Figures and Illustrations
J. Knodel, M. Naab:
http://pragmatic-architecting.com
© Fraunhofer IESE 2016 | © Springer International Publishing Switzerland 2016
Did we identify the right stakeholders?

Did we elicit the right concerns?

Did we set the right scope of delivery?

Did we design the right solution?

Did we build the product right?

Did the product satisfy the concerns?

Software Product Assessment
http://pragmatic-architecting.com
© Fraunhofer IESE 2016 | © Springer International Publishing Switzerland 2016
<table>
<thead>
<tr>
<th>Artifact Quality</th>
<th>Believed</th>
<th>Inspected</th>
<th>Measured</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Quality</td>
<td>Believed</td>
<td>Predicted</td>
<td>Probed</td>
</tr>
<tr>
<td>Human Capabilities</td>
<td>Product Properties</td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------------------</td>
<td>--------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Software System (Flaws/Weaknesses)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Technologies in Use (Misusage/Known Pitfalls)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Complexity (Comprehensibility/Deficiencies)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Evolution (Change-Proneness/History)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Cause

Baseline

Universal, Domain, Organization, Individual

http://pragmatic-architecting.com
© Fraunhofer IESE 2016 | © Springer International Publishing Switzerland 2016
<table>
<thead>
<tr>
<th>Balance of findings</th>
<th>Severity of findings</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mainly negative findings</td>
<td>Critical</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>Harmful</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Minor</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Harmless/Advantageous</td>
<td></td>
</tr>
<tr>
<td>Negative findings predominate</td>
<td>Partial</td>
<td></td>
</tr>
<tr>
<td>Positive findings predominate</td>
<td>Large</td>
<td></td>
</tr>
<tr>
<td>Mainly positive findings</td>
<td>Full</td>
<td></td>
</tr>
</tbody>
</table>

Legend:
- Critical
- Harmful
- Minor
- Harmless / Advantageous
- N/A

http://pragmatic-architecting.com
© Fraunhofer IESE 2016 | © Springer International Publishing Switzerland 2016
Is the architecture adequate for the requirements?

Is the architecture documentation adequate?

Is the code consistent with the architecture as it was planned?

Does the code have good overall quality?
# Architecture Driver Template

<table>
<thead>
<tr>
<th>Categorization</th>
<th>Responsibilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Driver Name</td>
<td>Application startup time</td>
</tr>
<tr>
<td>Driver ID</td>
<td>AD.01.PERFORMANCE</td>
</tr>
<tr>
<td>Status</td>
<td>Realized</td>
</tr>
<tr>
<td>Priority</td>
<td>High</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
<th>Quantification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Environment</td>
<td>Previous starts $\geq 1$</td>
</tr>
<tr>
<td>Stimulus</td>
<td>Initial startup time $&lt; 1s$</td>
</tr>
<tr>
<td>Response</td>
<td>Full startup time $&lt; 5s$</td>
</tr>
</tbody>
</table>

The application is installed on the system and has been started before at least once. The application is currently closed and the system is running on normal load.

A user starts the application from the Windows start menu.

The application starts and is ready for inputting search data in less than 1 second. The application is ready for fast answers to search queries after 5 seconds.
## Driver Solution Template

<table>
<thead>
<tr>
<th>Driver Name</th>
<th>Application startup time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Driver ID</td>
<td>AD.01.PERFORMANCE.</td>
</tr>
</tbody>
</table>

| Steps | 1. Application always stores preprocessed index-structures on updates of searchable items  
      | 2. On startup, loading of search data is moved to a separate thread  
      | 3. The UI is started and ready for user input while loading of search data is ongoing  
      | 4. After loading the search data, searches can be done without the user noticing that search was not available before |

| Related Design Decisions | DD.01 Decoupled loading of search data  
                          | DD.12 Preprocessed index-structures of search data |

<table>
<thead>
<tr>
<th>Pros &amp; Opportunities</th>
<th>• Very fast startup time, application directly usable by user</th>
</tr>
</thead>
</table>

| Cons & Risks | • More effort in realization  
              | • Loading in separate thread requires synchronization and makes implementation more difficult |

| Assumptions & Quantifications | • Data can be loaded in 5s  
                                | • User rarely sends a search in less than 4s after start is completed |
| Trade-Offs | • Maintainability, understandability |
## Decision Rationale Template

<table>
<thead>
<tr>
<th><strong>Decision Name</strong></th>
<th>Decoupled loading of search data</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Design Decision ID</strong></td>
<td>DD.01</td>
</tr>
<tr>
<td><strong>Explanation</strong></td>
<td>Loading the search data is done in a separate thread. The application’s UI can be started and used for typing in search queries before the search data is actually loaded.</td>
</tr>
</tbody>
</table>

**Pros & Opportunities**
- Data loading time does not add on startup time

**Cons & Risks**
- Loading in separate thread requires synchronization and makes implementation more difficult

**Assumptions & Quantifications**
- Data can be loaded in 5s

**Trade-Offs**
- Maintainability, understandability
**Driver Integrity Check (DIC)**

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

**Input**
- Requirements documentation
- Architecture documents (if available)

**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

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

**Rating**
- Severity and balance of findings

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

**Confidence Levels**
- Predicted
- Probed

**Tools**
- Documentation tools

**Evaluators**
- Architect
- Peers
- External auditor

http://pragmatic-architecting.com
© Fraunhofer IESE 2016 | © Springer International Publishing Switzerland 2016
Confidence vs. Effort

- Low Confidence, Low Effort: Self-Controlled
- Low Confidence, High Effort: Estimated
- High Confidence, Low Effort: Reviewed 3rd Party
- High Confidence, High Effort: Probed

**Applicability**
(Diameter: Applicability to number of drivers)
**Solution Adequacy Check (SAC)**

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

**Input**
- Architecture drivers
- Architecture documentation

**Involved Stakeholders**
- Architects of system under evaluation
- Further stakeholders of system (optional)

**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

**Rating**
Severity and balance of findings

**Confidence Levels**
- Predicted
- Probed
- Tested

http://pragmatic-architecting.com
© Fraunhofer IESE 2016 | © Springer International Publishing Switzerland 2016
http://pragmatic-architecting.com
© Fraunhofer IESE 2016 | © Springer International Publishing Switzerland 2016
**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

**Rating**
Severity and balance of findings

**Output**
Findings on adequacy of documentation and adherence to best practices

**Evaluators**
- Architect
- Peers
- External auditor

**Tools**
- Best practice and style checkers

**Confidence Levels**
- Inspected
- Measured
http://pragmatic-architecting.com
© Fraunhofer IESE 2016 | © Springer International Publishing Switzerland 2016
Architecture Compliance Check (ACC)

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

**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

**Involved Stakeholders**
- Architects and developers of the system under evaluation

**Rating**
Severity and balance of findings

**Output**
Findings on the compliance of the implementation with respect to the intended architecture
- Convergences
- Divergences (violation)
- Absences (violation)

**Evaluators**
- Architect
- Peers
- External auditor

**Confidence Levels**
- Inspected
- Measured

**Tools**
- Compliance checking tools

http://pragmatic-architecting.com
© Fraunhofer ISE 2016 | © Springer International Publishing Switzerland 2016
Intended Architecture
(Structural Model)

Implemented Architecture
(Model Extracted from Source Code)

Compliance Checking
(Comparison of Models)

http://pragmatic-architecting.com
© Fraunhofer IESE 2016 | © Springer International Publishing Switzerland 2016
Compliance Checking

Behavioral Model vs.
Runtime Trace Model

Instance
Convergence
Divergence
Absence
**Code Quality Check (CQC)**

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

**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

**Input**
- Source code
- (Build scripts)

**Output**
- Findings on quality of the source code
  - Best practice violations
  - Code clones
  - Quality warnings (maintainability, security, ...)
  - Code metrics
  - ...

**Evaluators**
- Architect / Quality Engineer
- Peers
- External auditor

**Tools**
- Code quality tools (style checker, clone detection, quality warning checker, ...)

**Rating**
- Severity and balance of findings

**Confidence Levels**
- Inspected
- Measured

http://pragmatic-architecting.com
© Fraunhofer IESE 2016 | © Springer International Publishing Switzerland 2016
<table>
<thead>
<tr>
<th>Architecture Requirements</th>
<th>DIC</th>
<th>Example 1 System</th>
<th>Example 2 System</th>
<th>Example 3 System 1</th>
<th>Example 3 System 2</th>
<th>Example 4 System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture</td>
<td>SAC</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Architecture Documentation</td>
<td>DQC</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Architecture Compliance</td>
<td>ACC</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Code Quality</td>
<td>CQC</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Quality Attribute</td>
<td>Product 1</td>
<td>Product 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>-----------</td>
<td>-----------</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Availability</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reliability</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Performance</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Resource Utilization</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usability</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Security</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scalability</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maintainability / Extensibility</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flexibility</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Integrateability</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Migration Support</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Portability</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

http://pragmatic-architecting.com
© Fraunhofer IESE 2016 | © Springer International Publishing Switzerland 2016
Accuracy
Availability
Businessgoal
Consistency
Flexibility
Interoperability
Monitoring
Operability
Performance
Reliability
Updatability
User Experience

F : Future Scenario

http://pragmatic-architecting.com
© Fraunhofer IESE 2016 | © Springer International Publishing Switzerland 2016
<table>
<thead>
<tr>
<th>Quality Attribute</th>
<th>Solution Adequacy</th>
<th>Solution Concepts</th>
<th>Compliance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td><img src="image" alt="Red" /></td>
<td>Shared Framework</td>
<td><img src="image" alt="Green" /></td>
</tr>
<tr>
<td>Performance</td>
<td><img src="image" alt="Red" /> <img src="image" alt="Yellow" /></td>
<td>Layering</td>
<td><img src="image" alt="Red" /></td>
</tr>
<tr>
<td>Resource Utilization</td>
<td><img src="image" alt="Green" /> <img src="image" alt="Yellow" /></td>
<td>Service-Orientation</td>
<td><img src="image" alt="Red" /></td>
</tr>
<tr>
<td>Operability</td>
<td><img src="image" alt="Green" /> <img src="image" alt="Orange" /></td>
<td>Data Model Separation</td>
<td><img src="image" alt="Red" /></td>
</tr>
<tr>
<td>Security</td>
<td><img src="image" alt="Orange" /> <img src="image" alt="Yellow" /></td>
<td>Business Process and GUI Design</td>
<td><img src="image" alt="Red" /></td>
</tr>
<tr>
<td>Scalability</td>
<td><img src="image" alt="Green" /></td>
<td>Event-Orientation</td>
<td><img src="image" alt="Yellow" /></td>
</tr>
<tr>
<td>Maintainability / Extensibility</td>
<td><img src="image" alt="Yellow" /> <img src="image" alt="Orange" /> <img src="image" alt="Red" /></td>
<td>Clustering</td>
<td><img src="image" alt="Red" /></td>
</tr>
<tr>
<td>Flexibility</td>
<td><img src="image" alt="Red" /></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reusability</td>
<td><img src="image" alt="Orange" /> <img src="image" alt="Red" /></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Migration Support</td>
<td><img src="image" alt="Orange" /> <img src="image" alt="Red" /></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Testability</td>
<td><img src="image" alt="Yellow" /> <img src="image" alt="Red" /></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Point in Time</td>
<td>Framework 1st Generation</td>
<td>Framework 2nd Generation</td>
<td>Framework 3rd Generation</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------------------</td>
<td>--------------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>t1</td>
<td>4.3%</td>
<td>10.2%</td>
<td>27.3%</td>
</tr>
<tr>
<td>t2</td>
<td>2.1%</td>
<td>1.9%</td>
<td>2.0%</td>
</tr>
<tr>
<td>t3</td>
<td>1.6%</td>
<td>1.5%</td>
<td>1.9%</td>
</tr>
<tr>
<td>t4</td>
<td>0,9%</td>
<td>5,8%</td>
<td></td>
</tr>
<tr>
<td>t5</td>
<td>1.7%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Emergency
Keep or Let Die, Crisis, Clash

Decision Support
Business Goals, Quality, Migration, Technologies, Reuse

Quality Management
Strategy, Institutionalization

Rescue
Evolution vs. Revolution

System “on plan”
System “out of hand”

Evaluation only

Evaluation & Improvement
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td># Persons Involved</td>
<td>Max-Min</td>
<td>[Mean]</td>
<td>Max-Min</td>
<td>[Mean]</td>
</tr>
<tr>
<td></td>
<td>∑ Total Person-Days</td>
<td></td>
<td></td>
<td>∑ Total Person-Days</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>#</td>
<td></td>
<td></td>
<td>∑ Total Person-Days</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10-1</td>
<td></td>
<td>10-0</td>
<td>5-0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[2]</td>
<td></td>
<td>[1]</td>
<td>[0]</td>
</tr>
<tr>
<td>Scope Evaluation Context</td>
<td></td>
<td># 2-0</td>
<td></td>
<td># 15-1</td>
<td>[1]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[1]</td>
<td></td>
<td>[1]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>∑ 10-0</td>
<td></td>
<td>∑ 10-1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[1]</td>
<td></td>
<td>[1]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>∑ 10</td>
<td></td>
<td>∑ 10</td>
<td></td>
</tr>
<tr>
<td>Setup Evaluation Project</td>
<td></td>
<td># 2-1</td>
<td></td>
<td># 15-1</td>
<td>[1]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[1]</td>
<td></td>
<td>[1]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>∑ 10-1</td>
<td></td>
<td>∑ 10-1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[1]</td>
<td></td>
<td>[1]</td>
<td></td>
</tr>
<tr>
<td>Conduct Evaluation</td>
<td></td>
<td># 6-1</td>
<td></td>
<td># 15-3</td>
<td>[5]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[2]</td>
<td></td>
<td>[5]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>∑ 200-4</td>
<td></td>
<td>∑ 60-2</td>
<td>[20]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[30]</td>
<td></td>
<td>[20]</td>
<td></td>
</tr>
<tr>
<td>Package Evaluation Results</td>
<td></td>
<td># 6-1</td>
<td></td>
<td># 15-1</td>
<td>[10]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[2]</td>
<td></td>
<td>[10]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>∑ 20-1</td>
<td></td>
<td>∑ 5-1</td>
<td>[4]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[10]</td>
<td></td>
<td>[4]</td>
<td></td>
</tr>
<tr>
<td>TOTAL</td>
<td></td>
<td># 15-1</td>
<td></td>
<td># 15-1</td>
<td>[10]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[10]</td>
<td></td>
<td>[10]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>∑ 10-1</td>
<td></td>
<td>∑ 10</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[4]</td>
<td></td>
<td>[4]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>∑ 40-4</td>
<td></td>
<td>∑ 240-4</td>
<td>[42]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[12]</td>
<td></td>
<td>[42]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>∑ 240-4</td>
<td></td>
<td>∑ 80-4</td>
<td>[25]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[42]</td>
<td></td>
<td>[25]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>∑ 80-4</td>
<td></td>
<td>∑ 240-4</td>
<td>[42]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[25]</td>
<td></td>
<td>[42]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>∑ 240-5</td>
<td></td>
<td>∑ 240-5</td>
<td>[25]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[25]</td>
<td></td>
<td>[25]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>∑ 240-5</td>
<td></td>
<td>∑ 240-5</td>
<td>[25]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[25]</td>
<td></td>
<td>[25]</td>
<td></td>
</tr>
</tbody>
</table>

http://pragmatic-architecting.com
© Fraunhofer IESE 2016 | © Springer International Publishing Switzerland 2016
<table>
<thead>
<tr>
<th>Typical Findings</th>
<th>Potential Gains if Fixed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unknown requirements or constraints</td>
<td>Time and effort saved in software engineering; late surprises (if requirements are discovered later) causing additional overhead are avoided.</td>
</tr>
<tr>
<td>Unknown architecture drivers</td>
<td>Critical requirements and complex problems requiring design excellence are not addressed appropriately. If detected early, more design effort can be spent on coming up with adequate solutions.</td>
</tr>
<tr>
<td>Architecture typically not thoroughly defined</td>
<td>Risks are mitigated and flaws in the architecture design can be fixed or circumvented before the need for change is imminent.</td>
</tr>
<tr>
<td>Architecture often not fully adequate for requirements (or not any longer for older systems)</td>
<td>Need for change for (future) requirement becomes obvious and can be addressed accordingly. Effort for designing is applied effectively.</td>
</tr>
<tr>
<td>Documentation incomplete, inconsistent, or contradictory</td>
<td>Assumptions and decisions based on missing or wrong information are avoided.</td>
</tr>
<tr>
<td>Documentation not available</td>
<td>Documentation can serve as communication and information vehicle for architects and other stakeholders.</td>
</tr>
<tr>
<td>Missing uniformity in implementation</td>
<td>Easier transfer of developers within the project and less time and effort for getting familiar with the source code.</td>
</tr>
<tr>
<td>Lack of architecture compliance or chaotic implementation</td>
<td>Improved maintainability and architecture can be used as abstraction and mediator for decision making and for principles guiding the evolution of the software system.</td>
</tr>
<tr>
<td>Limited architecture awareness among developers</td>
<td>Better knowledge of the architecture, especially in the part(s) of the system on which they are working (i.e., implementing or maintaining) and those part(s) that are dependent on the current working part (i.e., the context).</td>
</tr>
</tbody>
</table>
Why?
- To make informed business decisions
- To make informed technical decisions
- To make informed evolution decisions
- To increase the confidence in decisions
- To answer questions about software systems
- To check and assure product quality

What?
- Systematic analysis of product quality
- Identification of risks
- Evaluation of alternatives

How?
- Concerns
  - Knowledge Models
  - Documents
  - Source Code
  - Code Metrics
- Architectural Drivers
- Interpretation
  - Rating
  - Solution Adequacy Check (SAC)
  - Documentation Quality Check (DQC)
  - Architecture Compliance Check (ACC)
  - Code Quality Check (CQC)

When?
- At any time in the product lifecycle: New development, evolution, creation of variants, retirement
- At any time in development: Elicitation of architectural requirements, early design, implementation, evolution
- For different types of decisions: Product strategy, selection of technologies, acquisitions

How much?
- Influencing factors: Evaluation goals, stakeholders, organizational complexity, system size, criticality
- Confidence levels: Believed, predicted, probed, tested, inspected, measured
- Risk-based approach: Selection of adequate techniques and confidence levels | Design to cost and time