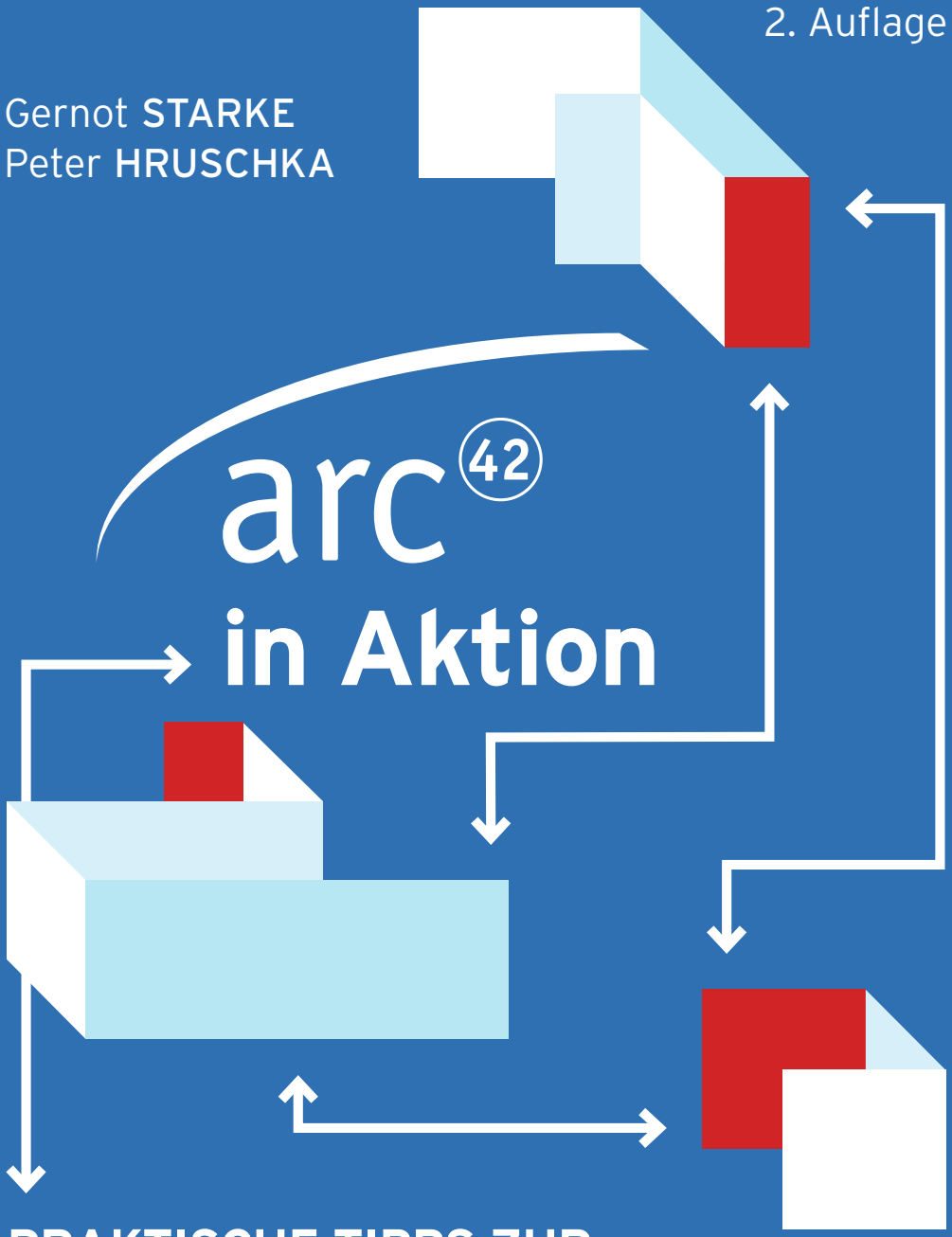
2. Auflage

Gernot **STARKE**
Peter **HRUSCHKA**



arc 42 in Aktion

PRAKTISCHE TIPPS ZUR ARCHITEKTURDOKUMENTATION

HANSER

INNOQ

Starke/Hruschka

arc42 in Aktion

Gernot Starke
Peter Hruschka

arc42 in Aktion

Praktische Tipps zur Architekturdokumentation

2., überarbeitete Auflage

HANSER

Dr. Gernot Starke, Köln
Dr. Peter Hruschka, Aachen

Bibliografische Information der Deutschen Nationalbibliothek:

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über *http://dnb.d-nb.de* abrufbar.

Lektorat: Brigitte Bauer-Schiewek
Copy editing: Petra Kienle, Fürstenfeldbruck
Layout: Manuela Treindl, Fürth
Umschlagdesign: Marc Müller-Bremer, *www.rebranding.de*, München
Coverillustration: Andreas Steinbrecher (*www.andreassteinbrecher.de*), Düsseldorf
Umschlagrealisation: Max Kostopoulos
Druck und Bindung: Hubert & Co. GmbH & Co. KG BuchPartner, Göttingen
Printed in Germany

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

Inhalt

1 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*.

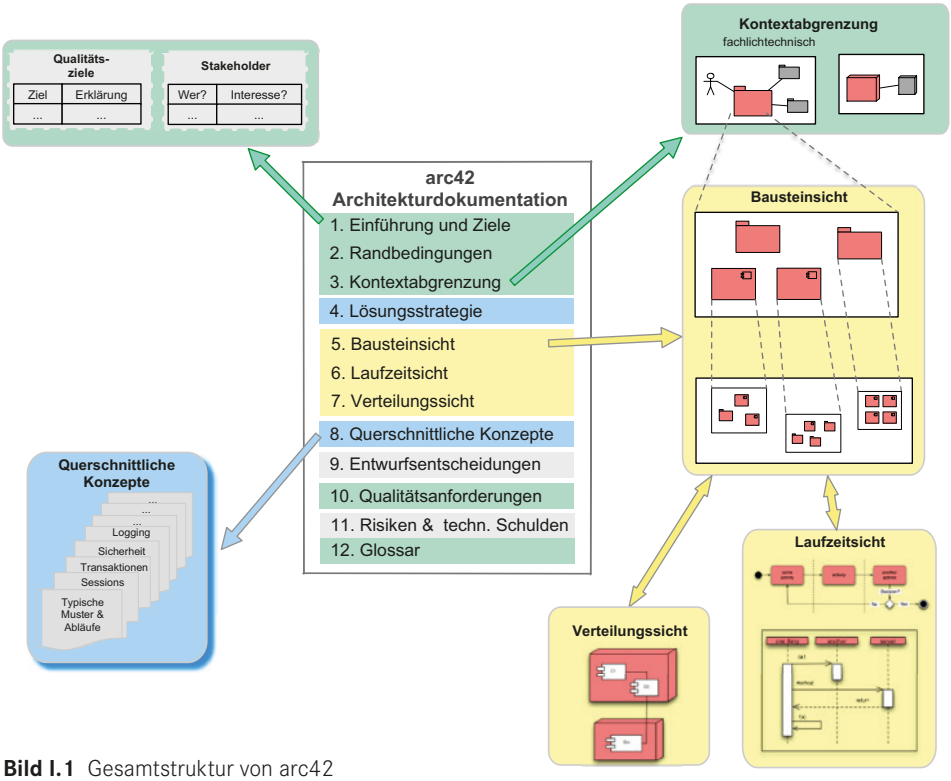


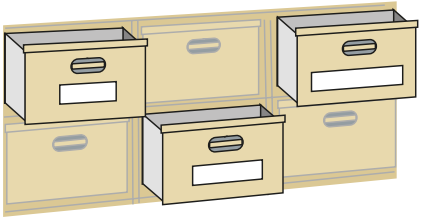
Bild I.1 Gesamtstruktur von arc42

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 ...

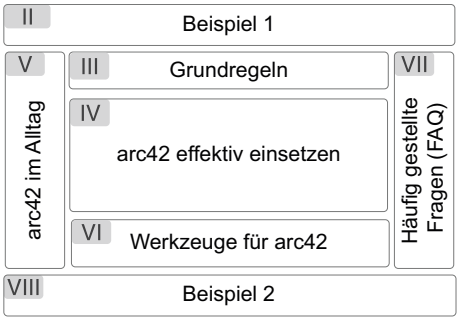
Als Autoren haben wir einige (möglicherweise törichte) Annahmen über Sie als Leser:in getroffen:

- 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.

II arc42 am Beispiel (1)

Dieses Kapitel zeigt Ihnen arc42 anhand des kleinen Open-Source-Systems HtmlSanityCheck [hsc] – einem Tool zum Auffinden von Fehlern in HTML-Seiten. Zu Beginn jedes Abschnitts finden Sie kurze Erläuterungen, formatiert in Kästen wie diesem hier. Sie erhalten darin Vorschläge für Inhalt und Form.

Praxistipps zu den Bestandteilen von arc42 finden Sie in den jeweiligen Abschnitten von Kapitel IV.

1 Einführung und Ziele

Sie finden in diesem Abschnitt die wesentlichen Anforderungen an das System und die treibenden Kräfte für die Entwicklung der Architektur, damit Sie die Lösung und Architekturentscheidungen einordnen und verstehen können.

1.1 Aufgabenstellung

Inhalt und Motivation

Sie möchten zentrale Ziele und Aufgaben des Systems, wichtige Anwendungsfälle oder Features in wenigen Sätzen erklären. Falls vorhanden, verweisen Sie auf die Anforderungsdokumentation.

Unabhängig von der gewählten Form der Beschreibung: Hauptsache, Ihre Leser können die wesentlichen Aufgaben des Systems verstehen, bevor Sie zur eigentlichen Architektur des Systems („die Lösung“, ab arc42-Abschnitt 3) kommen.

Der HTML-Sanity Checker (HtmlSC, [hsc]) soll Autoren, die HTML als digitales Ausgabeformat erzeugen, bei der Prüfung von Querverweisen, Bild- und Dateiverweisen sowie sonstigen Referenzen (Hyperlinks) unterstützen.

1. Autoren schreiben in Formaten wie AsciiDoc[1], Markdown[2] oder anderen Sprachen, die per Generator in HTML konvertiert werden.
2. HtmlSC führt auf den generierten HTML-Seiten semantische Prüfungen aus und identifiziert beispielsweise fehlerhafte Querverweise oder fehlende Bilder.
3. Als Resultat erzeugt HtmlSC einen Prüfbericht, analog den bekannten Unit-Test-Reports, etwa von JUnit.

Der HTML-Sanity Checker (HtmlSC, [hsc]) überprüft HTML-Files auf semantische Fehler. Dazu gehören beispielsweise Prüfungen auf falsche Querverweise, fehlende Bilder, fehlende lokale Ressourcen, duplizierte Target-Links, defekte externe Links, fehlerhafte URIs (*Uniform Resource Identifier*), fehlende Alternativtexte für Bilder oder ähnliche mögliche Probleme in HTML-Dateien. Details finden Sie in Tabelle II.1.

Tabelle II.1 Funktionale Anforderungen an HtmlSC

ID	Anforderung	Erklärung
M-1	Lese HTML-Dateien	HtmlSC soll eine oder mehrere (konfigurierbare) HTML-Dateien als Eingabe lesen können.
M-2	Erzeuge HTML-Report	Die Prüfergebnisse von HtmlSC sollen in einem (JUnit-ähnlichen) HTML-Report zusammengefasst werden.
M-3	Verschiedene Benutzungsarten	HtmlSC soll sowohl über die Kommandozeile wie auch als Gradle-Plug-in genutzt werden können.
M-4	Konfigurierbarer Input	Namen der Eingabedateien sind konfigurierbar.
M-5	Notwendige Prüfungen	▪ Fehlende Bilder (missing images) ▪ Falsche Querverweise (broken internal links) ▪ Doppelte Verweisziele (duplicate link targets) ▪ Falsche Links (malformed links)
O-1	Sonstige Prüfungen	▪ Fehlende Alt-Attribute ▪ Fehlerhafte Image-Maps ▪ Unbenutzte Bilder (unused images) ▪ Falsche Querverweise (broken hyperlinks)
O-2	Vorschläge (Suggestions)	HtmlSC soll bei Fehlern Vorschläge unterbreiten, welche Werte korrekt sein könnten.

Legende: M = Muss, O = Optional (*nice-to-have*)

Als Ergebnis der Prüfungen erzeugt HtmlSC einen detaillierten Report, ein Beispiel finden Sie in Bild II.2.

[1] *http://asciidoctor.org/* Markup-Sprache, wird von vielen Open-Source-Projekten zur Dokumentation verwendet. Eines unserer favorisierten Mittel zur Architekturdokumentation – siehe auch Kapitel VI.5.1.

[2] *https://daringfireball.net/projects/markdown/* Eine einfache Auszeichnungssprache bzw. Konverter von Text nach HTML. Sehr einfache Syntax, weite Verbreitung bei Webautoren.

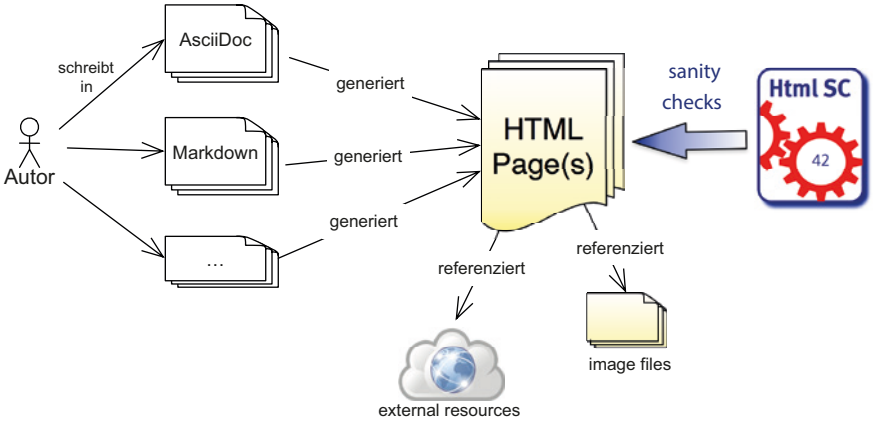


Bild II.1 Ziel von HtmlSC: semantische Prüfung von HTML-Seiten

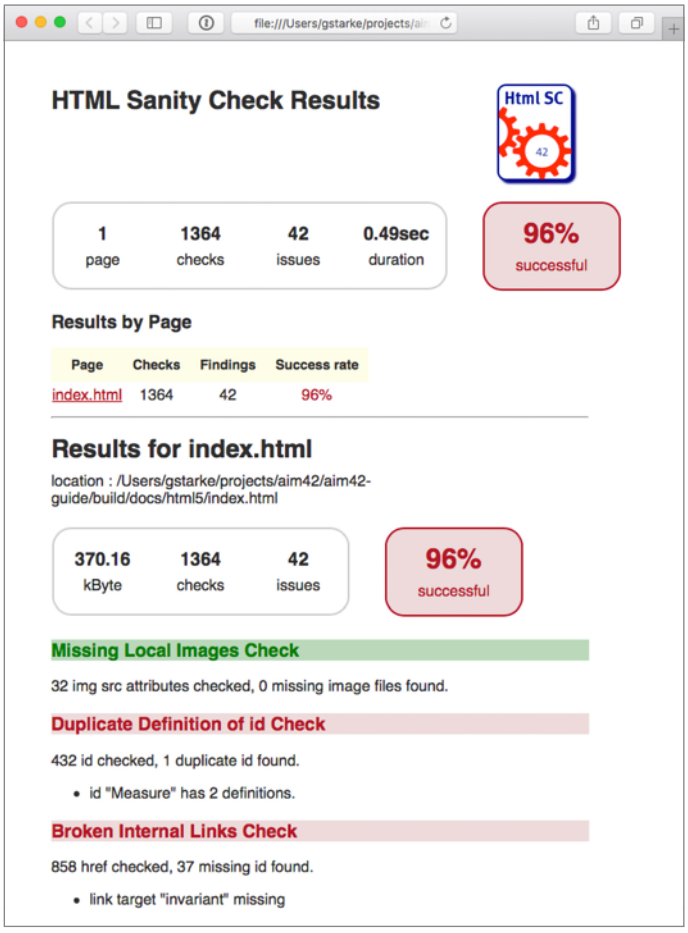


Bild II.2 Beispiel eines Reports von HtmlSC

1.2 Qualitätsziele

Inhalt und Motivation

Sie möchten die Qualitätsziele (auch als Architekturziele bezeichnet) Ihres Systems möglichst genau kennen, damit Sie Entwurfs- oder Technologieentscheidungen daran orientieren können.

Diese (tendenziell langfristigen) Qualitätsziele weichen in der Realität manchmal von den (tendenziell kurzfristigen) Zielen laufender Projekte ab. Beachten Sie den Unterschied.

Siehe auch arc42-Abschnitt 10 (Qualitätsbaum und Szenarien).

ID	Prio	Qualitätsziel	Erläuterung
Q-1	1	Korrektheit	HtmlSC findet alle in den Anforderungen spezifizierten HTML-Fehler.
Q-2	2	Sicherheit	Der Inhalt der zu prüfenden Seiten und Dokumente wird niemals verändert.
Q-3	3	Performanz	Die vollständige Prüfung eines 100-kB-HTML-Files wird in weniger als fünf Sekunden ausgeführt.
Q-4	4	Flexibilität	HtmlSC unterstützt verschiedene Prüfalgorithmen und Ausgabeformate. Die Erweiterung um einen weiteren Prüfalgorithmus soll innerhalb von 1–2 PT möglich sein.

1.3 Stakeholder

Inhalt und Motivation

Sie möchten einen Überblick über Ihre Stakeholder haben, d. h. über alle Personen, Rollen oder Organisationen, die die Architektur kennen sollten oder mit Architektur oder Code arbeiten.

Stellen Sie die konkrete Erwartungshaltung dieser Stakeholder bezüglich der Architektur und deren Dokumentation in Form einer Tabelle dar.

Anmerkung: Für unser winziges HtmlSC-Beispiel gibt es viel weniger Stakeholder als für *normal* umfangreiche Systeme ...

Rolle	Ziel	Erwartungshaltung
Autor digitaler Dokumente	Möchte Dokumente mit korrekten Querverweisen und Hyperlinks, frei von typischen HTML-Fehlern	Hat an Architektur des Systems kein Interesse, lediglich an von diesem Tool erzeugten Reports
arc42-Nutzer	Möchte ein kleines, aber praktisches Beispiel der Anwendung von arc42	Sucht eine Vorlage zum Abschreiben

2 Randbedingungen

Inhalt und Motivation

Sie möchten die Fesseln kennen, die Ihre Freiheiten bezüglich Entwurf, Implementierung oder Ihres Entwicklungsprozesses einschränken. Randbedingungen gelten manchmal organisations- oder firmenweit über die Grenzen einzelner Systeme hinweg.

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

ID	Beschreibung
RB-1	HtmlSC muss in Java oder Groovy entwickelt werden.
RB-2	HtmlSC muss mit dem Gradle (siehe [gradle]) Build-Werkzeug integrierbar sein.
RB-3	HtmlSC muss unter einer liberalen Open-Source-Lizenz stehen, etwa einer Creative-Commons Sharealike (siehe [CCSA]).
RB-4	Code und Dokumentation sollen auf einer der namhaften Open-Source-Plattformen (etwa: Github; siehe [github]), Bitbucket oder Sourceforge gehostet werden.
RB-5	Fehler und neue Anforderungen sollen mit einem Issue-Tracker (etwa Github-Issues) gepflegt werden.
RB-6	Eine erste Version (0.9.x) soll bis November 2015 veröffentlicht sein. Diese initiale Version kann auf einige der Prüfalgorithmen verzichten, beispielsweise die Prüfung externer Verweise.

3 Kontextabgrenzung

Inhalt und Motivation

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

An den Schnittstellen zwischen einem System und seiner Umgebung treten Fehler besonders gerne und häufig auf. Die Klärung der externen Schnittstellen gehört daher zu den kritischen Architekturaufgaben.

In arc42 differenzieren wir zwischen fachlichem und technischem Kontext.

3.1 Fachlicher Kontext

Inhalt und Motivation

Sie benötigen eine grobe Erklärung der fachlich relevanten Nachbarsysteme und Daten sowie der Nutzer und/oder Nutzerrollen.

Manchmal kann es zusätzlich helfen, die Formate oder Protokolle der Kommunikation mit Nachbarsystemen und der Umwelt zu zeigen.

Verwenden Sie ein Kontextdiagramm zusammen mit einer tabellarischen Erläuterung.

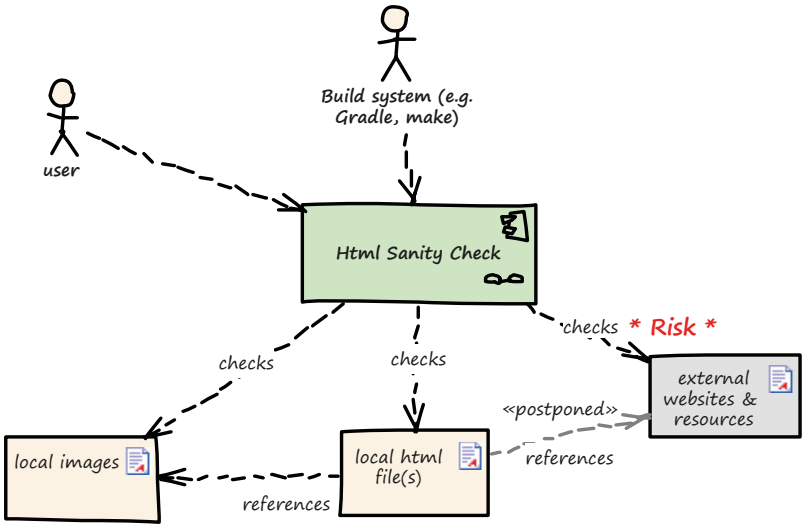


Bild II.3 Fachlicher Kontext

Nachbar	Beschreibung
user	Erstellt Dokumentation mit einem Werkzeug, das HTML als Ausgabeformat liefert. Erhält vom System die Prüfergebnisse.
local html files	HtmlSC liest und überprüft lokale HTML-Dateien.
local images	HtmlSC überprüft, ob referenzierte Bilder als Dateien (lokal) vorliegen bzw. ob Bilddateien überhaupt referenziert werden.
external websites & resources	HtmlSC kann zukünftig auch zur Prüfung externer Links verwendet werden. **Risiko:** Die Dauer der Zugriffe auf externe Websites hängt von Netzlatenz und der Reaktionszeit der externen Systeme ab und könnte ggfs. die Performanceanforderung (siehe Qualitätsziele oben, insbesondere Q-3) gefährden.

Schnittstelle zum Build-System, insbesondere Gradle

Die Schnittstelle von HtmlSC zum Gradle-Build-System ist in Abschnitt II.8.3 (Gradle-Plug-in-Konzept) beschrieben.