 Template

Mai 2022

**Über arc42**

arc42, das Template zur Dokumentation von Software- und Systemarchitekturen.

Template Version 8.1 DE. (basiert auf AsciiDoc Version), Mai 2022

Created, maintained and © by Dr. Peter Hruschka, Dr. Gernot Starke and contributors. Siehe <https://arc42.org>.

Diese Version des Templates enthält Hilfen und Erläuterungen. Sie dient der Einarbeitung in arc42 sowie dem Verständnis der Konzepte. Für die Dokumentation eigener System verwenden Sie besser die *plain* Version.

# Einführung und Ziele

## Aufgabenstellung

**Inhalt**

Dieses Dokument beschreibt die Middleware Komponente der Advanced Client – Server von Tron.

Die Middleware soll folgende Anforderungen erfüllen:

### Use Cases

**Nummer: UC-1**

**Titel:** Nameserver

**Beschreibung:**

* Die Middleware benötigt einen Nameserver, bei dem sich Komponenten mit ihren Methoden registrieren können.
* Der Nameserver sollte dann alle Methoden kennen und wo sie zu finden sind.
* Andere Komponenten sollen dann Anfragen an den Nameserver stellen können, um zu erfragen, wo man die jeweiligen registrierten Methoden findet

**Nummer: UC-2**

**Titel:** Marshaling & unmarshiling

**Beschreibung:** Die Middleware soll Nachrichten so formatieren, dass sie über das Netzwerk geschickt werden können um beispielsweise Methodenaufrufe zu initieren. Diese Nachrichten sollen einmal “gepackt” und auch wieder “entpackt” werden.

**Nummer: UC-3**

**Titel:** Remote Invocation

**Beschreibung:**

* Die Middleware soll Nachrichten über das Netzwerk verschicken können. Die Nachrichten sollen Methodenaufrufe beinhalten. Die Nachrichten sollen im richtigen format geschickt werden und beim Empfänger einen methodenaufruf auslösen.
* Dabei soll sowohl Synchrone als auch asynchrone aufrufe gedeckt werden.
* Dabei sollen auch zeitkritische aufrufe mitbedacht werden (TCP / UDP)

## Motivation

Die wesentliche Motivation für uns, diese Middleware zu implementieren ist es, die PVL zu erhalten. Weitere Motivationspunkte wären aber auch, neues zu lernen und unser bisheriges Wissen zu vertiefen.

## Qualitätsziele

|  |
| --- |
| **Qualitätsziele:** |
| Gut definierte Schnittstellen |
| Kompatibilität zu einer anderen Gruppe (Mindestens zwei Teams müssen miteinander Spielen können) |
| Fehlertoleranz (Wenn ein Spieler abstürzt, egal welcher Spieler, dann geht das Spiel trotzdem weiter) -> Stabilität |
| Die Transparenz Kriterien sollten so gut wie moeglich erfuellt werden. |

## Stakeholder

**Inhalt**

Unsere Stakeholder sind die Entwickler (Studenten), der Kunde (Professor) und die Spieler (Studenten).

Die Stakeholder mit deren Kontakt werden in der unteren Tabelle aufgelistet:

| Rolle | Kontakt | Erwartungshaltung |
| --- | --- | --- |
| *Entwickler* | [Dominik.martin@haw-hamburg.de](mailto:Dominik.martin@haw-hamburg.de) | *Ein gutes Spiel zu programmieren & die PVL zu erhalten* |
| *Entwickler* | [*Can.heintze@haw-hamburg.de*](mailto:Can.heintze@haw-hamburg.de) | *Ein gutes Spiel zu programmieren & die PVL zu erhalten* |
| *Entwickler* | [*Dominik.mueller@haw-hamburg.de*](mailto:Dominik.mueller@haw-hamburg.de) | *Ein gutes Spiel zu programmieren & die PVL zu erhalten* |
| *Kunde* | [*Martin.becke@haw-hamburg.de*](mailto:Martin.becke@haw-hamburg.de) | *Ein Lauffaehiges Spiel, bei dem 6 Spieler gleichzeitig ein komplettes Spiel ohne fehler durchspielen koennen.* |
| *Spieler / Nutzer* | *n/a* | *Ein funktionierendes Spiel spielen und dabei Spaß haben* |

# 

# Randbedingungen

**Inhalt**

Randbedingungen und Vorgaben, die ihre Freiheiten bezüglich Entwurf, Implementierung oder Ihres Entwicklungsprozesses einschränken. Diese Randbedingungen gelten manchmal organisations- oder firmenweit über die Grenzen einzelner Systeme hinweg.

**Motivation**

Für eine tragfähige Architektur sollten Sie genau wissen, wo Ihre Freiheitsgrade bezüglich der Entwurfsentscheidungen liegen und wo Sie Randbedingungen beachten müssen. Sie können Randbedingungen vielleicht noch verhandeln, zunächst sind sie aber da.

**Form**

Einfache Tabellen der Randbedingungen mit Erläuterungen. Bei Bedarf unterscheiden Sie technische, organisatorische und politische Randbedingungen oder übergreifende Konventionen (beispielsweise Programmier- oder Versionierungsrichtlinien, Dokumentations- oder Namenskonvention).

Siehe [Randbedingungen](https://docs.arc42.org/section-2/) in der online-Dokumentation (auf Englisch!).

# Kontextabgrenzung

## Fachlicher Kontext

## Technischer Kontext

# Lösungsstrategie

### Funktionale Anforderungen

|  |  |  |  |
| --- | --- | --- | --- |
| **Methoden Signatur** | **Beschreibung** | **Fehlersemantik** | **UseCase** |
| Void register(String methodName, Array<String> parameter) | * Methode, damit sich der caller an den Nameservice verbindet * Registriert die jeweilige Methode, in einer map unter dem Port / IP Adresse vom jeweiligen Caller |  | UC-1 |
| Array<String> [Ip, Port] query(String methodName, Array<String> parameter) | * Methode, damit der Nameservice die richtige Komponente findet * Gibt den port & die IP Adresse zurueck, an welcher stelle die methode gefunden wird |  | UC-1 |
| String message : marshaling(String methodName, Array<String> parameter) | * Erstellt eine Nachricht in JSON Format, welche dann ueber das Netzwerk geschickt werden kann |  | UC-2 |
| Object returnValue : unmarshaling(String message) | * Nimmt die message & ruft die methode mit den jeweiligen parametern auf |  | UC-2 |
| String message : marhsalReturnValue(Object returnValue) | * Baut einen JSON String um den returnValue einer aufgerufenen Methode zurueck zu senden |  | UC-2 |
| Object returnValue : invokeSynchron(String methodName, Array<String> parameter) | * Ruft query auf * Ruft marshaling auf * Sendet mit TCP |  | UC-3 |
| Void : invokeAsynchron(String methodName, Array<String> parameter) | * Ruft query auf * Ruft marshaling auf * Sendet mit UDP bei zeitkritischen aufrufen (Spiele logik), sendet mit TCP bei nicht zeitkritischen aufrufen |  | UC-3 |
| Void : receive(String message) | * Ruft unmarshaling auf * Sendet ggf. Den returnValue an den sender zurueck, mithilfe von marshalReturnValue() |  | UC-3 |

# Bausteinsicht

**Komponenten Diagramm auf oberster Ebene:**

Graphical user interface, application

Description automatically generated

**Komponenten Diagramm auf zweiter Ebene:**

Diagram

Description automatically generated

# Laufzeitschicht

**Sequenzdiagramme**

Diagram, schematic

Description automatically generatedDiagram

Description automatically generatedDiagram

Description automatically generatedDiagram

Description automatically generatedDiagram

Description automatically generated

**Ebene 1** ist die Whitebox-Beschreibung des Gesamtsystems, zusammen mit Blackbox-Beschreibungen der darin enthaltenen Bausteine.

**Ebene 2** zoomt in einige Bausteine der Ebene 1 hinein. Sie enthält somit die Whitebox-Beschreibungen ausgewählter Bausteine der Ebene 1, jeweils zusammen mit Blackbox-Beschreibungen darin enthaltener Bausteine.

**Ebene 3** zoomt in einige Bausteine der Ebene 2 hinein, usw.

Siehe [Bausteinsicht](https://docs.arc42.org/section-5/) in der online-Dokumentation (auf Englisch!).

## Whitebox Gesamtsystem

An dieser Stelle beschreiben Sie die Zerlegung des Gesamtsystems anhand des nachfolgenden Whitebox-Templates. Dieses enthält:

* Ein Übersichtsdiagramm
* die Begründung dieser Zerlegung
* Blackbox-Beschreibungen der hier enthaltenen Bausteine. Dafür haben Sie verschiedene Optionen:
  + in *einer* Tabelle, gibt einen kurzen und pragmatischen Überblick über die enthaltenen Bausteine sowie deren Schnittstellen.
  + als Liste von Blackbox-Beschreibungen der Bausteine, gemäß dem Blackbox-Template (siehe unten). Diese Liste können Sie, je nach Werkzeug, etwa in Form von Unterkapiteln (Text), Unter-Seiten (Wiki) oder geschachtelten Elementen (Modellierungswerkzeug) darstellen.
* (optional:) wichtige Schnittstellen, die nicht bereits im Blackbox-Template eines der Bausteine erläutert werden, aber für das Verständnis der Whitebox von zentraler Bedeutung sind. Aufgrund der vielfältigen Möglichkeiten oder Ausprägungen von Schnittstellen geben wir hierzu kein weiteres Template vor. Im schlimmsten Fall müssen Sie Syntax, Semantik, Protokolle, Fehlerverhalten, Restriktionen, Versionen, Qualitätseigenschaften, notwendige Kompatibilitäten und vieles mehr spezifizieren oder beschreiben. Im besten Fall kommen Sie mit Beispielen oder einfachen Signaturen zurecht.

***<Übersichtsdiagramm>***

Begründung

*<Erläuternder Text>*

Enthaltene Bausteine

*<Beschreibung der enthaltenen Bausteine (Blackboxen)>*

Wichtige Schnittstellen

*<Beschreibung wichtiger Schnittstellen>*

Hier folgen jetzt Erläuterungen zu Blackboxen der Ebene 1.

Falls Sie die tabellarische Beschreibung wählen, so werden Blackboxen darin nur mit Name und Verantwortung nach folgendem Muster beschrieben:

| **Name** | **Verantwortung** |
| --- | --- |
| *<Blackbox 1>* | *<Text>* |
| *<Blackbox 2>* | *<Text>* |

Falls Sie die ausführliche Liste von Blackbox-Beschreibungen wählen, beschreiben Sie jede wichtige Blackbox in einem eigenen Blackbox-Template. Dessen Überschrift ist jeweils der Namen dieser Blackbox.

### <Name Blackbox 1>

Beschreiben Sie die <Blackbox 1> anhand des folgenden Blackbox-Templates:

* Zweck/Verantwortung
* Schnittstelle(n), sofern diese nicht als eigenständige Beschreibungen herausgezogen sind. Hierzu gehören eventuell auch Qualitäts- und Leistungsmerkmale dieser Schnittstelle.
* (Optional) Qualitäts-/Leistungsmerkmale der Blackbox, beispielsweise Verfügbarkeit, Laufzeitverhalten o. Ä.
* (Optional) Ablageort/Datei(en)
* (Optional) Erfüllte Anforderungen, falls Sie Traceability zu Anforderungen benötigen.
* (Optional) Offene Punkte/Probleme/Risiken

*<Zweck/Verantwortung>*

*<Schnittstelle(n)>*

*<(Optional) Qualitäts-/Leistungsmerkmale>*

*<(Optional) Ablageort/Datei(en)>*

*<(Optional) Erfüllte Anforderungen>*

*<(optional) Offene Punkte/Probleme/Risiken>*

### <Name Blackbox 2>

*<Blackbox-Template>*

### <Name Blackbox n>

*<Blackbox-Template>*

### <Name Schnittstelle 1>

…

### <Name Schnittstelle m>

## Ebene 2

Beschreiben Sie den inneren Aufbau (einiger) Bausteine aus Ebene 1 als Whitebox.

Welche Bausteine Ihres Systems Sie hier beschreiben, müssen Sie selbst entscheiden. Bitte stellen Sie dabei Relevanz vor Vollständigkeit. Skizzieren Sie wichtige, überraschende, riskante, komplexe oder besonders volatile Bausteine. Normale, einfache oder standardisierte Teile sollten Sie weglassen.

### Whitebox *<Baustein 1>*

…zeigt das Innenleben von *Baustein 1*.

*<Whitebox-Template>*

### Whitebox *<Baustein 2>*

*<Whitebox-Template>*

…

### Whitebox *<Baustein m>*

*<Whitebox-Template>*

## Ebene 3

Beschreiben Sie den inneren Aufbau (einiger) Bausteine aus Ebene 2 als Whitebox.

Bei tieferen Gliederungen der Architektur kopieren Sie diesen Teil von arc42 für die weiteren Ebenen.

### Whitebox <\_Baustein x.1\_>

…zeigt das Innenleben von *Baustein x.1*.

*<Whitebox-Template>*

### Whitebox <\_Baustein x.2\_>

*<Whitebox-Template>*

### Whitebox <\_Baustein y.1\_>

*<Whitebox-Template>*

# Laufzeitsicht

**Inhalt**

Diese Sicht erklärt konkrete Abläufe und Beziehungen zwischen Bausteinen in Form von Szenarien aus den folgenden Bereichen:

* Wichtige Abläufe oder *Features*: Wie führen die Bausteine der Architektur die wichtigsten Abläufe durch?
* Interaktionen an kritischen externen Schnittstellen: Wie arbeiten Bausteine mit Nutzern und Nachbarsystemen zusammen?
* Betrieb und Administration: Inbetriebnahme, Start, Stop.
* Fehler- und Ausnahmeszenarien

Anmerkung: Das Kriterium für die Auswahl der möglichen Szenarien (d.h. Abläufe) des Systems ist deren Architekturrelevanz. Es geht nicht darum, möglichst viele Abläufe darzustellen, sondern eine angemessene Auswahl zu dokumentieren.

**Motivation**

Sie sollten verstehen, wie (Instanzen von) Bausteine(n) Ihres Systems ihre jeweiligen Aufgaben erfüllen und zur Laufzeit miteinander kommunizieren.

Nutzen Sie diese Szenarien in der Dokumentation hauptsächlich für eine verständlichere Kommunikation mit denjenigen Stakeholdern, die die statischen Modelle (z.B. Bausteinsicht, Verteilungssicht) weniger verständlich finden.

**Form**

Für die Beschreibung von Szenarien gibt es zahlreiche Ausdrucksmöglichkeiten. Nutzen Sie beispielsweise:

* Nummerierte Schrittfolgen oder Aufzählungen in Umgangssprache
* Aktivitäts- oder Flussdiagramme
* Sequenzdiagramme
* BPMN (Geschäftsprozessmodell und -notation) oder EPKs (Ereignis-Prozessketten)
* Zustandsautomaten
* …

Siehe [Laufzeitsicht](https://docs.arc42.org/section-6/) in der online-Dokumentation (auf Englisch!).

## *<Bezeichnung Laufzeitszenario 1>*

* <hier Laufzeitdiagramm oder Ablaufbeschreibung einfügen>
* <hier Besonderheiten bei dem Zusammenspiel der Bausteine in diesem Szenario erläutern>

## *<Bezeichnung Laufzeitszenario 2>*

…

## *<Bezeichnung Laufzeitszenario n>*

…

# Verteilungssicht

**Inhalt**

Die Verteilungssicht beschreibt:

1. die technische Infrastruktur, auf der Ihr System ausgeführt wird, mit Infrastrukturelementen wie Standorten, Umgebungen, Rechnern, Prozessoren, Kanälen und Netztopologien sowie sonstigen Bestandteilen, und
2. die Abbildung von (Software-)Bausteinen auf diese Infrastruktur.

Häufig laufen Systeme in unterschiedlichen Umgebungen, beispielsweise Entwicklung-/Test- oder Produktionsumgebungen. In solchen Fällen sollten Sie alle relevanten Umgebungen aufzeigen.

Nutzen Sie die Verteilungssicht insbesondere dann, wenn Ihre Software auf mehr als einem Rechner, Prozessor, Server oder Container abläuft oder Sie Ihre Hardware sogar selbst konstruieren.

Aus Softwaresicht genügt es, auf die Aspekte zu achten, die für die Softwareverteilung relevant sind. Insbesondere bei der Hardwareentwicklung kann es notwendig sein, die Infrastruktur mit beliebigen Details zu beschreiben.

**Motivation**

Software läuft nicht ohne Infrastruktur. Diese zugrundeliegende Infrastruktur beeinflusst Ihr System und/oder querschnittliche Lösungskonzepte, daher müssen Sie diese Infrastruktur kennen.

**Form**

Das oberste Verteilungsdiagramm könnte bereits in Ihrem technischen Kontext enthalten sein, mit Ihrer Infrastruktur als EINE Blackbox. Jetzt zoomen Sie in diese Infrastruktur mit weiteren Verteilungsdiagrammen hinein:

* Die UML stellt mit Verteilungsdiagrammen (Deployment diagrams) eine Diagrammart zur Verfügung, um diese Sicht auszudrücken. Nutzen Sie diese, evtl. auch geschachtelt, wenn Ihre Verteilungsstruktur es verlangt.
* Falls Ihre Infrastruktur-Stakeholder andere Diagrammarten bevorzugen, die beispielsweise Prozessoren und Kanäle zeigen, sind diese hier ebenfalls einsetzbar.

Siehe [Verteilungssicht](https://docs.arc42.org/section-7/) in der online-Dokumentation (auf Englisch!).

## Infrastruktur Ebene 1

An dieser Stelle beschreiben Sie (als Kombination von Diagrammen mit Tabellen oder Texten):

* die Verteilung des Gesamtsystems auf mehrere Standorte, Umgebungen, Rechner, Prozessoren o. Ä., sowie die physischen Verbindungskanäle zwischen diesen,
* wichtige Begründungen für diese Verteilungsstruktur,
* Qualitäts- und/oder Leistungsmerkmale dieser Infrastruktur,
* Zuordnung von Softwareartefakten zu Bestandteilen der Infrastruktur

Für mehrere Umgebungen oder alternative Deployments kopieren Sie diesen Teil von arc42 für alle wichtigen Umgebungen/Varianten.

***<Übersichtsdiagramm>***

Begründung

*<Erläuternder Text>*

Qualitäts- und/oder Leistungsmerkmale

*<Erläuternder Text>*

Zuordnung von Bausteinen zu Infrastruktur

*<Beschreibung der Zuordnung>*

## Infrastruktur Ebene 2

An dieser Stelle können Sie den inneren Aufbau (einiger) Infrastrukturelemente aus Ebene 1 beschreiben.

Für jedes Infrastrukturelement kopieren Sie die Struktur aus Ebene 1.

### *<Infrastrukturelement 1>*

*<Diagramm + Erläuterungen>*

### *<Infrastrukturelement 2>*

*<Diagramm + Erläuterungen>*

…

### *<Infrastrukturelement n>*

*<Diagramm + Erläuterungen>*

# Querschnittliche Konzepte

**Inhalt**

Dieser Abschnitt beschreibt übergreifende, prinzipielle Regelungen und Lösungsansätze, die an mehreren Stellen (=*querschnittlich*) relevant sind.

Solche Konzepte betreffen oft mehrere Bausteine. Dazu können vielerlei Themen gehören, beispielsweise:

* Modelle, insbesondere fachliche Modelle
* Architektur- oder Entwurfsmuster
* Regeln für den konkreten Einsatz von Technologien
* prinzipielle — meist technische — Festlegungen übergreifender Art
* Implementierungsregeln

**Motivation**

Konzepte bilden die Grundlage für *konzeptionelle Integrität* (Konsistenz, Homogenität) der Architektur und damit eine wesentliche Grundlage für die innere Qualität Ihrer Systeme.

Manche dieser Themen lassen sich nur schwer als Baustein in der Architektur unterbringen (z.B. das Thema „Sicherheit“).

**Form**

Kann vielfältig sein:

* Konzeptpapiere mit beliebiger Gliederung,
* übergreifende Modelle/Szenarien mit Notationen, die Sie auch in den Architektursichten nutzen,
* beispielhafte Implementierung speziell für technische Konzepte,
* Verweise auf „übliche“ Nutzung von Standard-Frameworks (beispielsweise die Nutzung von Hibernate als Object/Relational Mapper).

**Struktur**

Eine mögliche (nicht aber notwendige!) Untergliederung dieses Abschnittes könnte wie folgt aussehen (wobei die Zuordnung von Themen zu den Gruppen nicht immer eindeutig ist):

* Fachliche Konzepte
* User Experience (UX)
* Sicherheitskonzepte (Safety und Security)
* Architektur- und Entwurfsmuster
* Unter-der-Haube
* Entwicklungskonzepte
* Betriebskonzepte



Siehe [Querschnittliche Konzepte](https://docs.arc42.org/section-8/) in der online-Dokumentation (auf Englisch).

## *<Konzept 1>*

*<Erklärung>*

## *<Konzept 2>*

*<Erklärung>*

…

## *<Konzept n>*

*<Erklärung>*

# Architekturentscheidungen

**Inhalt**

Wichtige, teure, große oder riskante Architektur- oder Entwurfsentscheidungen inklusive der jeweiligen Begründungen. Mit "Entscheidungen" meinen wir hier die Auswahl einer von mehreren Alternativen unter vorgegebenen Kriterien.

Wägen Sie ab, inwiefern Sie Entscheidungen hier zentral beschreiben, oder wo eine lokale Beschreibung (z.B. in der Whitebox-Sicht von Bausteinen) sinnvoller ist. Vermeiden Sie Redundanz. Verweisen Sie evtl. auf Abschnitt 4, wo schon grundlegende strategische Entscheidungen beschrieben wurden.

**Motivation**

Stakeholder des Systems sollten wichtige Entscheidungen verstehen und nachvollziehen können.

**Form**

Verschiedene Möglichkeiten:

* ADR ([Documenting Architecture Decisions](https://cognitect.com/blog/2011/11/15/documenting-architecture-decisions)) für jede wichtige Entscheidung
* Liste oder Tabelle, nach Wichtigkeit und Tragweite der Entscheidungen geordnet
* ausführlicher in Form einzelner Unterkapitel je Entscheidung

Siehe [Architekturentscheidungen](https://docs.arc42.org/section-9/) in der arc42 Dokumentation (auf Englisch!). Dort finden Sie Links und Beispiele zum Thema ADR.

# Qualitätsanforderungen

**Inhalt**

Dieser Abschnitt enthält möglichst alle Qualitätsanforderungen als Qualitätsbaum mit Szenarien. Die wichtigsten davon haben Sie bereits in Abschnitt 1.2 (Qualitätsziele) hervorgehoben.

Nehmen Sie hier auch Qualitätsanforderungen geringerer Priorität auf, deren Nichteinhaltung oder -erreichung geringe Risiken birgt.

**Motivation**

Weil Qualitätsanforderungen die Architekturentscheidungen oft maßgeblich beeinflussen, sollten Sie die für Ihre Stakeholder relevanten Qualitätsanforderungen kennen, möglichst konkret und operationalisiert.

**Weiterführende Informationen**

Siehe [Qualitätsanforderungen](https://docs.arc42.org/section-10/) in der online-Dokumentation (auf Englisch!).

## Qualitätsbaum

**Inhalt**

Der Qualitätsbaum (à la ATAM) mit Qualitätsszenarien an den Blättern.

**Motivation**

Die mit Prioritäten versehene Baumstruktur gibt Überblick über die — oftmals zahlreichen — Qualitätsanforderungen.

* Baumartige Verfeinerung des Begriffes „Qualität“, mit „Qualität“ oder „Nützlichkeit“ als Wurzel.
* Mindmap mit Qualitätsoberbegriffen als Hauptzweige

In jedem Fall sollten Sie hier Verweise auf die Qualitätsszenarien des folgenden Abschnittes aufnehmen.

## Qualitätsszenarien

**Inhalt**

Konkretisierung der (in der Praxis oftmals vagen oder impliziten) Qualitätsanforderungen durch (Qualitäts-)Szenarien.

Diese Szenarien beschreiben, was beim Eintreffen eines Stimulus auf ein System in bestimmten Situationen geschieht.

Wesentlich sind zwei Arten von Szenarien:

* Nutzungsszenarien (auch bekannt als Anwendungs- oder Anwendungsfallszenarien) beschreiben, wie das System zur Laufzeit auf einen bestimmten Auslöser reagieren soll. Hierunter fallen auch Szenarien zur Beschreibung von Effizienz oder Performance. Beispiel: Das System beantwortet eine Benutzeranfrage innerhalb einer Sekunde.
* Änderungsszenarien beschreiben eine Modifikation des Systems oder seiner unmittelbaren Umgebung. Beispiel: Eine zusätzliche Funktionalität wird implementiert oder die Anforderung an ein Qualitätsmerkmal ändert sich.

**Motivation**

Szenarien operationalisieren Qualitätsanforderungen und machen deren Erfüllung mess- oder entscheidbar.

Insbesondere wenn Sie die Qualität Ihrer Architektur mit Methoden wie ATAM überprüfen wollen, bedürfen die in Abschnitt 1.2 genannten Qualitätsziele einer weiteren Präzisierung bis auf die Ebene von diskutierbaren und nachprüfbaren Szenarien.

**Form**

Entweder tabellarisch oder als Freitext.

# Risiken und technische Schulden

**Inhalt**

Eine nach Prioritäten geordnete Liste der erkannten Architekturrisiken und/oder technischen Schulden.

Risikomanagement ist Projektmanagement für Erwachsene.

— Tim Lister Atlantic Systems Guild

Unter diesem Motto sollten Sie Architekturrisiken und/oder technische Schulden gezielt ermitteln, bewerten und Ihren Management-Stakeholdern (z.B. Projektleitung, Product-Owner) transparent machen.

**Form**

Liste oder Tabelle von Risiken und/oder technischen Schulden, eventuell mit vorgeschlagenen Maßnahmen zur Risikovermeidung, Risikominimierung oder dem Abbau der technischen Schulden.

Siehe [Risiken und technische Schulden](https://docs.arc42.org/section-11/) in der online-Dokumentation (auf Englisch!).

# Glossar

**Inhalt**

Die wesentlichen fachlichen und technischen Begriffe, die Stakeholder im Zusammenhang mit dem System verwenden.

Nutzen Sie das Glossar ebenfalls als Übersetzungsreferenz, falls Sie in mehrsprachigen Teams arbeiten.

**Motivation**

Sie sollten relevante Begriffe klar definieren, so dass alle Beteiligten

* diese Begriffe identisch verstehen, und
* vermeiden, mehrere Begriffe für die gleiche Sache zu haben.
* Zweispaltige Tabelle mit <Begriff> und <Definition>
* Eventuell weitere Spalten mit Übersetzungen, falls notwendig.

Siehe [Glossar](https://docs.arc42.org/section-12/) in der online-Dokumentation (auf Englisch!).

| Begriff | Definition |
| --- | --- |
| *<Begriff-1>* | *<Definition-1>* |
| *<Begriff-2* | *<Definition-2>* |