# Contents

Preface ................................................................. i
  Acknowledgements ..................................................... i
  Conventions ............................................................. v
  This is Just a Sample of the Full Book .............................. vi

I - Introduction ...................................................... 1
  I.1 What is arc42? ..................................................... 2
  I.2 Why this Book? .................................................... 5
  I.3 What this Book is Not ............................................. 5
  I.4 Overview of the Examples ....................................... 6

II. HTML Sanity Checking ............................................ 8
  II.1. Introduction and Goals ......................................... 9
  II.2 Constraints ..................................................... 14
  II.3 Context .......................................................... 15
  II.4 Solution Strategy ............................................... 18
  II.5 Building Block View ........................................... 19
  II.6 Runtime View .................................................. 22
  II.7 Deployment view ............................................... 25

III - Mass Market Customer Relationship Management ............. 30
  III.1. Introduction and Requirements .............................. 31

IV - biking2 ......................................................... 44
  IV.4 Solution Strategy ............................................... 45
  IV.8 Technical and Crosscutting Concepts .......................... 47

V - DokChess .......................................................... 50
Preface

Acknowledgements

Gernot

Long ago, on a winters’ day in 2004, I sat together with Peter Hruschka, a long-time friend of mine and discussed one\(^1\) of our mutual favorite subjects - structure and concepts of software systems.

We reflected about an issue we both encountered often within our work, independent of client, domain and technology: Developers know their way around implementation technologies, managers theirs’ around budgets and risk management. But when forced to communicate (or even document) the *architecture* of systems, they often started inventing their own specific ways of articulating structures, designs, concepts and decisions.

Peter talked about his experience in requirements engineering: He introduced me to a *template* for requirements, a pre-structured *cabinet* (or document) called VOLERE\(^2\) which contains placeholders for everything that might be important for a specific requirements document. When working on requirements, engineers therefore needn’t think long before they could *dump* their results in the right place - and others would be able to retrieve it later on... (as long as they knew about VOLERE’s structure). This requirements template had been in industry use since several years, there even was a *book available*\(^3\) on its usage and underlying methodology.

“If we only had a similar template for software architecture”, Peter complained, and continued “countless IT projects could save big time and money”... My developer-soul added a silent wish: “If this was great, it could even take the ugliness out of documentation”.

\(^1\)Peter and Gernot both share a passion for cooking too, but you probably wouldn’t share our sometimes exotic taste
\(^2\)http://volere.co.uk
\(^3\)http://www.volere.co.uk/book/mastering-the-requirements-process-getting-requirements-right
We both looked at each other, and within this second decided to create exactly this:
A template for software architecture documentation (and communication), that is
highly practical, allows for simple and efficient documentation, is usable for all kinds
of stakeholders and could facilitate software architecture documentation. And of
course, it had to be open source, completely free for organizations to use.

That’s how the arc42 journey started.

Since then, Peter and myself have used arc42 in dozens of different IT systems within
various domains. It has found significant acceptance within small, medium and large
organizations throughout the world. We wrote more many articles around it, taught
it to more than 1000 (!) IT professionals and included it in several of our software
architecture-related books.

Thanx Peter for starting this wild ride with me. And, of course, for your lectures on
cooking.

Thanx to my customers and clients - I learned an incredible lot by working together
with you on your complex, huge, hard, interesting and sometimes stressful problems.
Due to all these nondisclosure agreements I signed all my life, I’m not officially
allowed to mention you all by name.

Thanx to my wife Cheffe Uli and my kids Lynn and Per for allowing dad to (once
more) sit on the big red chair and ponder about another book project... You’re the
best, and I call myself incredibly lucky to have you!

Thanx to my parents, who, back in 1985, when the computer stuff was regarded
to be something between crime and witchcraft, they encouraged my to buy one (an
Apple-2, by the way) and didn’t even object when I wanted to study computer science
instead of something (by that time) more serious. You’re great!

Michael

I’ve met Gernot and Peter (Hruschka) as instructors on a training at the end of 2015.
The training was called “Mastering Software Architectures” and I learned an awful
lot from both of them, not only the knowledge they shared but how they both shared
it. By the end of the training I could call myself “Certified Professional for Software
Architecture”, but what I really took home was the wish structure, document and
communicate my own projects like Peter and Gernot proposed and that’s why the
current documentation of my pet project biking2 was created.
Since then I used arc42 based documentations several times: As well as for in-house products and also at projects and consultancy gigs. The best feedback was something along the lines: “Wow, now we’ve got an actual idea about what is going on in this module.” What helped that special project a lot was the fact that we set fully on Asciidoctor and the “Code as documentation and documentation as code”⁴ approach I described in depth in my blog.

So here’s to Gernot and Peter: Thanks for your inspiration and the idea for arc42.

StefanZ

Originally, DokChess is a case study from my German book⁵ on documenting software architecture. Gernot has encouraged me to write it after I had told him about the chess example on a conference in Munich in 2011. Thank you especially for that, Gernot! (besides many valuable discussions and good times since then in Cologne, Zurich ...)

Thanks to my embarc-colleague Harm Gnoyke for trying to save my miserable English. All mistakes are my fault, of course.

Ralf D. Müller

Quite a while ago, I discovered the arc42 template as MS Word document. It didn’t take long to see how useful it is and I started to use it in my projects. Soon, I discovered that MS Word wasn’t the best format for me as a developing architect. I started to experiment with various text-based formats like Markdown, AsciiDoc but also with Wikis. The JAX conference was the chance to exchange my ideas with Gernot. He told me that Jürgen Krey already created an AsciiDoc version of the arc42 template. We started to consider this template as the golden master (this is now five years ago, 2014) and tried to generate all other formats needed (at this time, mainly MS Word and Confluence) from this template. The arc42-generator⁶ was born, and a wonderful journey was about to start. The current peak of this journey is docToolchain⁷ - the evolution of the arc42-generator. Read more about its architecture in this book.

---

⁴https://info.michael-simons.eu/2018/12/05/documentation-as-code-code-as-documentation/
⁵http://www.swadok.de
⁶https://github.com/arc42/arc42-generator
⁷https://doctoolchain.github.io/docToolchain/
On this journey, I have met many people who helped me along this way - impossible to name all of them.

However my biggest “Thanx!” goes out to Gernot who always encouraged me to do my next step and helped me along the way.

Thank you, Peter and Gernot for pushing my architectural skills to the next level through their superb training workshops.

Last but not least, I have to thank my family for their patience while I spent too much time with my notebook!

**All of us**

Thanx to our thorough reviewers that helped us improve the examples, especially Jerry Preissler, Roland Schimmack and Markus Schmitz.
Conventions

Chapter and Section Numbering:

We use roman chapter numbers (I, II, III etc), so we can have the arabic numbers within chapters in alignment with the arc42 sections...

In the sections within chapters, we add the chapter-prefix only for the top-level sections. That leads to the following structure:

Chapter II: HtmlSC

II.1 Introduction and Goals
II.2 Constraints
II.3 Context
...

Chapter III: Mass Market CRM

III.1 Introduction and Goals
III.2 Constraints
III.3 Context
...

Explanations:

The first example (HTML Sanity Checking) contains short explanations on the arc42 sections, formatted like this one.

In this book, we keep these explanations to a bare minimum, as there are other books extensively covering arc42 background and foundations.
This is Just a Sample of the Full Book

Dear Readers,

thankx for downloading this sample. It contains chapter I, most of chapter II and some excerpts of other chapters, to give you an impression what to expect in the full book

Some internal references (hyperlinks) will not work in this sample ...
I - Introduction

This chapter explains the following topics

- I.1 What is arc42?
- I.2 Why this book?
- I.3 What this book is not!
- I.4 Overview of the examples
- I.5 Table of arc42 sections

This book contains several examples of Software Architecture Documentation based upon the practical, economical, well-established and systematic arc42 approach.

It shows how you can communicate about software architectures, but it does not show how to develop or implement systems!
I.1 What is arc42?

arc42 is a template for architecture documentation.
It answers the following two questions in a pragmatic way, but can be tailored to your specific needs:

- *What* should we document/communicate about our architecture?
- *How* should we document/communicate?

Figure I.1 gives you the big picture: It shows a (slightly simplified) overview of the structure of arc42.

Compare arc42 to a cabinet with drawers: the drawers are clearly marked with labels indicating the content of each drawer. arc42 contains 12 such drawers (a few more than you see in the picture above). The meaning of these arc42 drawers is easy to understand.
Therefore, arc42 offers you a simple and clear structure to document and communicate your (complex!) system. Starting with the goals and requirements for your system and its embedding into its environment you can provide the important stakeholder of your system with adequate information about the architecture.

arc42 is optimized for understandability and adequacy. It naturally guides you to explain any kind of architecture information or decision in an understandable and reproducible context.

Individuals and organizations using arc42 especially like two things about it:

1. the *understandability* of the documentation resulting from its standardized structure (the *drawers*) and
2. the *manageable effort* to create such documentation. We call it “painless documentation”.

**Why 42?**

You’re kidding, aren’t you? Ever heard of Douglas Adams, the (very British) and already deceased sci-fi writer... his novel “Hitchhikers Guide to The Galaxy” calls 42 the:

*Answer to the Ultimate Question of Life, The Universe, and Everything*⁹

arc42 aims at providing the answer to *everything* around your software architecture. (Yes, I know it’s a little pretentious, but we couldn’t think of a better name back in 2005.)

---
Where to get additional information

With the latest release (V7) of arc42 in January 2017, the project now has extensive online help and FAQ (frequently asked questions) available:

- The documentation site docs.arc42.org\(^{10}\) contains the full documentation plus many (more than 100) tips for efficient and effective usage.
- The FAQ site faq.arc42.org\(^{11}\) answers more than 100 typical questions.

\(^{10}\)https://docs.arc42.org
\(^{11}\)https://faq.arc42.org
I.2 Why this Book?

Examples are often better suited to show *how things can work* than lengthy explanations.

arc42 users have often asked for examples to complement the (quite extensive) conceptual documentation of the template that was, unfortunately, only available in German for several years.

There were a few approaches to illustrate how arc42 can be used in real-world applications, but those were (and still are) scattered around numerous sources, and not carefully curated.

After an incredibly successful (again, German only) experiment to publish one single example as a (very skinny) 40-page booklet we decided to publish a *collection of examples* on a modern publishing platform - so we can quickly react to user feedback and add further samples without any hassle.

I.3 What this Book is Not

In this book we focus on examples for arc42 to document and communicate software architectures. We only give a brief introduction to arc42 (in case you like more, see appendix B for details)

This book is *no* introduction to:

- Software architecture
- Architecture and Design Patterns
- Modeling, especially UML
- Chinese cooking (but probably you didn’t expect that here...)
1.4 Overview of the Examples

We encountered exciting software and system architectures during our professional lifes. Several of those would have made instructive, interesting and highly relevant examples - too bad they were declared confidential by their respective owners. Our nondisclosure agreements stated that we’re not allowed to even mention clients’ names without violating these contracts...

Therefore it has been quite difficult to find practical examples which we can freely write and talk about.

Now let’s get an overview of the examples that are waiting in the rest of this book...

HTML Sanity Checking

A tiny open source system to check HTML files for semantic errors, like broken cross references, missing images, unused images and similar stuff.

Gernot wrote the initial version of this system himself when working on a rather large (100+pages) documentation based upon the AsciiDoctor¹² markup language.

HtmlSanityCheck (or HtmlSC for short) is described in chapter II of this book.

Mass Market Customer Relationship Management

Gosh, what an awkward name - so let’s shorten it to MaMa-CRM at first: Mass market refers to consumers of severl kinds of enterprises, like health insurance, mobile telecommunication or large real-estate companies.

We are reasonably sure you know these 86×54mm plastic cards (experts call them ISO/IEC 7810¹³). Chances are you even own several of these cards. In case it contains your portrait photo, chances are high it was produced by a system like MaMa-CRM...

MaMa-CRM takes the burden of (usually paper-based) customer contacts from the organizations working in such mass markets. It has been initially build by an independent mid-sized data center to support the launch of the German (government-

¹²https://asciidoc.org
enforced) e-Health-Card - and later on used to support campaigns like telephone billing, electrical-power metering and similar stuff.

The architecture of MaMa-CRM is covered in chapter III.

**biking2**

biking2 or “Michis milage” is a web based application for tracking biking related activities. It allows the user to track the covered milage, collect GPS tracks of routes, convert them to different formats, track the location of the user and publish pictures of bike tours.

The architecture of biking2 is covered in chapter IV

**DokChess - a Chess Engine**

DokChess is a fully functional chess engine.

The architecture of DokChess is covered in chapter V

**docToolchain**

(docToolchain\(^{14}\) is a heavily used and highly practical implementation of the docs-as-code\(^{15}\) approach: The basic idea is to facilitate creation and maintenance of technical documentation.

The architecture of docToolchain is covered in chapter VI

**MiniMenu**

Surely one of the leanest architecture documentations you will ever encounter. Captures some design and implementation decisions for a tiny Mac-OS menu bar application.

The main goal is to show that arc42 even works for very small systems.

Skip to chapter VII to see for yourself.

\(^{14}\)https://doctoolchain.github.io/docToolchain

\(^{15}\)https://docs-as-co.de
II. HTML Sanity Checking

By Gernot Starke.

Convention for this example
At the beginning of each section you find short explanations, formatted in boxes like this.

The system documented here is a small open source tool hosted on Github¹⁶.

The full sourcecode is available - you might even configure your Gradle build to use this software. Just in case you're writing documentation based on Asciidoc, that would be a great idea!

But enough preamble. Let's get started...

¹⁶https://github.com/aim42/htmlSanityCheck
II.1. Introduction and Goals

Content and Motivation

This section shows the driving forces for architecturally relevant decisions and important use-cases or features, summarized in a few sentences. If possible, refer to existing requirements documentation.

The main goal of this section is enabling your stakeholders to understand the solution, which is detailed in arc42-sections 3 to 12.

HtmlSC supports authors creating digital formats by checking hyperlinks, images and similar resources.

1.1 Requirements Overview

Content and Motivation

You like to briefly explain important goals and requirements, use-cases or features of the system. If available, refer to existing requirements documentation.

Most important: Readers can understand the central tasks of the system, before they encounter the architecture of the system (starting with arc42-section 3).

The overall goal of HtmlSC is to create neat and clear reports, showing errors within HTML files. Below you find a sample report.
Sample Report

HtmlSanityCheck (HtmlSC) checks HTML for semantic errors, like broken links and missing images. It has been created to support authors who create HTML as output format.

1. Authors write in formats like AsciiDoc\(^\text{17}\), Markdown\(^\text{18}\) or other formats, which are transformed to HTML by the corresponding generators.
2. HtmlSC checks the generated HTML for broken links, missing images and other semantic issues.
3. HtmlSC creates a test report, similar to the well-known unit test report.

\(^{17}\)https://asciidoctor.org/docs/what-is-asciidoc/
\(^{18}\)https://www.daringfireball.net/projects/markdown/syntax
II. HTML Sanity Checking

**Basic Usage**

1. A user configurations the location (directory and file name) of one or several HTML file(s), and the corresponding images directory.
2. HtmlSC performs various checks on the HTML and
3. reports its results either on the console or as HTML report.

HtmlSC can run from the command line or as Gradle plugin.

**Basic Requirements**

<table>
<thead>
<tr>
<th>ID</th>
<th>Requirement</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>G-1</td>
<td>Check HTML for semantic errors</td>
<td>HtmlSC checks HTML files for semantic errors, like broken links.</td>
</tr>
<tr>
<td>G-2</td>
<td>Gradle and Maven Plugin</td>
<td>HtmlSC can be run/used as Gradle and Maven plugin.</td>
</tr>
<tr>
<td>G-3</td>
<td>Multiple input files</td>
<td>Configurable for a set of files, processed in a single <em>run</em>, HtmlSC produces a joint report.</td>
</tr>
</tbody>
</table>
II. HTML Sanity Checking

<table>
<thead>
<tr>
<th>ID</th>
<th>Requirement</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>G-4</td>
<td>Suggestions</td>
<td>When HtmlSC detects errors, it shall identify suggestions or alternatives that might repair the error.</td>
</tr>
<tr>
<td>G-5</td>
<td>Configurable</td>
<td>Several features of checks shall be configurable, especially input files/location, output directory, timeouts and status-code behavior for checking external links etc.</td>
</tr>
</tbody>
</table>

**Required Checks**

HtmlSC shall provide the following checks in HTML files:

<table>
<thead>
<tr>
<th>Check</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Missing images</td>
<td>Check all image tags if the referenced image files exist.</td>
</tr>
<tr>
<td>Broken internal links</td>
<td>Check all internal links from anchor-tags (‘href=’”#XYZ”) if the link targets “XYZ” are defined.</td>
</tr>
<tr>
<td>Missing local resources</td>
<td>Check if referenced files (e.g. css, js, pdf) are missing.</td>
</tr>
<tr>
<td>Duplicate link targets</td>
<td>Check all link targets (... id=”XYZ”) if the id’s (“XYZ”) are unique.</td>
</tr>
<tr>
<td>Malformed links</td>
<td>Check all links for syntactical correctness.</td>
</tr>
<tr>
<td>Illegal link targets</td>
<td>Check for malformed or illegal anchors (link targets).</td>
</tr>
<tr>
<td>Broken external links</td>
<td>Check external links for both syntax and availability.</td>
</tr>
<tr>
<td>Broken ImageMaps</td>
<td>Though ImageMaps are a rarely used HTML construct, HtmlSC shall identify the most common errors in their usage.</td>
</tr>
</tbody>
</table>

**1.2 Quality Goals**

**Content and Motivation**

You want to understand the quality goals (aka architecture goals), so you can align architecture and design decisions with these goals.

These (usually long term) quality goals diverge from the (usually short term) goals of development projects. Mind the difference! See also arc42-section 10.
II. HTML Sanity Checking

<table>
<thead>
<tr>
<th>Priority</th>
<th>Quality Goal</th>
<th>Scenario</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Correctness</td>
<td>Every broken internal link (cross reference) is found.</td>
</tr>
<tr>
<td>1</td>
<td>Correctness</td>
<td>Every potential semantic error is found and reported. In case of doubt(^\text{19}), report and let the user decide.</td>
</tr>
<tr>
<td>1</td>
<td>Safety</td>
<td>Content of the files to be checked is <em>never</em> altered.</td>
</tr>
<tr>
<td>2</td>
<td>Flexibility</td>
<td>Multiple checking algorithms, report formats and clients. At least Gradle and command-line have to be supported.</td>
</tr>
<tr>
<td>2</td>
<td>Correctness</td>
<td>Correctness of every checker is automatically tested for positive AND negative cases.</td>
</tr>
<tr>
<td>3</td>
<td>Performance</td>
<td>Check of 100kB html file performed under 10 secs (excluding Gradle startup)</td>
</tr>
</tbody>
</table>

1.3 Stakeholders

**Content and Motivation**

You want an overview of persons, roles or organizations that affect, are affected by or can contribute to the system and its architecture. Make the concrete expectations of these stakeholders with respect to the architecture and its documentation explicit. Collect these in a simple table.

Remark: For our simple HtmlSC example we have an extremely limited number of stakeholders, in real-life you will most likely have many more stakeholders!

<table>
<thead>
<tr>
<th>Role</th>
<th>Description</th>
<th>Goal, Intention</th>
</tr>
</thead>
<tbody>
<tr>
<td>Documentation author</td>
<td>writes documentation with HTML output</td>
<td>wants to check that the resulting document contains good links, image references.</td>
</tr>
<tr>
<td>arc42 user</td>
<td>uses arc42 for architecture documentation</td>
<td>wants a small but practical example of <em>how to apply arc42</em>.</td>
</tr>
<tr>
<td>software developer</td>
<td></td>
<td>wants an example of pragmatic architecture documentation</td>
</tr>
</tbody>
</table>

\(^{19}\)Especially when checking external links, the correctness of links depends on external factors, like network availability, latency or server configuration, where HtmlSC cannot always identify the root cause of potential problems.
II.2 Constraints

**Content and Motivation**

You want to know the constraints that restrict your freedom of design decisions or the development process. Such constraints are often imposed by organizations across several IT systems.

HtmlSC shall be:

- platform-independent and should run on the major operating systems (Windows(TM), Linux, and Mac-OS(TM))
- implemented in Java or Groovy
- integrated with the Gradle build tool
- runnable from the command line
- have minimal runtime and installation dependencies (a Java(TM) runtime may be required to run HtmlSC)
- developed under a liberal open-source license. In addition, all required dependencies/libraries shall be compatible with a CreativeCommons license.
II.3 Context

Content and Motivation

You want to know the boundaries and scope of the system to distinguish it from neighboring systems. The context identifies the systems relevant external interfaces.

3.1 Business Context

You want to identify all neighboring systems and the different kinds of (business) data or events that are exchanged between your system and its neighbors.
Neighbor | Description
--- | ---
user | documents software with toolchain that generates html. Wants to ensure that links within this HTML are valid.
build system | mostly Gradle²⁰
local HTML files | HtmlSC reads and parses local HTML files and performs sanity checks within those.
local image files | HtmlSC checks if linked images exist as (local) files.
external web resources | HtmlSC can be configured to optionally check for the existence of external web resources. **Risk:** Due to the nature of web systems and the involved remote network operations, this check might need significant time and might yield invalid results due to network and latency issues.

## 3.2 Deployment Context

You like to know about the technical or physical infrastructure of your system, together with physical channels or protocols.

The following diagram shows the participating computers (nodes) with their technical connections plus the major artifacts of HtmlSC, the hsc-plugin-binary.

²⁰[https://gradle.org](https://gradle.org)
## Deployment context

<table>
<thead>
<tr>
<th>Node / Artifact</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hsc-development</td>
<td>where development of HtmlSC takes place</td>
</tr>
<tr>
<td>hsc-plugin-binary</td>
<td>compiled and packaged version of HtmlSC including required dependencies.</td>
</tr>
<tr>
<td>artifact repository</td>
<td>A global public cloud repository for binary artifacts, similar to MavenCentral(^{21}), the Gradle Plugin Portal(^{22}) or similar. HtmlSC binaries are uploaded to this server.</td>
</tr>
<tr>
<td>hsc user computer</td>
<td>where arbitrary documentation takes place with html as output formats.</td>
</tr>
<tr>
<td>build.gradle</td>
<td>Gradle build script configuring (among other things) the HtmlSC plugin to perform the HTML checking.</td>
</tr>
</tbody>
</table>

For details see the deployment-view.

\(^{21}\)https://search.maven.org/
\(^{22}\)https://plugins.gradle.com
II.4 Solution Strategy

**Content and Motivation**

You need a brief summary and explanation of the fundamental solution ideas and strategies. These key ideas should be familiar to everyone involved in development and architecture.

Briefly explain how you achieve the most important quality requirements.

1. Implement HtmlSC mostly in the Groovy programming language and partially in Java with minimal external dependencies.
2. We wrap this implementation into a Gradle plugin, so it can be used within automated builds. Details are given in the [Gradle userguide]²³. (The Maven plugin is still under development).
3. Apply the [template-method-pattern]²⁴ to enable:
   - multiple checking algorithms. See the [concept for checking algorithms],
   - both HTML (file) and text (console) output. See the [reporting-concept].
   - For the Maven plugin, this might lead to problems.

---

²³ https://docs.gradle.org/current/userguide/userguide.html
²⁴ https://sourcemaking.com/design_patterns/template_method/
II.5 Building Block View

Content and Motivation

The building block view explains the static decomposition of the system into building blocks (modules, components, subsystems, packages...) and their relationships. It shows the overall structure of the source code.

This view is organized in a top-down hierarchy.

5.1 Whitebox HtmlSanityChecker

Rationale: We used functional decomposition to separate responsibilities:
• HSC Core shall encapsulate checking logic and HTML parsing/processing.
• Plugins and GraphicalUI encapsulate all usage aspects

Contained Blackboxes:

<table>
<thead>
<tr>
<th>Building block</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HSC Core</td>
<td>HTML parsing and sanity checking</td>
</tr>
<tr>
<td>HSC Gradle Plugin</td>
<td>Exposes HtmlSC via a standard Gradle plugin, as described in the <a href="https://docs.gradle.org/current/userguide/userguide.html">Gradle user guide</a>. Source: Package org.aim42.htmlsanitycheck, classes: HtmlSanityCheckPlugin and HtmlSanityCheckTask</td>
</tr>
<tr>
<td>NetUtil</td>
<td>package org.aim42.inet, checks for internet connectivity, configuration of http status codes</td>
</tr>
<tr>
<td>FileUtil</td>
<td>package org.aim42.filesystem, file extensions etc.</td>
</tr>
<tr>
<td>HSC Graphical UI</td>
<td>(planned, not implemented)</td>
</tr>
</tbody>
</table>
The full book contains a much more extensive description of the building blocks of HtmlSC...
II.6 Runtime View

**Content and Motivation**

The runtime view shows behavior, interactions and runtime dependencies of the building blocks in form of concrete scenarios.

It helps you to understand *how* the building blocks of your systems fulfill their respective tasks at runtime, and *how* they communicate/interact with each other at runtime.

### II.6.1 Execute all checks

A typical scenario within HtmlSC is the execution of *all* available checking algorithms on a set of HTML pages.
II. HTML Sanity Checking

Explanation:

1. User or build calls `htmlSanityCheck` build target.
2. Gradle (from within build) calls `sanityCheckHtml`
3. HSC configures input files and output directory
4. HSC creates an `AllChecksRunner` instance
5. gets all configured files into `allFiles`
6. (planned) get all available Checker classes based upon annotation
7. perform the checks, collecting the results
II.6.2 Report checking results

Reporting is done in the natural hierarchy of results (see the corresponding concept in section 8.2.1 for an example report).

1. per “run” (PerRunResults): date/time of this run, files checked, some configuration info, summary of results
2. per “page” (SinglePageResults):
3. create page result header with summary of page name and results
4. for each check performed on this page create a section with SingleCheckResults
5. per “single check on this page” report the results for this particular check
II.7 Deployment view

Content and motivation
You like to know the technical infrastructure where your system and its building blocks will be executed. That’s especially important if your software is distributed or deployed on several different machines, application-servers or containers.

Sometimes you need to know about different environments (like dev, test, production).

In large commercial or web systems, aspects like scalability, clustering, automatic deployment, firewalls and load-balancing play important roles - which we definitely don’t need for our small example.
HtmlSC deployment (for use with Gradle)

<table>
<thead>
<tr>
<th>Node / Artifact</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hsc plugin binary</td>
<td>Compiled version of HtmlSC, including required dependencies.</td>
</tr>
<tr>
<td>hsc-development</td>
<td>Development environment</td>
</tr>
<tr>
<td>hsc-development</td>
<td>Development environment</td>
</tr>
<tr>
<td>artifact repository</td>
<td>Global public <em>cloud</em> repository for binary artifacts, similar to mavenCentral(^26). HtmlSC binaries are uploaded to this server.</td>
</tr>
<tr>
<td>hsc user computer</td>
<td>Where documentation is created and compiled to HTML.</td>
</tr>
<tr>
<td>build.gradle</td>
<td>Gradle build script configuring (among other things) the HtmlSC plugin.</td>
</tr>
</tbody>
</table>

The three nodes (*computers*) shown in the diagram above are connected via Internet.

\(^{26}\)https://search.maven.org/
Prerequisites:

- HtmlSC developers need a Java development kit, Groovy, Gradle plus the JSoup HTML parser.
- HtmlSC users need a Java runtime (> 1.6) plus a build file named `build.gradle`. See below for a complete example.

**Example for `build.gradle`**

```java
buildscript {
    repositories {
        mavenLocal()
        maven {
            url "https://plugins.gradle.org/m2/"
        }
    }
    dependencies {
        // in case of mavenLocal(), the following line is valid:
        classpath(group: 'org.aim42',
        // in case of using the official Gradle plugin repository:
        //classpath (group: 'gradle.plugin.org.aim42',
        // name: 'htmlSanityCheck', version: '1.0.0-RC-3')
    }
    plugins {
        id 'org.asciidoctor.convert' version '1.5.8'
    }
    // ==== path definitions =====
    ext {
        srcDir = "$projectDir/src/docs/asciidoc"
        // location of images used in AsciiDoc documentation
    }
}
```
srcImagesPath = "$srcDir/images"

// (input for htmlSanityCheck)
htmlOutputPath = "$buildDir"

targetImagesPath = "$buildDir/images"
}

// ==== asciidoctor ========
apply plugin: 'org.asciidoctor.convert'

asciidoc {  
  outputDir = file(buildDir)  
  sourceDir = file(srcDir)  

  sources {  
    include "many-errors.adoc", "no-errors.adoc"  
  }

  attributes = [  
    doctype : 'book',  
    icons : 'font',  
    sectlink : true,  
    sectanchors: true  
  ]

  resources {  
    from(srcImagesPath) { include '**' }  
    into "./images"  
  }

// ==============================================================
apply plugin: 'org.aim42.htmlSanityCheck'

htmlSanityCheck {  
  // ensure asciidoctor->html runs first
// and images are copied to build directory

dependsOn asciidoctor

sourceDir = new File("${buildDir}/html5")

// files to check, in Set-notation
sourceDocuments = ["many-errors.html", "no-errors.html"]

// fail the build if any error is encountered
failOnErrors = false

// set the http connection timeout to 2 secs
httpConnectionTimeout = 2000

ignoreLocalHost = false
ignoreIPAddresses = false

defaultTasks 'htmlSanityCheck'
MaMa-CRM takes the burden of (usually paper-based) customer contacts from organizations working in mass markets, like insurance, credit-card providers, mobile telecommunication providers, energy and water providers or large real-estate companies (in MaMa speak these are called «Mandator»)

It has been initially ordered by an independent mid-sized data center to support the launch of the German (government-enforced) e-Health-Card - and later on used to support campaigns like telephone billing, electrical-power metering and similar stuff.

For every mandator, there is at least one completely independent MaMa-CRM instance running, which is specifically configured for its mandator and a campaign.

MaMa-CRM architecture documentation is quite heavy in the requirements part, describing several aspects of flexibility that triggered many central architecture decisions.

The team that built the system consisted of 7-10 persons working in 2-4 week iterations for about 15 months.

Me (Gernot Starke) had the honor to be part of that team in a responsible role. The original client allowed me to talk and write about the system without disclosing the company name. I was not allowed to use any of the original documentation or source code.

Thanx to Thorsten, Sven, Arno and a few unnamed other guys for great cooperation and a successful finish.
III.1 Introduction and Requirements

This document describes the MaMa-CRM platform, a software product family to coordinate processes between mass-market-service-providers and their clients. MaMa-CRM provides the foundation for individual systems, which have to be customized or configured for specific CRM domains.

MaMa-CRM has been built and is being operated by an innovative datacenter (let’s call it InDAC²⁷) – a company providing a variety of hosted software services to its clients. Within business process outsourcing initiatives, InDAC is operating specific instances of MaMa-CRM for its customers.

Let’s clarify MaMa-CRM’s scope with a few brief examples (you’ll find more details in the requirements overview):

- Telecommunication providers offer new or updated tariffs to their customers. These have several possibilities, like email, phone, letter or fax, to respond or issue support queries regarding such offers.
- Retail organizations send specific advertising or marketing material to certain parts of their customers. Again, these can react on several different communication channels. These kinds of advertisements are called target group marketing.
- Insurance companies propose modifications or enhancements of existing contracts to specific customer groups. Insured persons or organizations can react over different communication channels, like phone, mail, email or even in person. In case they accept this proposal, they have to submit a legally binding signature.

The following figure shows a generalized overview:

²⁷The real company behind the MaMa-CRM system prefers not to be disclosed here. The name InDAC (= Innovative data center) is therefore pure fantasy. The original system wasn’t called MaMa, but had a different name – I hope you don’t mind.
In that figure, the mandator represents an arbitrary company or organization serving mass market customers. MaMa, our system, can host CRM campaigns for many different mandators from different industries.

For an explanation of the terms campaign, activity, mandator and partner, please refer to the glossary in chapter 12.
1.1 Requirements Overview

MaMa-CRM controls customer relationship CRM campaigns, which InDAC (the contractor of MaMa) conducts for his mandators. InDACs’ mandators are companies like insurance or telco enterprises – which themselves offer services or products to mass market customers. These mandators outsource their CRM related business processes to InDAC.

MaMa-CRM shall support the following campaign types:

- Changes to tariffs or contracts for insurance companies, telecom and internet service providers and energy suppliers. You’ll find a detailed example for this process below.
- Processing of billing information for call-by-call telephone service providers: Mandator submits billing data to MaMa, which coordinates printing, submission and support for the resulting invoices. Management or handling of payments is out-of-scope for MaMa-CRM.
- Management of binary images for credit card and insurance companies, which print cardholders’ images onto the cards to prevent misuse. The German e-Health card belongs to this type.
- Meter reading for either energy or water providing companies or large-scale real-estate enterprises.

As MaMa is the foundation for a whole family of CRM systems, it shall be adaptable to a variety of different input and output interfaces and channels without any source code modification! This interface flexibility is detailed in section 1.1.2.

1.1.1 Campaign Example: Mobile Phone Contract Modification

An example shall clarify the complex business and technical requirements to MaMa. MoPho is a (hypothetical) mobile phone provider with a large customer base and a variety of different tariff options (flat fee, time-based, volume-based etc.). Some of these tariffs or tariff combinations are not marketable any longer (a shorthand for “MoPho does not earn its’ desired profit from them... - but that’s another story). Others are technically outdated.
In their ongoing effort to optimize their business, MoPho management decided to streamline their tariff landscape. Within a large campaign they contact every customer with outdated tariffs and offer upgrades to new and mostly more beneficial tariff options. Some customers get to pay less, but have to accept a longer contract duration. Others will have to pay a little more, but receive additional services, increased capacity, bandwidth or other benefits. In all cases where customers accept the new tariff offer, the formal contract between MoPho and the customer has to be updated, which requires a valid signature to be legally binding²⁸.

MoPho intends to inform certain customers via printed letters of the new tariff offerings. Customers can react via letter, fax, phone or in person at one of MoPho’s many sales outlets. As MoPho’s core competency are phone services, it doesn’t want to bother with printing letters, scanning and processing replies, answering phone inquiries and dealing with other out-of-scope activities. Therefore they employ MaMa-CRM as an all-around carefree solution. Look at MaMa’s approach to this scenario:

The 9 steps depicted in that scenario need some explanation:

1. MoPho selects pertained customers from its internal IT systems. We’re talking about a 30+ million customer base and approximately 10 million of those customers will be part of this campaign. MoPho exports their address-, contract- and tariff data and transmits these to MaMa-CRM. MaMa imports this customer data.

²⁸In some countries or for some contract types there might be simpler solutions than a written signature. Please ignore that for the moment.
2. MaMa forwards only the relevant parts of customer data to the print service provider (a «Partner»). This company creates personalized letters from address and tariff data, prints those letters and finally delivers them to the postal service (which ensures the letters finally end up in customers’ mailboxes).

3. MaMa now informs the call center (also a «Partner») participating in this campaign, so they can prepare for the customers’ reaction.

4. In the meantime the print service has delivered the letters to the postal service, which delivers letters to the customers. This is more difficult than it sounds: 1-5% of addresses tend to change within 12 months, depending on the customer base. Some customers refuse to accept marketing letters. The postal service informs MaMa about all the problematic cases (address unknown, forwarding request etc.), so MaMa can decide on further activities.

5. The customers decide what to do with the letter: There’s a multitude of options here:
   - Customer fills out the enclosed reply form and signs it.
   - Customer fills out the reply form, but forgets to sign it.
   - Customer does not understand this offer and sends written inquiry by letter.
   - Customer inquires by phone call.
   - Customer ignores letter and does not react at all.
   - There are additional special cases (customer is not contractually capable, has a custodian or legal guardian, is under-age, deceased or temporarily unavailable...).

6. Let’s assume the customer accepts and sends the letter back via postal service. These letters will be forwarded to the scan service provider (again, a «Partner»), which scans them and performs optical character recognition.

7. MaMa imports the scan-results from the scan service provider. Again, several cases are possible.

8. MaMa distinguishes between all possible cases and takes appropriate actions:
   - The form is completely filled and signed.
   - The form is completely filled, but customer forgot to sign.
   - Customer signed, but forgot to fill other important fields in the form...
   - If the form is not returned within an appropriate period of time, MaMa might decide to resend the letter, maybe even with a different wording, layout or even a different contractual offer.
Surely you can imagine several other possible options...

9. Finally, for every customer who accepted the changed tariff agreement, MaMa sends back the appropriate information, so the mandator MoPho can internally change the corresponding master data, namely contracts and tariffs and take all other required actions (activities in MaMa).

For the phone service provider MoPho, the primary advantage is the centralized interface to all customer interaction: All processes required to conduct this crucial campaign are handled by MaMa-CRM, without intervention by MoPho. Very practical for MoPho is the nearly unlimited flexibility of MaMa-CRM to import and export a variety of different data formats. This flexibility enables MaMa-CRM to easily incorporate new campaign partners (like a second scan service provider, an additional email-service provider or web hosting partner).

1.1.2 Campaign Configuration

MaMa provides the software foundation for campaign instances that have to be extensively configured to operate for a specific mandator with a number of specific partners.

The following section gives an overview of such a configuration for the MoPho campaign described in the previous section.

1. Configure client master data

Define and configure required data attributes and additional classes, usually this information is completely provided by the mandator.

This data (e.g. contract-, tariff-, offer- and other business-specific entities or attributes) are modeled as extensions of the MaMa base model in UML.

In the MoPho telecommunications example, this data includes:

- Existing tariff
- List of potential new tariffs
- Contract number
- Validity period of existing tariff
- Validity period of new tariff
2. Configure communication channels

- Define and exchange security credentials (passwords, certificates, user-ids).
- Determine contact address, especially for handling technical and process issues.

3. Define campaign meta data

- start- and end dates,
- target dates for specific activities,
- reply address for letters, reply email address,
- phone number for call center,

4. Define campaign activities

Campaign activities\textsuperscript{29} can be either input-, output- and MaMa-internal activities.

- Output-activities \textit{export} data to partners or the mandator, e.g.:  
  - PrintLetter  
  - DataToCallCenter  
  - DataToScanServiceProvider  
  - ResultsToMandator  
  - ProductionOrderToCardProvider
- Input-activities \textit{import} data provided by the mandator or partners, e.g.:  
  - MasterDataFromMandatorImport  
  - ScanDataImport  
  - CallCenterImport  
  - SalesOutletImport
- Internal activities define data maintenance operations, e.g.:  
  - Delete customer data 60 days after completion.
  - Delete all campaign data 120 days after campaign termination.

5. Model campaign flows

What is the required flow of activities, what needs to happen under which conditions (including plausibility checks).

\textsuperscript{29}See the MaMa-CRM Glossary for definitions of these term.
1.1.3 Activities Subject to Charge

Some activities result in charges, for example:

- sending letters via postal service results in postal charges
- producing e-Health cards results in card production charges

As we are talking about potentially millions of clients respectively activities, these charges will amount to a significant flow of money:

MaMa-CRM does NOT handle billing and payment processes.

Instead these are handled parallel to MaMa-campaigns by InDACs’ finance department.

1.1.4 Additional Requirements

Status and Financial Reporting

MaMa has to campaign reports containing all campaign activities and their results (e.g. how many letters were printed, how many calls from clients were processes in the call center, how many clients sent their return letter etc.)

It’s especially important to report about activities subject to a charge (aka having financial consequences). Although MaMa-CRM is not responsible for the resulting billing and payment processes, mandators and partners need reliable and trustworthy reports about such activities.

MaMa shall offer adequate standard reports covering these cases.

Client Data Belongs To Mandators

MaMa-CRM follows a very strict data protection policy: All data processes in campaigns is somehow related to clients and needs to be kept strictly confidential. Mandators always remain owners of all such data, from whatever partner it is transferred to MaMa by whatever activities.

---

In the real system, billing and payment was usually processed by the mandators (!) - but these 3-party processes are not significant for the software architecture, and therefore left out of this documentation.
After a campaign officially ends, all (!) data including backups, configurations, data and metadata needs to be deleted\textsuperscript{31}.

\textsuperscript{31}Effectively this requirement forbade the use of the well-known version control systems “subversion” (svn), as by development time of MaMa-CRM, svn could not reliably purge files from version history! Therefore a campaign-spanning svn repository was not allowed.
1.2 Quality Goals

1.2.1 Flexibility First

Simply stated, MaMa is a data hub with flexibility in several aspects: It imports data (aspect 1) coming over various transmission channels (aspect 2) executes a set of specific business rules (aspect 3) to determine further activities to be performed with this data. Such activities consist of data exports (aspect 4) to certain business partners (like print-, scan- or fax service providers, call centers and the like).

MaMa has to be flexible concerning all these aspects.

**Aspect 1: Flexibility in Data Structure and Format**

MaMa always deals with variants of client data - the most simple version could look like an instance of the following class definition:

```java
public class Client
{
    String primaryKey;
    String lastName;
    String firstName;
    Date birthDate;
    String title;
    Address postalAddress;
}
```

MaMa needs to import such data from various campaign partners. These partners operates their own IT systems with their own data formats. They will export the
affected data records in some format (see below) of their choice. MaMa needs to handle many such formats:

- Comma separated value (CSV): attributes are separated by a (configurable) delimiter, e.g.: “id01”, “starke”, “gernot”, “23-jul-1963”, “willi-lauf”, “D-50858”, “cologne”, “germany”
- Field order and delimiter can vary.
- Elements could be enclosed in either “ or ‘ or even «».
- Fix length formats, where every data record has a fixed length. Shorter entries are padded with empty strings or zeroes.
- XML dialects, either conforming to an appropriate schema or DTD.

Regardless of format, data elements need to comply to validation or plausibility rules. Good news: No serialized objects from programming language runtimes (e.g. from a java virtual machine) have to be processed.

**Aspect 2: Flexibility in Transmission Formats and Modes**

MaMa needs to handle the following transmission-related aspects:

- Standard transmission protocols (ftp, sftp, http, https) as both client and server
- Compressed,
- Encrypted
- Creation of checksums, at least with counters, MD5 and SHA-1
- Mandator- or campaign specific credentials for secure transmission
- Transmission metadata, i.e.: Count the number of successful transmissions and send the number of the last successful transmissions as a prefix to the next transmission.

**Aspect 3: Flexibility in Business / Campaign Rules**

MaMa needs to support several categories of campaign rules (sometimes called business rules)

- What are required and possible activities in this campaign?
• In what order shall these activities be executed?
• Certain types of activities (especially data import activities) are sometimes trig-
gerated by «Partner» organizations. What kind of event triggers what activities?
• How often to remind clients/customers by letter?
• When to call clients/customers by phone?
• What data is mandatory from the customer and what is optional?
• How long to keep data records, when to delete or archive data records?
• Whom to contact when certain (data) conditions arise?

Aspect 4: Flexibility in Data Export

Now that MaMa has received and processed data (by import, transmission and rule
processing), it needs to send data to the campaign partners. Analogous to data import,
these receiving partners have very specific requirements concerning data formats.

It is pretty common for MaMa to receive data from a partner company or mandator
in CSV format, import it into its own relational database, process it record-by-record,
export some records in XML to one partner and some other records in a fix format
to another partner.

1.2.2 Quality Goals (Scenarios)

Prio 1: Flexibility

MaMa-CRM can:

• import and export (configurable) CSV and fix-record-length data.
• import and export data via ftp, sftp, http and https, as client and server.
• handle compressed data. Compression algorithm is configurable per activity,
standard lossless have to be supported.
• handle encrypted data. Security credentials need to be exchanged between
parties prior to campaigns during campaign setup. PGP or derivatives are
recommended.
• handle hashes or similar transmission metadata, configurable per campaign.

For all these topics, InDAC administrators can fully configure all campaign-specific
aspects within one workday.
**Prio 2: Security**

MaMa will keep all its client and campaign related data safe. It shall never be possible for campaign-external parties to access data or metadata.

InDAC Administrators need to sign nondisclosure contracts to be allowed to administer campaign data. This organizational issue is out-of-scope for MaMa.

**Prio 3: Runtime Performance**

MaMa can fully process 250,000 scanned return letters within 24 hours.

**1.2.3 Non-Goals (Out of Scope)**

- MaMa shall be exclusively operated by InDAC. It shall not evolve into a marketable product.
- MaMa shall not replace conventional CRM systems, like Siebel™, salesforce.com™ or others.
- MaMa does not handle billing and payment of any charges generated by its activities, e.g. sending letters or others. See section 1.1.3

In the full book, MaMa-CRM is completely documented. Especially the architecture decisions and solution concepts that support the enormous flexibility may be worth a read :-)  

[https://leanpub.com/arc42byexample](https://leanpub.com/arc42byexample)
biking2 or “Michis milage” is primarily a web based application for tracking biking related activities. It allows the user to track the covered milage, collect GPS tracks of routes, convert them to different formats, track the location of the user and publish pictures of bike tours.

The secondary goal if this application is to have a topic to experiment with various technologies, for example Spring Boot on the server side and AngularJS, JavaFX and others on the client side.

As such biking2 has been around since 2009 and in it’s current Java based form since 2014. Defining production as full filling the primary goal it’s been in production ever since.

The project is published under Apache License on GitHub, use it however you like. Though I’ve been blogging regularly about this pet project, the documentation in its current form was created after I met Gernot and Peter at an awesome workshop in Munich.

³²https://github.com/michael-simons/biking2
IV.4 Solution Strategy

At the core of biking2 is a simple yet powerful domain model based on a few entities of which a “Bike” and it’s “Milage” are the most important.

Although data centric, the application resigns from using too much SQL for creating reports, summaries and such but tries to achieve that with new Java 8 features around streams, lambdas and map/reduce functions.

Building the application with Spring Boot is an obvious choice as one of the main quality goals is learning about it. But furthermore using Spring Boot as a “framework” for Spring Framework allows concentration on the business logic. On the one hand there is no need to understand a complex XML configuration and on the other hand all building blocks are still put together using dependency injection.

Regarding dependency injection and testability: All injection should be done via constructor injection, setter injection is only allowed when there’s no other technical way. This way units under tests can only be correctly instantiated. Otherwise one tends to forget collaborators or even worse: 20 injected attributes may not hurt, but a constructor with 20 parameters will. This hopefully prevents centralized “god classes” that control pretty much every other aspect of the application.

Spring Boot applications can be packaged as single, “fat jar” files. Since Spring Boot 1.3 those files contain a startup script and can be directly linked to /etc/init.d on a standard Linux systems which serves TC4.

Interoperability will be achieved by using JSON over simple http protocol for the main API. Security is not the main focus of this application. It should be secure enough to prevent others from tempering with the data, confidentiality is not a main concern (read: passwords can be transmitted in plain over http).

The internal single page application shall be implemented using AngularJS. The deployable artifact will contain this application so there is no need for hosting a second webserver for the static files.

For real time tracking the MQTT protocol will be used which is part of Apache ActiveMQ, supported out of the box by Spring Messaging.

Graphing should not be implemented here but instead the Highcharts\(^3\) library should

[^3]: https://www.highcharts.com
be used. The configuration for all charts should be computed server side.
IV.8 Technical and Crosscutting Concepts

Domain Models

biking2 is a datacentric application, therefore everything is based around this Entity Relationship Diagram (ER-Diagram):
Entities in biking2

Those entities are manifested as tables:
### Tables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bikes</td>
<td>Stores the bikes. Contains dates when the bike was bought and decommissioned, an optional link, color for the charts and also an auditing column when a row was created.</td>
</tr>
<tr>
<td>milages</td>
<td>Stores milages for a bike (when and how much).</td>
</tr>
<tr>
<td>tracks</td>
<td>Stores GPS tracks recorded and uploaded with an optional description. For each day the track names must be unique. The columns minlat, minlon, maxlat and maxlon store the encapsulating rectangle for the track. The type column is constrained to “biking” and “running”.</td>
</tr>
<tr>
<td>assorted_trips</td>
<td>Stores a date and a distance on that day. Multiple distances per day are allowed.</td>
</tr>
<tr>
<td>locations</td>
<td>Stores arbitrary locations (latitude and longitude based) for given timestamp with an optional description.</td>
</tr>
<tr>
<td>biking_pictures</td>
<td>Stores pictures collected from Daily Fratze together with their original date of publication, their unique external id and a link to the page the picture originally appeared.</td>
</tr>
<tr>
<td>gallery_pictures</td>
<td>Stores all pictures uploaded by the user with a description and the date the picture was taken. The filename column contains a single, computed filename without path information.</td>
</tr>
</tbody>
</table>

In the full book⁴, each drawer of the arc42 cabinet is filled.

You’ll read more about the requirements and the constraints that have been put on the application as well as the full context.

Especially the full chapters 8 and 9 contains several interesting architecture decisions.

⁴https://leanpub.com/arc42byexample
This chapter describes the architecture of the chess program DokChess. I originally created it as an example for presentations and trainings on software architecture and design. Later on I implemented it in Java and refined it for a German book on documenting software architectures (see www.swadok.de\(^{34}\)). The source code is available on GitHub, more information can be found on www.dokchess.de\(^{35}\).

With the following architectural overview you are able to understand the significant design decisions of DokChess. It shows the requirements relevant for the architecture, basic solution approaches, the structure of the software and the interaction of key elements. The outline of the content follows the arc42 template.

Target audience of this architectural overview are primarily software architects seeking suggestions and examples on how to document architecture appropriately.

Developers who plan to write a chess program on their own receive valuable tips, and learn something about software architecture en passant.

The full book\(^a\) contains the documentation for this chess engine, optimized for understandability (not for execution speed or playing strength) Surely worth a

---

\(^{34}\)http://www.swadok.de  
\(^{35}\)http://www.dokchess.de
read!!

https://leanpub.com/arc42byexample
The Authors

Gernot Starke

Dr. Gernot Starke (innoQ\textsuperscript{36} Fellow) is co-founder and longstanding user of the (open source) arc42\textsuperscript{37} documentation template. For more than 20 years he works as software architect, coach and consultant, conquering the challenges of creating effective software architectures for clients from various industries.

In 2008 Gernot co-founded the International Software Architecture Qualification Board (iSAQB e.V.\textsuperscript{38}) and since then supports it as an active member. 2014 he founded the (open source) Architecture Improvement Method aim42\textsuperscript{39}.

Gernot has authored several (German) books on software architecture and related topics.

Gernot studied computer science at the Institute of Technology in Aachen (RWTH Aachen) and finished with a Diploma. He then worked as a developer and consultant for smaller software companies, before coming back to university for international research on methodical software engineering. 1995 he received his PhD from Johannes Kepler University of Linz, Austria (Prof. Gerhard Chroust for his thesis on “Software Process Modeling”.

He then joined Schumann AG in Cologne and did consulting and development work for several years. He became technical director of the “Object Reality Center”, a joint-venture of Sun Microsystems and Schumann Consulting
AG and lead the first European Java Project (the Janatol project for Hypobank in Munich).

Since then he has consulted and coached numerous clients from various domains, mainly finance, insurance, telecommunication, logistics, automotive and industry on topics around software engineering, software development and development process organization.

Gernot was an early adopter of the agile movement and has successfully worked as Scrum master in agile projects.

He lives in Cologne with his wife (Cheffe Uli) and his two (nearly grown-up) kids, two cats and a few Macs.

Email Gernot Starke⁴⁰ or contact him via Twitter @gernotstarke⁴¹.

---

⁴⁰mailto:gernot.starke@innoq.com?subject=arc42%20By%20Example
⁴¹https://twitter.com/gernotstarke
Michael Simons

Michael Simons is a father of two, husband, geek, programmer and passionate cyclist.

Michael took his apprenticeship at the FZ Jülich\(^{42}\) and studies at FH Aachen: Campus Jülich. He is a PRINCE2 ® Registered Practitioner and sometimes torn between the roles of an architect and project manager. In 2018, he was announced a Java Champion.

Nowadays, Michael works as a Senior Software engineer at Neo4j\(^{43}\) in the Spring Data team. In a previous life he worked at ENERKO INFORMATIK, an Aachen based company dealing with GIS system and has a background focussed on geographic information systems for utilities and price calculation at the energy market. In his brief time at INNOQ he helped customers modernize their application systems. Michael is known for having a certain passion for SQL and Spring. You can buy Michaels book about modern software development with Spring Boot here: here\(^{44}\)

Michael is a dedicated blogger and engaged in various open source projects. You’ll find his stuff starting at michael-simons.eu\(^{45}\).

You can reach Michael via email\(^{46}\) or on Twitter as @rotnroll666\(^{47}\).

---

\(^{42}\)https://www.fz-juelich.de/portal/DE/Home/home_node.html
\(^{43}\)https://neo4j.com
\(^{44}\)http://springbootbuch.de
\(^{45}\)http://michael-simons.eu
\(^{46}\)mailto:michael@simons.ac
\(^{47}\)https://twitter.com/rotnroll666
Stefan Zörner

From the Bayer AG via IBM and oose to embarc. Stefan Zörner⁴⁸ represents twenty years of experience in IT and still looks forward excitedly. He supports clients in solving architecture and implementation issues. In vivid workshops he demonstrates practical design tools and spreads enthusiasm for real architecture work.

About embarc

Hamburg-based embarc⁴⁹ is your first choice for architectural topics. We are an independent consulting company acting open-minded to find the best solution for you. Our customers trust us because we actively shape their systems, technologies and platforms as needed. This serves the long-term goal to gain optimum value from your IT investments.

Our team of consultants are generally recognized experts with hands-on knowledge from manifold international projects. Technical and professional excellence combined with pragmatic approaches are the main characteristics of our architectural and engineering work.

Document your software architecture decisions and principles – effectively and comprehensibly!

You want to build a strong foundation for architecture documentation across your organisation for different kinds of projects? Do you want to use a standardized template like e.g. arc42?

Architecture documentation isn’t what you might think of first: baroque diagrams, bulky concepts, etc. Start your architectural overview with only a few basic ‘ingredients’. Our cheat sheet⁵⁰ gives you a quick and convenient introduction.

⁴⁸mailto:stefan.zoerner@embarc.de
⁴⁹https://www.embarc.de
⁵⁰https://www.embarc.de/architektur-spicker/
On top of that we support you with specific coaching, workshops or kickoff presentations for documentation topics. Key benefits for your team:

- common understanding of architecture decisions / concepts / solutions
- appropriate communication to different target audiences
- a proper basis for discussions / reflection / architecture reviews
- get new team members efficiently up-to-speed
- learn from our best practices from many projects

Please find more information on our [website](https://www.embarc.de/themen/architekturdokumentation/).
Ralf D. Müller

More than twenty years of experience in web development have shaped Ralf D. Müller’s experience and thinking. And as trained Six Sigma Black Belt, the productivity of the whole is most important for him. That’s why he dedicated much of his spare time to make the process to document a software project even easier. No question that the arc42 template is at the heart of this process. In addition, docToolchain helps development and architecture teams to implement the Docs-as-Code approach.

Contacting the Authors

In case you have questions or suggestions concerning the examples, arc42 or software architecture stuff in general, please let us know.

Gernot Starke
Just email Gernot Starke or contact him via Twitter @gernotstarke.

Michael Simons
You can reach Michael via email or on Twitter as @rotnroll666.

Stefan Zörner
Just email Stefan Zörner or contact him via Twitter: @StefanZoerner.

Ralf D. Müller
Just email Ralf D. Müller or contact him via Twitter: @RalfDMueller.

52mailto:ralf.d.mueller@docs-as-co.de
53https://github.com/docToolchain/docToolchain
54https://docs-as-co.de
55mailto:gernot.starke@innoq.com?subject=arc42%20By%20Example
56https://twitter.com/gernotstarke
57mailto:misi@planet-punk.de?subject=arc42%20By%20Example
58https://twitter.com/rotnroll666
59mailto:stefan.zoerner@embarc.de
60https://twitter.com/StefanZoerner
61mailto:ralf.d.mueller@docs-as-co.de
62https://twitter.com/RalfDMueller
Further Reading

If you liked the way we explained, communicated and documented software architectures, you might want to know more about arc42.

Communicating Software Architectures

This practical guide shows how you can effectively apply the practical and well-proven arc42 template to design, develop and document your software architecture.

It contains more than 200 practical tips how to improve your architecture communication and documentation:

- immediately actionable tips
- arc42 for practical software development
- effective communication and documentation of software architectures
- arc42 to construct, design and implement new systems
- arc42 to document existing system
- arc42 for { large | medium | small } systems
Further Reading

- tools for arc42: wikis, asciidoc, modeling tools an others
- frequently asked questions around arc42

You find it on https://leanpub.com/arc42inpractice⁶³.

⁶³https://leanpub.com/arc42inpractice
**arc42 FAQ (Frequently Asked Questions)**

More than 150 questions around arc42 topics - available for free as eBook and online\(^6^4\).

NEW: FAQ has now a searchable public (free!) website at [https://faq.arc42.org](https://faq.arc42.org)\(^6^5\).

Contains questions in the following categories:

- General: Cost, license, contributing
- Methodology: minimal amount of documentation, where-does-what-info-belong, notations, UML
- arc42 sections: quality requirements, context, building blocks, runtime scenarios, deployment, concepts etc.
- Modelling: UML, diagrams, interfaces, ports, understandability, consistency, clarity
- arc42 and Agility: Scrum, Kanban, definition-of-done, minimal, lean, economical documentation
- Tools for arc42
- Versioning and variants: versioning documents, variants of systems
- Traceability: tracing requirements to solution decisions and vice-versa

\(^6^4\)[https://leanpub.com/arc42-faq/read]  
\(^6^5\)[https://faq.arc42.org]
- Management: very large systems, standardization, governance, checklists, access-rights
- Customizing arc42: tailoring and customizing, known adaptations of arc42
Glossary of Software Architecture Terminology

An extensive (and free!) collection of terms used in software architecture. Created and maintained by the iSAQB e.V., the association for software architecture education.

In addition to definitions this eBook also contains translation tables, currently for German and English.

You find it on https://leanpub.com/isaqbglossary⁶⁶.

⁶⁶https://leanpub.com/isaqbglossary
(German:) arc42 in Aktion

The German original of the previously mentioned eBook... for those who both understand German and like to hold books in their hands...

You get it at your local bookstore or online.67

---

67https://www.amazon.de/arc42-Aktion-Praktische-Tipps-Architekturdokumentation/dp/3446448012