youlearn days 8–9 Nov

Let‘s start hands on – IT architecture

Bernd Rederlechner
Principal Lead Architect
Phone +491607090492
E-mail bernd.rederlechner@t-systems.com
LinkedIn https://www.linkedin.com/in/brederle/

ATTENTION: Experimental session – please don’t expect perfection !

Github account: https://github.com/signup

Agenda

01 **IT Architects in the wild**
15 min – Why architecture
15 min - discover my workspace

02 **Business Context**
15 min - Purpose, Goal, Stakeholder
15 min - Context, Constraints

Biobreak?

03 **Building blocks**
15min - Services, Black/Whitebox
15min - System, Conceptual

04 **Quality & implementation**
15min - Quality scenarios
15min - Steering & consistency

“

What is the thingy in a project...

... everybody cries for

... magically disappears when needed

... used as popular excuse for delays ?

Architecture

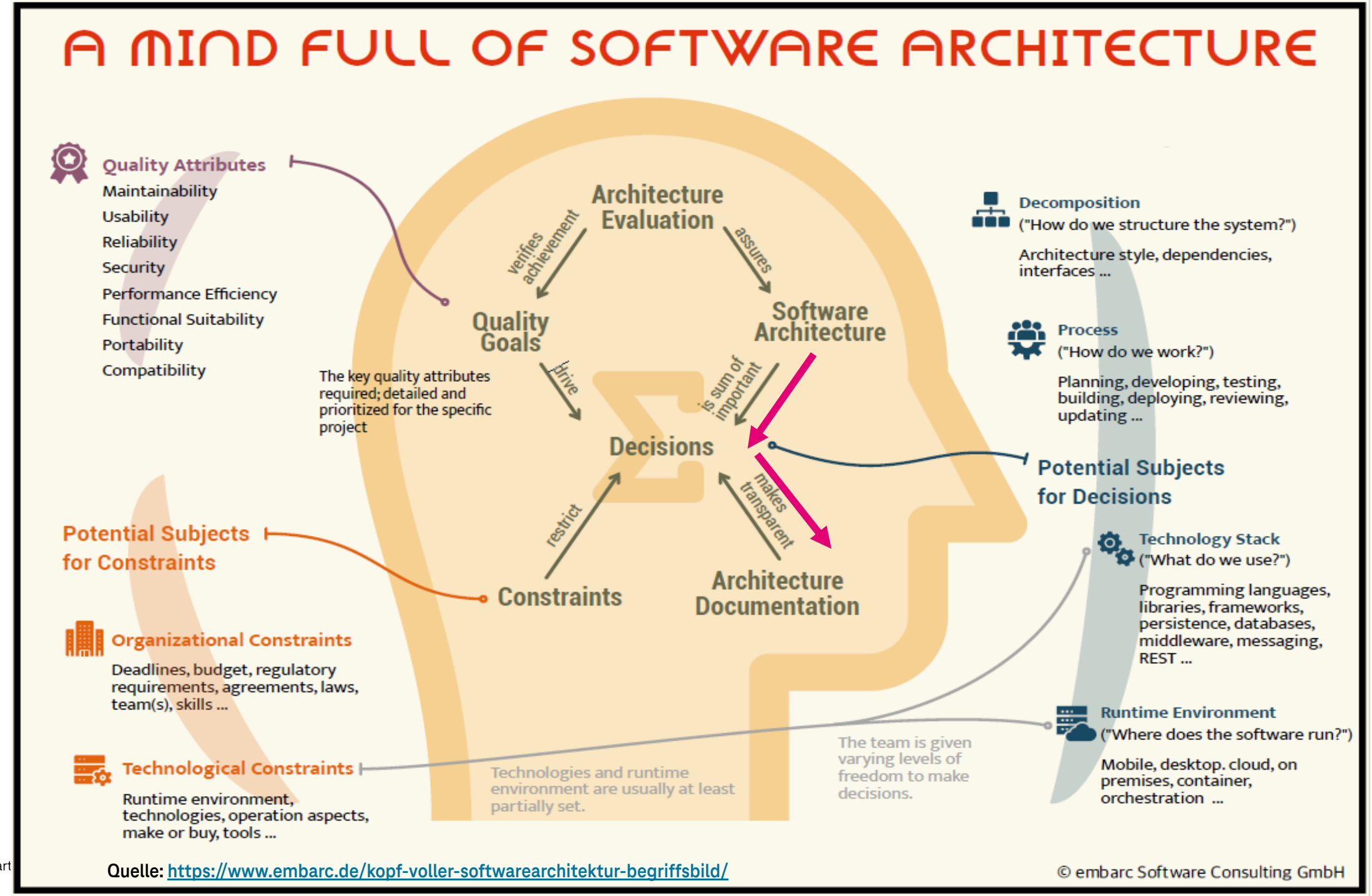
Is the balance of 3 aspects:

- **Usefulness („utilitas“)**
  The system contributes to business by fulfilling the functional and non-functional requirements of all important stakeholders using it and the ones who pay for (in this sequence 😉).

- **Solidity („firmitas“)**
  The software/system/service is stable „by itself“ in terms of the specified quality requirements (and not permanently under intensive care).

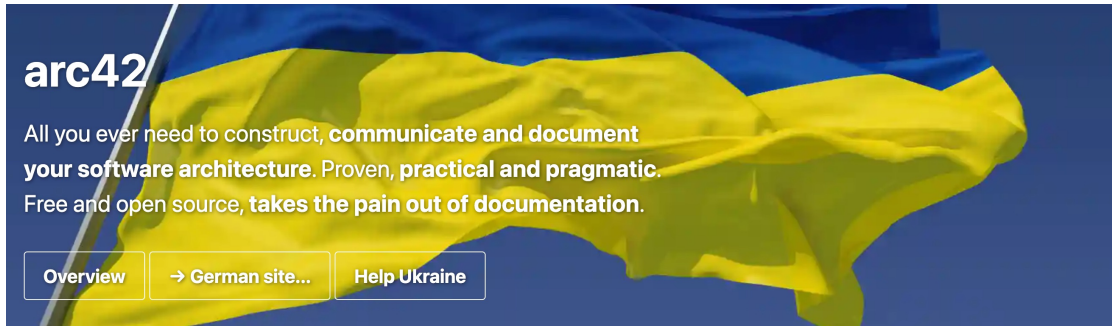
- **Elegance („venustas“)**
  The software/system/service structures makes it intuitive to use, but also easy to maintain and develop.

A MIND FULL OF SOFTWARE ARCHITECTURE

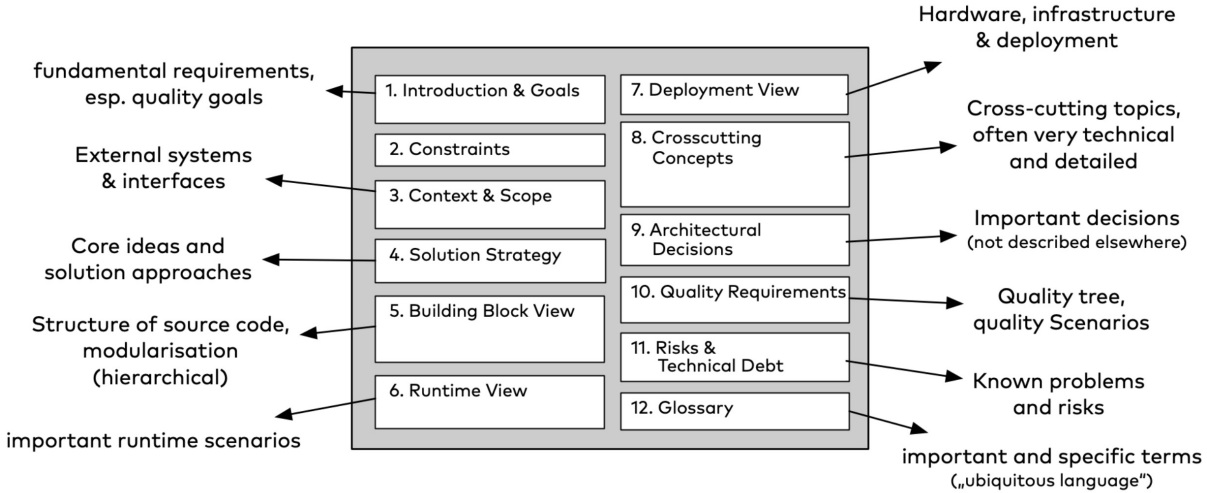


Quelle: https://www.embarc.de/kopf-voller-softwarearchitektur-begriffsbild/

Widespread approach – arc42



https://arc42.org



keep arc42 section, reuse all the experienced tips

- structured around software work (history) → **leverage system architecture**
- „tear down" customer to system level language vs. bring nerd speach to customer level language → **change reading flow: from biz to tech**
- not magenta 😞 → **Magentaify**

Why is it not there when needed?

- **Office tools ←→ „live in a different place, feel like a different world“**
- **Powerpoint ←→ inprecise in detail**
- **Wiki ←→stakeholders with different views and information needs**
- **typical project mechanisms not easily applicable to documentation:**
  - parallel work on different parts
  - keeping old versions (in GIT)
  - Continuous delivery: tag all fragments for a deployment
  - replicated information from JIRA, ProjectNOW, ConfigMgmt, Automation

! outdated before useful

IT architecture in the wild

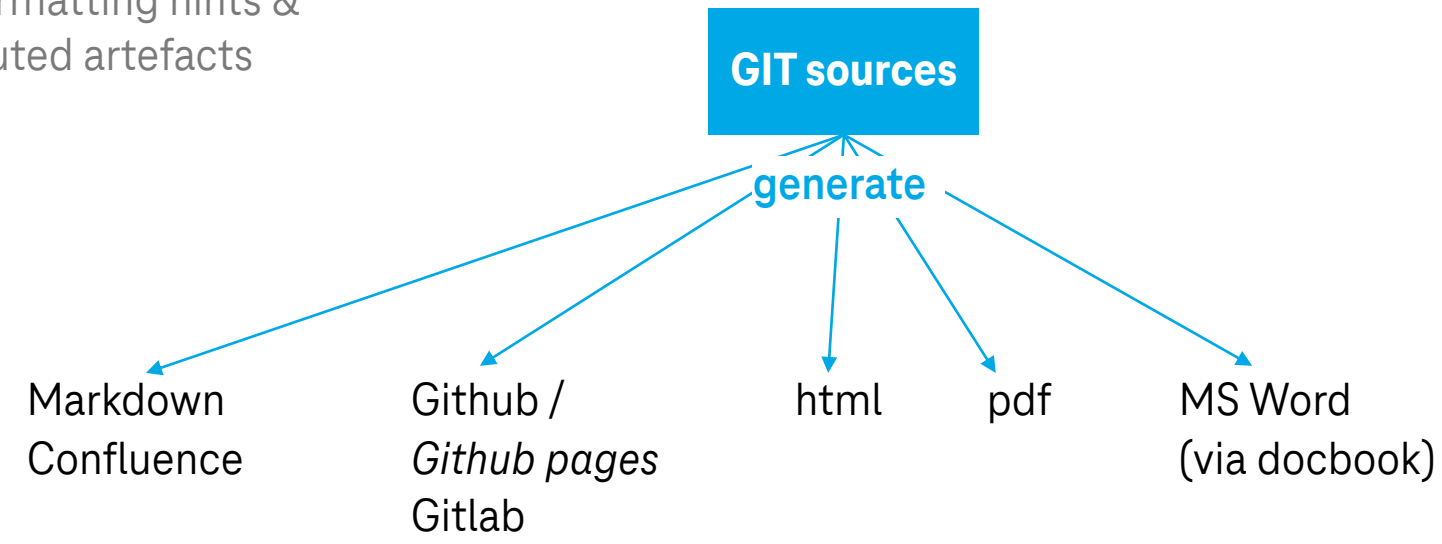
If digital is about coding why not code architecture?

Architecture as OpenSource project - documentation as code

AsciiDoc

= Markdown / wiki language
with formatting hints &
distributed artefacts

Asciidoctor



Dirty hands ahead! (Demo)

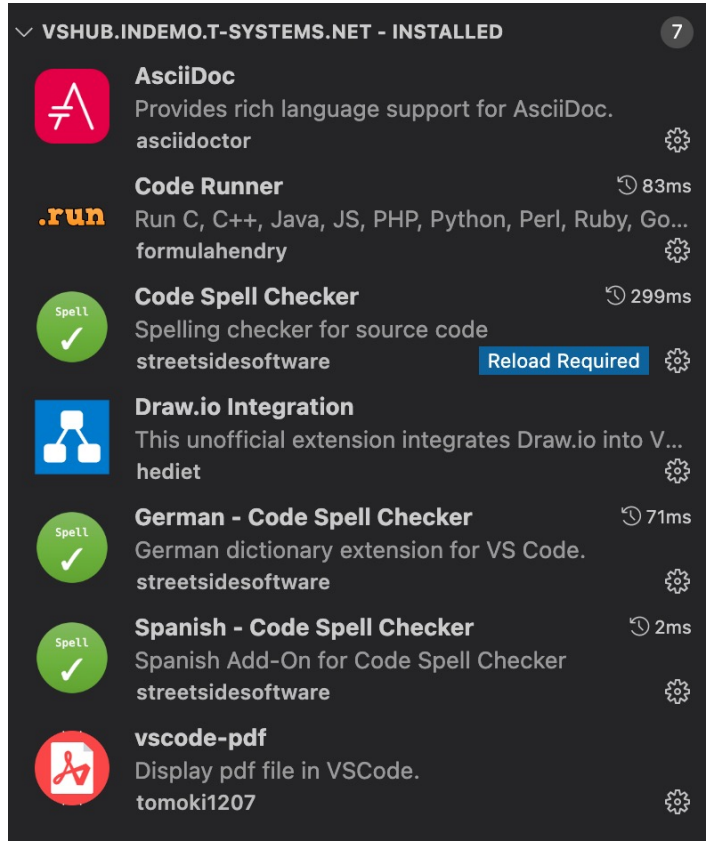
https://vshub.indemo.t-systems.net

git clone https://github.com/tsdicloud/arcdoc-template

Self-installation steps

- **Visual Studio Code, git (OpenSource)**
- **Install AsciiDoctor:**
  https://docs.asciidoctor.org/asciidoctor/latest/install/
- **recommended Visual Studio Code extensions**
  - AsciiDoc (`asciidoctor.asciidoctor-vscode`)
  - Code Spell Checker (`streetsidesoftware.code-spell-checker`)
    + Language packages
    (`e.g. streetsidesoftware.code-spell-checker-german`)
  - Draw.io integration (`hediet.vscode-drawio`)
  - vscode-pdf (`tomoki1207.pdf`)
- **add Preview/spell checker settings to settings.json**
  See Appendix B „Tips, tricks, tools“ in architecture template as copy template.

[git is a problem on T-Workstations]



Business Context

Purpose, goal, stakeholder

Architecture purpose / goals

Purpose

Design and build a 2 family appartment building
with integrated architecture studio
at <parcel>, Pustertal, Südtirol

Goals (not more than 5 +/-2)

- Work out a modern design following „Vorarlberger Schule“
- use sustainable materials wherever affordable
- building produces at least 60% of required energy itself
- Grounding/isolation withstands extreme wet conditions

Purpose drives relevance and priorities,
goals are „general decision trajectories“

Generate your first documentation

```
cd arcdoc-template

asciidoctor-pdf -a toc\
--destination-dir="gen/en/pdf"\
-a imagesdir=../../images\
-a pdf-fontsdir="themes/telefluid-pdf"\
-a pdf-themesdir="themes/telefluid-pdf"\
-a pdf-theme="telefluid-pdf" --trace -verbose\
src/en/architecture-book.adoc
```

→ **copy**

or

→ **use** `adocpdf`

Sidenote: Architecture doc vs README

Many architecture documentations are a mixture of actual architecture (conceptual) artefacts and technical details.

Architecture documentation is NOT an installation manual.

Architecture documentation is NOT a collection of HowTos.

Recommendations

1) Use top-level repo README for installation and HowTo details

`Readme.md` … `Readme.adoc`

2) Shift „HowTo“ content to appendices in architecture documentation

Example: `tools.adoc`

TIP: Keep it where it is most „natural“ to keep information up to date.

Context, Constraints

Architecture models reality

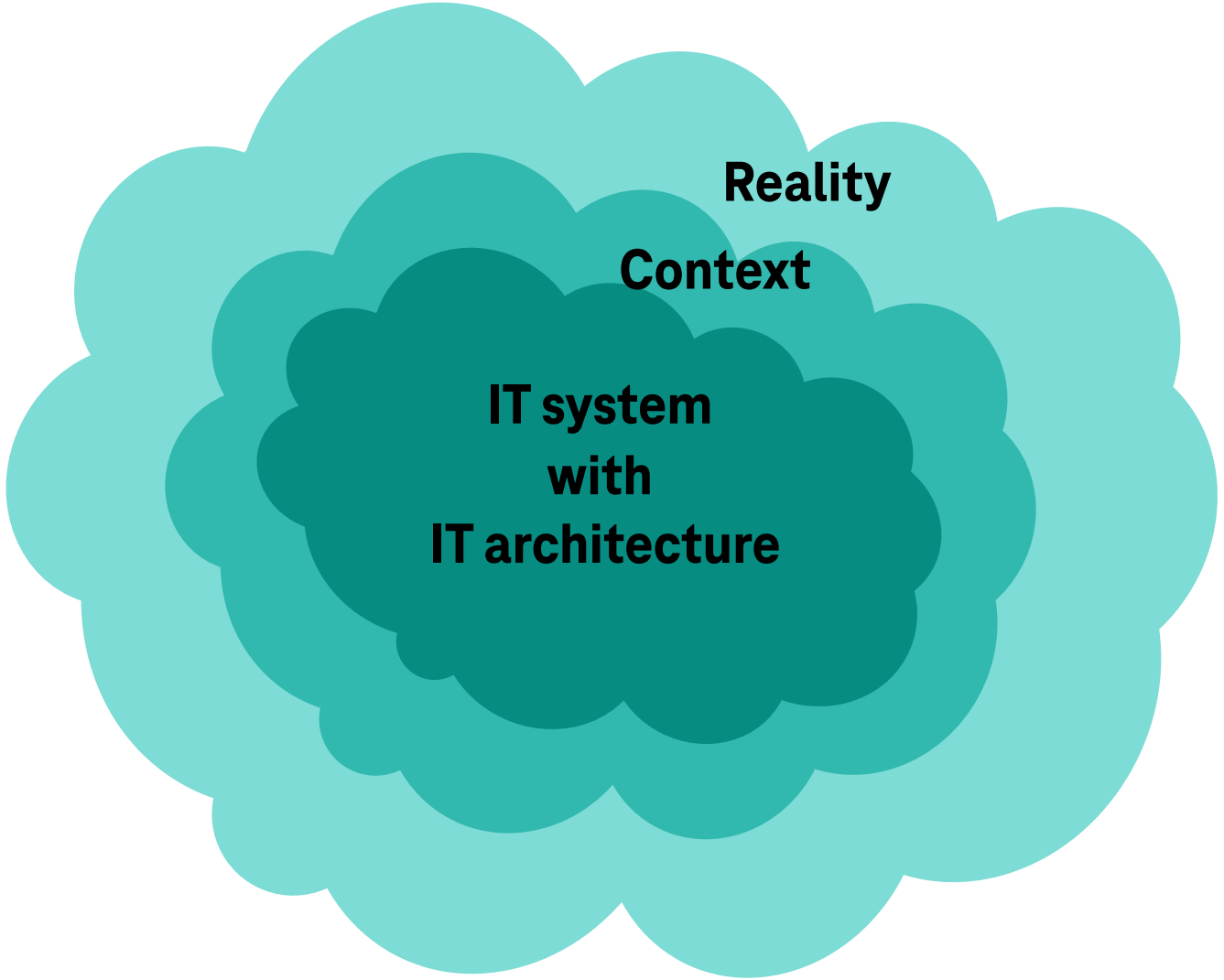
A **model** is
a simplified representation of reality.

Every IT system realizes a model.

Due to this simplification, the model is
only valid in a defined context.

Thus, defining the context of the
architecture model is essential.

Stakeholders are also part of context.



“

The most wanted thing in architecture doc is clean and consistent diagrams.

The most boring and time consuming part in architecture is keeping diagrams clean and consistent.

UML component diagrams ++

→ **Feel free to use any modelling tool - that can produce scalable vector graphics (SVG)**
Note: You can edit any .svg file using `Inkscape` OpenSource tool.

→ **UML is still a good way to sketch IT system details**

→ **„Formal“ UML is usually too restrictive**

→ **System modelling: Hypescaler have their own iconsets most experts expect to see.**

Recommendation

- `diagram.net`

- `PlantUML`

Remember: Diagrams help, but the most powerful modelling tool is code.

Tip: Safeguard work on own Github repo

(Hint: works similar for MagentaCICD, different URL path)

- **Github: Create public/private project**
- **Github: Personal auth2 token:**
  Settings > Developer settings > Personal access token > Fine-grained token > Generate new token
- **Push to GitHub:**

```
# config for your repo
git remote set-url origin https://oauth2:<mytoken>@github.com/<myusername>/<myproject>
git remote user.name  <myemail or myusername>
git remote user.email <myemail>
# commit changes
git commit –m „Change xyz“ <files or wildcards>
# push to remote
git push origin main
```

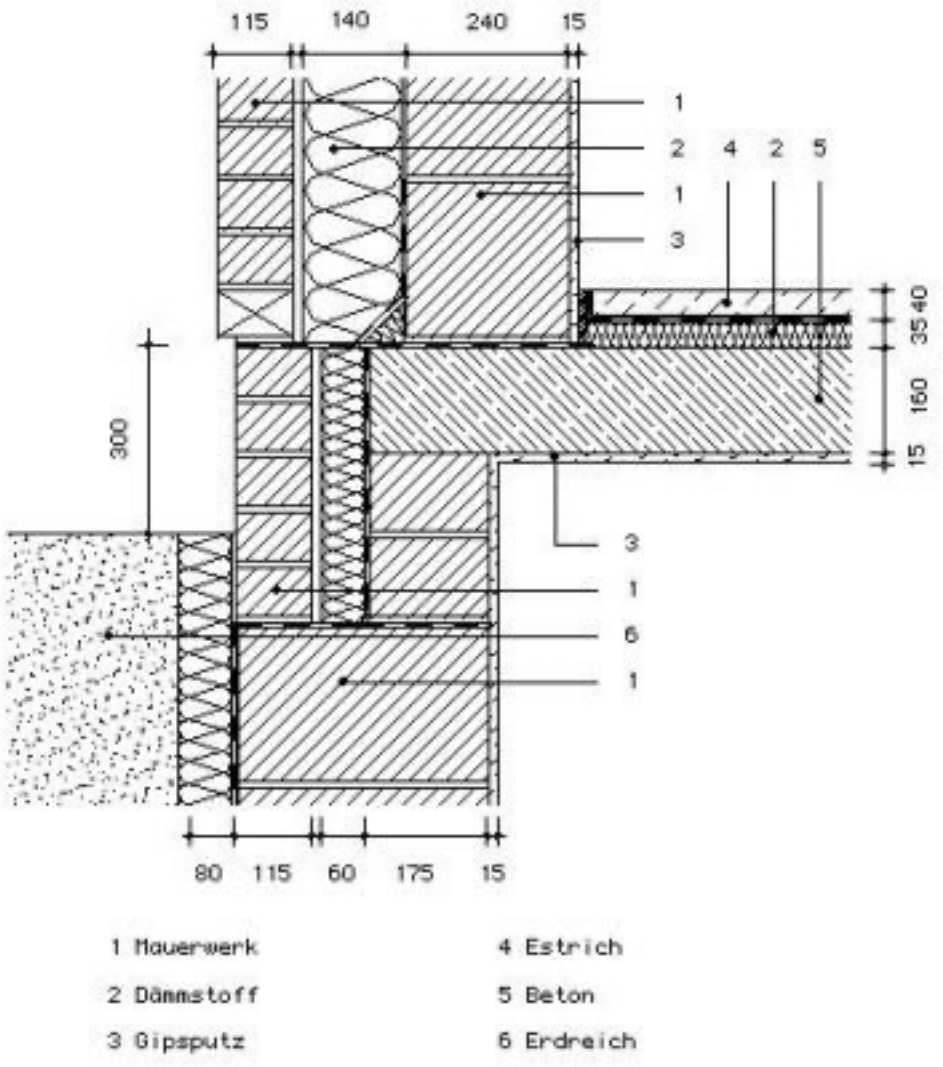
Congrats, you own your work now!

[For MagentaCICD:]

```
git remote set-url origin https://oauth2:<mytoken>@gitlab.devops.telekom.de/<path_to_myproject>
```

Building blocks

Views = architecture (de-)construction drawings



An architecture view [...] expresses the architecture of the system of interest from the perspective of one or more stakeholders to address specific concerns,[...]. An architecture view consists of one or more architecture models [ISO/IEC/IEEE 42010].

→ Take care that the „perspective“ is clear, i.e. what view is effective for a section?

“

An architecture tapestry can be nice and impressive for discussions.

But tapestry is never a complete architecture documentation.

“Picture puzzles“ ←→ structured architecture

Blackbox / Whitebox

Blackbox – Whitebox metaphore

Blackbox

Whitebox

Services view

An IT architecture template

T-Systems, PU Digital - BA Emerging Industries ,, : internal

T Systems

System architecture / concepts walkthrough

Quality & Implementation

Steer with principles

„Form follows function.“

„Simplicity
Friendliness
Minimalism
Precision
Focus“

Quality scenarios

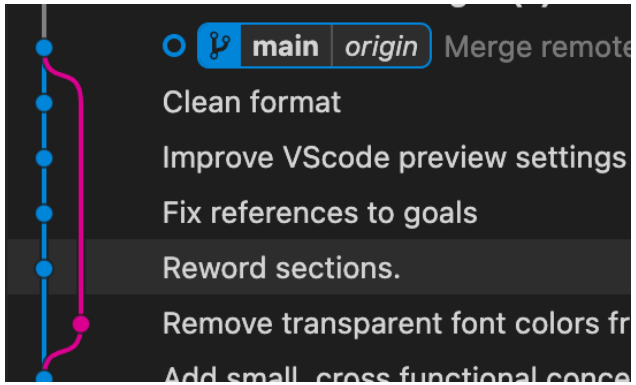
OpenSource-like architecture

(OpenSource) project mechanisms

Parallel work

„this section is new but incomplete“

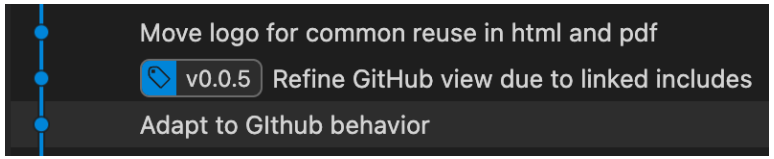
`git branch …`



Historical versions / version mapping

„different concept
in the older release“

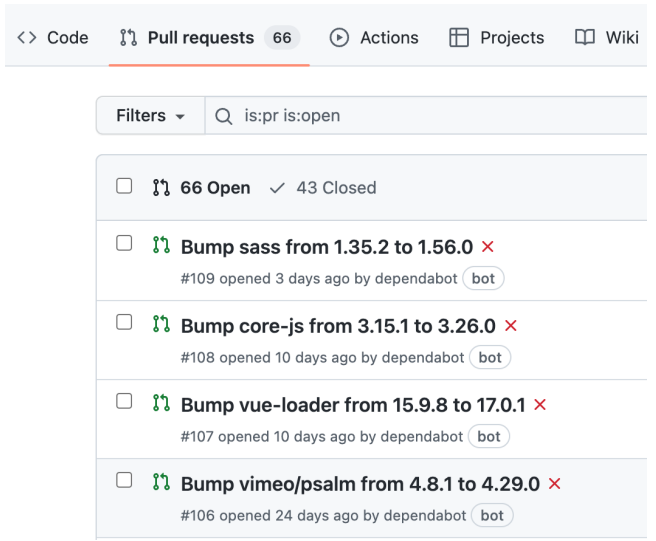
`git tag …`



Pull requests

Project community
architecture review process

New pull request



Generators

Attribute / commandline

by attribute & commandline

```
alias arcnumber="git tag …"
alias arcdate="git log -q -- …"

asciidoctor-pdf … \
-a revdate="$(arcdate)"\
-a revnumber=", $(arcnumber)" …
```

Text template generators

by text template generators

e.g. Jinja2 et al.

xyz.adoc.tmpl:

```
|====
|Id|Subject
{% for ticket in tickets %}
|{{ticket.id}}|{{ticket.subject}}
{% endfor %}
|====
```

Scripting

by (ruby) scripting

e.g. JIRA integration

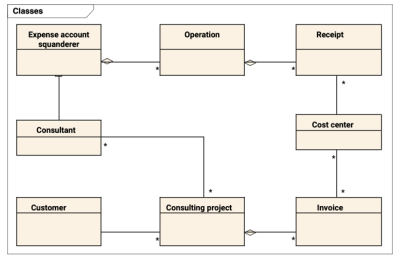
example.adoc:

```
The effect is related to
jira:DOC-1234[] .

=== Technical debts
jiraIssues:DOC[jql=
"resolution='Unresolved' ORDER BY
priority DESC"]
```

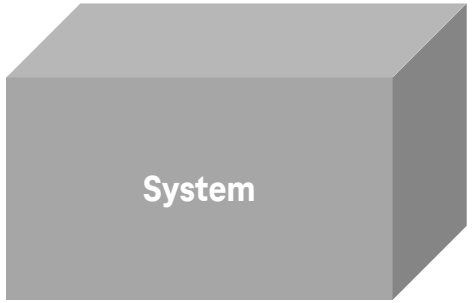
Generated diagrams

Old: truth in diagram („model-driven“)



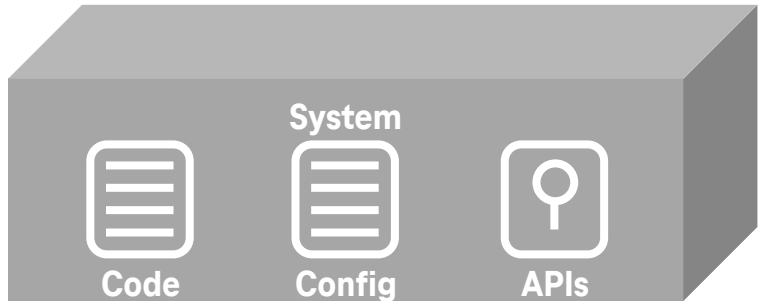
Semantic precision
(→ metadata)
„Tapestry programming“

generate



„Visual debugging“

New: truth in code



generate

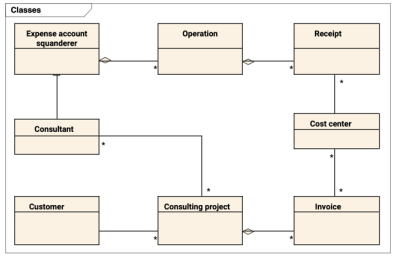


diagram.net/draw.io: it is a code library!

https://jgraph.github.io/mxgraph/

https://github.com/plantuml/plantuml

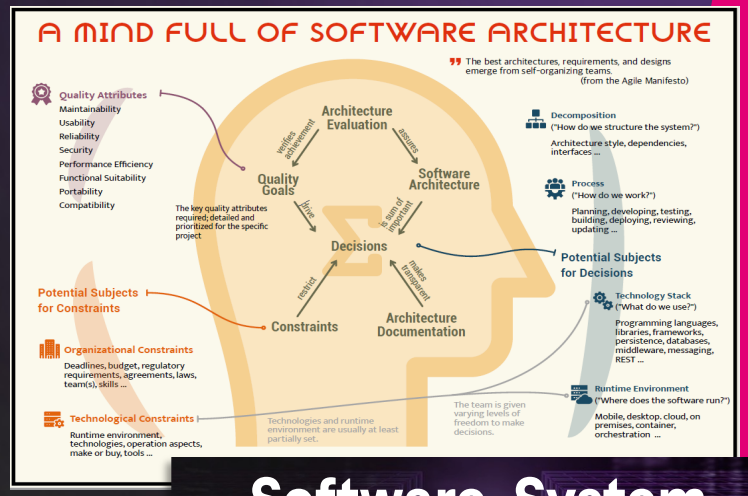
Finalize architecture template walkthrough

THANK YOU!

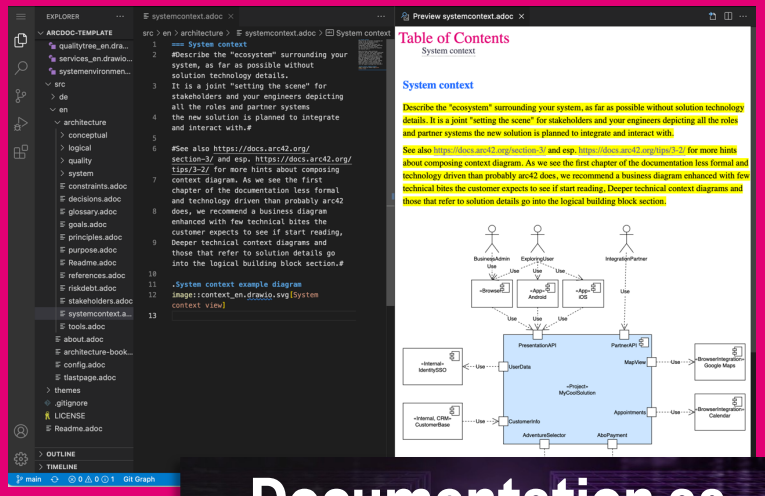
youlearn days 8–9 Nov

Let‘s start hands on – IT architecture

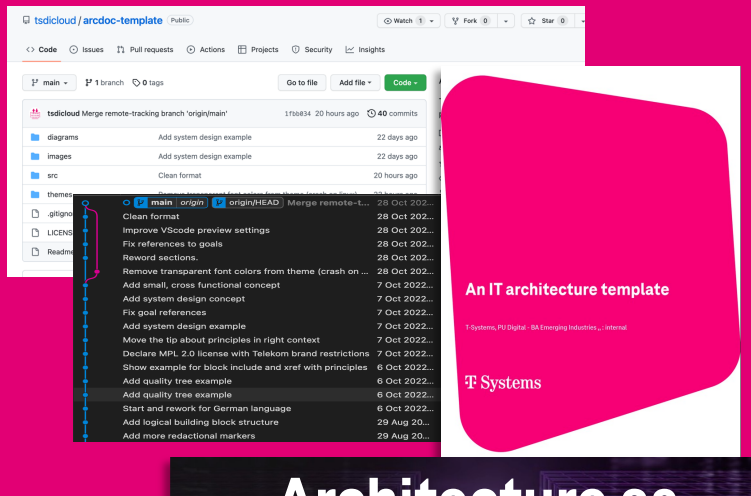
Wednesday, 9th, 11:55 – 14:00
TSI Hub (please register)



Software, System & Quality architecture



Documentation as Code



Architecture as OpenSource