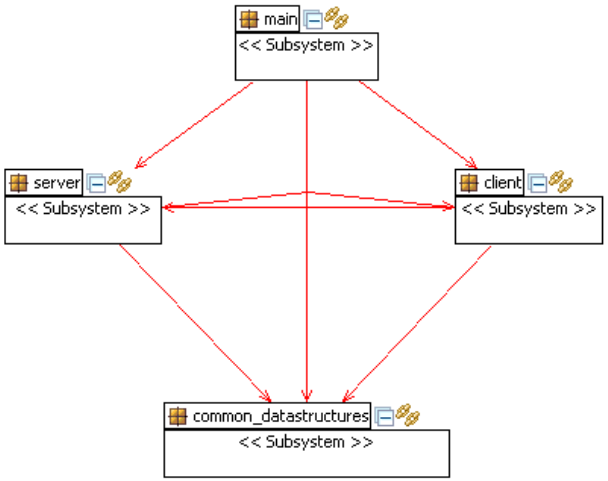
RATE: Architecture Compliance Check

Architecture Evaluation with Fraunhofer RATE

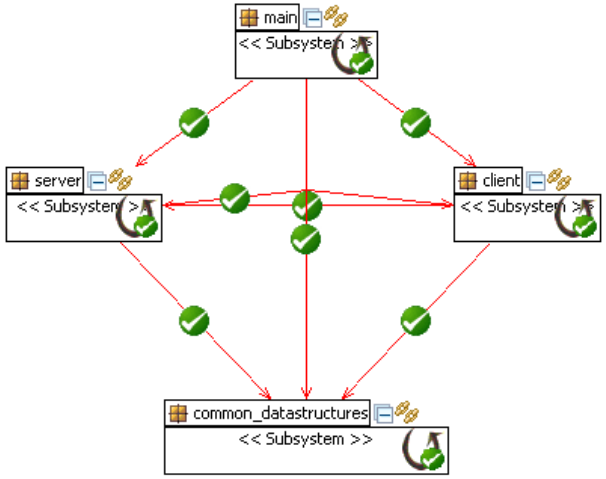


Architecture Compliance

- Architectures have to be implemented as they were planned. Otherwise, their value disappears
  - Implemented system must conform to the specified architecture
  - Traceability between architecture and source code is ensured



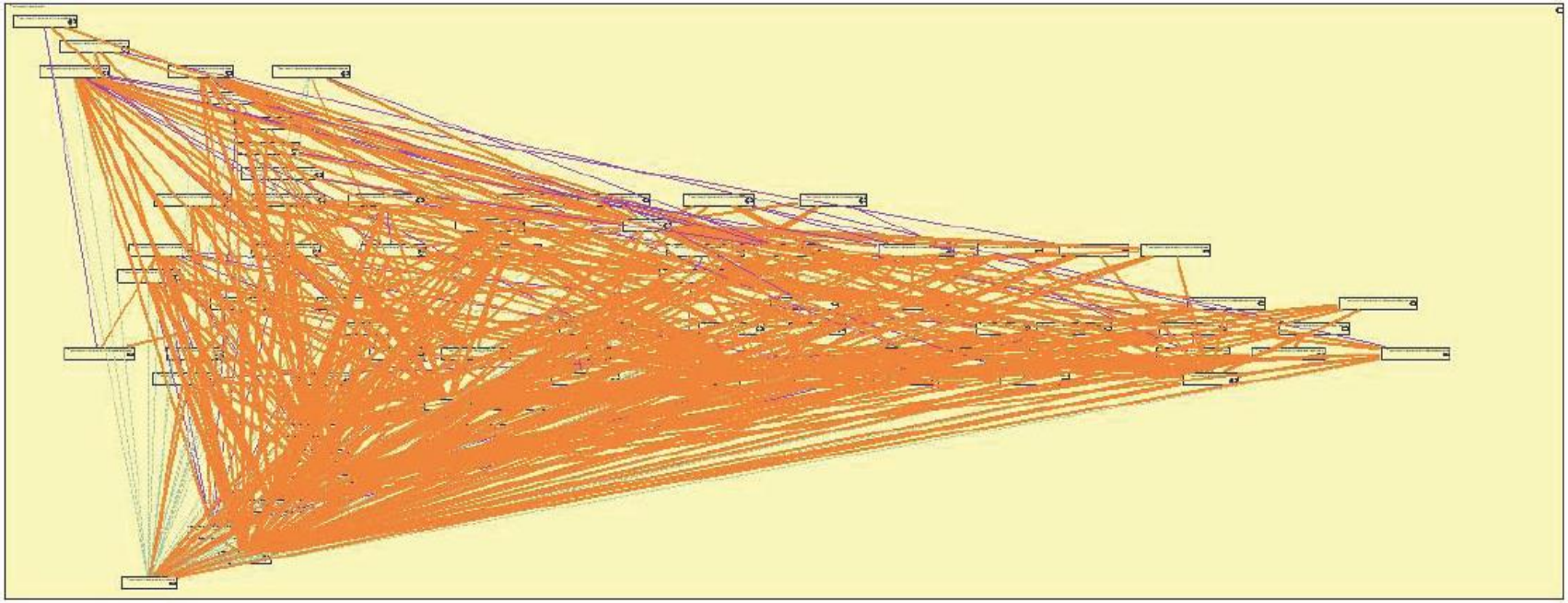
Planned structure



Implemented structure

Fraunhofer IESE

Industry Implementations Lack Structural Compliance



Just ONE subsystem (out of 20) of a real system

Fraunhofer IESE

Architecture Compliance Checking

Architecture Compliance Check (ACC)

Serves to check the manifestation of solution concepts in source code and/or in executables of the system.

Involved Stakeholders

- Architects and developers of the system under evaluation

Rating

Severity and balance of findings

Input

- Architecture documents, models, wikis, sketches, API documentation
- Source code
- (Running system)

Execution

- Identification of solution concepts to be checked for compliance
- Extraction of relevant facts from the code / running system
- Mapping of extracted facts to solution concepts
- Comparison of implemented architecture (extracted facts) and intended architecture (solution concepts)
- Interpretation of compliance checking results

Output

Findings on the compliance of the implementation with respect to the intended architecture

- Convergences
- Divergences (violation)
- Absences (violation)

Evaluators

- Architect
- Peers
- External auditor

Tools

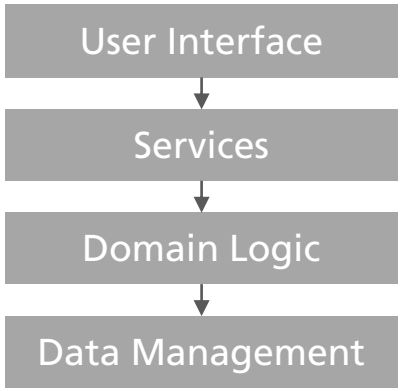
- Compliance checking tools

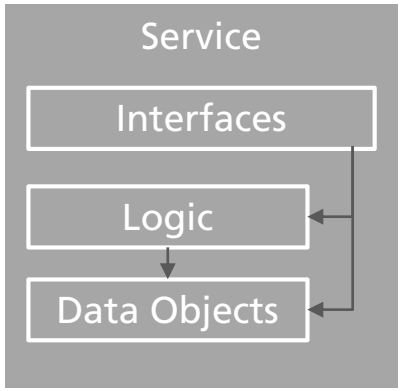
Confidence Levels

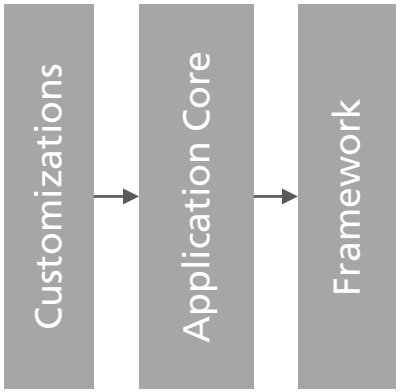
- Inspected
- Measured

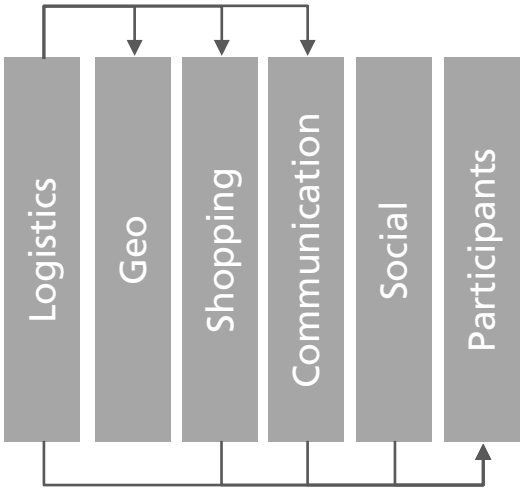
Fraunhofer IESE

Typical Concepts to Check for Structural Compliance



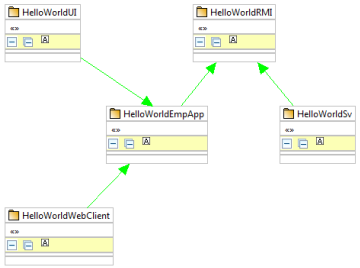






Fraunhofer IESE

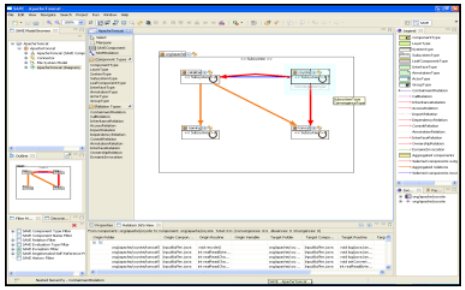
Comparison and Visualization of Results



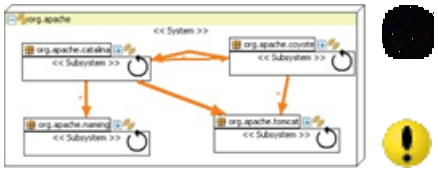
System artifacts
- Implementation (code)
- Execution (runtime) traces

Experts
- Documentation
- Reverse engineering
- Reconstruction

Mapping



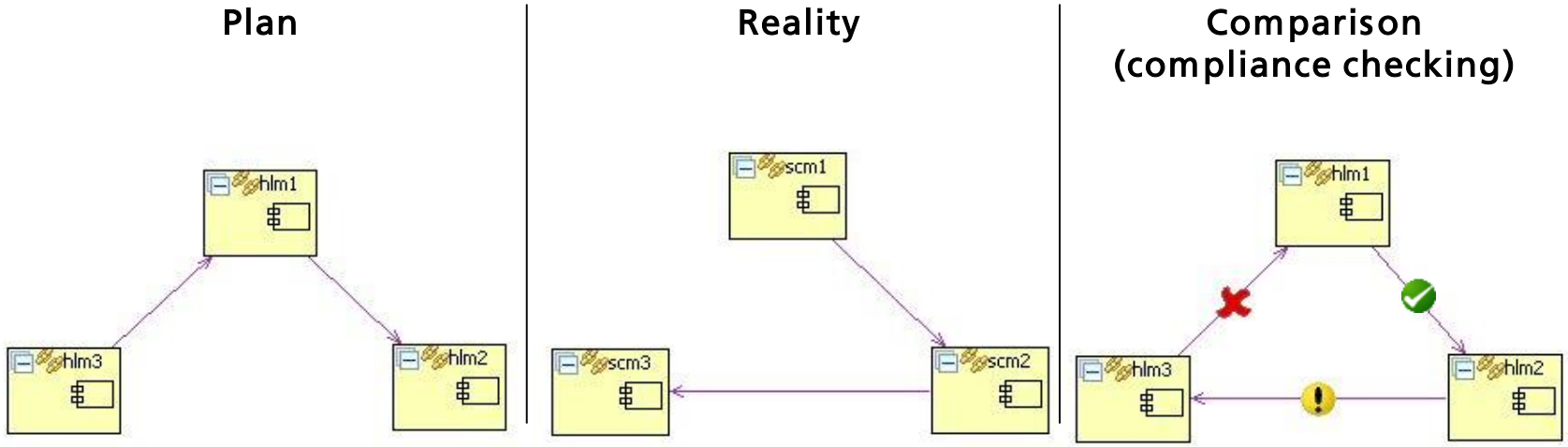
Compliance checking tool

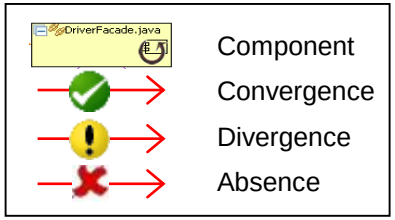


Result report

Fraunhofer IESE

Compliance Checking - Structure





Fraunhofer IESE

Compliance Checking - Tools

- Axivion Bauhaus
- CAST
- jDepend
- jRMTool
- Klocwork Insight
- Lattix
- Hello2morrow SonarJ
- Hello2morrow Sotograph
- Semmle .QL
- Structure101

- Fraunhofer SAVE

Rating of Architecture Compliance

N/A means that the architecture compliance for a solution concept has not (yet) been checked.

NO Architecture Compliance

- systemic misunderstanding that has been manifested in the code
- affects the fulfillment of architecture drivers and requires great dedicated effort for correction.
- no counterparts found on code level for architecture solution concepts

PARTIAL Architecture Compliance

- large gap between the solution concept and the source code
- does not break the architecture but the number of violations is drastically high
- estimated effort for fixing these violations does not fit into the next evolution cycle; rather, fixing the violations requires dedicated effort for redesigning, restructuring, and refactoring

LARGE Architecture Compliance

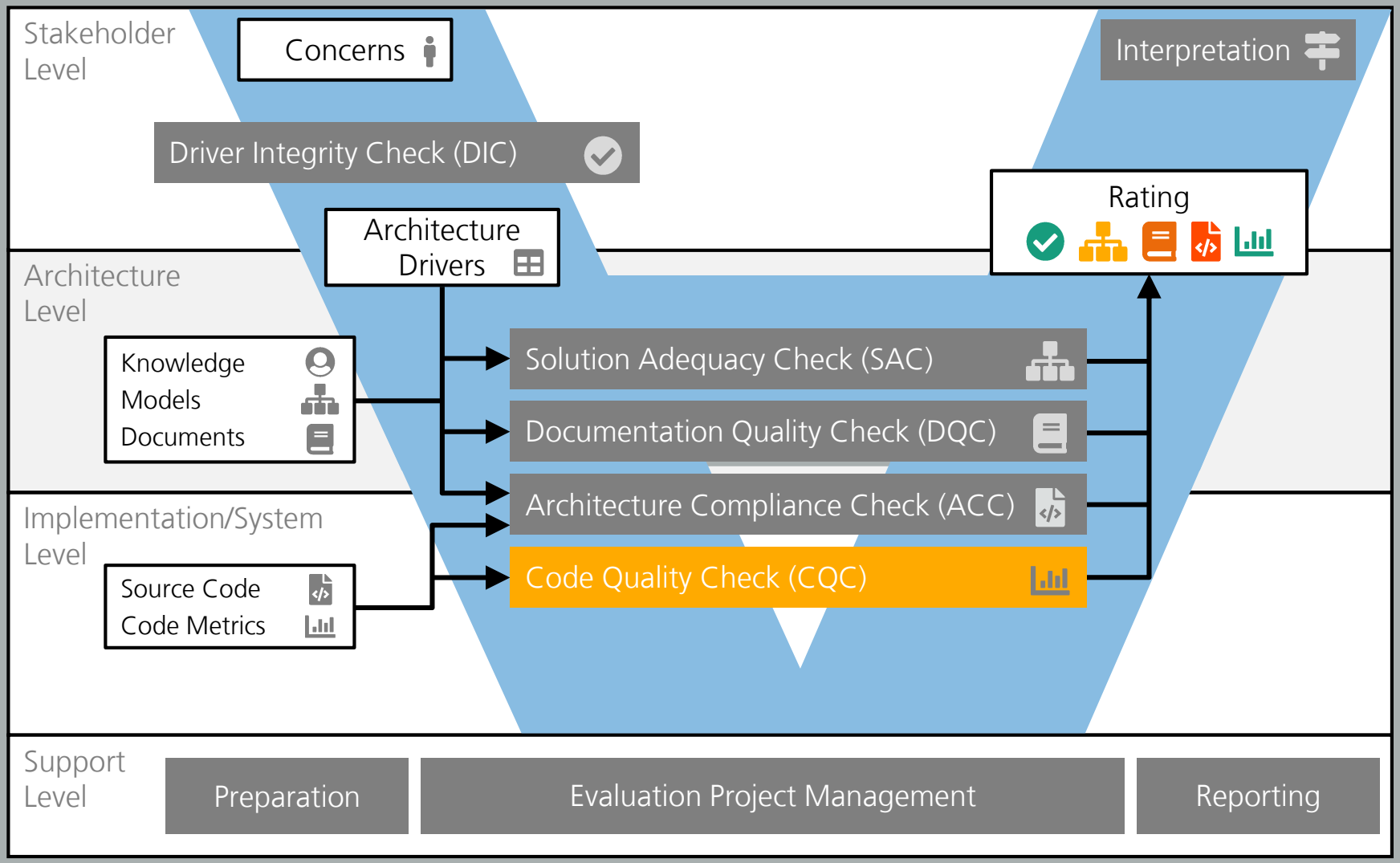
- small or medium gap between the solution concept and the source code
- does not break the architecture but has a significant adverse impact on some architecture drivers
- violations should be addressed explicitly and the estimated effort for fixing does fit into the next evolution cycle.

FULL Architecture Compliance

- no or almost no violations in the source code (short distance to the architectural solution concepts)
- having no violations at all is unrealistic for non-trivial software systems; there will always be exceptions for good reasons (technical limitations, optimizations of quality attributes, etc.). It is rather important to have a low number of violations (e.g., less than one percent violations of all dependencies) that are known explicitly and revisited regularly to keep them under control.

RATE: Code Quality Assessment

Architecture Evaluation with Fraunhofer RATE



Code Quality Check

Code Quality Check (CQC)

Serves to check the implementation for the adherence to coding best practices and quality models.

Input

- Source code
- (Build scripts)

Involved Stakeholders

- Developers of the system under evaluation

Execution

- Identification of goals for checks
- Setup and configuration of code quality checks
- Measurement of the selected metrics and checks
- Interpretation of code quality results

Output

Findings on quality of the source code

- Best practice violations
- Code clones
- Quality warnings (maintainability, security, …)
- Code metrics
- …

Rating

Severity and balance of findings

Evaluators

- Architect / Quality Engineer
- Peers
- External auditor

Tools

Code quality tools (style checker, clone detection, quality warning checker, …)

Confidence Levels

- Inspected
- Measured

Software Measurement and Metrics

- Multiple metrics exist
  - Design
    - Coupling
    - Cohesion
    - Inheritance depth
    - …
  - Implementation
    - Code style
    - McCabe
    - Maintainability index
    - …
  - Testing
    - Test success
    - Code coverage
    - …
  - …

Example: Measurement Tools

- Weighted aggregation of metrics of all areas
  - Code quality (architecture, design)
  - Test
- Result
  - One indicator for the whole system
- What does that indicate?
  - Best practice measurement
  - Interpretation is difficult!

Total Quality
[TQ=0.25*ARCH + 0.25*DESIGN + 0.25*CODE + 0.25*TESTS] = 68,9%

Unit Tests
[TS=TMR] = 41,6%
Method Test Reference
[TMR=COVERAGE] = 41,6%

Architecture
[ARCH=0.50*COH + 0.50*ADI] = 66,0%
Distance
[ADI=count(distance<=20)/packages] = 54,4%
Cohesion
[COH=1-(count(cycles=true)/packages)] = 77,5%

Design
[DES=0.2*NOM + 0.3*RFC + 0.3*CBO + 0.2*DIT] = 76,5%
Number of Methods
[NOM=count(Complexity/method < 20)/classes] = 86,4%
Response for Class
[RFC=count(rfc<50)/classes] = 80,2%
Coupling Between Objects
[CBO=count(cbo<5)/classes] = 59,7%
Deph of Inherance Tree
[DIT=count(dit<5)/classes] = 86,4%

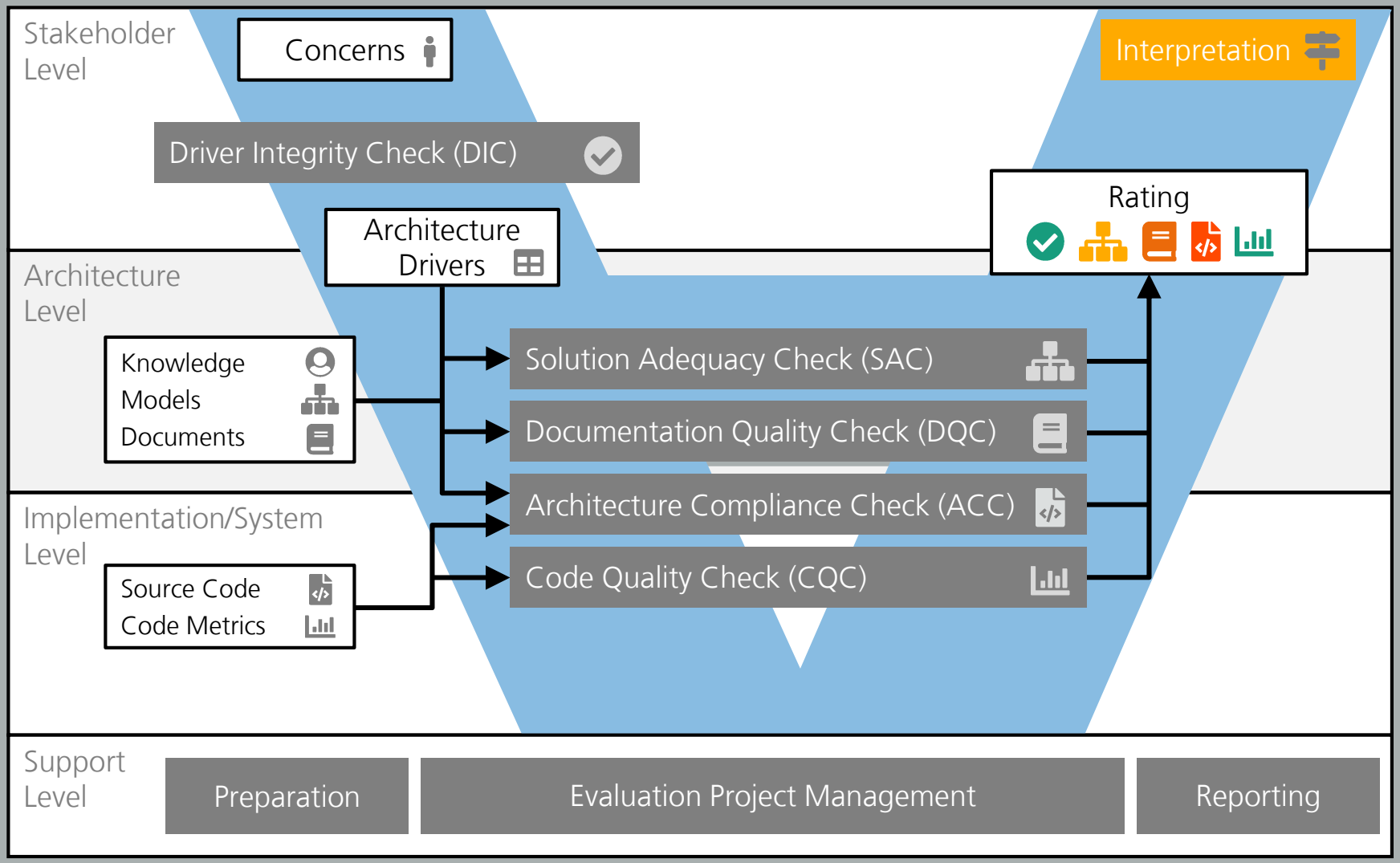
Code Quality
[CODE=0.15*DOC + 0.40*DRY + 0.45*RULES] = 91,3%
Documentation
[DOC=COMMENT_LINES_DENSITY] = 77,5%
DRYness
[DRY=1 - DUPLICATED_LINES_DENSITY] = 95,1%
Rules
[RULES=MAND_VIOLATIONS_DENSITY] = 92,5%

Rating of Code Quality

- **N/A** means that the code quality for a criterion has not (yet) been checked.
- **NO Code Quality** indicates major parts of the code base exceed the thresholds that have been defined for the criterion at hand.
- **PARTIAL Code Quality** means for some parts of the source code, the thresholds defined and the impact of the anomalies is considered harmful. The estimated effort for fixing these anomalies does not fit into the next evolution cycle; rather, dedicated effort for refactoring is required to fix the anomalies.
- **LARGE Code Quality** means that only limited anomalies were found with respect to the defined criterion. The existing anomalies should be addressed explicitly and the estimated effort for fixing them does fit into the next evolution cycle.
- **FULL Code Quality** means there are no or only few anomalies (e.g., condoned exceptions).

RATE: Packaging and Presentation

Architecture Evaluation with Fraunhofer RATE



Packaged Example Results from Different Projects

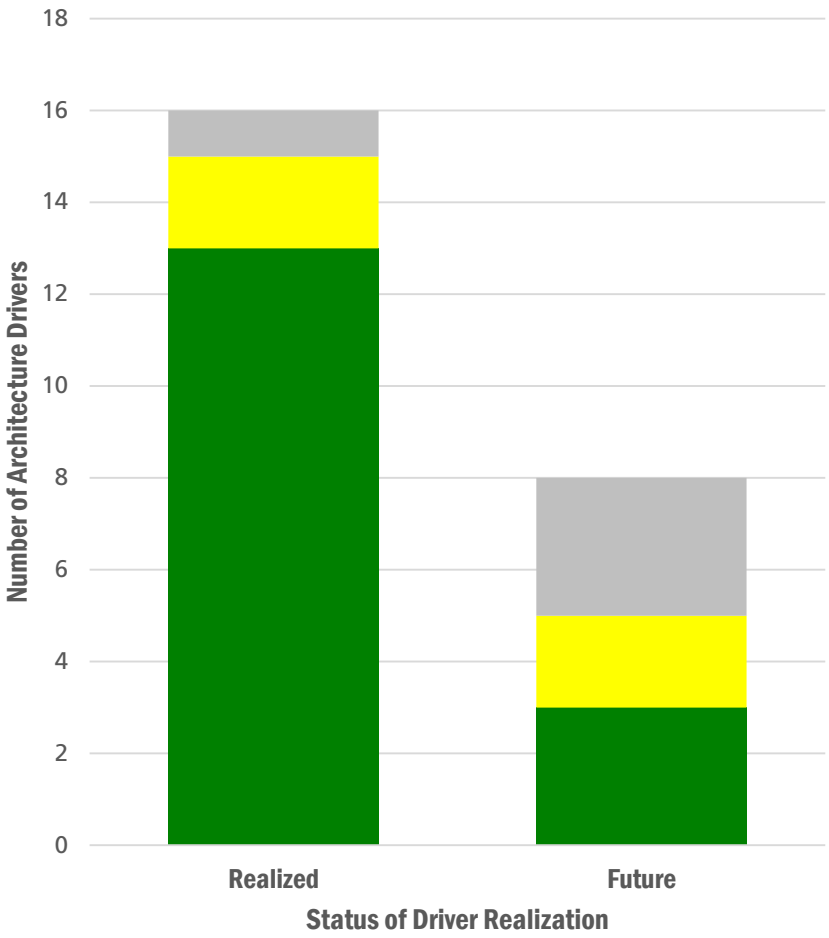
		Example 1 System	Example 2 System	Example 3 System 1	Example 3 System 2	Example 4 System
Architecture Requirements	DIC					N/A
Architecture	SAC					N/A
Architecture Documentation	DQC	N/A		N/A	N/A	N/A
Architecture Compliance	ACC					
Code Quality	CQC	N/A		N/A	N/A	N/A

Fraunhofer IESE

Results for Different Quality Attributes

Accuracy	●
Availability	● ● ● ●
Businessgoal	F F F F F
Consistency	● ● ●
Flexibility	F
Interoperability	F
Monitoring	●
Operability	● ●
Performance	F
Reliability	● ● ●
Updatability	●
User Experience	●

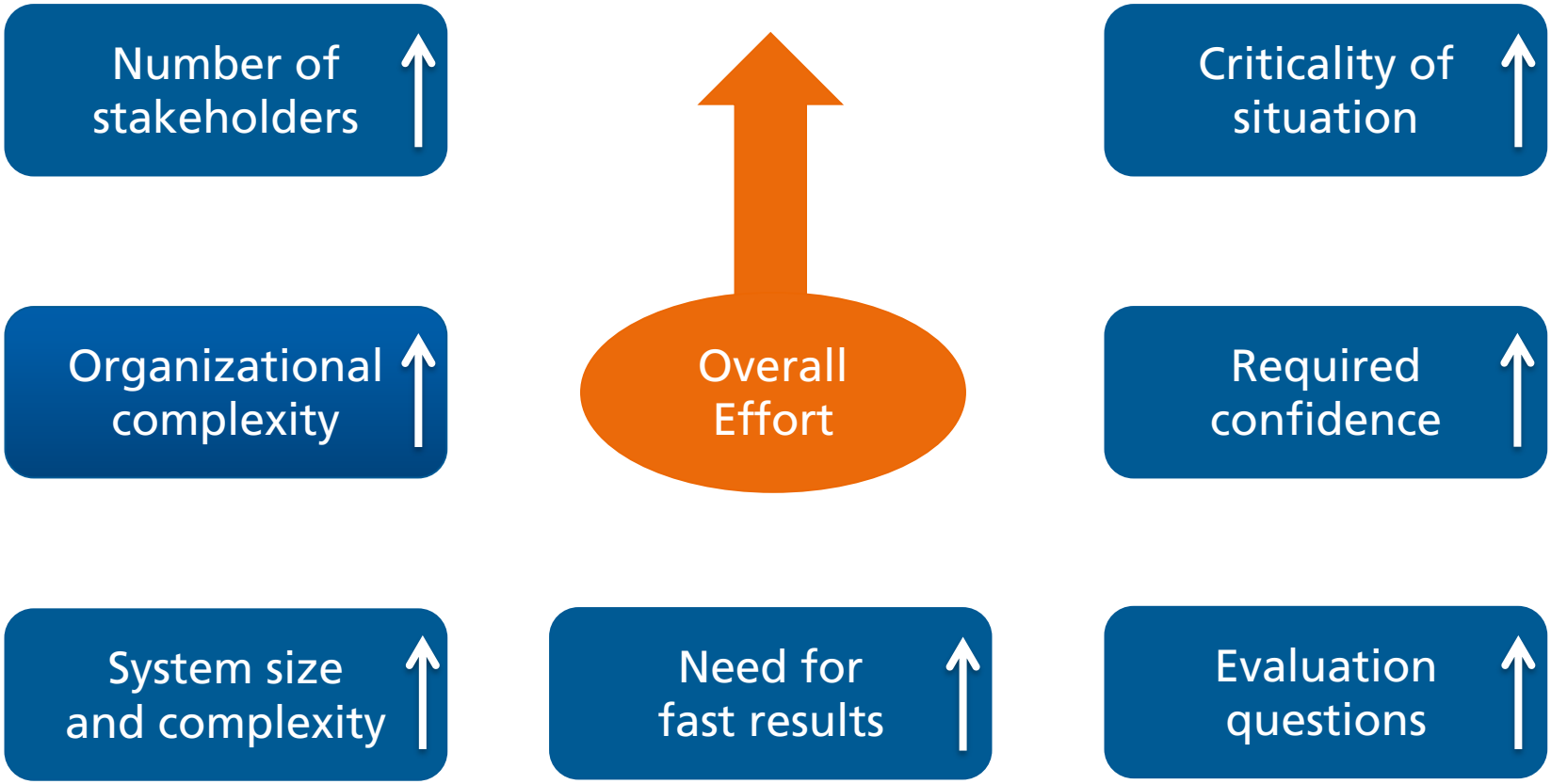
F : Future Architecture Driver



Fraunhofer IESE

Audits and Application

Factors Driving Effort for Architecture Evaluation



Fraunhofer IESE

Findings: Requirements that are Often Neglected

Runtime Quality Attributes

- Typically known
- Partially missing quantification
- Often addressed well

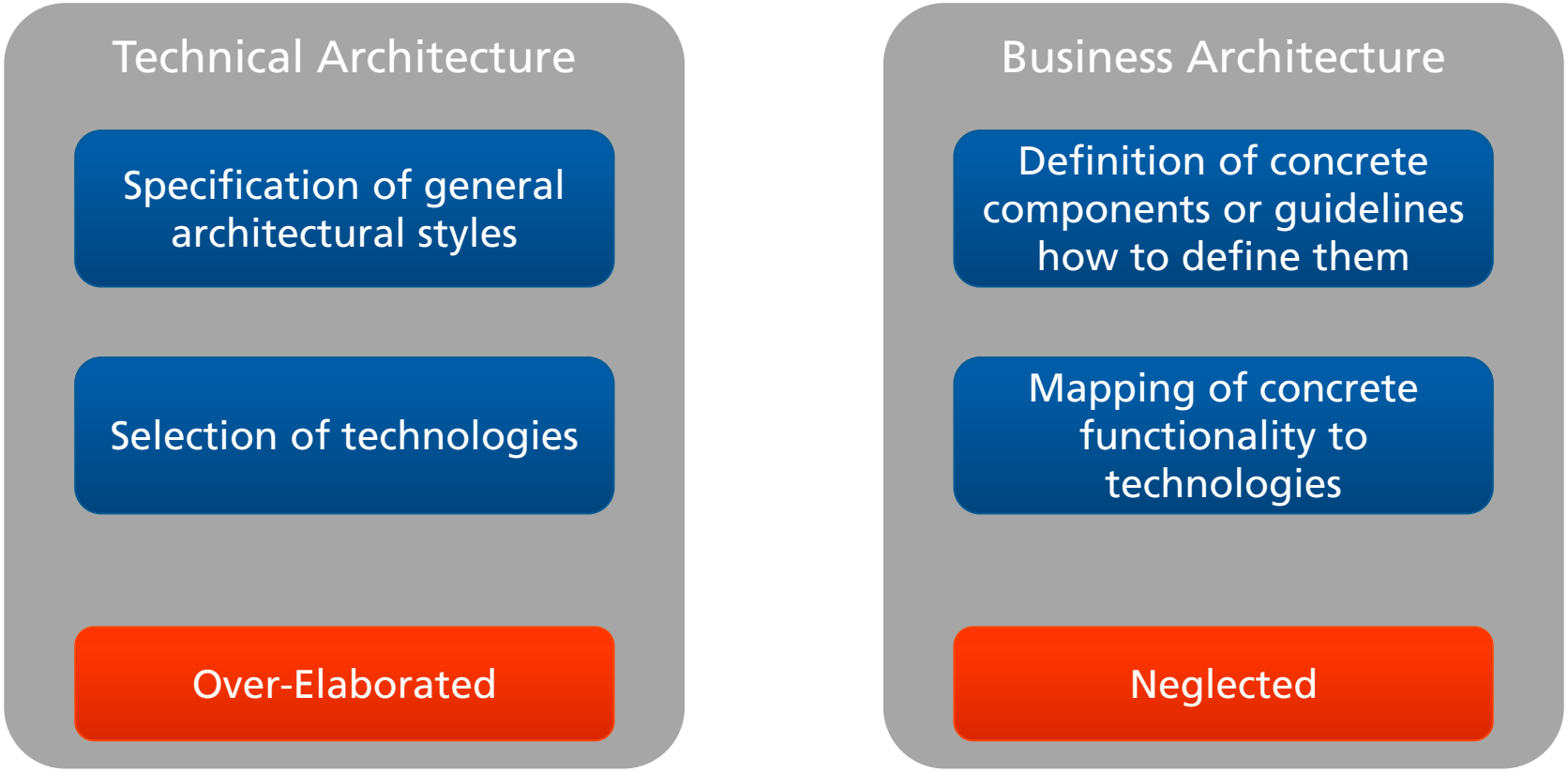
Devtime Quality Attributes

- Often not explicitly known
- Often hard to quantify
- Often not addressed well

Operation Quality Attributes

- Typically not explicitly known
- Partially missing quantification
- Often not addressed well

Findings: Aspects that are „Over-Elaborated“



Fraunhofer IESE

Findings: Architecture Documentation

Architectural Requirements	Often not available	Often very good knowledge
Architecture	Often not available	Often very good knowledge
Implementation	→ Missing uniformity, lack of compliance, quality problems	

Reconstruction is essential as basis for evaluation

D. Rost, M. Naab: Architecture Documentation for Developers: A Survey, ECSA 2013

Fraunhofer IESE

Interpretation of Evaluation Results

No standard interpretation possible

Stakeholders partially try to influence the interpretation for their goals

Interpretation has to consider evaluation questions + many context factors

Architecture Evaluation often not fully objective and quantitative

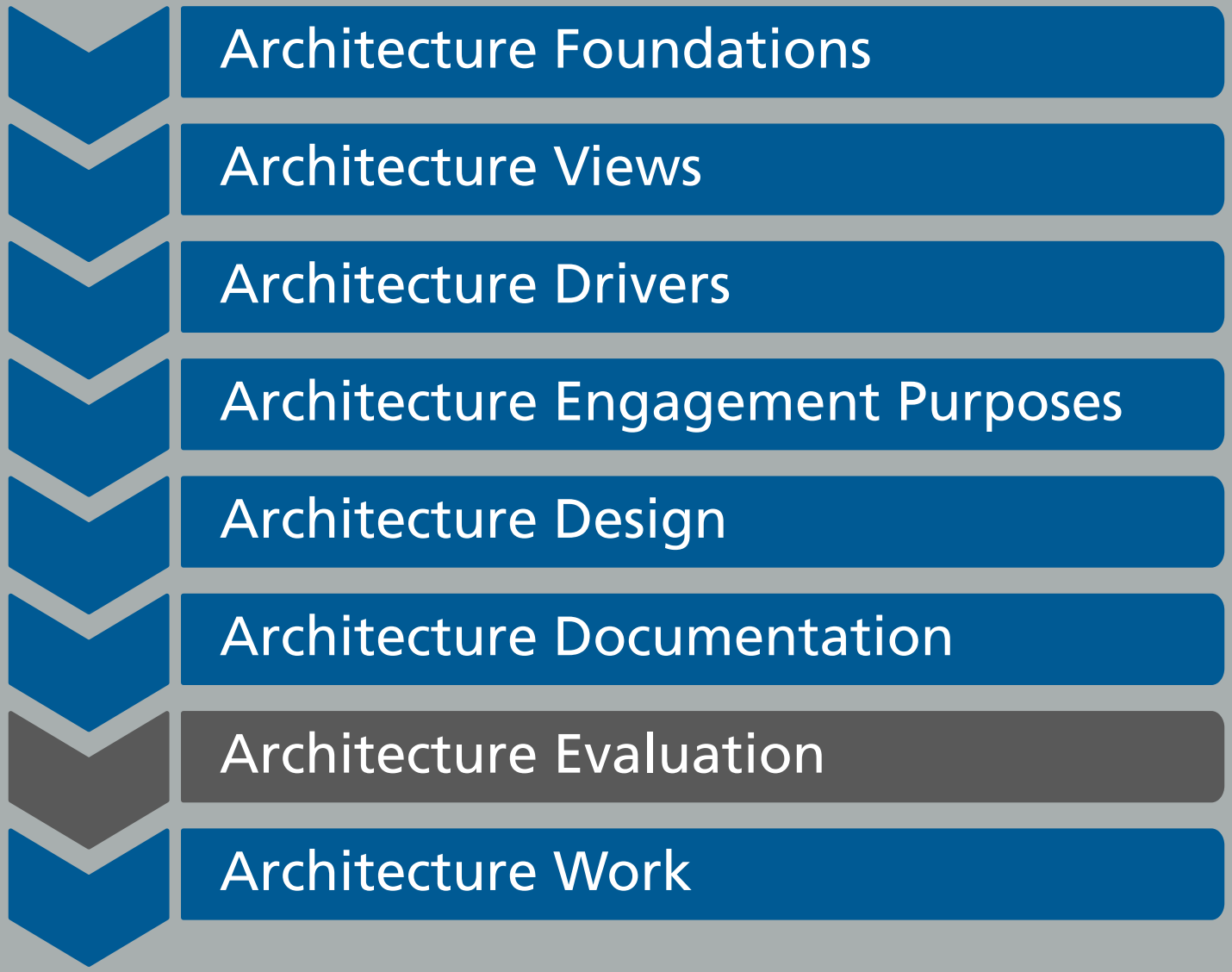
Tool-based reverse engineering often leads to nice but useless visualizations

Even quantitative data (e.g. number of incompliant relationships) often hard to interpret

Representation of results for management is challenging (→ actions?)

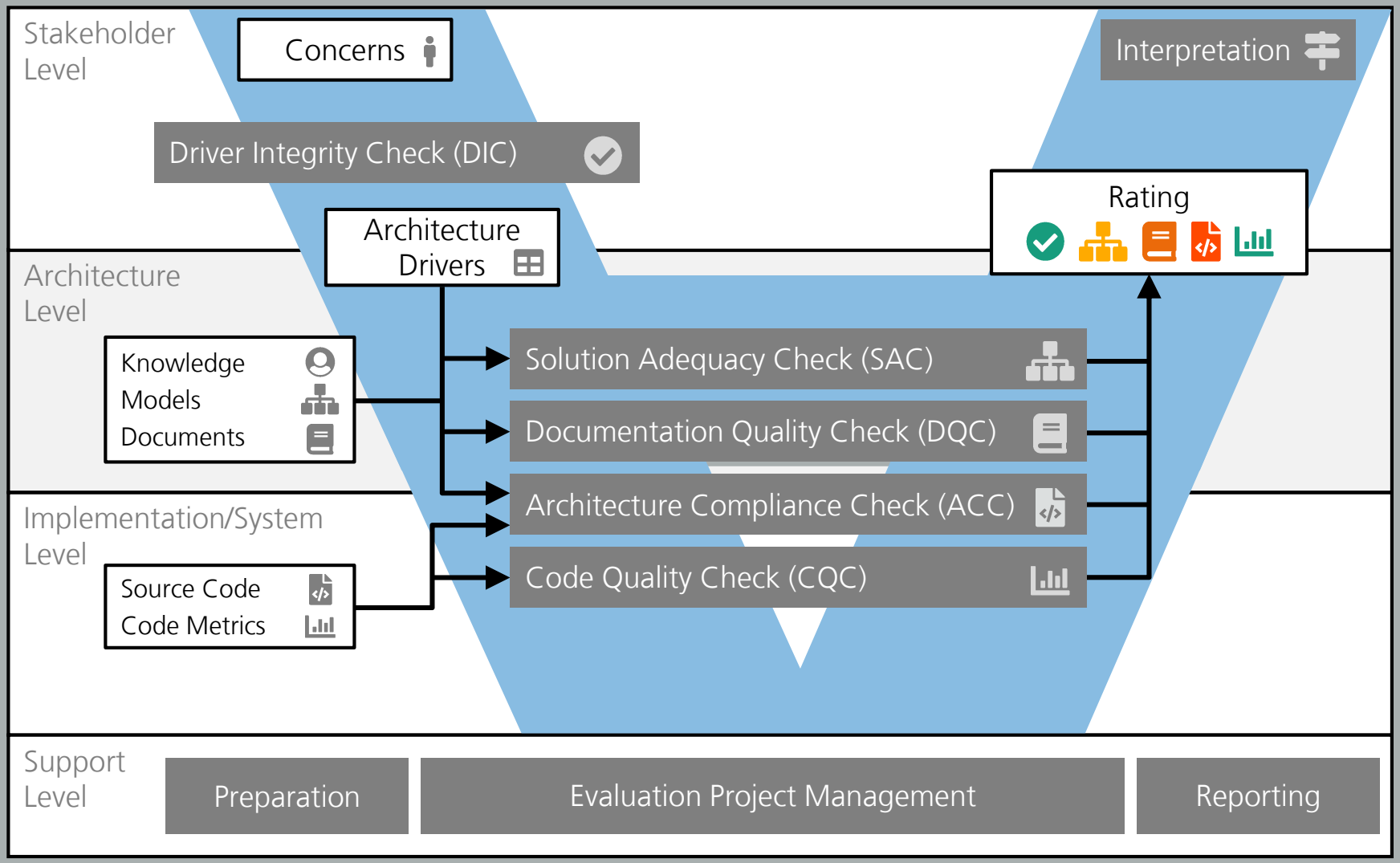
Fraunhofer IESE

Wrap Up


Architecture Foundations
Architecture Views
Architecture Drivers
Architecture Engagement Purposes
Architecture Design
Architecture Documentation
Architecture Evaluation
Architecture Work

Points in Time for Architecture Evaluation

Architecture Evaluation with Fraunhofer RATE



Architekturbewertung

Systematische Analyse von Produktqualität

Wissen
KNOW-HOW FÜR IT-PROFESSIONALS

OBJEKTspektrum

Dr. Matthias Naab, Dr. Jens Knodel
Fraunhofer IESE, http://architecture.iese.fraunhofer.de/

Was ist das?
- Systematische Analyse von Produktqualität
- Identifikation von Risiken
- Bewertung von Alternativen

Warum?
- Um fundierte Geschäftsentscheidungen zu treffen
- Um Produktqualität zu überprüfen und sicherzustellen
- Um Technologieentscheidungen zu treffen
- Um Evolutions- und Migrationsentscheidungen zu treffen

Wer braucht es?
- **Intern:** Entscheider, Projektleiter, Chefs, Entwickler, ...
- **Extern:** Kunden (potentiell, bestehend), Auftraggeber ...

Wann?
- Zu jeder Phase im Lebenszyklus: Neuentwicklung, Evolution, Variantenbildung, Ablösung
- Technologieentscheidungen, Akquisitionsentscheidungen

Wie?

Wie viel?
- Faktoren, die Aufwand beeinflussen (je höher, desto mehr Aufwand): Bewertungsziele, Anzahl Stakeholder, organisatorische Komplexität, Systemgröße, zeitnahe Ergebnisse, Kritikalität der Situation, benötigte Konfidenz
- Risikobasiertes Vorgehen: je nach Kritikalität und Unsicherheit Kombination verschiedener Bewertungsansätze
- Konfidenzlevel (je höher, desto mehr Aufwand): Vertrauen, Expertenerfahrung, externe Begutachtung, modell-basierte Simulation, Technischer Durchstich / Prototyp, Nachweis im realen System

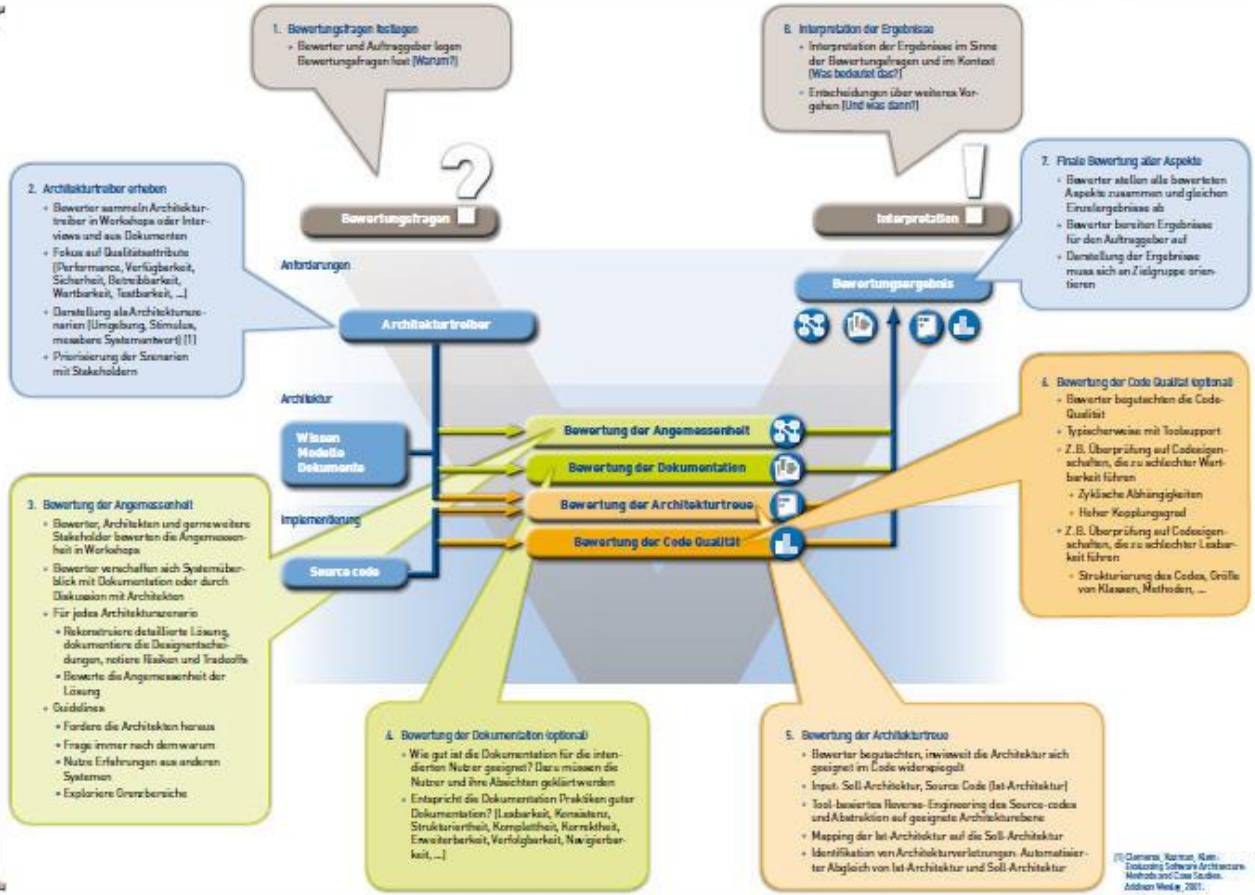
Wer macht es?
- **Intern:** Architekt, Entwickler, Architekt aus anderem Bereich
- **Extern:** Architekt des Kunden, neutrale Auditoren

Wer macht mit?
- Prinzipiell alle Stakeholder des Produkts: Produktmanager, Projektmanager, Kunde, Nutzer, Marketing, Betrieb, Architekt, Entwickler, Tester, ...

Womit?
- Modellierungswerkzeuge und Office-Tools
- Tools für Reverse Engineering und Compliance Checking
- Tools für Metrikerhebung und zur Erkennung von typischen Problemen

Was bedeutet das?
- Interpretation der Ergebnisse je nach Kontext und Bewertungszielen
- Bewertung meist nicht vollständig objektiv und quantitativ möglich
- Darstellung mit Ampelfarben hat sich bewährt

Und was dann?
- Maßnahmenauswahl: Priorisierung nach Kritikalität und Nutzen
- Maßnahmenkategorien: Entscheidungen treffen und umsetzen, Anforderungen schärfen, Lösungskonzepte überarbeiten, Dokumentation verbessern, Architekturverletzungen eliminieren, Architekturtraining



[1] Clements, Kazman, Klein: Evaluating Software Architectures. Methods and Case Studies. Addison Wesley, 2001.

Poster available from OBJEKTspektrum