Photo by AndRo Poplis on Unsplash

INNOQ

Gernot Starke

Fellow

arc42

The Smart Way to Describe Your Architecture

Thanx to Michael Simons

Twitter @rotnroll666

Photo by Jonada Molla on Unsplash

Missing in the picture

Technology

(internal) Structure

Interfaces

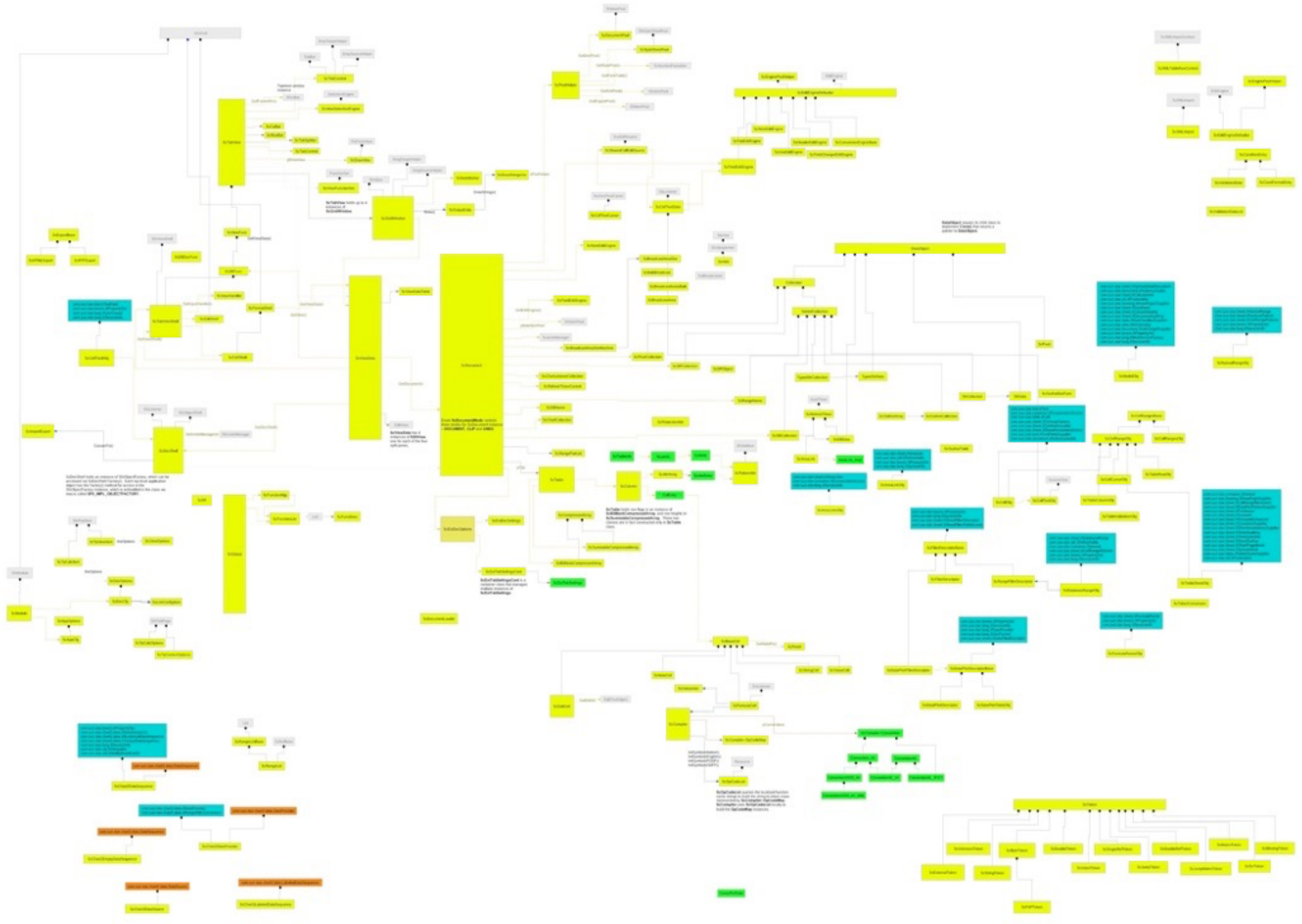
Reasons

Requirements

Architecture is (much) more than a diagram ...

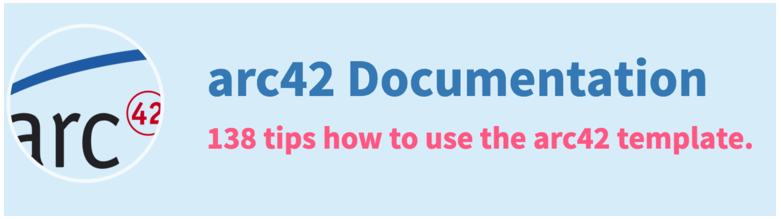
and more
than code ...

http://www.flickr.com/photos/mistressf/7373500025/sizes/l/



arc42 Template

Repository for (architecturally) relevant information


arc42 Documentation

138 tips how to use the arc42 template.

- Free and open source: arc42.org
- Documentation https://docs.arc42.org

Results of Architecture Work...

arc42

1. Introduction and Goals
1.1 Requirements Overview
1.2 Quality Goals
1.3 Stakeholders

2. Constrainst
2.1 Technical Constraints
2.2 Organisational Constraints
2.3 Conventions

3. Context and Scope
3.1 Business Context
3.2 Technical Context

4. Solution Strategy

5. Building Block View
5.1 Level 1
5.2 Level 2
….

6. Runtime View
6.1 Runtime Scenario 1
6.2 Runtime Scenario 2
….

7.Deployment View
7.1 Infrastructure Level 1
7.2 Infrastructure Level 2
….

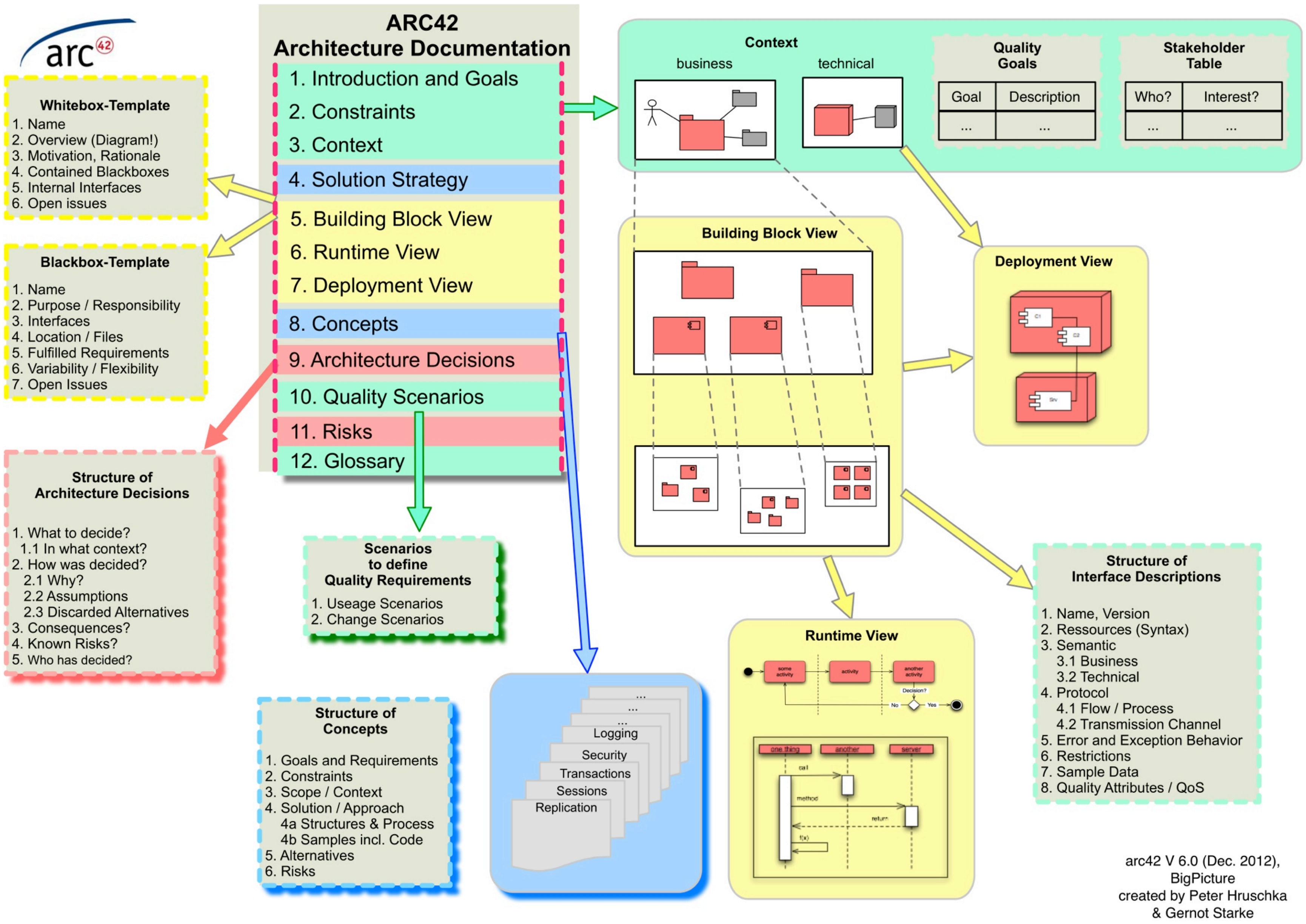
8. Crosscutting Concepts
8.1 Domain Structures and Models
8.2 Architectural/Design Patterns
8.3 Under the hood
8.4 User Experience
….

9. Architectural Decisions
9.1 Decision 1
9.2 Decision 2
….

10. Quality Requirements
10.1 Quality Tree
10.2 Quality Scenarios

11. Risks and technical debts

12. Glossary



ARC42 Architecture Documentation

1. Introduction and Goals
2. Constraints
3. Context
4. Solution Strategy
5. Building Block View
6. Runtime View
7. Deployment View
8. Concepts
9. Architecture Decisions
10. Quality Scenarios
11. Risks
12. Glossary

Whitebox-Template
1. Name
2. Overview (Diagram!)
3. Motivation, Rationale
4. Contained Blackboxes
5. Internal Interfaces
6. Open issues

Blackbox-Template
1. Name
2. Purpose / Responsibility
3. Interfaces
4. Location / Files
5. Fulfilled Requirements
6. Variability / Flexibility
7. Open Issues

Structure of Architecture Decisions
1. What to decide?
   1.1 In what context?
2. How was decided?
   2.1 Why?
   2.2 Assumptions
   2.3 Discarded Alternatives
3. Consequences?
4. Known Risks?
5. Who has decided?

Scenarios to define Quality Requirements
1. Useage Scenarios
2. Change Scenarios

Structure of Concepts
1. Goals and Requirements
2. Constraints
3. Scope / Context
4. Solution / Approach
   4a Structures & Process
   4b Samples incl. Code
5. Alternatives
6. Risks

Quality Goals

Goal	Description
...	...

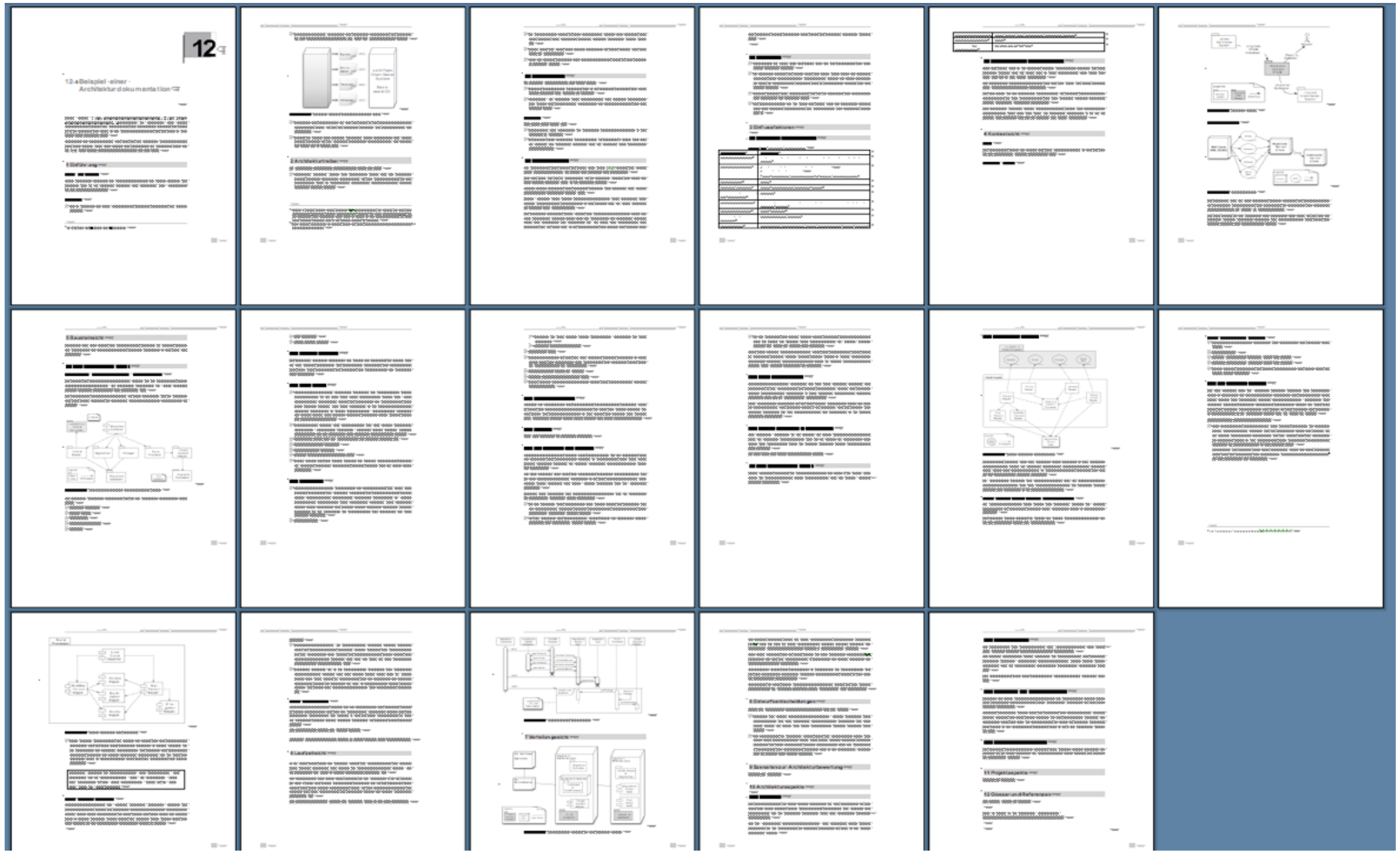
Stakeholder Table

Who?	Interest?
...	...

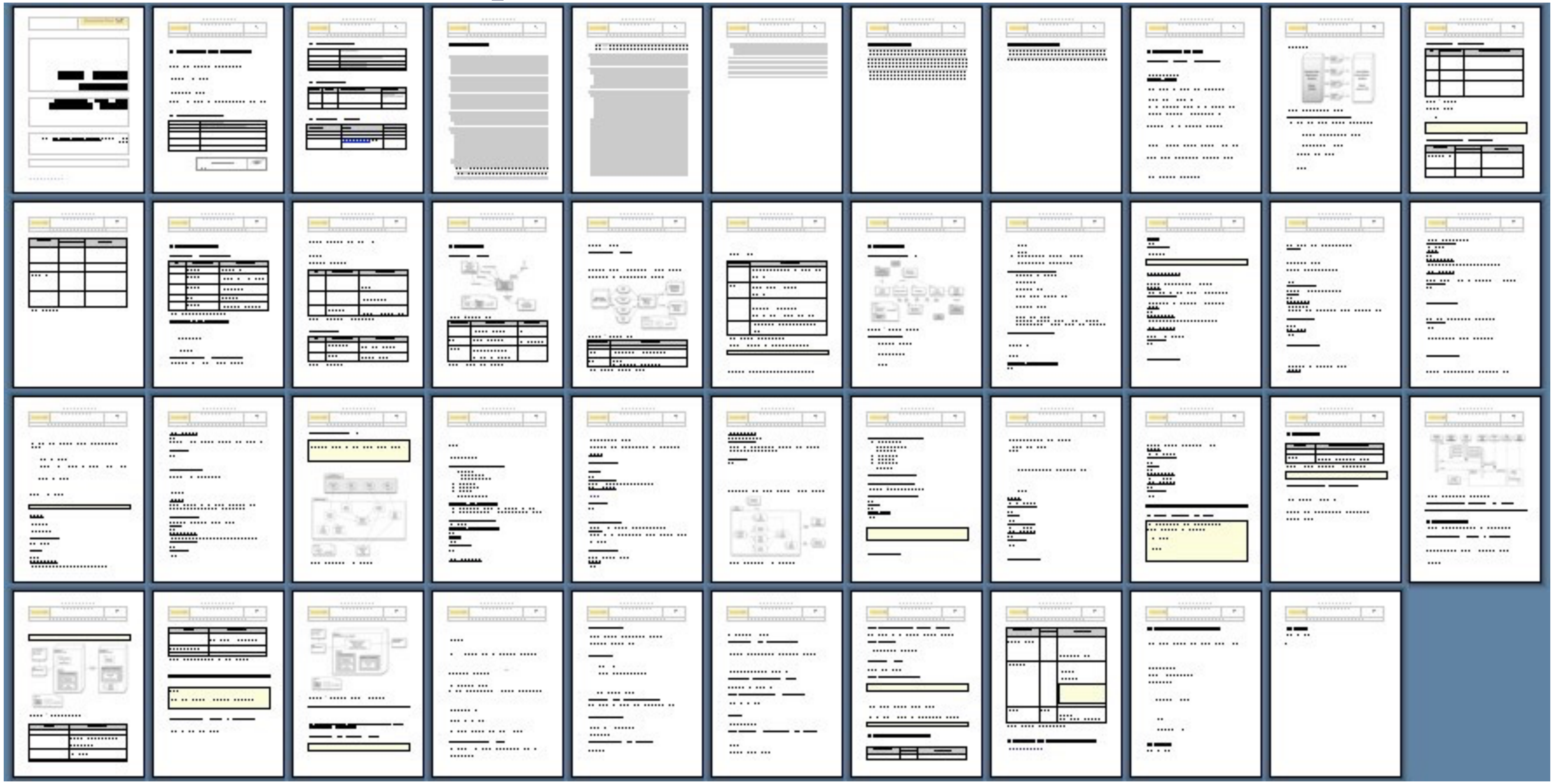
Structure of Interface Descriptions
1. Name, Version
2. Ressources (Syntax)
3. Semantic
   3.1 Business
   3.2 Technical
4. Protocol
   4.1 Flow / Process
   4.2 Transmission Channel
5. Error and Exception Behavior
6. Restrictions
7. Sample Data
8. Quality Attributes / QoS

arc42 V 6.0 (Dec. 2012),
BigPicture
created by Peter Hruschka
& Gernot Starke

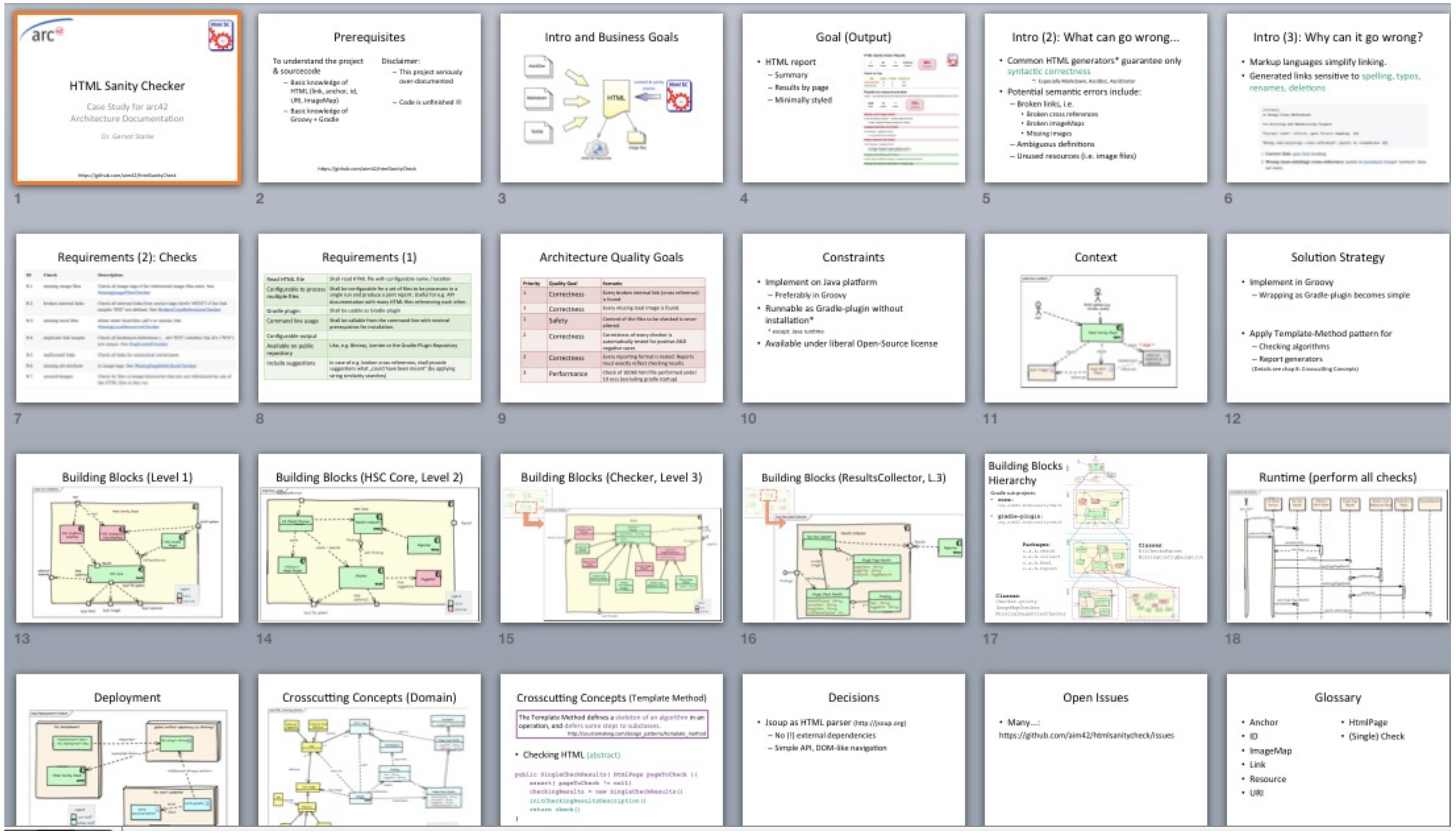
Sample (Data Migration, 4000+ PT)



Sample (CRM-System, 2000+ PT)



Sample „htmlSanityCheck"


1 HTML Sanity Checker
Case Study for arc42 Architecture Documentation
2 Prerequisites
3 Intro and Business Goals
4 Goal (Output)
5 Intro (2): What can go wrong...
6 Intro (3): Why can it go wrong?
7 Requirements (2): Checks
8 Requirements (1)
9 Architecture Quality Goals
10 Constraints
11 Context
12 Solution Strategy
13 Building Blocks (Level 1)
14 Building Blocks (HSC Core, Level 2)
15 Building Blocks (Checker, Level 3)
16 Building Blocks (ResultsCollector, L.3)
17 Building Blocks Hierarchy
18 Runtime (perform all checks)
Deployment
Crosscutting Concepts (Domain)
Crosscutting Concepts (Template Method)
Decisions
Open Issues
Glossary

https://github.com/aim42/htmlsanitycheck

HTML Sanity Checker

Case Study for arc42
Architecture Documentation

(containing miserable source code by Gernot Starke)

https://github.com/aim42/htmlSanityCheck

Prerequisites

Required:

- Basic knowledge of HTML (link, anchor, id, URI, ImageMap)
- Basic knowledge of Groovy + Gradle

Disclaimer:

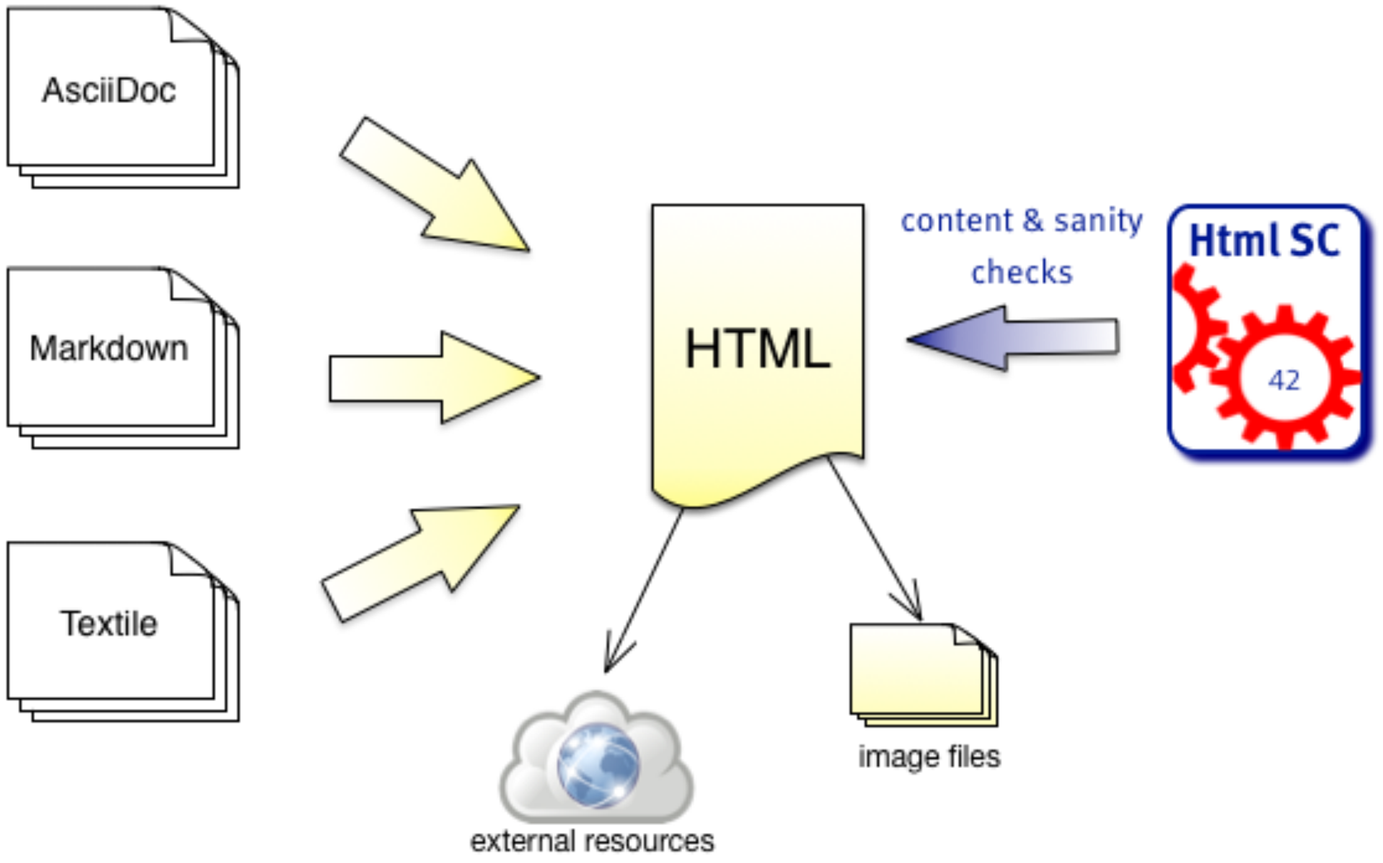
- System is seriously over-documented
- Code is unfinished ☹

Intro and (Business) Goal

Html SC 42

shall support authors creating digital formats with hyperlinks and integration of images and similar resources."

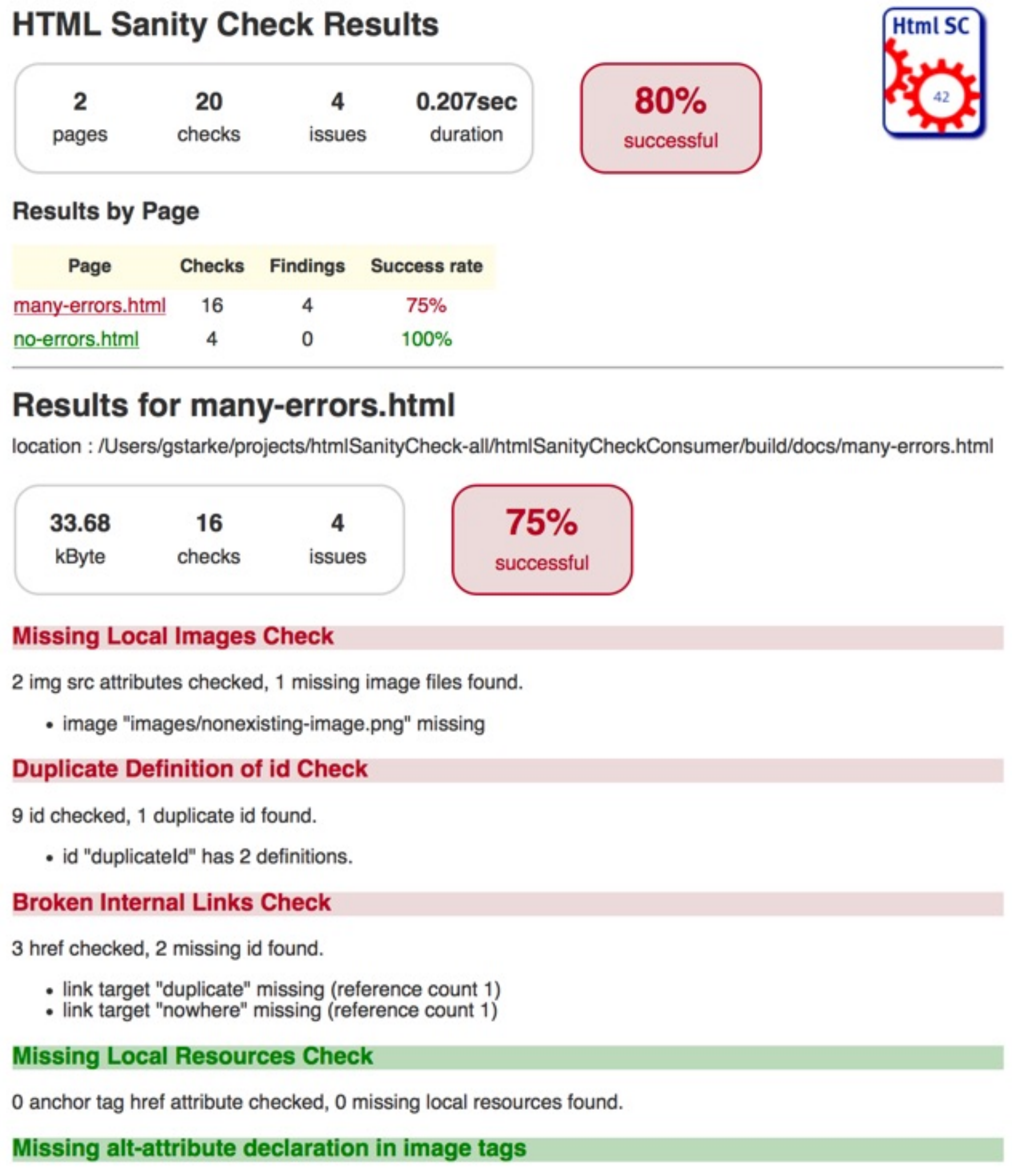
Overview


AsciiDoc
Markdown
Textile
HTML
content & sanity checks
Html SC
42
external resources
image files

Goal (Output)

HTML report

- Summary
- Results by page
- Minimally styled



HTML Sanity Check Results

2	20	4	0.207sec	80%
pages	checks	issues	duration	successful

Results by Page

Page	Checks	Findings	Success rate
many-errors.html	16	4	75%
no-errors.html	4	0	100%

Results for many-errors.html

location : /Users/gstarke/projects/htmlSanityCheck-all/htmlSanityCheckConsumer/build/docs/many-errors.html

33.68	16	4	75%
kByte	checks	issues	successful

Missing Local Images Check

2 img src attributes checked, 1 missing image files found.

- image "images/nonexisting-image.png" missing

Duplicate Definition of id Check

9 id checked, 1 duplicate id found.

- id "duplicateId" has 2 definitions.

Broken Internal Links Check

3 href checked, 2 missing id found.

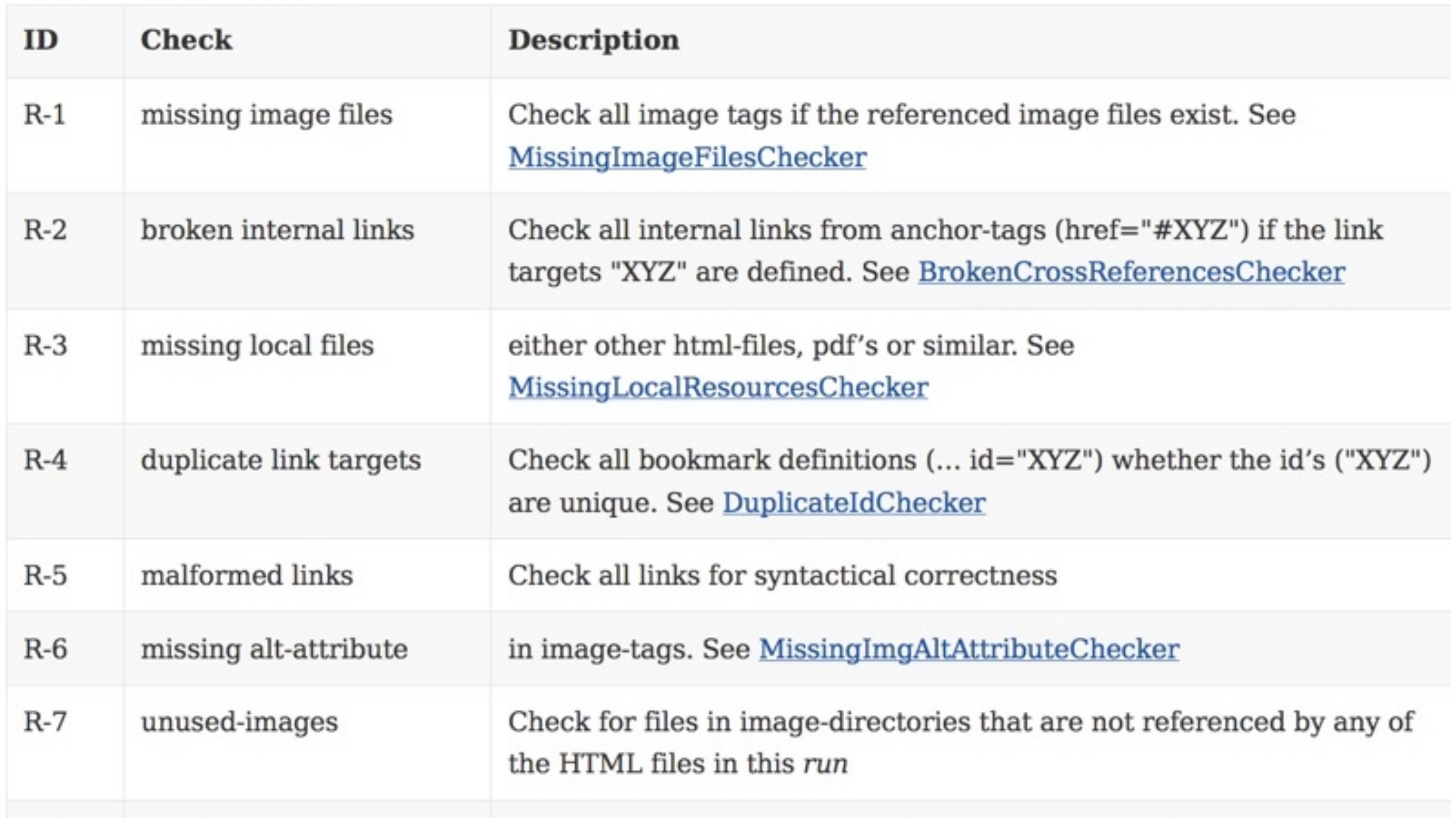
- link target "duplicate" missing (reference count 1)
- link target "nowhere" missing (reference count 1)

Missing Local Resources Check

0 anchor tag href attribute checked, 0 missing local resources found.

Missing alt-attribute declaration in image tags

Requirements (1): Checks

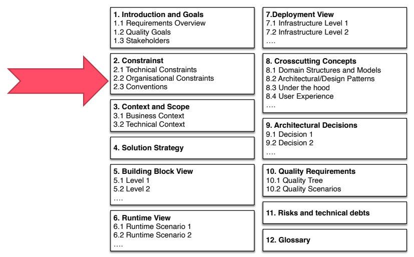


ID	Check	Description
R-1	missing image files	Check all image tags if the referenced image files exist. See MissingImageFilesChecker
R-2	broken internal links	Check all internal links from anchor-tags (href="#XYZ") if the link targets "XYZ" are defined. See BrokenCrossReferencesChecker
R-3	missing local files	either other html-files, pdf's or similar. See MissingLocalResourcesChecker
R-4	duplicate link targets	Check all bookmark definitions (... id="XYZ") whether the id's ("XYZ") are unique. See DuplicateIdChecker
R-5	malformed links	Check all links for syntactical correctness
R-6	missing alt-attribute	in image-tags. See MissingImgAltAttributeChecker
R-7	unused-images	Check for files in image-directories that are not referenced by any of the HTML files in this *run*

Architecture Quality Requirements

Priority	Quality Goal	Scenario
1	Correctness	Every broken internal link (cross reference) is found
1	Correctness	Every missing local image is found.
1	Safety	Content of the files to be checked is *never* altered.
2	Correctness	Correctness of every checker is automatically tested for positive AND negative cases
3	Correctness	Every reporting format is tested: Reports must exactly reflect checking results.
3	Performance	Check of 100kB html file performed under 10 secs (excluding gradle startup)

Constraints



1. Introduction and Goals
1.1 Requirements Overview
1.2 Quality Goals
1.3 Stakeholders

2. Constrainst
2.1 Technical Constraints
2.2 Organisational Constraints
2.3 Conventions

3. Context and Scope
3.1 Business Context
3.2 Technical Context

4. Solution Strategy

5. Building Block View
5.1 Level 1
5.2 Level 2
....

6. Runtime View
6.1 Runtime Scenario 1
6.2 Runtime Scenario 2
....

7.Deployment View
7.1 Infrastructure Level 1
7.2 Infrastructure Level 2
....

8. Crosscutting Concepts
8.1 Domain Structures and Models
8.2 Architectural/Design Patterns
8.3 Under the hood
8.4 User Experience
....

9. Architectural Decisions
9.1 Decision 1
9.2 Decision 2
....

10. Quality Requirements
10.1 Quality Tree
10.2 Quality Scenarios

11. Risks and technical debts

12. Glossary

Implemented on Java platform

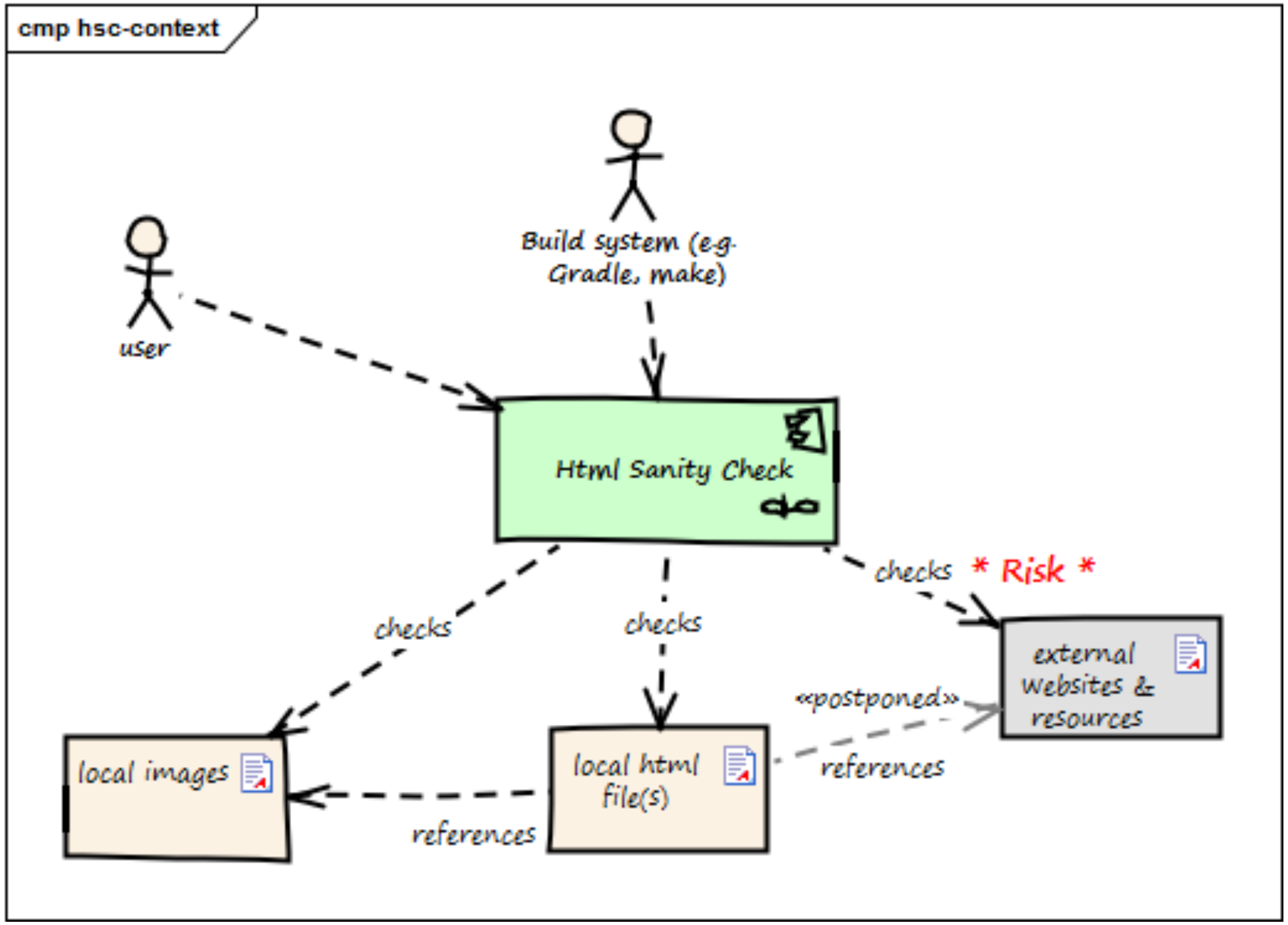
- Preferably in Groovy

Runnable as Gradle-plugin without installation*

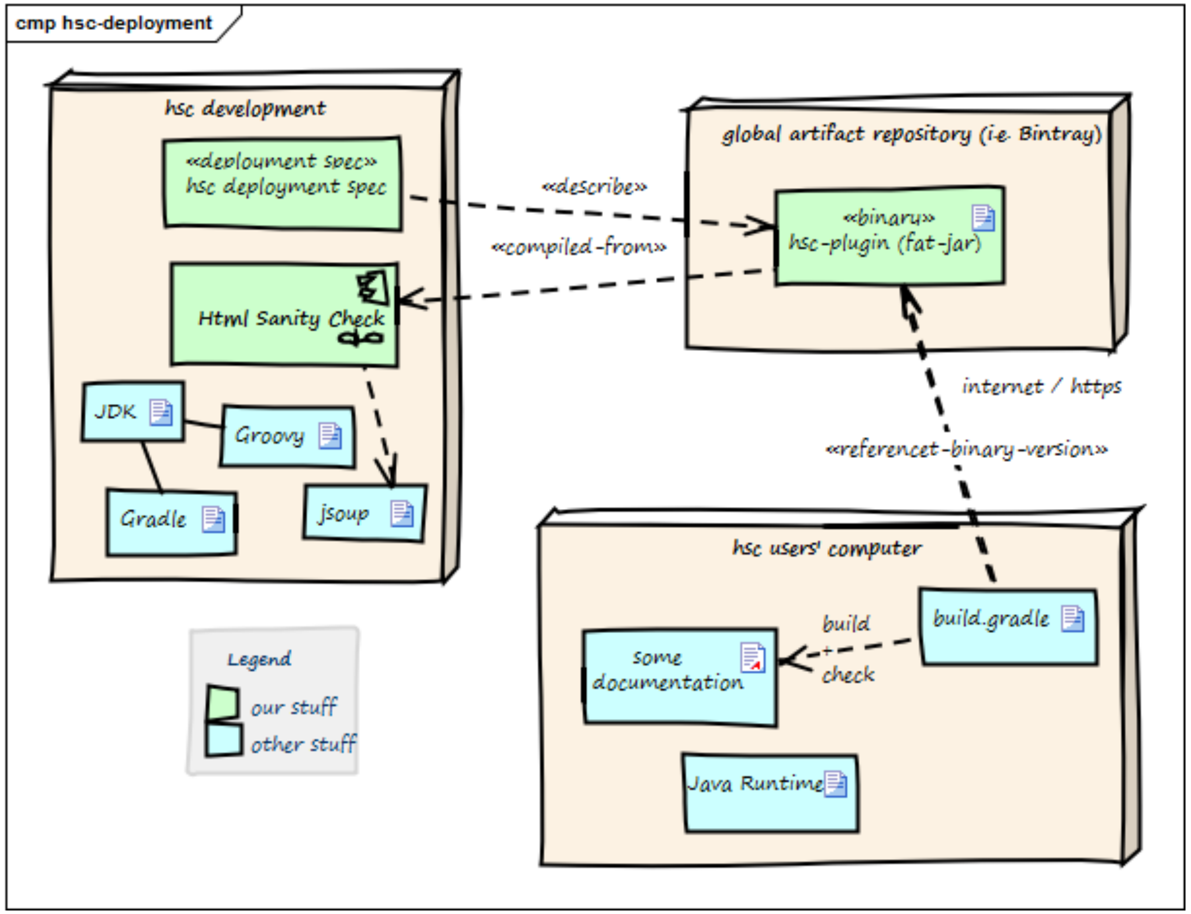
* except: Java runtime

Available under liberal Open-Source license

(Business) Context



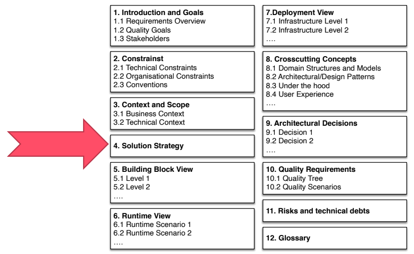
(Technical) Context



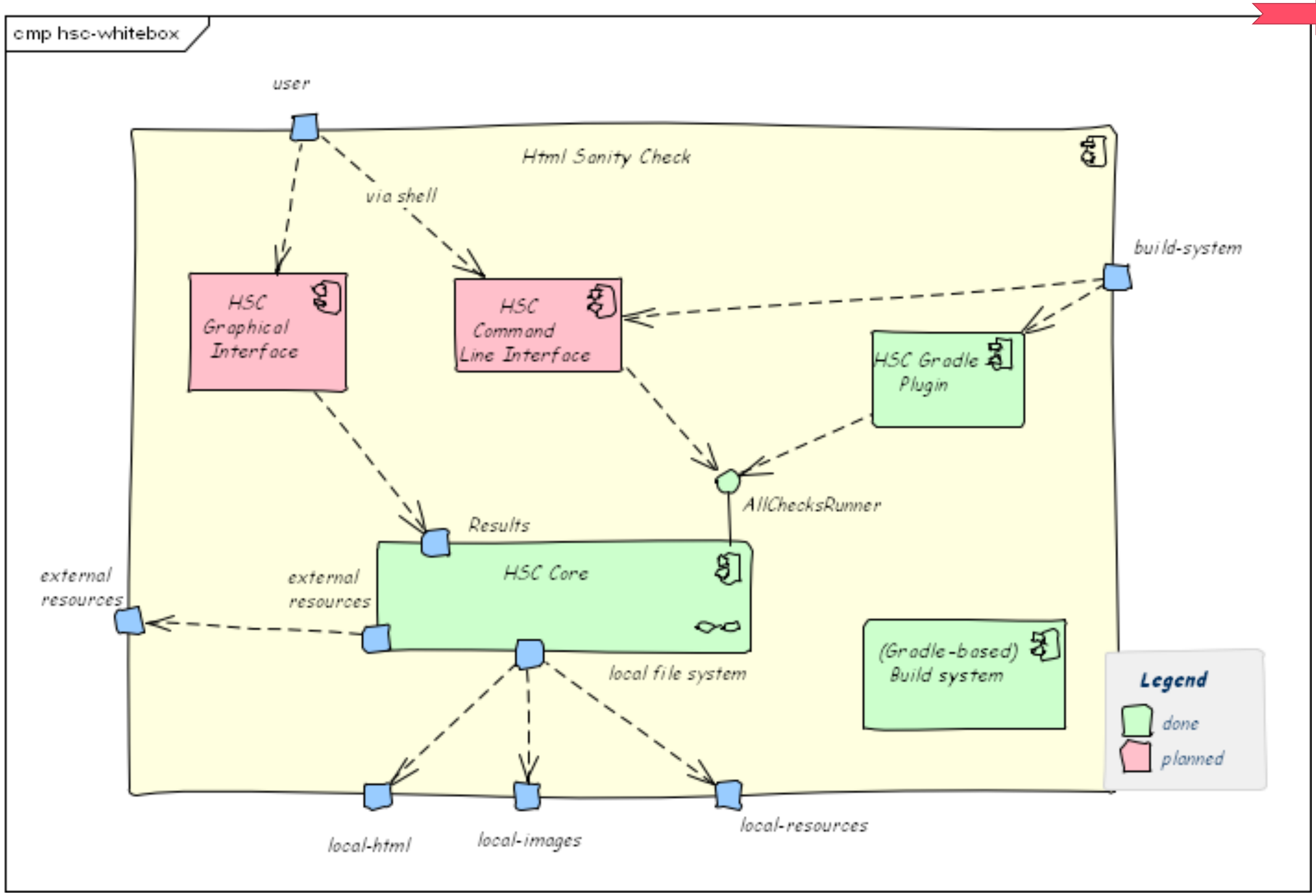
Solution Strategy

- Use HTML parser with minimal dependencies
- Implement in Groovy
  - Wrapping as Gradle-plugin becomes simple

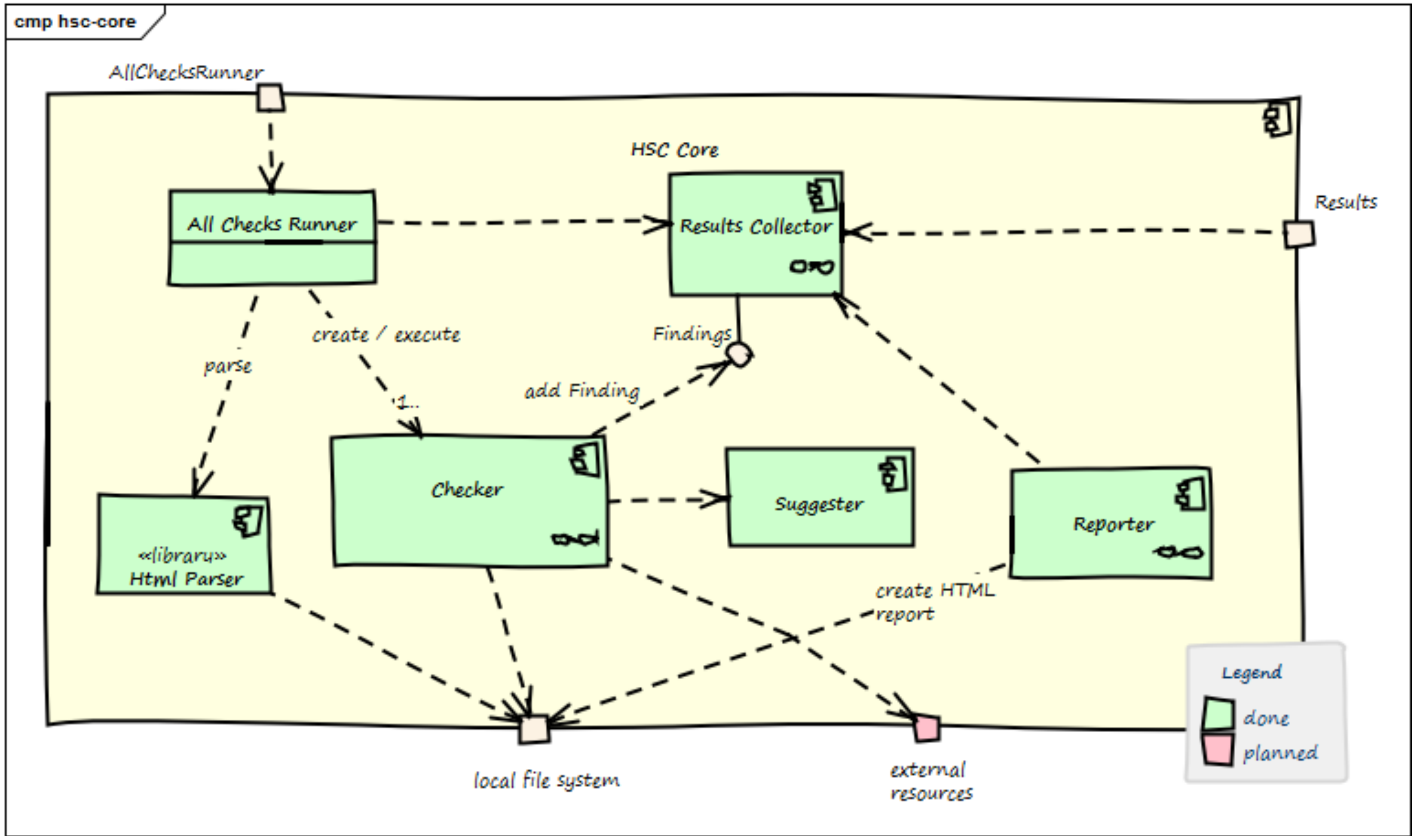
- Apply Template-Method pattern for
  - Checking algorithms
  - Report generators
  - (Details see chap 8: Crosscutting Concepts)



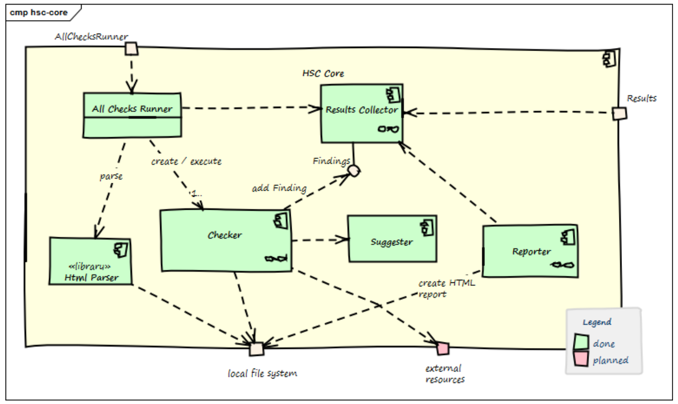
Building Blocks (Level 1)



Building Blocks (HSC Core, Level 2)



HSC Core, Level 2 (ff)

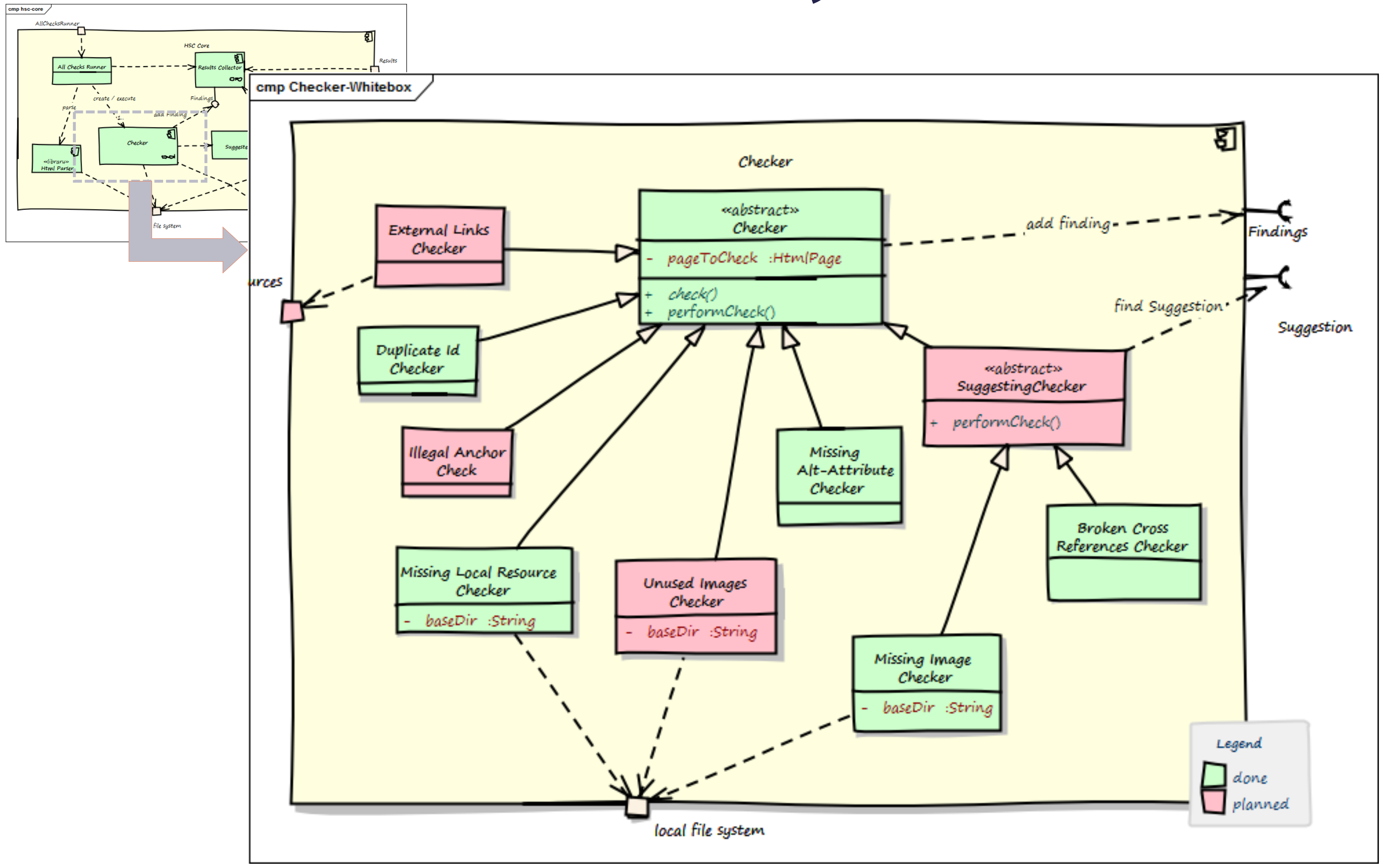


Contained Blackboxes

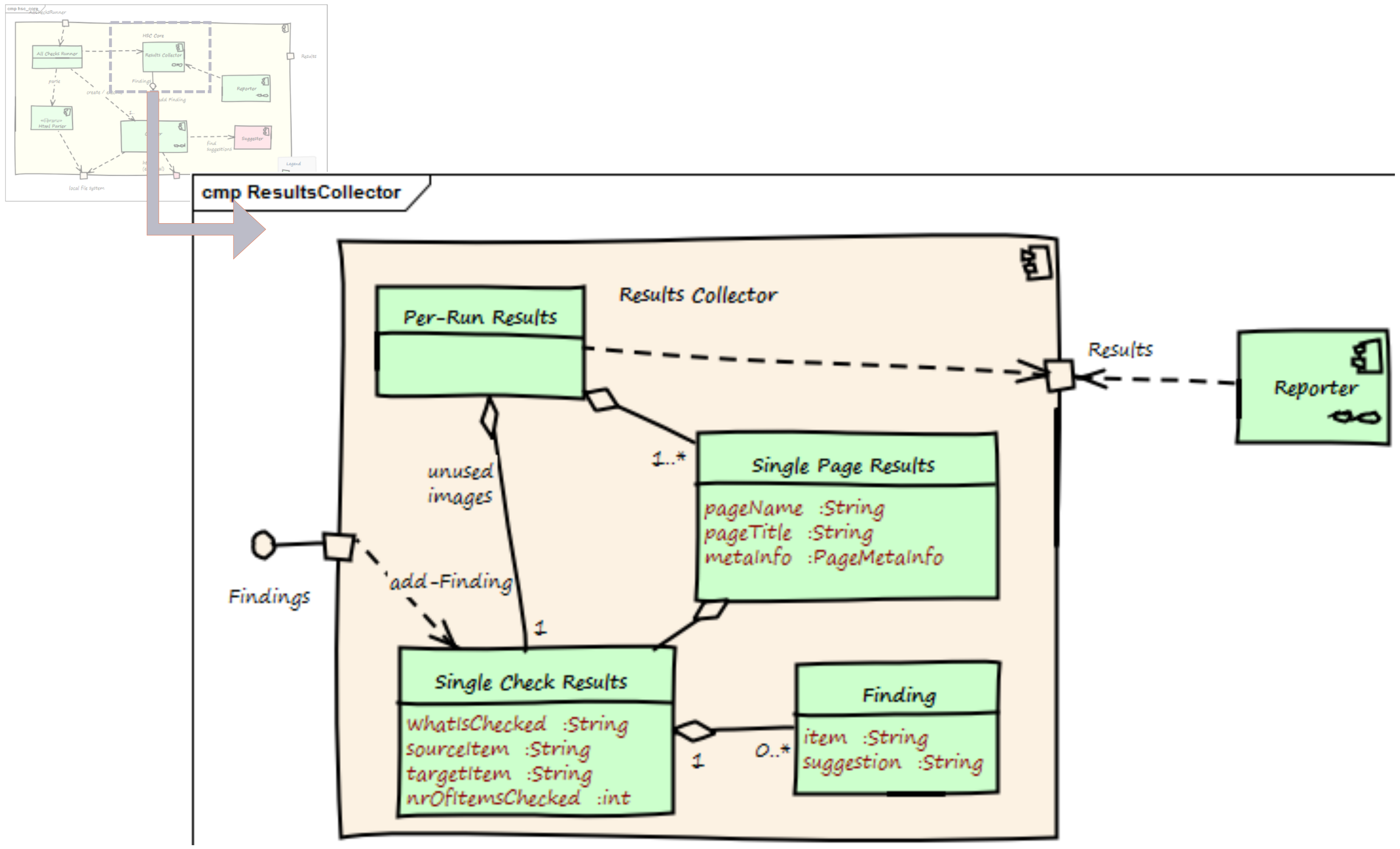
Table 12. HSC-Core building blocks

Checker	Abstract class, used in form of the template-pattern. Shall be subclassed for all checking algorithms.
AllChecksRunner	Facade to the different Checker instances. Provides a (parameter-driven) command-line interface.
ResultsCollector (Whitebox)	Collects all checking results. Its interface `Results` is contained in the whitebox description
Reporter	Reports checking results to either console or an html file.
HtmlParser	Encapsulates html parsing, provides methods to search within the (parsed) html.
Suggester	In case of checking issues, suggests alternatives by comparing the faulty element to the one present in the html file. Currently not implemented

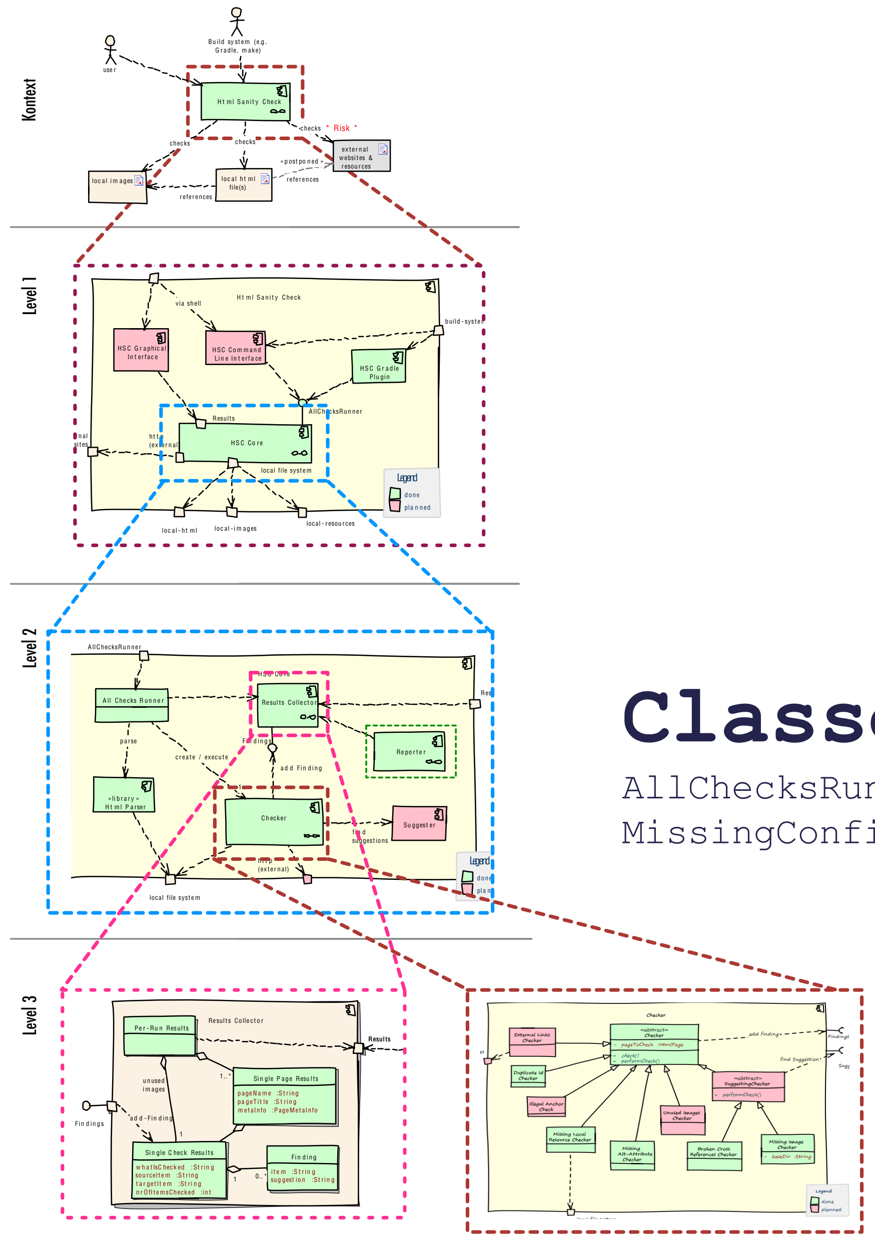
Building Blocks (Checker, Level 3)



Building Blocks (ResultsCollector, L.3)



Building Blocks Hierarchy



Packages:

o.a.h.check
o.a.h.collect
o.a.h.html
o.a.h.report

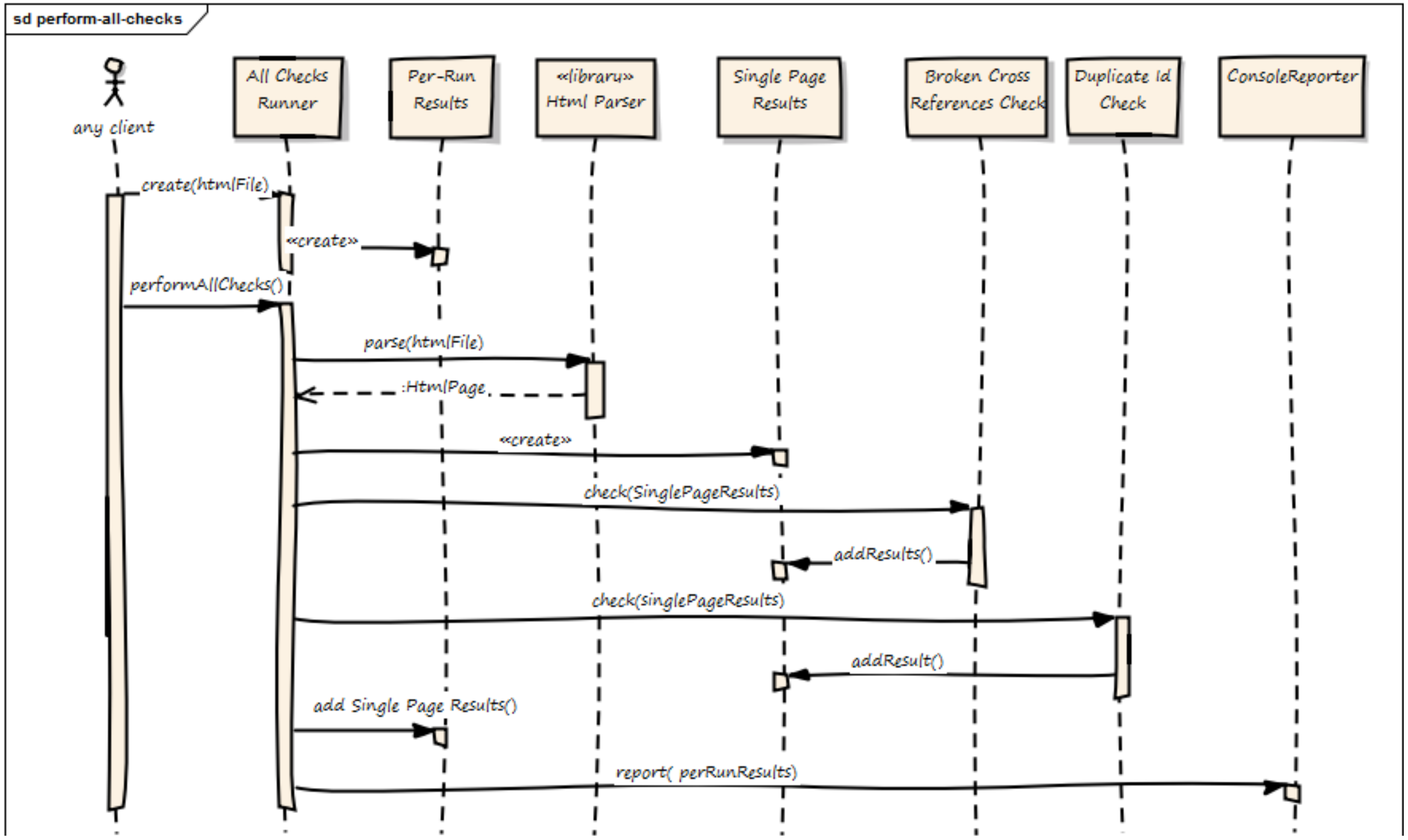
Classes:

AllChecksRunner
MissingConfigException

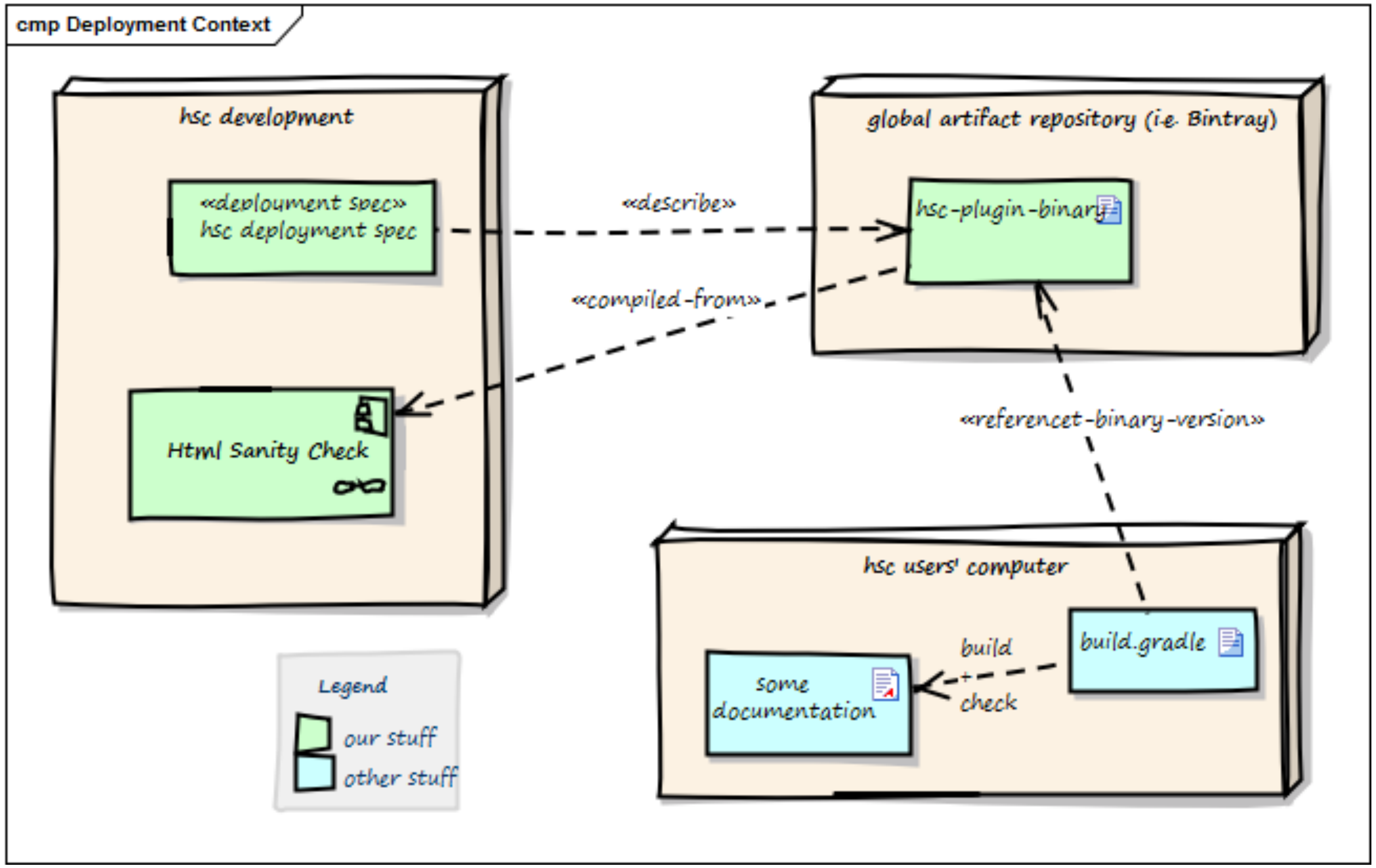
Classes:

Checker.groovy
ImageMapChecker
MissingImageFilesChecker

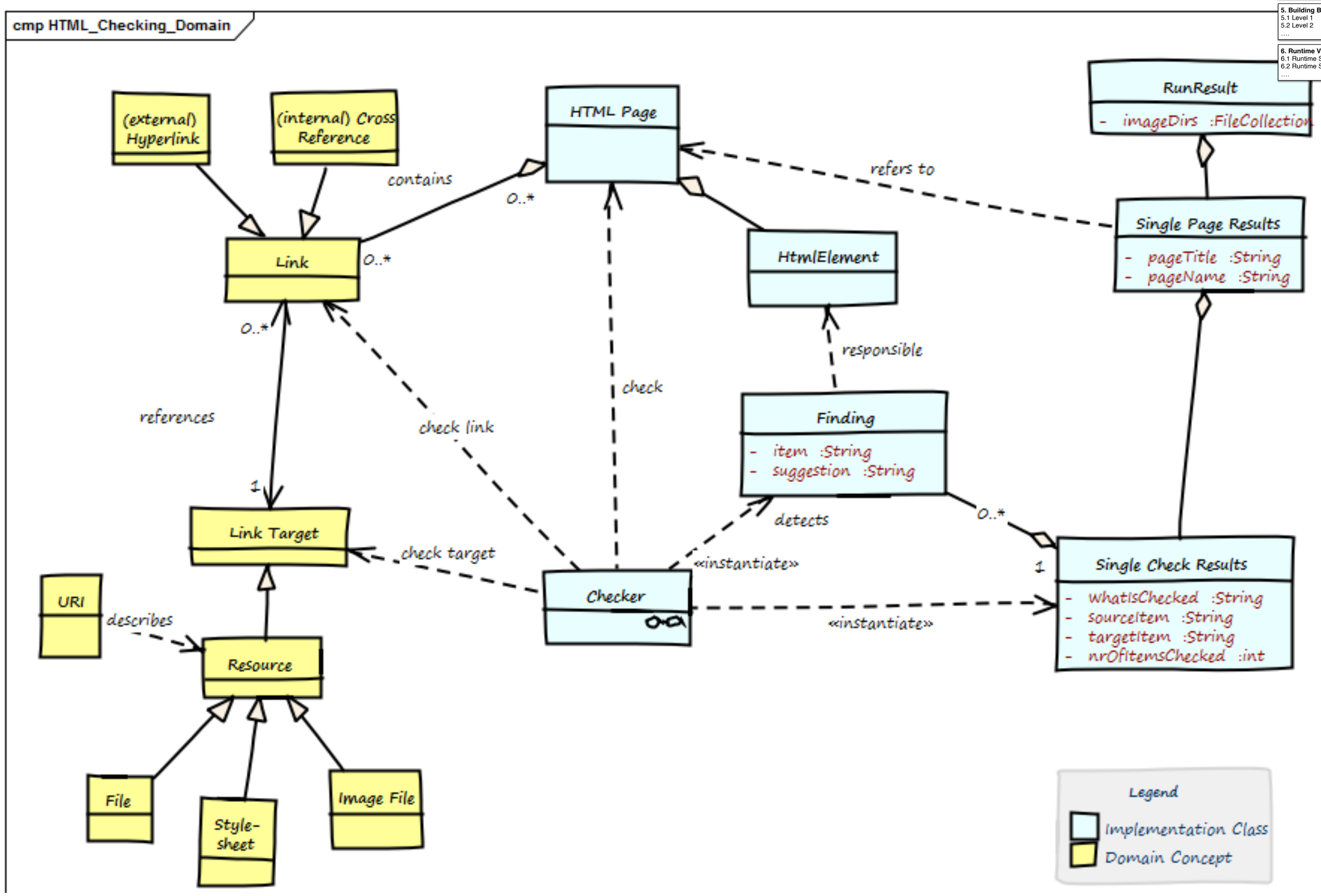
Runtime (perform all checks)

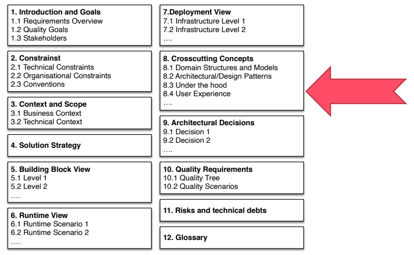


Deployment


cmp Deployment Context
hsc development
«deployment spec»
hsc deployment spec
Html Sanity Check
global artifact repository (i.e. Bintray)
hsc-plugin-binary
«describe»
«compiled-from»
«referencet-binary-version»
hsc users' computer
build.gradle
some documentation
build
+
check
Legend
our stuff
other stuff

Crosscutting Concepts



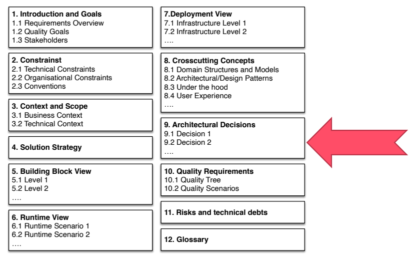


Crosscutting Concepts
(Template Method)

Checking HTML (abstract)

```
public SingleCheckResults( HtmlPage pageToCheck ){
    assert( pageToCheck != null)
    checkingResults = new SingleCheckResults()
    initCheckingResultsDescription()
    return check()
}
```

The Template Method defines a skeleton of an algorithm in an operation, and defers some steps to subclasses.



Decisions

Result: Jsoup as HTML parser (http://jsoup.org)

Criteria	A1: Jsoup	A2: HTMLUnit
No (!) external dependencies	No deps	>15 deps
Simple API	simple	simple
DOM-like navigation	yes	partially
Fast (1 MB page / sec)	yes	no

Start here: https://arc42.org/overview/

Introduction and Goals
Required Functions
Quality Goals
Goal
Description
...
...
Stakeholder
Who?
Expectation?
...
...

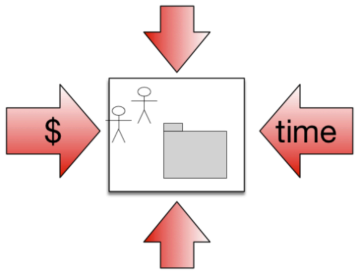
1. Introduction and Goals

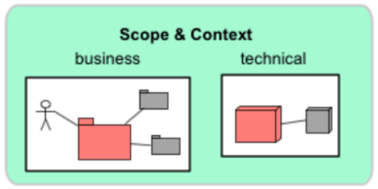
Short description of the **requirements**, driving forces, extract (or abstract) of requirements. Top three (max five) **quality goals** for the architecture which have highest priority for the major stakeholders. A table of important **stakeholders** with their expectation regarding architecture.

2. Constraints

Anything that constrains teams in design and implementation decisions or decision about related processes. Can sometimes go beyond individual systems and are valid for whole organizations and companies.

Read More

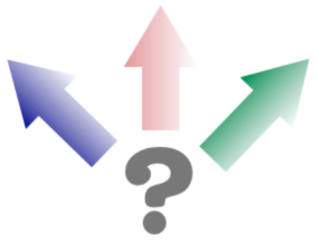




3. Context and Scope

Delimits your system from its (external) communication partners (neighboring systems and users). Specifies the external interfaces. Shown from a business/domain perspective (always) or a technical perspective (optional)

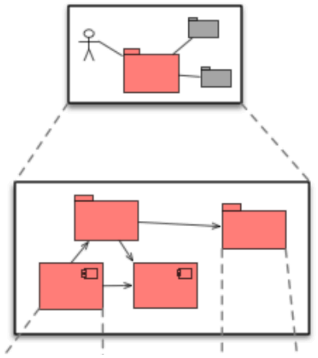
Read More



9. Architectural Decisions

Important, expensive, critical, large scale or risky architecture decisions including rationales.

Read More



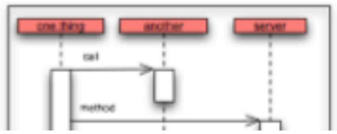
5. Building Block View

Static decomposition of the system, abstractions of source-code, shown as hierarchy of white boxes (containing black boxes), up to the appropriate level of detail.

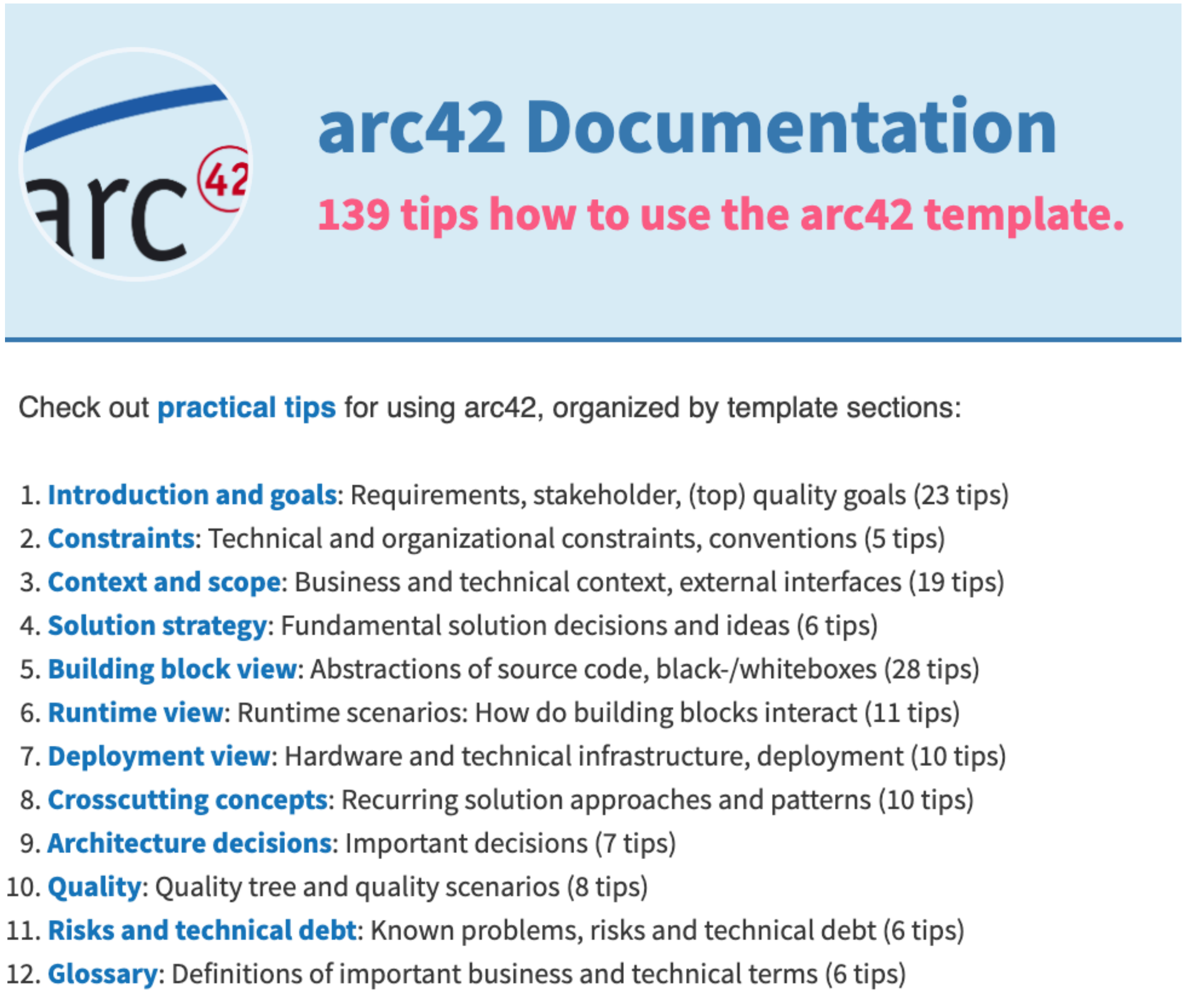
Read More

6. Runtime View

Behavior of building blocks as scenarios, covering important use cases or features, interactions at critical external interfaces, operation and administration plus error and exception behavior.



Continue here: https://docs.arc42.org



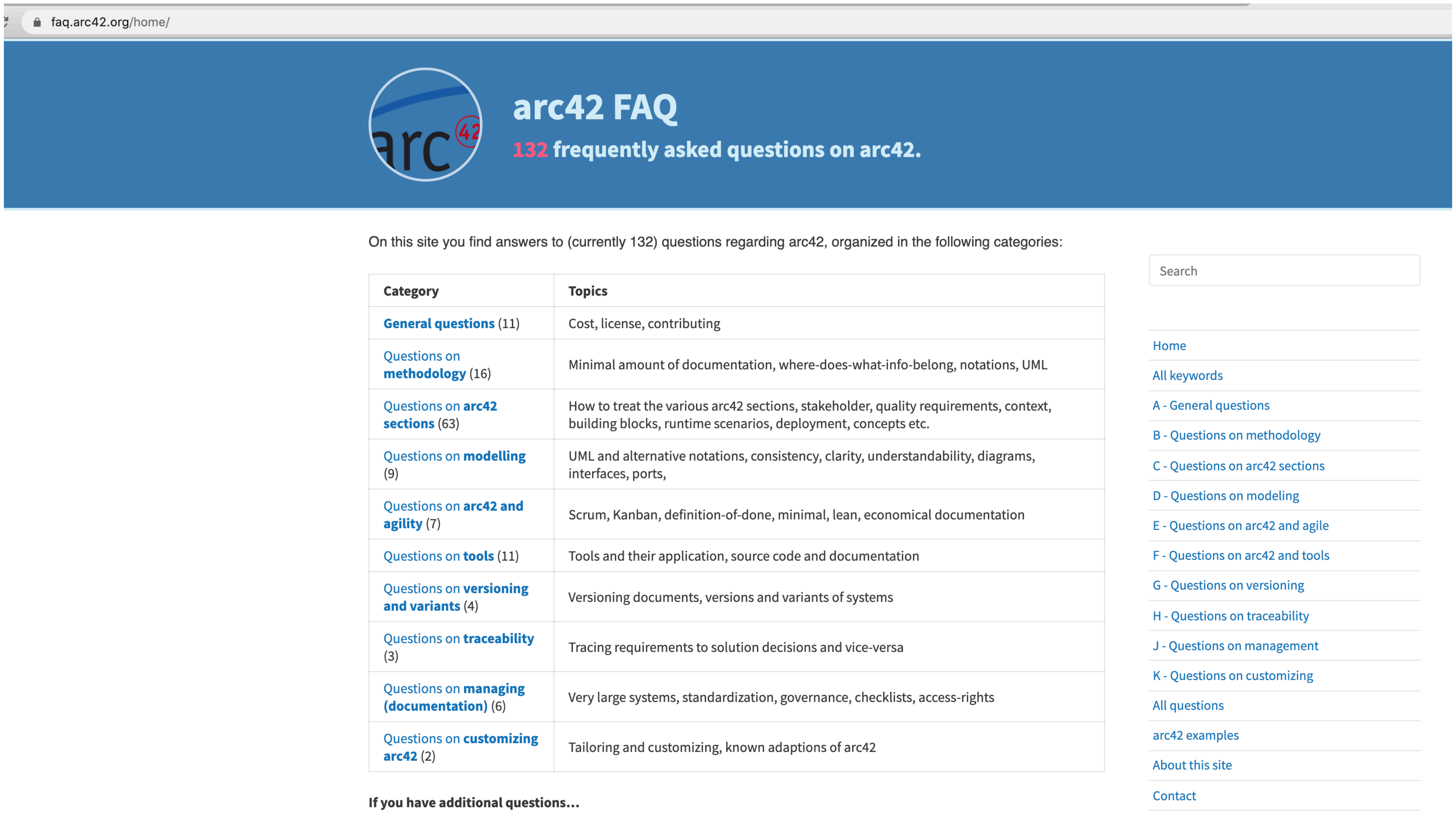
arc42 Documentation

139 tips how to use the arc42 template.

Check out **practical tips** for using arc42, organized by template sections:

1. **Introduction and goals**: Requirements, stakeholder, (top) quality goals (23 tips)
2. **Constraints**: Technical and organizational constraints, conventions (5 tips)
3. **Context and scope**: Business and technical context, external interfaces (19 tips)
4. **Solution strategy**: Fundamental solution decisions and ideas (6 tips)
5. **Building block view**: Abstractions of source code, black-/whiteboxes (28 tips)
6. **Runtime view**: Runtime scenarios: How do building blocks interact (11 tips)
7. **Deployment view**: Hardware and technical infrastructure, deployment (10 tips)
8. **Crosscutting concepts**: Recurring solution approaches and patterns (10 tips)
9. **Architecture decisions**: Important decisions (7 tips)
10. **Quality**: Quality tree and quality scenarios (8 tips)
11. **Risks and technical debt**: Known problems, risks and technical debt (6 tips)
12. **Glossary**: Definitions of important business and technical terms (6 tips)

Tipps and FAQ (2)



faq.arc42.org/home/

arc42 FAQ

132 frequently asked questions on arc42.

On this site you find answers to (currently 132) questions regarding arc42, organized in the following categories:

Category	Topics
General questions (11)	Cost, license, contributing
Questions on **methodology** (16)	Minimal amount of documentation, where-does-what-info-belong, notations, UML
Questions on **arc42 sections** (63)	How to treat the various arc42 sections, stakeholder, quality requirements, context, building blocks, runtime scenarios, deployment, concepts etc.
Questions on **modelling** (9)	UML and alternative notations, consistency, clarity, understandability, diagrams, interfaces, ports,
Questions on **arc42 and agility** (7)	Scrum, Kanban, definition-of-done, minimal, lean, economical documentation
Questions on **tools** (11)	Tools and their application, source code and documentation
Questions on **versioning and variants** (4)	Versioning documents, versions and variants of systems
Questions on **traceability** (3)	Tracing requirements to solution decisions and vice-versa
Questions on **managing (documentation)** (6)	Very large systems, standardization, governance, checklists, access-rights
Questions on **customizing arc42** (2)	Tailoring and customizing, known adaptions of arc42

If you have additional questions...

Search

- Home
- All keywords
- A - General questions
- B - Questions on methodology
- C - Questions on arc42 sections
- D - Questions on modeling
- E - Questions on arc42 and agile
- F - Questions on arc42 and tools
- G - Questions on versioning
- H - Questions on traceability
- J - Questions on management
- K - Questions on customizing
- All questions
- arc42 examples
- About this site
- Contact

Tooling?

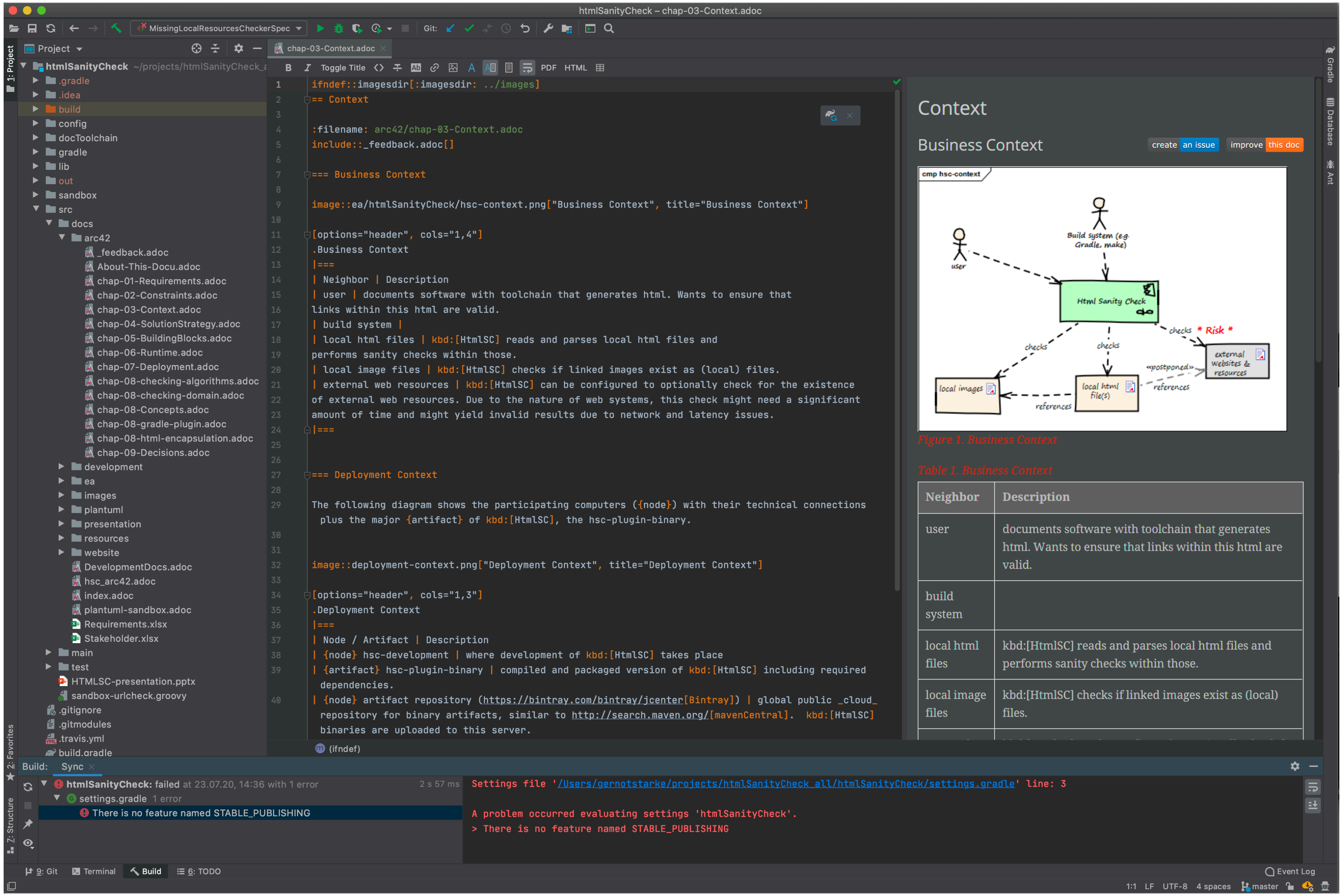
Whatever is available...

AsciiDoc in your IDE

Wiki

draw.io

Microsoft-Word...



```
ifndef::imagesdir[:imagesdir: ../images]
== Context

:filename: arc42/chap-03-Context.adoc
include::_feedback.adoc[]

=== Business Context

image::ea/htmlSanityCheck/hsc-context.png["Business Context", title="Business Context"]

[options="header", cols="1,4"]
.Business Context
|===
| Neighbor | Description
| user | documents software with toolchain that generates html. Wants to ensure that
links within this html are valid.
| build system |
| local html files | kbd:[HtmlSC] reads and parses local html files and
performs sanity checks within those.
| local image files | kbd:[HtmlSC] checks if linked images exist as (local) files.
| external web resources | kbd:[HtmlSC] can be configured to optionally check for the existence
of external web resources. Due to the nature of web systems, this check might need a significant
amount of time and might yield invalid results due to network and latency issues.
|===


=== Deployment Context

The following diagram shows the participating computers ({node}) with their technical connections
  plus the major {artifact} of kbd:[HtmlSC], the hsc-plugin-binary.


image::deployment-context.png["Deployment Context", title="Deployment Context"]

[options="header", cols="1,3"]
.Deployment Context
|===
| Node / Artifact | Description
| {node} hsc-development | where development of kbd:[HtmlSC] takes place
| {artifact} hsc-plugin-binary | compiled and packaged version of kbd:[HtmlSC] including required
  dependencies.
| {node} artifact repository (https://bintray.com/bintray/jcenter[Bintray]) | global public _cloud_
  repository for binary artifacts, similar to http://search.maven.org/[mavenCentral].  kbd:[HtmlSC]
  binaries are uploaded to this server.
```

Context

Business Context

Figure 1. Business Context

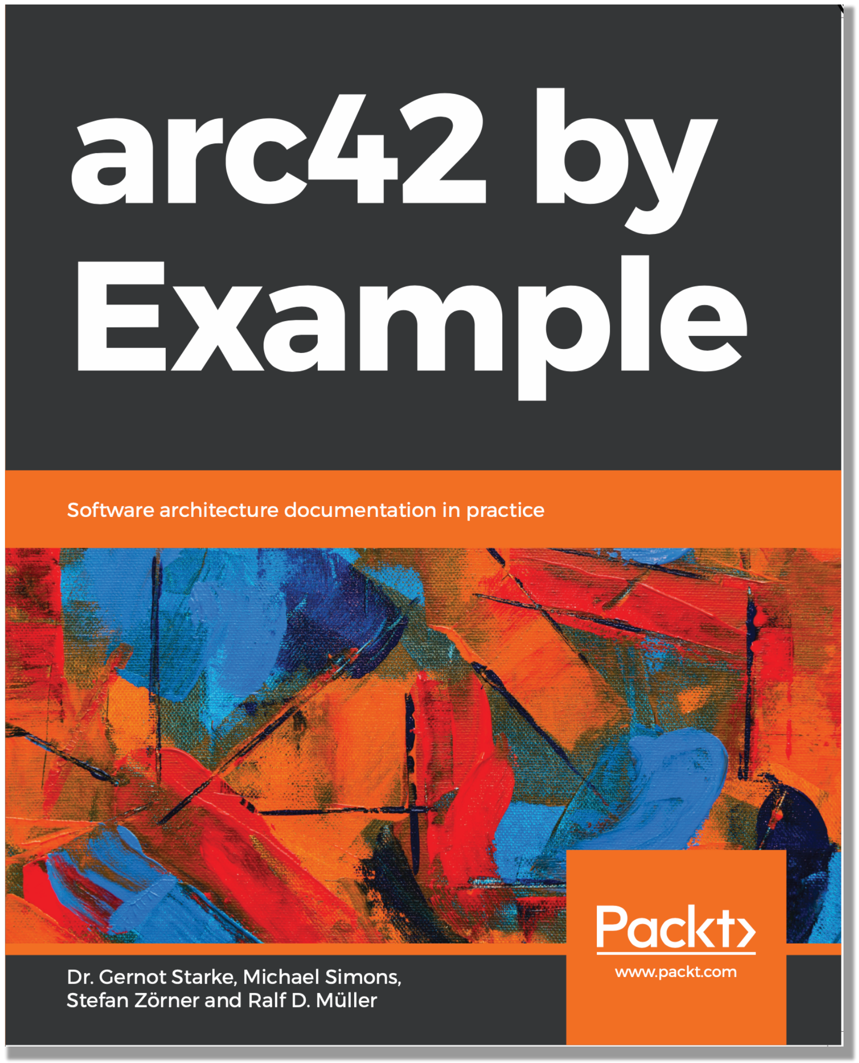
Table 1. Business Context

Neighbor	Description
user	documents software with toolchain that generates html. Wants to ensure that links within this html are valid.
build system	
local html files	kbd:[HtmlSC] reads and parses local html files and performs sanity checks within those.
local image files	kbd:[HtmlSC] checks if linked images exist as (local) files.

```
htmlSanityCheck: failed at 23.07.20, 14:36 with 1 error      2 s 57 ms
  settings.gradle  1 error
    There is no feature named STABLE_PUBLISHING
```

```
Settings file '/Users/gernotstarke/projects/htmlSanityCheck_all/htmlSanityCheck/settings.gradle' line: 3

A problem occurred evaluating settings 'htmlSanityCheck'.
> There is no feature named STABLE_PUBLISHING
```


arc42 by Example

Software architecture documentation in practice

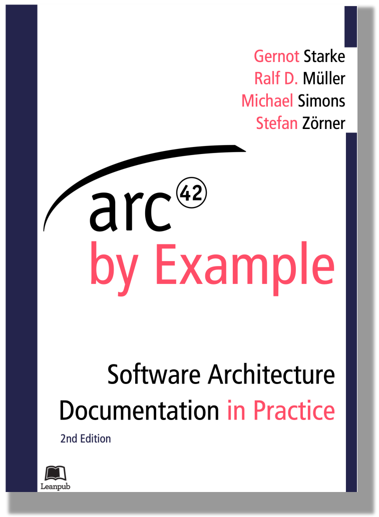
Dr. Gernot Starke, Michael Simons, Stefan Zörner and Ralf D. Müller

Packt›

www.packt.com

Further info:

leanpub.com/arc42byexample/c/JUG-Tirana


Gernot Starke
Ralf D. Müller
Michael Simons
Stefan Zörner

arc42 by Example

Software Architecture Documentation in Practice

2nd Edition

Leanpub

Thanx!

Questions?

Dr. Gernot Starke
gernot.starke@innoq.com

INNOQ
www.innoq.com