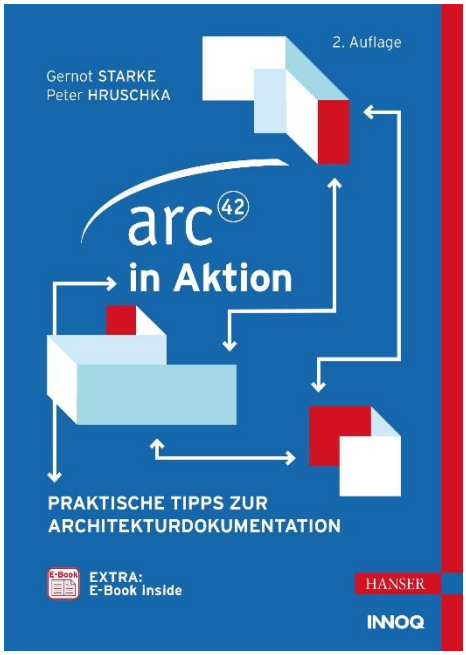
HANSER



Leseprobe

zu

arc42 in Aktion

von Gernot Starke und Peter Hruschka

Print-ISBN: 978-3-446-46380-6
E-Book-ISBN: 978-3-446-46555-8

Weitere Informationen und Bestellungen unter
https://www.hanser-kundencenter.de/fachbuch/artikel/9783446463806
sowie im Buchhandel

Inhalt

I Überblick

May the force of the proper word and diagram be with you.

Dieses Kapitel klärt folgende Themen:

- Grundprinzipien von arc42
- Warum dieses Buch?
- Wozu können Sie arc42 verwenden?
- Was dieses Buch *nicht* ist!
- Für wen haben wir dieses Buch geschrieben?
- Navigationshilfe für Eilige

Softwaresysteme bleiben oftmals viele Jahre im Einsatz – und unterliegen währenddessen kontinuierlicher Weiterentwicklung. Dieses „Leben" von Software endet leider manchmal tragisch: Mangelnde Wartbarkeit und unzureichende Dokumentation machen selbst kleine Änderungen zu einem riskanten Vabanquespiel mit ungewissem Ausgang. Oftmals lassen sich selbst marginale Erweiterungen nur mit massivem Aufwand bewältigen, der wirtschaftliche Nutzen des Systems schwindet dahin.

Sie als Softwarearchitekt:in haben es in der Hand, dieses „Verfaulen" von Software gründlich zu verhindern und Ihre Systeme langfristig wartbar, flexibel und verständlich zu konstruieren. Dafür müssen Sie, neben der selbstverständlichen Erfüllung der Anforderungen, langfristig auf die *innere Qualität* Ihrer Systeme achten und deren Architekturen *angemessen* (schriftlich und mündlich) kommunizieren.

Wir stellen Ihnen in diesem Buch arc42 vor, den bewährten, praxisnahen und frei verfügbaren Standard zur Dokumentation und Kommunikation von Softwarearchitekturen. arc42 basiert auf langjähriger Erfahrung und wird seit 2005 von vielen Unternehmen und Organisationen ganz unterschiedlicher Branchen erfolgreich eingesetzt.

arc42 ist in erster Linie ein Template zur Architekturdokumentation. Es beantwortet die beiden folgenden Fragen auf einerseits pragmatische, andererseits auch an spezielle Bedürfnisse anpassbare Weise:

- Was sollen wir über unsere Architektur aufschreiben?
- Wie sollen wir dokumentieren?

Bild I.1 gibt einen vereinfachten Überblick über die Struktur von arc42, zeigt Ihnen das *big picture*.

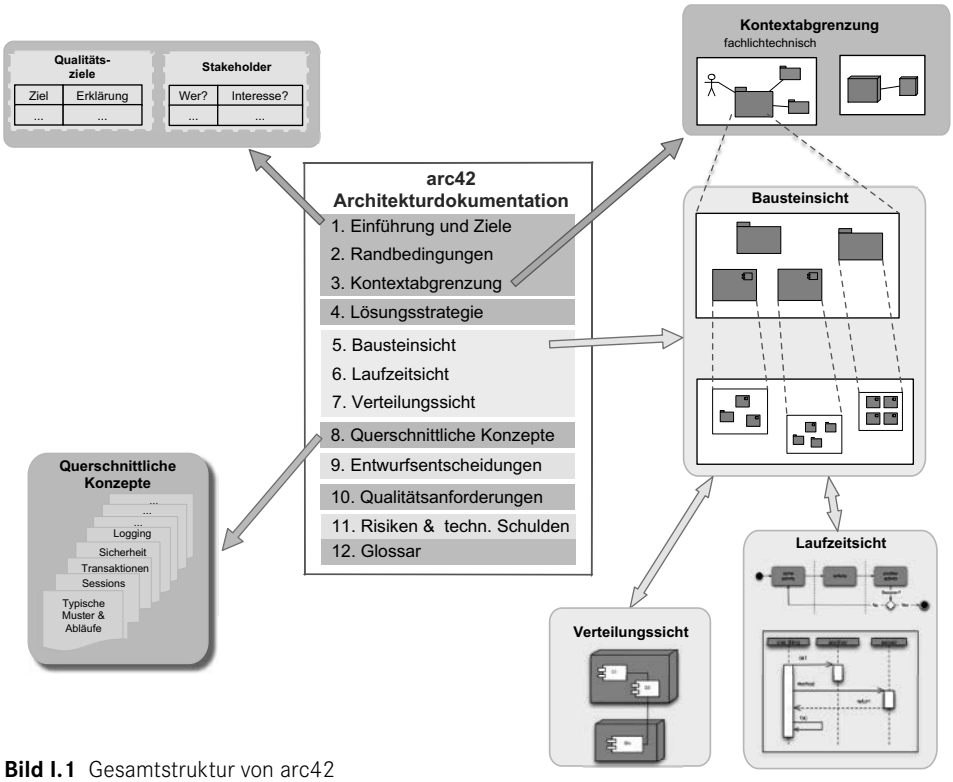


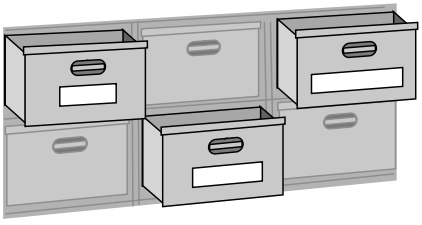
Bild I.1 Gesamtstruktur von arc42

Wenn Sie ungeduldig sind und arc42 anhand eines kleinen Beispiels *live* sehen möchten, blättern Sie kurz zu Kapitel II (das Ihnen zeigt, wie sich arc42-Dokumentation *anfühlt*). Das umfangreiche Beispiel in Kapitel VIII rundet das Buch ab.

Ansonsten möchten wir Ihnen kurz einige Hintergründe erklären, die Ihnen die Arbeit mit arc42 erleichtern.

I.1 Grundprinzipien von arc42

Klare Struktur



Vergleichen Sie arc42 mit einem Schubladenschrank: Die Schubladen sind ordentlich beschriftet und enthalten zusammengehörige Informationen. arc42 enthält zwölf solche Fächer (also ein paar mehr als auf dem Bild zu sehen). Die Bedeutung dieser arc42-Fächer ist leicht verständlich.

arc42 gibt Ihnen damit eine einfache, klare Struktur zur Beschreibung Ihrer (komplexen!) Systeme. Beginnend bei den Zielen und Anforderungen an Ihr System und der Einbettung in die fachliche und technische Umgebung können Sie nahezu alle Beteiligten oder Interessenten Ihres Systems mit passenden Informationen zur Architektur versorgen.

arc42 ist auf Verständlichkeit und Stakeholder-Eignung optimiert: Es leitet Sie auf ganz natürliche Weise an, jede Art von Architekturinformation und -entscheidung in einem verständlichen und nachvollziehbaren Kontext zu erklären.

Anwender:innen von arc42 loben die hohe Verständlichkeit der Ergebnisse, die aus diesem etablierten Aufbau resultiert, und den überschaubaren Aufwand, eine solche Dokumentation zu erstellen. „Schmerzfreie Dokumentation" – wie wir das nennen.

Unabhängig von Entwicklungsvorgehen

Sie können Ihren arc42-„Schrank" in jeder beliebigen Reihenfolge bearbeiten – ganz wie es Ihre konkrete Situation erfordert. Bei einer Neuentwicklung[1] werden Sie wahrscheinlich mit den Teilen rund um Anforderungen und Ziele beginnen, bei der Arbeit an bestehenden Systemen tauchen Sie möglicherweise sofort in die Tiefen der Bausteinsicht ab.

Sie können die Arbeit an arc42-basierter Architekturdokumentation jederzeit unterbrechen oder wieder aufnehmen – die feste Struktur des „Schranks" stellt eine problemlose Weiterarbeit sicher. Voraussetzung dafür ist natürlich, dass die beteiligten Mitarbeiter etwas Verständnis für die arc42-„Schrankfächer" besitzen.

Expert:innen nennen diese positive Eigenschaft von arc42 übrigens „prozessagnostisch".

Architekturdokumentation mit wenig Aufwand

arc42 steht unter einer liberalen Open-Source-Lizenz, daher ist die Nutzung von arc42 in jedem Fall kostenfrei.

Bei der Arbeit mit dem arc42-Template fällt kein zusätzlicher Aufwand für Sie oder Ihre Teams an: Sie

- beschreiben Sachverhalte, die Stakeholder des Systems kennen müssen.
- erklären Zusammenhänge, die Voraussetzungen für das Verständnis des Systems oder einzelner Entwurfsentscheidungen sind.
- heben nur wichtige Entscheidungen auf, die Sie ohnehin treffen müssen.

arc42 hilft Ihnen, für jede dieser Entscheidungen und Sachverhalte eine passende Schublade zu finden, in der alle Beteiligten diese Informationen leicht wiederfinden können.

Risiko: Formular-Zombies[2] …

Ein Risiko im Umgang mit dem arc42-Template möchten wir offenlegen: Menschen *könnten* arc42 mit einem Formular verwechseln und daraufhin versuchen, alle Teile des Templates „auszufüllen" ☺. Wir mögen den Begriff *ausfüllen* überhaupt nicht, genauso wenig wie das

[1] Kapitel V und VI unterbreiten Vorschläge, wie Sie arc42 in Ihrem Arbeitsalltag in verschiedenen Situationen einsetzen können.

[2] Der Begriff stammt aus dem lesenswerten Buch „Adrenalin-Junkies und Formular-Zombies" von Tom DeMarco, Peter Hruschka et al. (Hanser, 2022).

Formular: arc42 ist als leichtgewichtiges, an Ihre konkrete Situation adaptierbares, angemessenes Hilfsmittel gedacht, in keinem Fall als „Alles-ausfüllen-Formular“.

Unsere tiefe Abneigung gegen *Formulare* und Angst vor dem möglichen Missbrauch von arc42 hat uns bewogen, in Kapitel III einige entsprechende Grundregeln aufzustellen: Beachten Sie insbesondere Regel 2 (Sparsamkeit) und Regel 3 (Angemessenheit).

I.2 Warum dieses Buch?

Seit 2005 verwenden Unternehmen arc42, um Software- und Systemarchitekturen zu dokumentieren. Seither durften wir (die Autoren) dabei helfen, arc42 einzuführen, bestehende Dokumentation im Sinne von arc42 aufzubereiten und neue Systeme mit Hilfe von arc42 zu entwickeln und zu dokumentieren.

Das arc42-Template selbst enthält kurze Hinweise und Ratschläge zu den einzelnen Teilen („Schubladen“). Die Struktur ist übersichtlich und verständlich, es gibt keine Hürden bezüglich der Anwendung von arc42.

So einfach und klar strukturiert arc42 jedoch auch ist – im Projektalltag stellen sich immer wieder Fragen, die wir in diesem Buch im Sinne eines *missing manual*[3] beantworten.

[3] Die „the missing manuals®“ ist eine Buchreihe des O'Reilly-Verlags mit dem schönen Untertitel „the book that should have been in the box“. Siehe *https://www.oreilly.com/missingmanuals/*.

I.3 Was dieses Buch *nicht* ist

Wir fokussieren in diesem Buch auf die effektive und effiziente Nutzung von arc42 zur Kommunikation und Dokumentation von Softwarearchitekturen. Dazu grenzen wir es von einigen anderen Themen oder Disziplinen ausdrücklich ab.

Dieses Buch ist **keine** Einführung in:

- Softwarearchitektur und -entwurf: Wir setzen einige Kenntnisse in methodischem Entwurf und Entwicklung voraus. Sie können Problemraum (Anforderungen) und Lösungsraum (Architektur, Implementierung) differenzieren. Sie wissen um den Wert von Prinzipien wie *Separation-of-Concern*, das Geheimnisprinzip (*Information Hiding*), lose Kopplung, hohe Kohäsion, Einfachheit, hohe Konsistenz sowie die Trennung fachlicher und technischer Bausteine in der Architektur. Sie kennen Begriffe wie Architektursichten und querschnittliche Konzepte. Falls Sie mehr über Softwarearchitektur lesen möchten: [Starke-19] hilft weiter.
- Technologie und Frameworks: Sie werden für Entwurf und Implementierung Ihres Systems konkrete Implementierungstechnologien und Frameworks benötigen, und diese sicherlich auch in Ihrer Dokumentation referenzieren. Wir setzen voraus, dass Sie die für Ihr System notwendigen Technologien kennen.
- Patterns aller Art: Ob Analyse, Entwurf oder Architektur: Patterns sammeln etablierte Lösungsansätze und diskutieren deren Vor- und Nachteile, Konsequenzen und Risiken. Wir selbst bedienen uns sehr gerne der umfangreichen Pattern-Literatur und verweisen in unseren eigenen Dokumentationen häufig darauf.
- Modellierung: Sie sollten Modelle als Abstraktion statischer und dynamischer Sachverhalte anwenden können. Wir setzen voraus, dass Sie den Zusammenhang statischer Modelle (Bausteine oder Komponenten und deren Beziehungen) mit dem zugehörigen Quellcode ebenso kennen wie Grundbegriffe dynamischer Modelle (Ablauf- oder Prozessmodelle).
- UML: Zwar erklären wir an vielen Stellen, wie Sie UML pragmatisch und effektiv einsetzen können, wir setzen aber Grundkenntnisse von Klassen-, Komponenten-, Sequenz-, Aktivitäts- und Verteilungsdiagrammen voraus. Weitere Informationen finden Sie beispielsweise in [Kecher+15] oder [Rupp+12].
- Anforderungs- und Business-Analyse: Softwarearchitekt:innen müssen Anforderungen an Systeme verstehen und bei Bedarf klären. Dazu müssen sie Anforderungen erheben, beschreiben und managen – was in idealen Situationen Requirements Engineers erledigen. Wir setzen voraus, dass Sie sowohl funktionale wie auch Qualitätsanforderungen kennen und mit Ihren Stakeholdern diskutieren können. Gute Quellen für weitere Informationen sind [Hruschka-19], [Rupp+12] oder [Robertson-12].

Obwohl sich unsere Tipps auf den Einsatz von arc42 beziehen, helfen Ihnen viele dieser Ratschläge auch dabei, gemeinsam mit Ihren Teams bessere Systeme zu entwerfen und zu implementieren ☺.

I.4 Unsere Annahmen über Sie ...

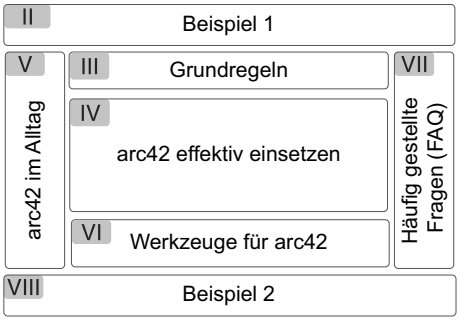
Als Autoren haben wir einige (möglicherweise törichte) Annahmen über Sie als Leser:in getroffen:

- Sie haben aktuell wenig Zeit und möchten nur die relevanten Teile dieses Buchs lesen ... (dafür haben wir die Navigationshilfe I.5 unten geschrieben!).
- Sie besitzen praktische Erfahrung in Softwareentwicklung und -architektur und kennen daher die Notwendigkeit angemessener Dokumentation.
- Sie haben mit der Entwicklung oder Pflege mittlerer bis großer, oftmals komplexer Softwaresysteme zu tun (dem *System*).
- Das eine oder andere Mal in Ihrem Berufsleben haben Sie unter schlechter, fehlender oder übertriebener Dokumentation gelitten.
- Sie möchten Informationen über Architektur, Aufbau und Implementierung des Systems kommunizieren oder dokumentieren.
- Dabei möchten Sie *angemessen* vorgehen: Für manche Systeme benötigen Sie eine gründliche und detaillierte Dokumentation, bei anderen genügt es, einige zentrale Themen kurz zu beschreiben.
- Vielleicht kennen Sie arc42 bereits und Sie möchten wissen, wie Sie das Template einfacher, pragmatischer oder effektiver einsetzen können.

I.5 Navigationshilfe für Eilige

Weil Sie viel zu tun haben, möchten Sie möglichst schnell die für Ihre konkreten Probleme relevanten Teile dieses Buchs identifizieren.

Folgende Abbildung zeigt Ihnen den weiteren Aufbau dieses Buchs. Wir haben die Nummern der Hauptkapitel römisch (d. h. I, II, III) vergeben, damit Sie die Buchkapitel leicht von den arc42-Abschnitten (1, 2, ..., 12) unterscheiden können ...



Kapitel II beschreibt ein kleines Open-Source-System anhand von arc42. Sie sehen exemplarisch, „wie es aussehen könnte“, und finden jeweils eine kurze Motivation der jeweiligen arc42-Abschnitte. Sie können dieses Kapitel unabhängig vom restlichen Buch lesen.

Kapitel III erklärt einige Grundregeln angemessener Architekturdokumentation, insbesondere unsere Aufforderung zu systematischer Sparsamkeit.

Kapitel IV gibt Ihnen praktische Tipps zu jedem Abschnitt von arc42 (also zu jeder Schublade, um die Metapher vom Anfang dieses Kapitels aufzugreifen). Auch hier motivieren wir kurz die einzelnen arc42-Abschnitte, weshalb es hier kleine und bewusste Redundanzen zu Kapitel II gibt. Dies ist der umfangreichste Teil dieses Buchs.

Kapitel V zeigt Ihnen, wie Sie arc42 im Alltag einsetzen können. Sie finden Hinweise für neue bzw. bestehende Systeme, für agile Projekte sowie Besonderheiten ganz großer Systeme.

Kapitel VI stellt Werkzeuge und Werkzeugkategorien vor, mit deren Hilfe Sie arc42 „zum Leben erwecken“ können.

Kapitel VII beantwortet häufig gestellte Fragen in diversen Kategorien (und verweist bei vielen Fragen auf Tipps aus den Kapiteln III bis VI).

Kapitel VIII zeigt ein umfangreiches Beispiel aus der Embedded-Welt – es geht um die automatisierte Unterstützung der Polizei bei der videobasierten Geschwindigkeitskontrolle.

I.6 Konventionen

	**Wichtig:** Trotz Sparsamkeit und Agilität gibt es einige Informationen über Ihre Systeme, die Sie in jedem Fall dokumentieren sollten, beispielsweise Ihre Qualitätsanforderungen.
	**Sparsam:** Sie suchen nach Möglichkeiten, Ihre Dokumentation zu vereinfachen oder pragmatisch abzukürzen. Sie möchten Ihre Dokumentation vereinfachen oder abkürzen. Sie möchten dabei Aufwand sparen, ohne Inhalt oder Wert zu verlieren. Sie arbeiten in einem agilen Umfeld und möchten schlanke, leichtgewichtige Dokumentation – getreu dem Motto travel light.
	**Gründlich:** Sie arbeiten in eher formalem Umfeld an größeren oder kritischen Systemen mit hohen Qualitätsanforderungen. Ihre Stakeholder legen Wert auf Gründlichkeit, Genauigkeit, Detailtreue oder Ausführlichkeit. Eventuell werden Ihre Systeme und deren Dokumentation auditiert.

Tipp I-1

In den Kapiteln III bis VII geben wir über 200 verschiedene Tipps rund um arc42. Diese Tipps sind über die römische Ziffer jeweils den Buchkapiteln zugeordnet.

I.7 Danke

Viele Personen haben uns in den letzten Jahren mit konstruktiver Kritik, Anregungen und Fragen rund um arc42 geholfen.

Ralf D. Müller ist die gute Seele und der unermüdliche Committer im arc42-Open-Source-Universum: Er beantwortet Supportanfragen, pflegt die Werkzeugkette, die arc42 aus AsciiDoc generiert, und hält unsere Github-Issues im Zaum. Danke Dir!

Danke an die Menschen, die arc42 übersetzt oder in andere Formate portiert haben – insbesondere Manfred Ferken, Peter Goetz, Oliver Lietz, Daniel Pozzi, Eduaro Rodriguez, Markus Schärtel, Boris Stumm, Fabian Wüthrich.

Wir möchten uns darüber hinaus bei Martin Dungs, Uwe Friedrichsen, Phillip Ghadir, Mahbouba Gharbi, Franz Hofer, Prof. Arne Koschel, Jürgen Krey, Anton Kronseder, Prof. Bernd Müller, Alex Nachtigall, Axel Noellchen, Robert Reiner, Roland Schimmack, Michael Simons, Boris Stumm, Daniel Takai, Eberhard Wolff, Oliver Wronka und Stefan Zörner für ihr Engagement in der Vorbereitung zu diesem Buch bedanken.

Danke an Pedro Lafuente Blanco, Stefan Paal, Christopher Schmidt, Silvia Schreier, Per Starke und Oliver Tigges für eure konstruktiven Reviews und Anregungen.

Ohne die tatkräftige Unterstützung durch Brigitte Bauer-Schiewek und Irene Weilhart vom Hanser-Verlag wären wir an der Textverarbeitung verzweifelt.

innoQ unterstützt und betreibt seit langer Zeit unser arc42-Confluence sowie die arc42.org-Website. Christian Sarazin räumt dabei die technischen Stolpersteine aus dem Weg.

Gernot: Uli, Lynn und Per: Ihr seid super, die beste Familie im Universum! Zeit mit Euch ist immer zu kurz. Danke auch an meine kundigen, kompetenten und kritischen Kollegen der innoQ.

Peter: Mein besonderer Dank gilt meiner Traumfrau Monika, die ein weiteres Buchprojekt durch ihre Kommentare aus einer Nicht-IT-Sicht bereichert hat.

IV arc42 effektiv einsetzen

In Kapitel II haben Sie das arc42-Template anhand eines Beispiels kennengelernt und pro arc42-Abschnitt erfahren, warum es existiert und was Sie darin festhalten können. In diesem Teil gehen wir tiefer.

Dieses Kapitel enthält für alle Abschnitte des arc42-Templates:

- unterschiedliche Darstellungsweisen,
- Beispiele und
- sparsamere und ausführlichere Varianten.

Vor allem aber finden Sie hier zahlreiche Praxistipps aus unserer eigenen Erfahrung und aus den vielseitigen Rückmeldungen, die uns arc42-Nutzer in den letzten Jahren gegeben haben.

Achtung: Es gibt mit arc42 oft mehrere Wege zum Ziel!

Im Folgenden finden Sie an einigen Stellen scheinbar widersprüchliche Ratschläge. So empfehlen wir bei der Aufgabenstellung (IV.1.1) einerseits die Nutzung von Aktivitätsdiagrammen, andererseits die Verwendung nummerierter Listen. Was wie ein Widerspruch klingt, ist als Aufzeigen verschiedener Alternativen gedacht. Für deren konkrete Auswahl bieten wir Ihnen teilweise explizite Kriterien an. An anderen Stellen bleibt die Wahl der Mittel auch Geschmackssache. Im Zweifel hilft iteratives Vorgehen: Beschaffen Sie sich Feedback zu einer (groben oder vorläufigen) Version eines arc42-Abschnitts – und passen Sie aufgrund dieser Rückmeldung Notation oder Detaillierung an (*siehe Tipp III-3*).

Mehr Dokumentation und Tipps zum Einsatz von arc42 finden Sie online auf *https://docs.arc42.org*.

Bevor wir in Details einsteigen, möchten wir nochmals an die grundlegenden Tipps zur Dokumentation aus Kapitel III erinnern: eine verantwortliche Person, methodische Sparsamkeit, frühzeitiges Feedback, top-down strukturieren, Begründungen geben, Anforderungen vor Prinzipien und die Trennung volatiler von bleibender Dokumentation!

1 Einführung und Ziele

Fassen Sie hier die wesentlichen Anforderungen und treibenden Kräfte zusammen, die Softwarearchitekten und Entwicklungsteams berücksichtigen müssen. Dazu gehören die

- zugrunde liegenden Geschäftsziele,
- wesentliche Aufgabenstellung bzw. fachliche Anforderungen des Systems,
- Qualitätsziele für die Architektur und
- relevante Stakeholder und deren Erwartungshaltung.

1.1 Aufgabenstellung

Inhalt

Kurzbeschreibung der fachlichen Aufgabenstellung, treibenden Kräfte, Extrakt (oder Abstract) der Anforderungen. Verweis auf (hoffentlich vorliegende) Anforderungsdokumente (mit Versionsbezeichnungen und Ablageorten).

Motivation

Aus Sicht der späteren Nutzer ist die Unterstützung einer fachlichen Aufgabe oder Verbesserung der Qualität der eigentliche Beweggrund, ein neues System zu schaffen oder ein bestehendes zu modifizieren.

Tipp IV-1: Erklären Sie möglichst kurz, worum es bei dem System geht

Dieser Abschnitt ist für manche Personen das Erste, was sie über das System lernen. Drücken Sie hier klar und kompakt die Business- oder Projektziele aus. Geben Sie einen kurzen Überblick, welches Problem das System löst.

Anmerkung: Manchmal betrifft ein Requirements-Dokument mehr als nur das eine System, das wir hier in der Architekturdokumentation im Fokus haben. Sollten Projektscope und Systemscope unterschiedlich sein, so konzentrieren Sie sich im jeweiligen Architekturdokument nur auf die Teile der Anforderungen, die dieses System betreffen.

Beschränken Sie sich auf das wirklich Wesentliche, die Essenz des Systems. Unsere Faustregel: möglichst weniger als eine Seite. Die „knappe Seite“ darf ein Diagramm enthalten, wenn das die inhaltliche Aussage unterstützt. Verweisen Sie auf Anforderungsdokumentation, sofern so etwas vorhanden ist.

Anmerkungen: In manchen Fällen müssen Sie wahrscheinlich diesen Rahmen sprengen:

- bei Systemen mit komplexen oder vielseitigen fachlichen Anforderungen,
- bei Systemen ohne bestehende (vernünftige) Anforderungsdokumentation.

Tipp IV-2: Beschränken Sie sich auf die wesentlichen Aufgaben/Anwendungsfälle

Skizzieren Sie hier die *wesentlichen* Anwendungsfälle, Abläufe, Prozesse oder User-Stories (wie immer Sie das in Ihrer Organisation nennen). Beschränken Sie sich auf einen Abstraktionsgrad, der auch Außenstehenden in kurzer Zeit einen Überblick über die Aufgabenstellung des Systems ermöglicht.

Wiederholen Sie so wenig wie möglich. Verweisen Sie, falls möglich. Fassen Sie ähnliche oder zusammengehörige Anforderungen zusammen (siehe nachfolgenden Tipp).

Nur wenn Sie gar keine expliziten Anforderungen haben, darf dieser Abschnitt länger sein ☺.

Tipp IV-3: Zeigen Sie die geschäftlichen Ziele des Systems auf

Sie sollten sicherstellen, dass die geschäftlichen Ziele explizit bekannt sind. Eigentlich sollte das schon im Projektauftrag stehen oder in einem Lasten- oder Pflichtenheft[1] ...

Geschäftliche Ziele sind oftmals globaler und den eher detaillierten übrigen Anforderungen übergeordnet.

Tipp IV-4: Schaffen Sie einen Überblick durch Anforderungsgruppen oder -cluster

Fassen Sie ähnliche Use-Cases, User-Stories, Abläufe, Funktionen, Prozesse, Aufgaben oder sonstige funktionale Anforderungen zu Gruppen oder Clustern zusammen.

Beschreiben Sie in Ihrer Architekturdokumentation lediglich die Bedeutung dieser Gruppen, ohne auf atomare oder detaillierte einzelne Anforderungen einzugehen. Damit geben Sie einen *Überblick* über die Funktionen Ihres Systems.

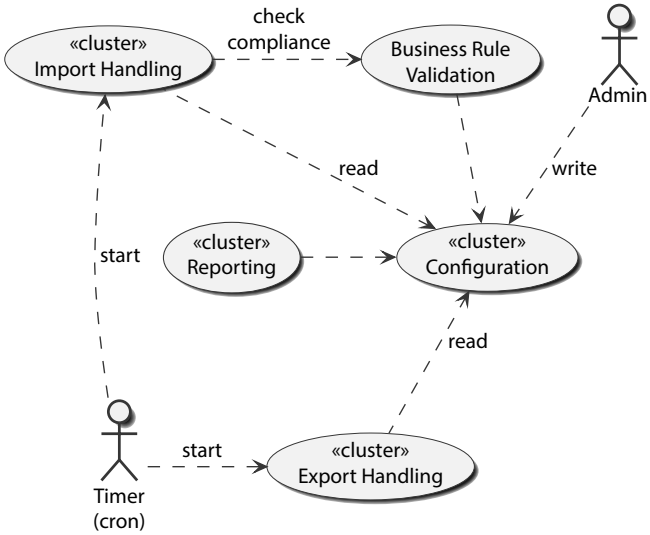


Bild IV.1 Anforderungscluster

[1] Steht es aber selten. Seufz.

In Bild IV.1 finden Sie dafür ein Beispiel: Einige Ellipsen gruppieren (*clustern*) jeweils mehrere Anforderungen oder Anwendungsfälle. Einige davon finden Sie in der Tabelle.

Cluster	Enthaltene Anwendungsfälle
Import Handling	Import-von-Mandant, Import-von-PrintShop, Import-von-Scanner, Import-von-CallCenter, Import-von-CAMS, ...
Configuration	Configure-Person, Configure-PrintJob, Configure-ScanOCR, Configure-Reports, ...

Tipp IV-5: Sorgen Sie für die Referenzierbarkeit der Anforderungen

In der Architekturdokumentation werden Sie an manchen Stellen Anforderungen referenzieren, beispielsweise in Begründungen von Entwurfsentscheidungen. Dazu müssen Anforderungen Identifikationsmerkmale besitzen, beispielsweise kurze Schlüssel.

- Manchmal können Sie solche IDs aus der Anforderungsdokumentation übernehmen.
- Sollten Sie Ihre Anforderungen in einem Tool (z. B. einem Issue-Tracker) verwalten, können Sie dessen IDs verwenden – bei manchen Werkzeugen haben Sie dann sogar stabile URLs.

Tipp IV-6: Beschreiben Sie funktionale Anforderungen als Aktivitätsdiagramm

Aktivitätsdiagramme können einerseits einen guten visuellen Überblick von Abläufen geben, andererseits auch die Behandlung von Sonderfällen, Alternativen, Parallelitäten oder Sequenzen erklären.

Potenzieller Nachteil ist ihr vergleichsweise hoher Erstellungs-/Pflegeaufwand.

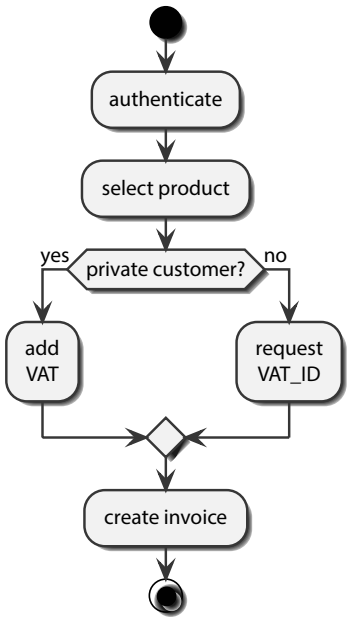


Bild IV.2 (Einfaches) Aktivitätsdiagramm

Tipp IV-7: Beschreiben Sie funktionale Anforderungen mit BPMN-Diagrammen

Falls Ihren Stakeholdern die Aktivitätsdiagramme (*siehe Tipp IV-6*) zu technisch sind, könnten BPMN-Diagramme helfen. Business Process Model and Notation richtet sich ja ausdrücklich an Business-Stakeholder. Für geschäftliche oder fachliche Abläufe könnte sie als Alternative zu den bisher genannten Möglichkeiten dienen.

Tipp IV-8: Beschreiben Sie funktionale Anforderungen als nummerierte Liste

Hier, im Rahmen der Aufgabenstellung, können Sie nummerierte Listen als einfache und pragmatische Möglichkeit verwenden, Aktivitäten, Abläufe oder Prozesse zu beschreiben. Der Ablauf aus Bild IV.2 könnte dann beispielhaft so lauten:

1) Authenticate
2) Selektiere ein Produkt

3a) Für private Kunden addiere die VAT (Mehrwertsteuer)

3b) Für Geschäftskunden: Erfrage VAT-ID

4) Erstelle die Rechnung

Falls Sie mit nebenläufigen Prozessen zu tun haben, klappt das mit Aktivitätsdiagrammen (*siehe Tipp IV-6*) allerdings besser.

Tipp IV-9: Beschreiben Sie funktionale Anforderungen mit (semi)formalem Text

Bild IV.2 haben wir mit PlantUML (*http://plantuml.com/*) erstellt (siehe Kapitel VI): Dieses Open-Source-Werkzeug generiert das Diagramm aus der folgenden textuellen Beschreibung:

```
@startuml
Start
:authenticate;

:select product;
if (private customer?) then (yes)
  :add\nVAT;
else (no)
  :request\nVAT_ID;
  endif

:create invoice;
stop

@enduml
```

Aktivitäten stehen zwischen : und ;, Verzweigungen können Sie wie Pseudocode lesen. Sie kombinieren damit die Vorteile von Plain-Text mit grafischer Repräsentation[2].

[2] In der Laufzeitsicht (Kapitel IV.6) bzw. in Kapitel VII über Werkzeuge werden wir auf diesen Ansatz noch näher eingehen.

Tipp IV-10: Beschreiben Sie funktionale Anforderungen als exemplarisches Geschäftsprozessmodell

eGPMs (exemplarische Geschäftsprozessmodelle) beschreiben Arbeitsabläufe aus Sicht der betroffenen Stakeholder, der eingesetzten Werkzeuge, Gegenstände und Materialien.

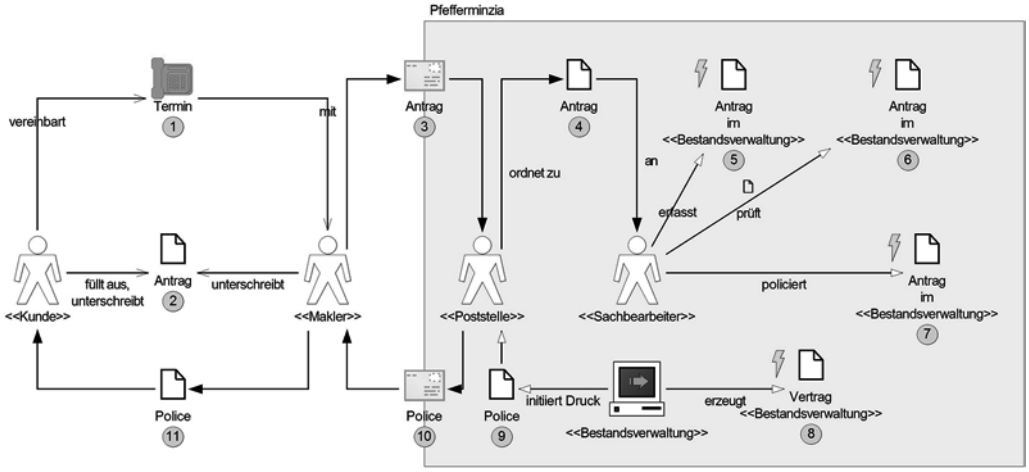


Bild IV.3 Exemplarisches Geschäftsprozessmodell

1.2 Qualitätsziele

Inhalt

Die Top-3 bis Top-5 der Qualitätsziele für die Architektur, deren Erfüllung oder Einhaltung den maßgeblichen Stakeholdern besonders wichtig sind.

Gemeint sind hier wirklich *Qualitätsziele*, die nicht unbedingt mit den Zielen des Projekts übereinstimmen. Beachten Sie den Unterschied.

Motivation

Sie müssen die für Ihre Stakeholder relevanten Qualitätsanforderungen an das System kennen, möglichst konkret und operationalisierbar. Wenn Sie als Architekt nicht wissen, woran Ihre Ergebnisse gemessen werden ...

Tipp IV-11: Arbeiten Sie grundsätzlich mit expliziten Qualitätsanforderungen

Anforderungsdokumente konzentrieren sich oft auf funktionale Anforderungen, Qualitätsziele bleiben implizit (und damit unklar, unsicher, interpretierbar ...). Dabei können Sie die gewünschten Qualitätsmerkmale relativ einfach systematisch erfassen, beispielsweise durch Szenarien.

Tipp IV-12: Erläutern Sie Qualitätsanforderungen durch Szenarien

Szenarien erklären in kurzen Sätzen, wie das System in bestimmten Situationen auf bestimmte Ereignisse reagieren soll. Szenarien beschreiben, wie das System auf Nutzung (durch Personen oder Systeme) bzw. auf Änderungen reagiert. Sie sollten diese Reaktion grundsätzlich durch eine Metrik beschreiben.

Von diesen Szenarien gibt es mehrere Kategorien:

- Anwendungsszenarien: Wie reagiert das System in bestimmten Nutzungsarten? Im Beispiel unten: Die Laufzeit einer HTML-Prüfung darf fünf Sekunden nicht überschreiten.
- Änderungsszenarien: Wie verhält sich das System, wenn Sie es erweitern oder ändern? Hiermit können Sie beispielsweise charakterisieren, welche Art von Änderungen oder Erweiterungen am System wie schnell oder mit welchem Aufwand möglich sein müssen.
- Ausfall- oder Fehlerszenarien: Wie verhält sich das System in gravierenden Fehlersituationen, etwa beim Ausfall zentraler Hard- oder Softwareteile.

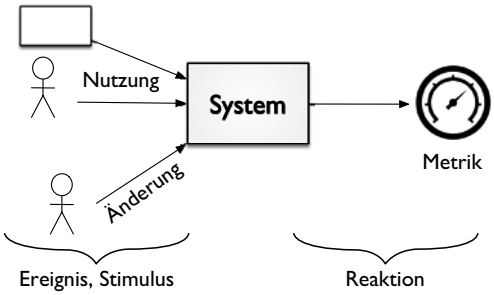


Bild IV.4 Szenarien zur Beschreibung von Qualitätsanforderungen

Szenarien können sich auf eine Vielzahl möglicher Qualitätseigenschaften von Systemen beziehen, die in gängigen Qualitätsmodellen (etwa: ISO 25010) hierarchisch gegliedert werden. Einige Beispiele:

Änderungsszenarien:

- Bei der Routenplanung der Roboter (im Hochregallager) soll ein neuer Algorithmus integriert werden. Ein Entwickler kann diese Änderung innerhalb von vier Stunden vornehmen, inklusive der Anpassung des Build-Systems und der Unit-/Integrationstests.
- Das Ausgabeformat des (jährlichen) Buchungsreports muss am Jahresende jeweils an neue gesetzliche Anforderungen angepasst werden. Alle für diesen Report notwendigen Ausgangsdaten liegen in der Datenbank vor, Anpassungen beziehen sich auf Aggregationen, Formatierung und Layout. Diese Änderungen sind in maximal 60 Personenstunden vollständig umgesetzt.

Anwendungsszenarien:

- Das System selektiert die für den XY-Prozess notwendigen Daten innerhalb von einer Sekunde (für bis zu 100 parallele Benutzern) bzw. innerhalb von drei Sekunden (für bis zu 1000 parallele Benutzer).
- Nach dem Einschalten ist das Navigationssystem innerhalb von vier Sekunden bereit, Eingaben über die GUI entgegenzunehmen.

Arbeiten Sie niemals an einer Architektur, deren Qualitätsanforderungen (synonym: Qualitätsziele) nicht explizit (schriftlich!) festgelegt und von den maßgeblichen Stakeholdern akzeptiert sind.

Tipp IV-13: Machen Sie Ihre Annahmen explizit, wenn Sie keine Qualitätsanforderungen bekommen

Immer wieder erleben wir in der Praxis, dass Entwicklungsteams keine Qualitätsanforderungen von Auftraggebern oder maßgeblichen Stakeholdern bekommen. Damit bleiben Qualitätsziele implizit, mit hohem Risiko von Missverständnis und Unzufriedenheit seitens aller Beteiligten.

Unser Rat: Sie können auf Basis Ihrer Kenntnisse und Erfahrungen *Annahmen* über angemessene Qualitätsanforderungen treffen, den sogenannten *educated guess*.

Schreiben Sie, gerne gemeinsam mit zwei bis drei Teammitgliedern, solche Annahmen als Szenarien auf und diskutieren Sie diese *educated guesses* mit Ihren Stakeholdern. Ihre Annahmen sind in jedem Fall besser, als keine expliziten Qualitätsanforderungen zu kennen!

Tipp IV-14: Verwenden Sie Checklisten für Qualitätsanforderungen

Der ISO-Standard 25010 bietet mit seiner hierarchischen Darstellung (siehe Bild IV.5) eine gute Checkliste. Alternativ enthält [arc42-QA] eine tabellarische Übersicht häufiger Qualitätsmerkmale.

- Verfügbarkeit (*availability*),
- Änderbarkeit (*modifiability*) oder Wartbarkeit (*maintainability*),
- Performanz (*performance*),
- Sicherheit (*security*, *safety*),
- Bedienbarkeit (*usability*),
- Testbarkeit (*testability*).

Tipp IV-15: Verwenden Sie Beispiele, um mit Ihren Stakeholdern Qualitätsziele zu erarbeiten

Das arc42-Subprojekt [arc42-QA] enthält mehr als 50 exemplarische Qualitätsszenarien, die Sie als Vorlage für die Qualitätsziele/-anforderungen Ihres Systems benutzen können. Auch in [Hruschka-19] finden Sie Formulierungsbeispiele für alle Kategorien von Qualitätsanforderungen.

Tipp IV-16: Halten Sie die Einführung kurz!

Zeigen Sie hier nur die „Top-Charts“ der Qualitätsanforderungen

Obwohl Sie alle Qualitätsanforderungen erfüllen müssen, die in Anforderungen und von Stakeholdern verlangt werden, sollten Sie diesen Abschnitt knapp halten. Zeigen Sie hier lediglich eine Handvoll („Hitparade“), möglichst mit kurzen Erläuterungen, nicht nur als Schlagwörter. (Die anderen Qualitäten stehen entweder im Lasten-/Pflichtenheft oder als Qualitätsbaum in arc42-Abschnitt 10.)

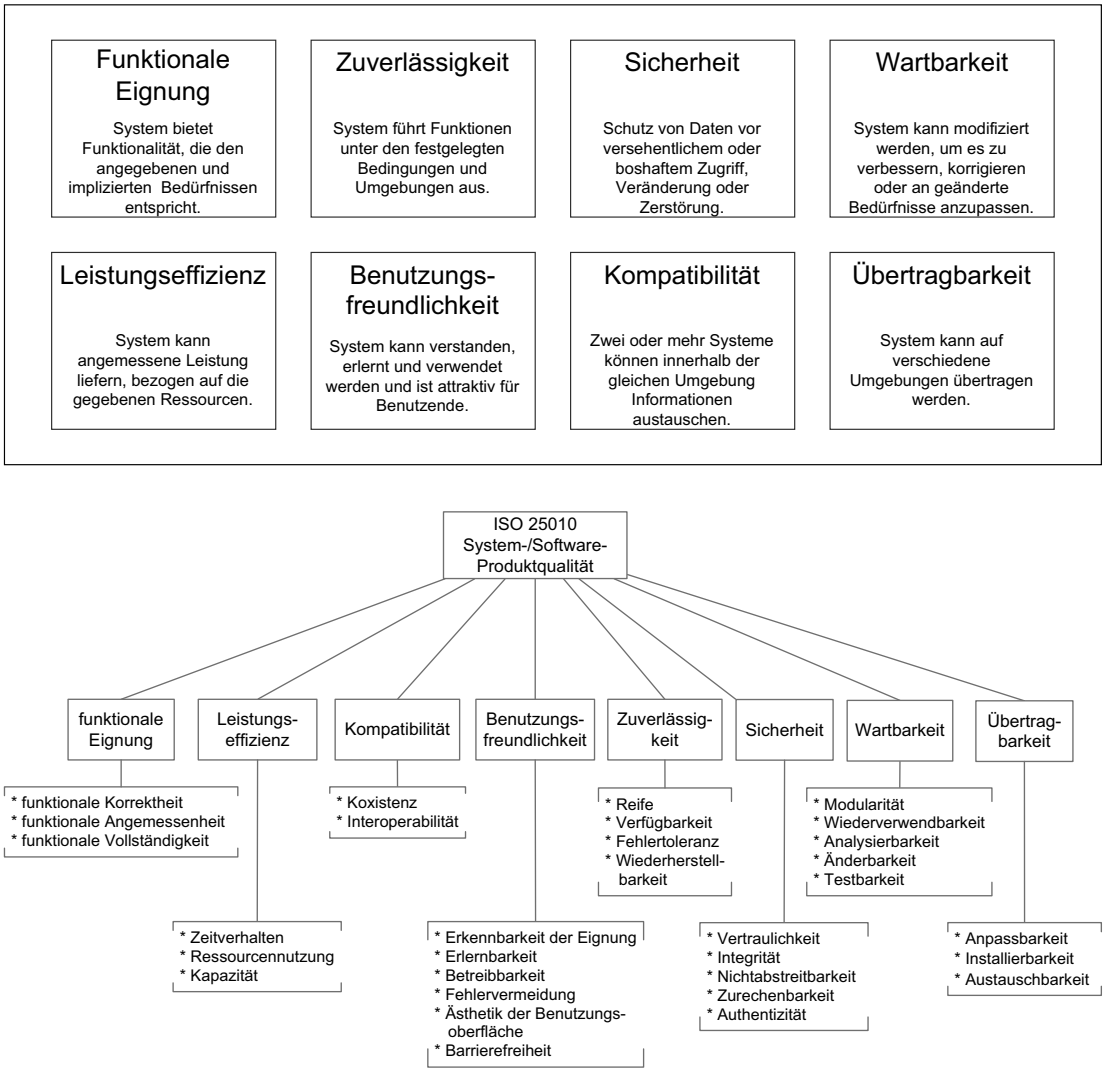


Bild IV.5 oben: Übersicht Qualitätsmerkmale nach ISO 25010 (Überblick)
unten: Qualitätsmerkmale nach ISO 25010 (Details)

Tipp IV-17: Kombinieren Sie Qualitätsziele mit Lösungsmaßnahmen im Abschnitt „Lösungsstrategie“

Manchmal treffen Sie Entscheidungen auf Basis konkreter, spezifischer Qualitätsanforderungen. In solchen Fällen hilft es, diese Qualitätsziele/-anforderungen und die resultierenden Entscheidungen gemeinsam in einer konsolidierten Tabelle zu dokumentieren. Wir schlagen dafür den arc42-Abschnitt 4 (Lösungsstrategie) vor. In arc42-Abschnitt 1.2 (Qualitätsziele) nehmen Sie dann nur einen Verweis auf.

Tipp IV-18: Zeigen Sie ausführliche Qualitätsanforderungen in arc42-Abschnitt 10

Eine ausführliche Übersicht aller Qualitätsanforderungen (Qualitätsbaum und Szenarien) sollten Sie in arc42-Abschnitt 10 (Qualitätsbaum) aufführen. Dort können Sie auch die Beziehungen zwischen den Qualitätszielen darstellen.

1.3 Stakeholder

Inhalt

Expliziter Überblick über die Stakeholder des Systems, d. h. über alle Personen, Rollen oder Organisationen, die

- die Architektur kennen sollten oder
- von der Architektur überzeugt werden müssen,
- mit Architektur oder Code arbeiten (z. B. Schnittstellen nutzen),
- Dokumentation der Architektur für ihre eigene Arbeit benötigen,
- Entscheidungen über das System und dessen Entwicklung treffen.

Motivation

Sie sollten die Projektbeteiligten und -betroffenen kennen, sonst erleben Sie später im Entwicklungsprozess Überraschungen. Diese Stakeholder bestimmen unter anderem Umfang und Detaillierungsgrad der von Ihnen zu leistenden Arbeit und Ergebnisse.

Tipp IV-19: Führen Sie eine Breitensuche nach Stakeholdern durch

Nur als Anregung haben wir aus [Clements+11] sowie aus dem arc42-Template mal eine erschreckend lange Liste möglicher Stakeholder mitgebracht. All diese Personen oder Rollen könnten ein Interesse an der Architektur oder deren Dokumentation haben ...

Analyst, Business-Analyst, (andere) Architekten, Auditor, Aufsichtsgremium, Auftraggeber, Behörde, Betriebsrat, Build-Manager, Business-Manager, Datenbankadministrator, Endbenutzer, Enterprise-Architekt, Entwickler, externe Dienstleister, externe Partner, Fachabteilung, Fachadministrator, Hardwareingenieur, Hotline/Support, Infrastrukturplaner, Integrator, IT-Strategie, Jurist, Konfigurationsmanager, Kontrollgremium, Kunde, Layouter, Lenkungskreis, Management, Nachbarsysteme, Netzwerkadministrator, Operator, Produktmanager, Product-Owner, Projektleiter, QS-Abteilung, Release-Manager, Rollout-Manager, Scrum-Master, Sicherheitsbeauftragte, Systemintegrator, Tester, TÜV, UX-Designer, verbundene Projekte, Wartungsteam, Webdesigner, Zulieferer

Tipp IV-20: Beschreiben Sie die Erwartungen Ihrer Stakeholder an Architektur und Dokumentation

Klären Sie die konkrete Erwartungshaltung dieser Stakeholder bezüglich der Architektur und deren Dokumentation. Fragen Sie dabei sowohl nach erwarteter Form wie auch nach Inhalten und notwendiger Detaillierung.

Stakeholder nach ihrer Erwartungshaltung zu befragen, kann eine Menge Nutzen stiften, auch wenn es zuerst nicht unbedingt nach Architekturarbeit klingt:

- Sie können spezifischer auf die Bedürfnisse der Beteiligten eingehen und damit bei Ihrem Zielpublikum mehr Zufriedenheit erreichen.
- Sie sparen sich Arbeit, weil Sie sich auf die Inhalte/Themen konzentrieren können, die Ihre Beteiligten wirklich benötigen. Sie vermeiden es, „auf Vorrat" zu dokumentieren.

Tipp IV-21: Pflegen Sie eine Stakeholder-Tabelle

Sie sollten die Erwartungshaltung dieser Stakeholder (siehe oben) bezüglich der Architektur und deren Dokumentation in Form einer Tabelle explizit darstellen.

In Tabelle IV.1 finden Sie eine Minimalversion, die lediglich die Erwartungshaltung bzw. benötigte Artefakte skizziert.

Tabelle IV.1 Kurze Stakeholder-Tabelle mit Rolle und Erwartungshaltung

Rolle	Erwartungshaltung
Administrator	Deployment-Übersicht, Installations- und Betriebshinweise, Firewalls
QM-Abteilung	Beschreibung der Schnittstellen zum Lasttest, mögliche Messpunkte zu Performancetests, technisches Konzept für Security und Ausfallsicherheit
...	...

Tabelle IV.2 enthält eine ausführlichere Version mit Abnahmerelevanz bzw. Kontaktdaten. Beachten Sie mögliche Veränderungen in Projektteams – und nehmen Sie neben konkreten Personen bei Bedarf auch deren Vertretungen bzw. Arbeitsbereiche/Abteilungen/Organisation mit auf.

Tabelle IV.2 Ausführlichere Stakeholder-Tabelle

Rolle	Kontakt	Abnahmerelevanz	Erwartungen
Projektleitung	Frau Dr. FooBar, *foobar@nsa.org*	Hoch	Übersicht technischer Risiken, externe Schnittstellen
Auftraggeber	Frau Dr. Lovelace	Hoch	Nachweis der Erfüllbarkeit der Top-3-Qualitätsziele
Backend-Entwickler	Bruno Batch	Keine	Persistenz- & Reporting-Konzept, Details der DWH-Schnittstelle
...	...	...	...

Tipp IV-22: Verzichten Sie auf die Stakeholder-Tabelle, sofern Ihr Management eine konsistente Stakeholder-Übersicht pflegt

Sofern Ihr Management (etwa: Projektleitung oder Product-Owner) die Übersicht der Stakeholder ernst nimmt und eine Stakeholder-Tabelle pflegt, sollten Sie in der Architekturdokumentation lediglich einen Querverweis aufnehmen.

Aber Vorsicht: Unserer Erfahrung nach fokussieren Projektleiter eher auf organisatorische Informationen über Stakeholder (z. B. Kontaktadressen und Ansprechpartner). In arc42 benötigen wir eher inhaltliche Informationen über die konkrete Erwartungshaltung der

Stakeholder an die Architektur und deren Dokumentation. Daher schlagen wir vor, die Stakeholder-Tabelle doch als Architekt selbst zu pflegen ...

Tipp IV-23: Klassifizieren Sie Ihre Stakeholder nach Interesse und Einfluss

Insbesondere bei Arbeit unter Zeitdruck können Sie sich wahrscheinlich nur um wenige Ihrer Stakeholder kümmern. Gerade in solchen Fällen kann statt einer Stakeholder-Tabelle eine visuelle Klassifikation nach Interesse und Einfluss helfen (siehe Bild IV.6). Eine solche Einordnung können Sie am Flipchart sehr informell im Team vornehmen und bei Bedarf dann in Ihre arc42-Dokumentation übernehmen.

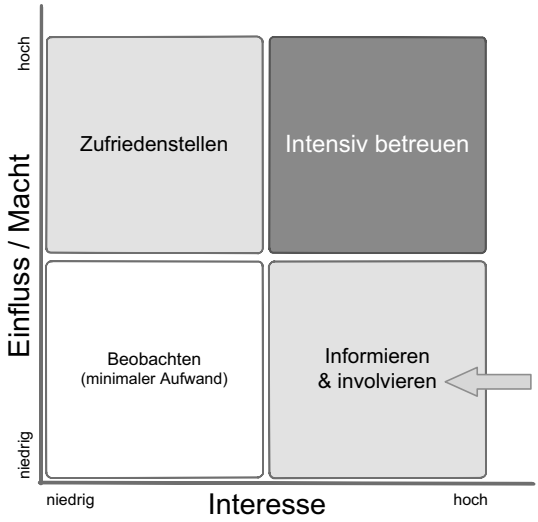


Bild IV.6 Klassifikation von Stakeholdern

- Hoher Einfluss und großes Interesse: Das sind Personen, die Sie intensiv involvieren und/oder betreuen sollten. Setzen Sie alles daran, diese Kategorie von Stakeholdern in allen Belangen zufriedenzustellen. Gehen Sie in der Kommunikation aktiv auf diese Personen zu.
- Hoher Einfluss, weniger großes Interesse: Investieren Sie gerade genug Aufwand, um diese Stakeholder zufriedenzustellen.
- Wenig Einfluss, großes Interesse: Diese Stakeholder können sehr hilfreich für die Arbeit am System sein: Involvieren Sie sie aktiv in die Arbeit am System, halten Sie sie angemessen informiert. Insbesondere können sie Feedback zu allen Arten von technischen oder fachlichen Details liefern.
- Wenig Einfluss, geringes Interesse: Hier können Sie Ihren Aufwand minimieren und beispielsweise Informationen bereitstellen, die diese Stakeholder nur bei Bedarf konsumieren.

Im Idealfall haben Sie eine Stakeholder-Tabelle plus eine solche Klassifizierung.

Anmerkung: Manche Stakeholder könnten ihren eigenen Einfluss/ihre Macht höher einschätzen, als es diese Einordnung widerspiegelt. Eine Veröffentlichung dieser Matrix könnte somit bei manchen Stakeholdern zu Verdruss führen. ☺

2 Randbedingungen

Inhalt

Fesseln 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

Als Architekt sollten Sie explizit wissen, wo Ihre Freiheitsgrade bezüglich Entwurfsentscheidungen liegen und wo Sie Randbedingungen beachten müssen. Sie können Randbedingungen vielleicht noch verhandeln, zunächst sind sie aber da.

Tipp IV-24: Suchen Sie bei anderen Systemen innerhalb der Organisation nach Randbedingungen

Falls Sie für Ihr System keine Randbedingungen kennen, beginnen Sie Ihre Suche bei anderen Systemen innerhalb der Organisation.

Tipp IV-25: Klären Sie die Konsequenzen von Randbedingungen

Sie sollten die *Konsequenzen* von Randbedingungen klären – beispielsweise resultierende (Mehr-)Kosten oder Aufwände.

Falls Randbedingungen *unangemessene* Konsequenzen mit sich bringen (beispielsweise nur mit übermäßig hohem Aufwand erfüllbar sind), sollten Sie mit den entsprechenden Stakeholdern darüber verhandeln.

Tipp IV-26: Beschreiben Sie organisatorische Randbedingungen

Organisatorische Randbedingungen, beispielsweise bezüglich Zeit und Budget, sind aus gutem Grund bei Entwicklungsteams unbeliebt – denn sie beschneiden die Freiheit bei Entwurfs- oder Implementierungsentscheidungen oftmals sehr stark. Legen Sie daher diese Art der Randbedingungen offen.

Manchmal beziehen sich solche Randbedingungen auf Entwicklungsprozesse, die Vergabe von Aufgaben oder Aufträgen an Dritte oder auch juristische Belange. Klären Sie solche Themen mit Ihrem Management.

Tipp IV-27: Beschreiben Sie Randbedingungen für Entwurf und Entwicklung

Für Entwurf und Entwicklung Ihres Systems gelten neben den organisatorischen wahrscheinlich auch technische Randbedingungen: Das können Vorgaben der Organisation oder des Managements bezüglich Hardware, Betrieb, Technologieauswahl, Verwendung von Produkten, Frameworks oder auch Referenzarchitekturen sein.

Legen Sie solche Randbedingungen offen, damit das Entwicklungsteam sich rechtzeitig darauf einstellen kann.

Tipp IV-28: Differenzieren Sie verschiedene Kategorien von Randbedingungen

Bei Bedarf unterscheiden Sie technische, organisatorische und politische Randbedingungen oder übergreifende Konventionen (beispielsweise Programmierrichtlinien, Dokumentations-, Namens- oder Ordnungskonventionen).

3 Kontextabgrenzung

Inhalt

Die Kontextabgrenzung grenzt das System gegen alle Kommunikationspartner (Nachbarsysteme und Benutzerrollen) ab.

Sie legt die externen Schnittstellen fest und zeigt damit auch die Verantwortlichkeit (scope) Ihres Systems: Welche Verantwortung trägt das System und welche Verantwortung übernehmen die Nachbarsysteme?

Differenzieren Sie fachlichen (Ein- und Ausgaben) und technischen Kontext (Kanäle, Protokolle, Hardware), falls nötig.

Motivation

Die fachlichen und technischen Schnittstellen zu Nachbarsystemen gehören zu den kritischsten Aspekten eines Systems. Stellen Sie rechtzeitig sicher, dass Sie diese komplett verstanden haben.

Die nächsten Tipps gelten zunächst für beide Arten der Kontextabgrenzung. Spezifische Ratschläge finden Sie unter den jeweiligen Überschriften IV.3.1 bzw. IV.3.2.

Tipp IV-29: Grenzen Sie explizit Ihr System von dessen Umfeld ab

Sie sollten Ihr System gegenüber allen Nachbarsystemen abgrenzen. Damit ordnen Sie die Aufgabe Ihres Systems in dessen Umfeld ein, andererseits zeigen Sie sämtliche externen Schnittstellen und Nutzer bzw. Nutzerrollen.

Sie zeigen damit auch die Verantwortlichkeit (*scope*) Ihres Systems: Welche Verantwortung trägt das System und welche Verantwortung übernehmen die Nachbarsysteme? In Bild IV.8 ist für einen Webshop die Abwicklung der Zahlungen an einen externen Provider delegiert, also aus dem Scope des Webshops herausgelöst.

Tipp IV-30: Stellen Sie den Kontext grafisch dar

In der grafischen Kontextabgrenzung sollten Sie das gesamte System als eine Blackbox zeigen, beispielsweise als eine einzige Komponente. Weiter unten finden Sie einige Beispiele für Kontextdiagramme (Bild IV.7, Bild IV.8).

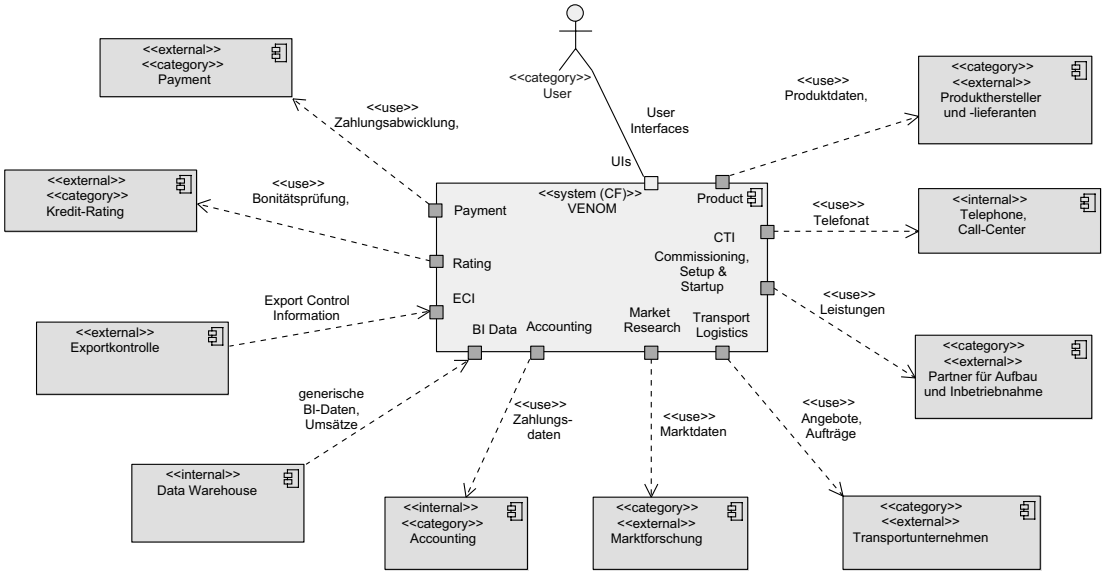


Bild IV.7 Umfangreiche Kontextabgrenzung

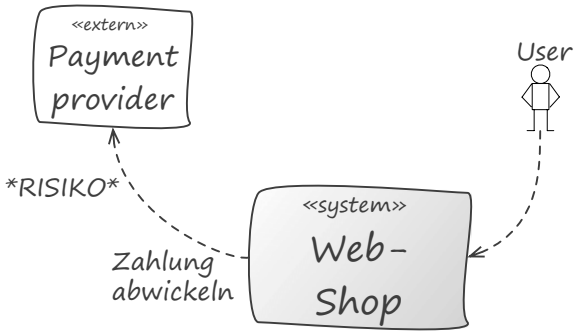


Bild IV.8 Kontextabgrenzung mit Hinweis auf Risiko

Sie haben eine Reihe grafischer Möglichkeiten:

- Unser Favorit: UML-Komponenten- oder Paketdiagramme,
- UML-Use-Case-Diagramme,
- Kästen und Pfeile.

In jedem Fall sollten Sie das System visuell in der Mitte platzieren, die Nachbarn außen herum.

Tipp IV-31: Kombinieren Sie das Kontextdiagramm mit einer Tabelle

Sie sollten das Kontextdiagramm immer durch eine Tabelle ergänzen. Auf diese Weise können Sie die Menge an Beschriftungen im Diagramm reduzieren und ganz einfach Erläuterungen, Begründungen oder Querverweise aufnehmen.

Im Bild IV.7 finden Sie ein umfangreiches Kontextdiagramm, das unbedingt eine solche tabellarische Erläuterung benötigt.

Wir geben die zugehörige Tabelle hier nur auszugsweise wieder, aber einige Merkmale dieses typischen „Grafik/Tabelle“-Pärchens sollten Sie beachten:

- In der Grafik können Sie kurze Bezeichner verwenden, im Sinne von Oberbegriffen oder Abstraktionen. Im Beispiel verwenden wir „Leistungen“ oder „Marktdaten“ – die wir in der Tabelle dann ausführlicher erläutern.
- Verweisen Sie in der Tabelle auf detaillierte Erläuterungen. Falls Sie beispielsweise Begriffe aus Ihrer Fachdomäne verwenden, kann deren Erklärung hier im Kontext entfallen und Sie verweisen auf den entsprechenden Abschnitt (etwa: arc42-Abschnitt 8.1, Domänenmodell).

Tabelle IV.3 Ergänzende Tabelle zu Bild IV.7 (Auszug)

Element	Bedeutung
User	Fasst sämtliche Arten von Benutzern zusammen: interne (Backoffice), externe (Kunden, Partner)
Produktdaten	Produktdaten bestehen aus den Katalogdaten, Abbildungen, aber auch Verfügbarkeiten, Konfigurationsregeln, Bestell- und Lieferinformationen sowie teilweise auch Preisen und Bezugsquellen.
Leistungen	Enthalten mögliche Transport- und Aufbauleistungen, Angebote, Termine sowie die verbindlichen Bestellungen

Tipp IV-32: Weisen Sie schon im Kontext ausdrücklich auf Risiken hin

Nutzen Sie dafür beispielsweise auffällige Farben oder Symbole. In der Abbildung nutzt ein Webshop einen Payment-Provider zur Abwicklung von Zahlungen. Ganz offensichtlich ist das für den Webshop eine wirklich wichtige Aufgabe: Die Verfügbarkeit des Webshops ist möglicherweise an die Verfügbarkeit des Zahlungsdienstleisters gekoppelt.

Solche Risiken sollten Sie im Diagramm markieren und im begleitenden Text erläutern. Im Idealfall verweisen Sie auf die Risikoliste Ihres Projekts bzw. auf arc42-Abschnitt 11.

Weitere Beispiele für Risiken, die Sie im Kontext deutlich kennzeichnen sollten:

- Verfügbarkeitsrisiko bei Ausfall externer Systeme: Ein externes System entscheidet über die Verfügbarkeit Ihres eigenen Systems mit.
- Kostenrisiken: Die Nutzung eines externen Systems ist teuer, einzelne Aufrufe oder Nutzungen des externen Systems kosten Geld. Beispiele hierfür sind etwa Kreditkartenprüfungen oder Zahlungs-/Buchungsdienstleistungen.
- Sicherheitsrisiken: Sie erhalten von externen Systemen sensible Daten bzw. übertragen sensible Daten an externe Systeme. Das könnte diese Schnittstellen besonders interessant für mögliche Angreifer machen.
- Volatilität (hohe Änderungswahrscheinlichkeit) bei Nachbarn: Die Schnittstellen der Nachbarsysteme befinden sich „in Arbeit“. Syntax und Semantik übertragener Daten könnten sich kurzfristig ändern (was entweder Arbeitsaufwand für Sie bedeutet oder aber Sie müssen genügend Flexibilität auf Ihrer Seite dieser Schnittstelle vorsehen).
- Komplexitätsrisiken: Die Nutzung dieser Schnittstelle ist besonders aufwendig oder schwierig. Das kann an komplexen Datenstrukturen liegen, an aufwendigen Handshake-Verfahren, an esoterischen Frameworks oder einer beliebigen Mischung.

Tipp IV-33: Schaffen Sie im Kontext Übersicht und verzichten Sie auf Details

Zeigen Sie Details, beispielsweise von externen Schnittstellen oder Nachbarsystemen, in der Bausteinsicht oder in externen Schnittstellendokumenten.

Das Beispiel von Bild IV.9 enthält zu viele Detailschnittstellen – Sie könnten die sieben einzelnen Schnittstellen zur Billing-Komponente auch durch eine einzige „Billing"-Schnittstelle ersetzen und die Details erst in der Bausteinsicht zeigen.

Zu viele Details können Kommunikation und Feedback über die Kontextabgrenzung erschweren.

Allerdings sollen Sie fachliche Ein- und Ausgaben und Protokolle hier zumindest benennen.

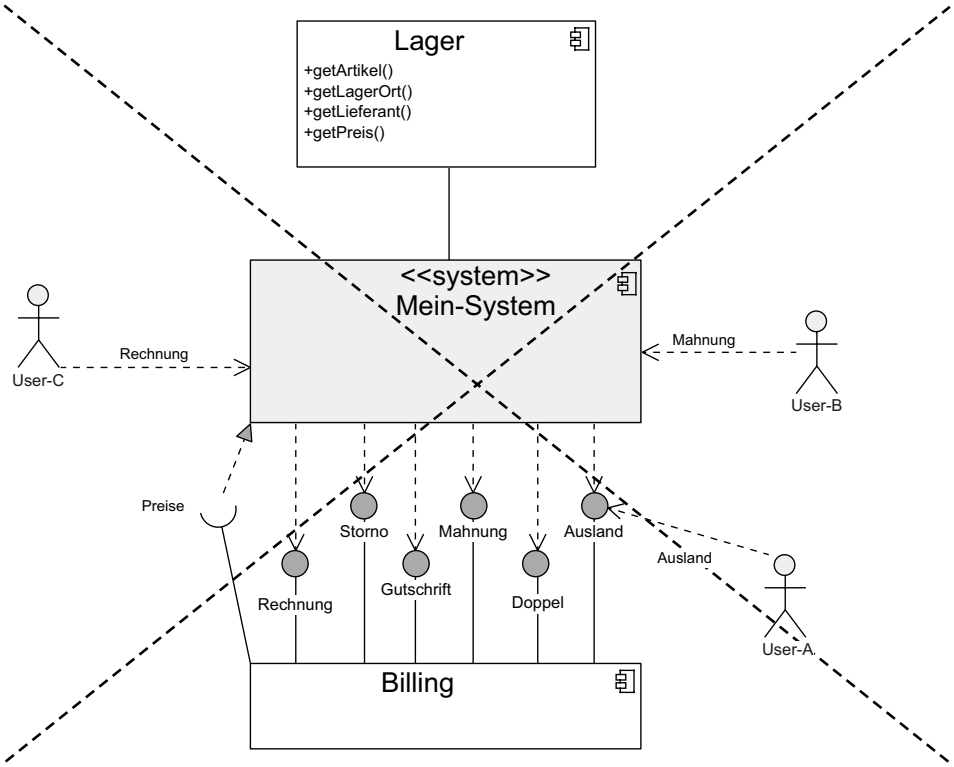


Bild IV.9 Zu viele Details im Kontext

Tipp IV-34: Vereinfachen Sie die Kontextabgrenzung durch Kategorisierung

Halten Sie die Kontextabgrenzung einfach: Fassen Sie externe Schnittstellen, Systeme oder Benutzerrollen zusammen, die große Ähnlichkeiten besitzen. Zeigen Sie explizit, dass es sich um Kategorien oder Zusammenfassungen handelt.

In Bild IV.10 übermittelt das System „Foo" *Reports* an User. In der Verfeinerung Level-1 rechts erkennen Sie zwei unterschiedliche Arten dieser Reports für zwei unterschiedliche Gruppen von Benutzern. Diese Art der Verfeinerung ist erwünscht und konsistent. In der Kontextabgrenzung sind *Reports* entsprechend als *category* markiert.

Solche Kategorien können Sie für vielerlei Kriterien finden (siehe folgenden Tipp).

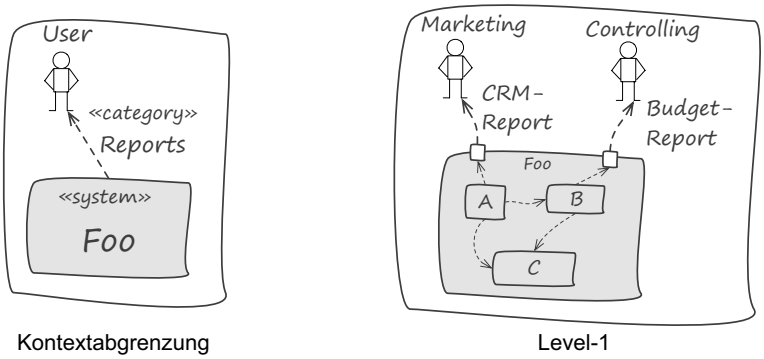


Bild IV.10 Abstraktion im Kontext

Tipp IV-35: Bei vielen Nachbarsystemen: Fassen Sie Nachbarn nach expliziten Kriterien zusammen

Sollte Ihr System mit vielen Nachbarsystemen interagieren, so können Sie mehrere dieser Nachbarn nach bestimmten Kriterien zusammenfassen. Sie sollten diese Kriterien explizit darlegen.

Solche Kriterien können beispielsweise sein:

- Nachbarn, mit denen wir gemeinsame oder ähnliche Daten austauschen,
- Nachbarn, mit denen wir im Rahmen derselben Anwendungsfälle oder zur selben Zeit kommunizieren,
- Nachbarn, mit denen wir mit Hilfe identischer oder ähnlicher Technologien kommunizieren (etwa: kategorisiert nach ftp- und WebService-Nachbarn),
- Nachbarn, die zu gleichen oder ähnlichen Organisation gehören, etwa: alle Systeme von Kfz-Werkstätten, Versicherungen, Glasereien und Steuerberatern,
- Nachbarn, die ähnliche fachliche oder technische Aufgaben lösen (alle Nachbarn, die mit dem Thema Dokumentendrucken zu tun haben, Nachbarn, die mit dem Thema „Chipkarte“ zu tun haben).

Im Beispiel finden Sie links das Nachbarsystem „Logistics“ als «category» gekennzeichnet. Auf der rechten Seite sehen Sie die ausführliche Version, die drei konkrete Instanzen der Kategorie «logistics» zeigt.

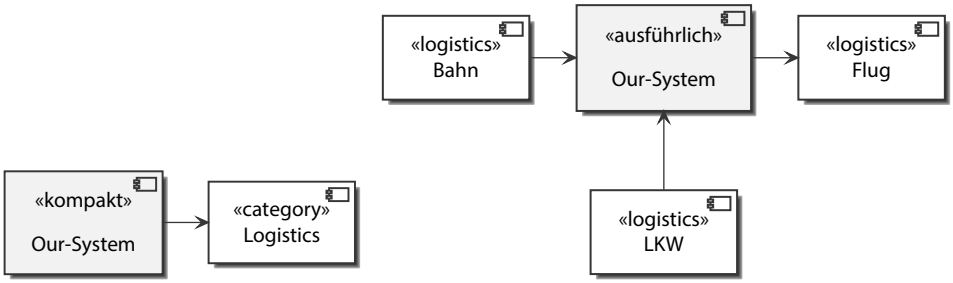


Bild IV.11 Zusammenfassung/Kategorisierung von Nachbarsystemen

Tipp IV-36: Fassen Sie „viele Nachbarsysteme“ über Ports zusammen

Sollte Ihr System mit vielen Nachbarsystemen interagieren, so können Sie an der Außenschnittstelle Ihres Systems *Ports* zeigen, statt die Nachbarn alle als eigene Symbole darzustellen (siehe Bild IV.12).

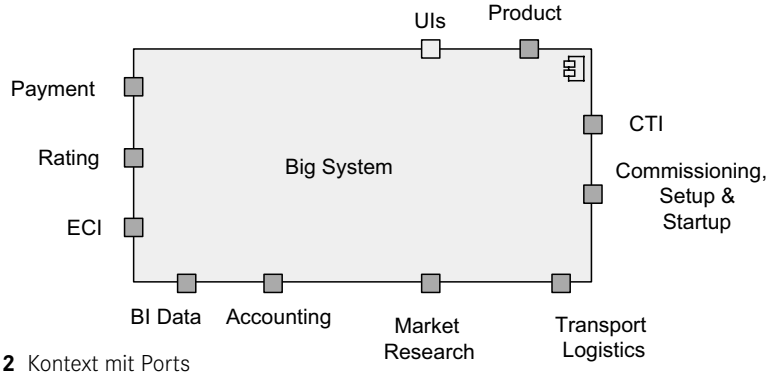


Bild IV.12 Kontext mit Ports

Das sieht auf den ersten Blick ungewöhnlich aus, kann aber Aufwand sparen. Vergleichen Sie die Diagramme aus Bild IV.7 und Bild IV.12 – es handelt sich um dasselbe System.

Tipp IV-37: Zeigen Sie sämtliche (alle!) externen Schnittstellen

Wir halten Vollständigkeit in der Regel für ein schlechtes Ziel – mit (mindestens) einer Ausnahme: Sie sollten alle, wirklich alle, externen Nachbarn Ihres Systems explizit in der Kontextabgrenzung aufnehmen.

Dabei können Sie, um Aufwand zu sparen, gerne Kategorien, Gruppen oder Cluster von Nachbarn bilden (*siehe Tipp IV-34 sowie Tipp IV-35*).

Tipp IV-38: Falls Sie externe Schnittstellen nach außen anbieten, erzeugen Sie eigenständige Schnittstellendokumente

Die Entwicklungsteams dieser externen Schnittstellen möchten eine kompakte und bedarfsgerechte Dokumentation der von ihnen benötigten Schnittstellen. Erklären Sie daher die Benutzung (!) der Schnittstellen, weniger die grundlegenden Konzepte oder Begründungen.

REST-APIs könnten Sie ansatzweise mit Tools wie swagger (*http://swagger.io/*) oder Apiary (*https://apiary.io/*) dokumentieren[3]. Für externe Programmierschnittstellen bietet es sich an, über Unit-Tests die Nutzung der API zu beschreiben[4].

Tipp IV-39: Differenzieren Sie fachlichen und technischen Kontext

Insbesondere bei Informationssystemen können Sie in der Kontextabgrenzung oft auf technische Details der Infrastruktur verzichten und sich auf die fachliche Umgebung konzentrieren.

Falls Hardware oder Technik jedoch eine große Rolle für Ihr System spielt, werden Sie sowohl einen fachlichen wie auch einen technischen Kontext benötigen. Siehe dazu Abschnitt 3.2 (Technischer Kontext).

[3] Korrekterweise merkt Silvia Schreier hierzu an, dass diese Tools Hypermedia leider außen vor lassen.
[4] In den Abschnitten IV.5 (Bausteinsicht) und IV.6 (Laufzeitsicht) gehen wir auf diesen Rat näher ein.

Bild IV.13 zeigt ein kleines Beispiel eines (Web-)Informationssystems mit fachlichem und technischem Kontext. Sie erkennen im technischen Kontext Protokolle wie HTTPS, SSH, die im fachlichen Kontext absichtlich nicht erwähnt werden. Ferner läuft die (fachlich externe) Komponente „Reporting“ in derselben technischen Infrastruktur wie das eigentliche System.

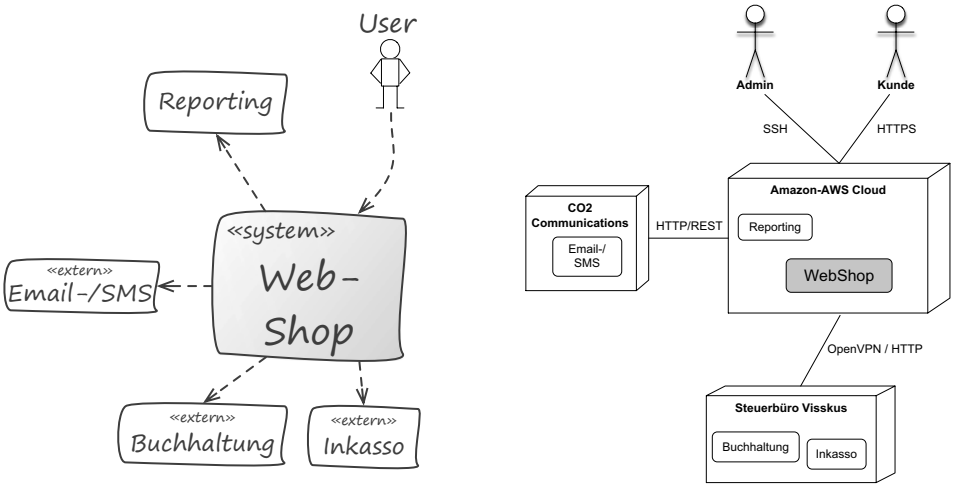


Bild IV.13 Fachliche und technische Kontextabgrenzung (Informationssystem)

Bild IV.14 zeigt ein vereinfachtes Beispiel aus der Embedded-Welt: Links sehen Sie den fachlichen Kontext, rechts den technischen.

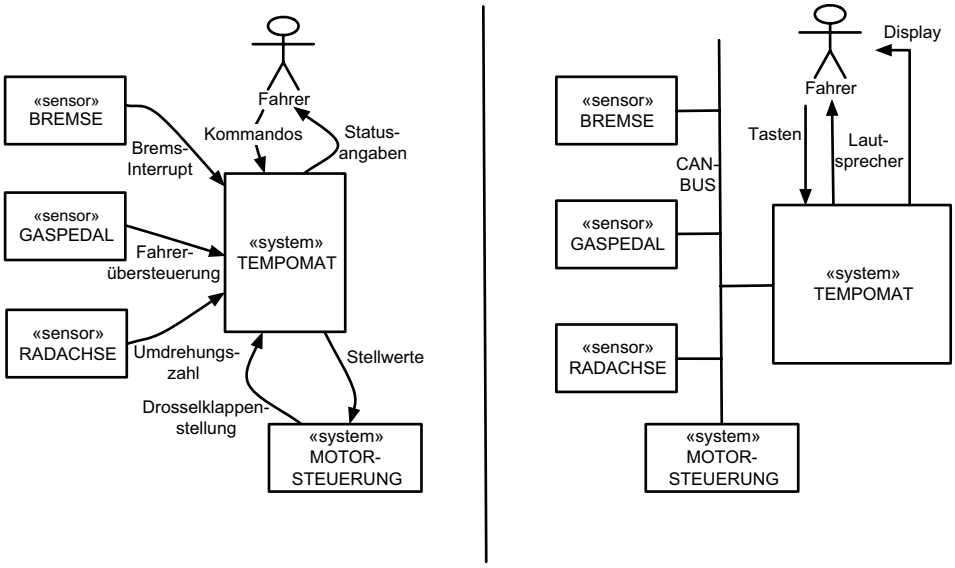


Bild IV.14 Fachliche und technische Kontextabgrenzung (Embedded-System)

Stichwortverzeichnis