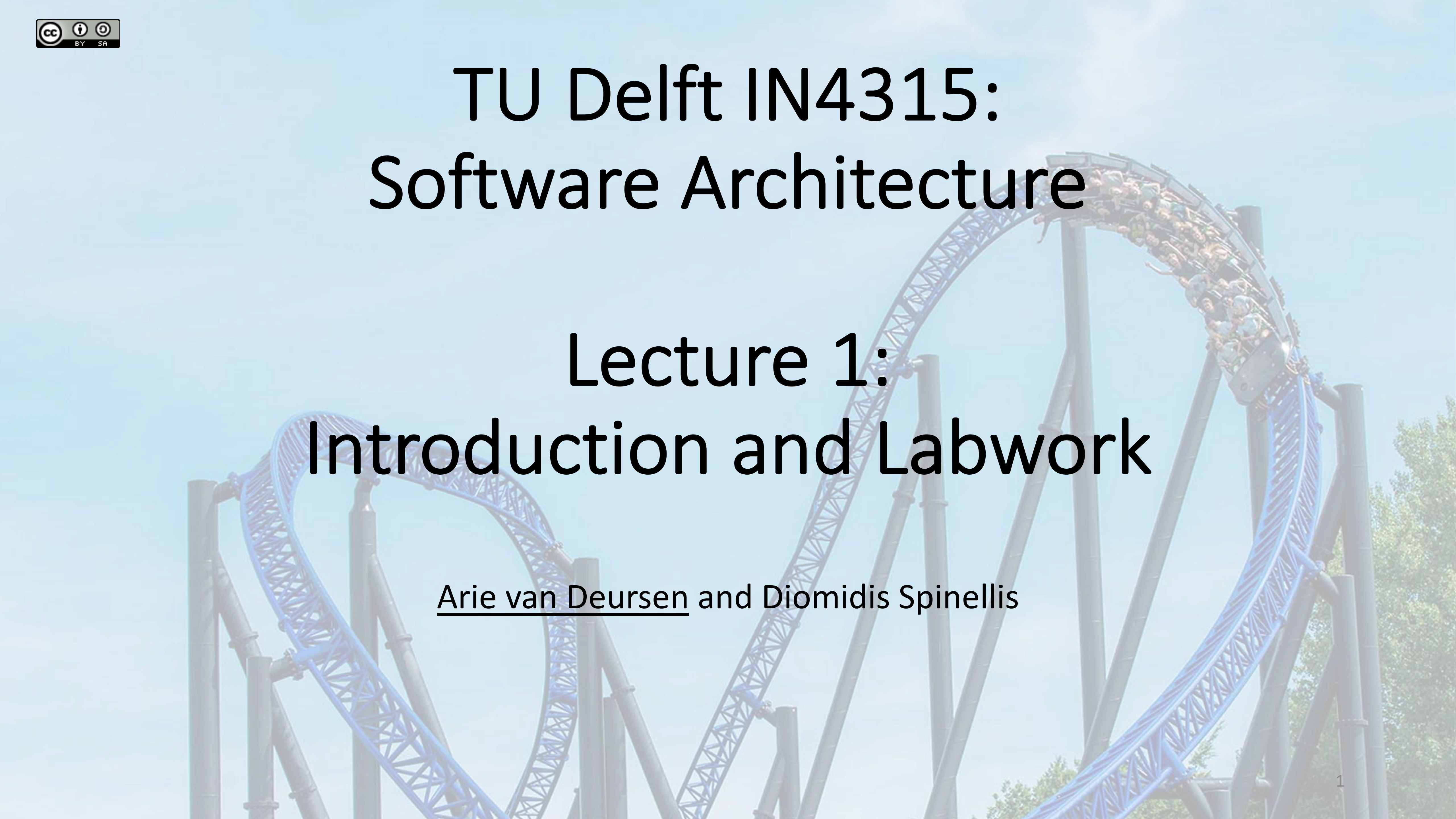


TU Delft IN4315: Software Architecture

Lecture 1: Introduction and Labwork

Arie van Deursen and Diomidis Spinellis

IN4315 and Covid

Covid isn’t over yet

Be respectful of everybody’s way of coping with Covid

Follow the rules set by the government and university

Contact teachers / teaching assistants if you’re affected

Help each other

Hybrid Online/On campus Lectures

On campus

- Max 75 room capacity
- Enroll via the queue
- No symptoms / negative test
- Please wear a mask when:
  - Walking
  - Sitting < 1.5m distance

Remote

- Please participate via chat
  - Answer teacher's questions
  - Ask questions
- Login with TU Delft credentials
- Lectures and chat will be recorded
- TAs will monitor chat and relay

Erik Sennema

Raoul Kalisvaart

Covid and IN4315 group work

- All lab-work in teams of four
- Encouraged and permitted to meet in person frequently
- Fallback: Doing everything remote is possible

- You or your team members may get sick this quarter
- Please start in time with assignments
- Plan for need to step in if one team member gets sick
- If needed, discuss fallback options with TAs / teachers

Lean Architecture
for Agile Software Development
by James O. Coplien & Gertrud Bjørnvig
Building software as if people mattered
WILEY

The Lean Secret: Everybody, All together, Early On

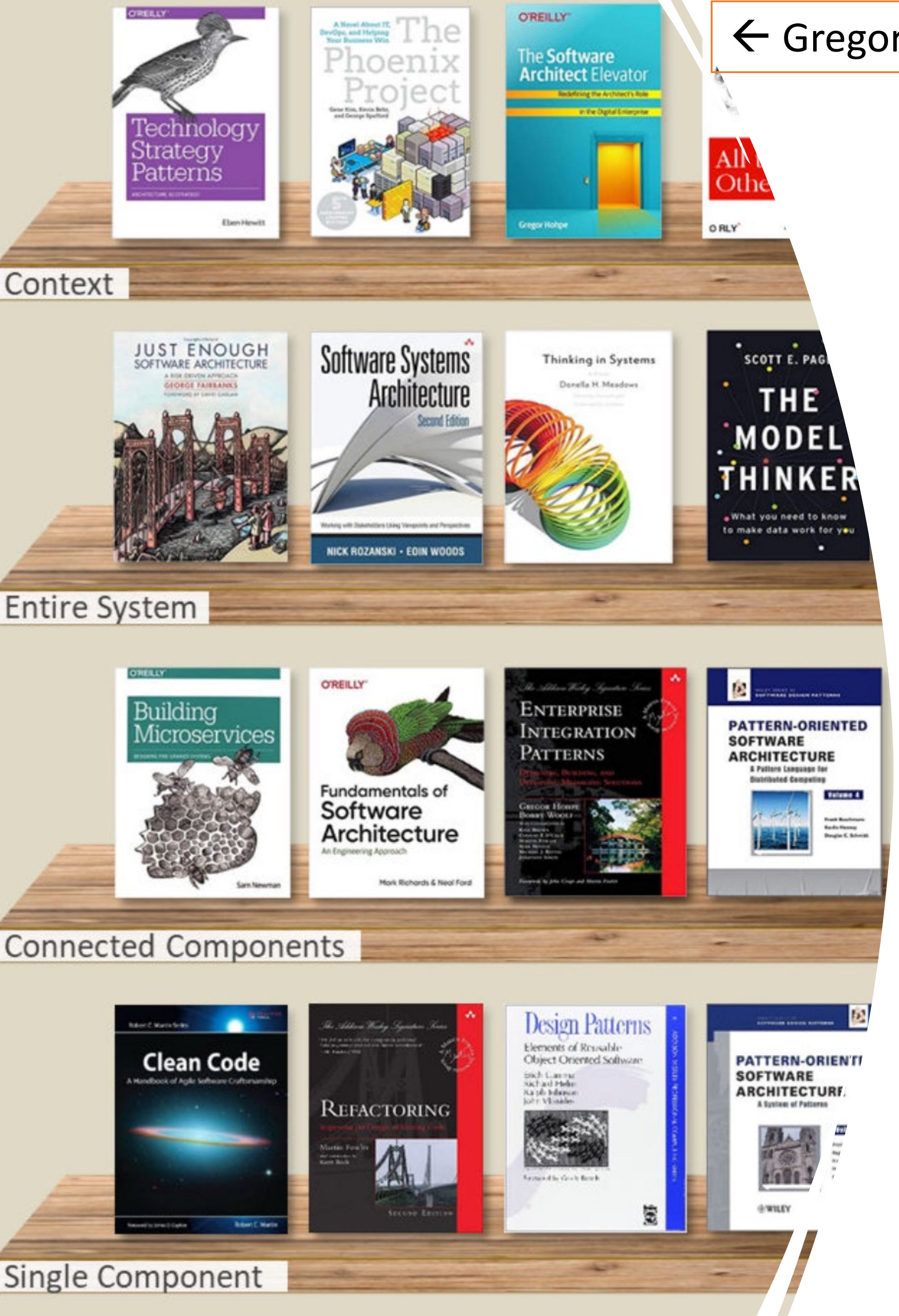
Who am I?

- Professor in Software Engineering at TU Delft
- Research interests:
  - developer productivity, software testing, trustworthy AI, SE4AI, AI4SE
- Scientific director: AI for Fintech Research (AFR)
  - 5 year project with ING, 10 PhD students, 10+ MSc/Bsc students per year
- Head of the Department of Software Technology at TU Delft
  - ±250 people working on algorithmics, embedded systems, programming languages, web information systems, distributed systems, sw. engineering
- Co-founder of two companies: SIG and PerfectXL

Today’s Kickoff Lecture

1. Welcome
2. What is it that software architects do?
3. What are we going to do together?
   - Labwork, coding, writing
   - Way of working
   - Peer review and grading
4. What is the structure of this course?
   - Schedule & deadlines
5. [A sneak preview of the first assignment]

← Gregor Hohpe's "Architect's Bookshelf"

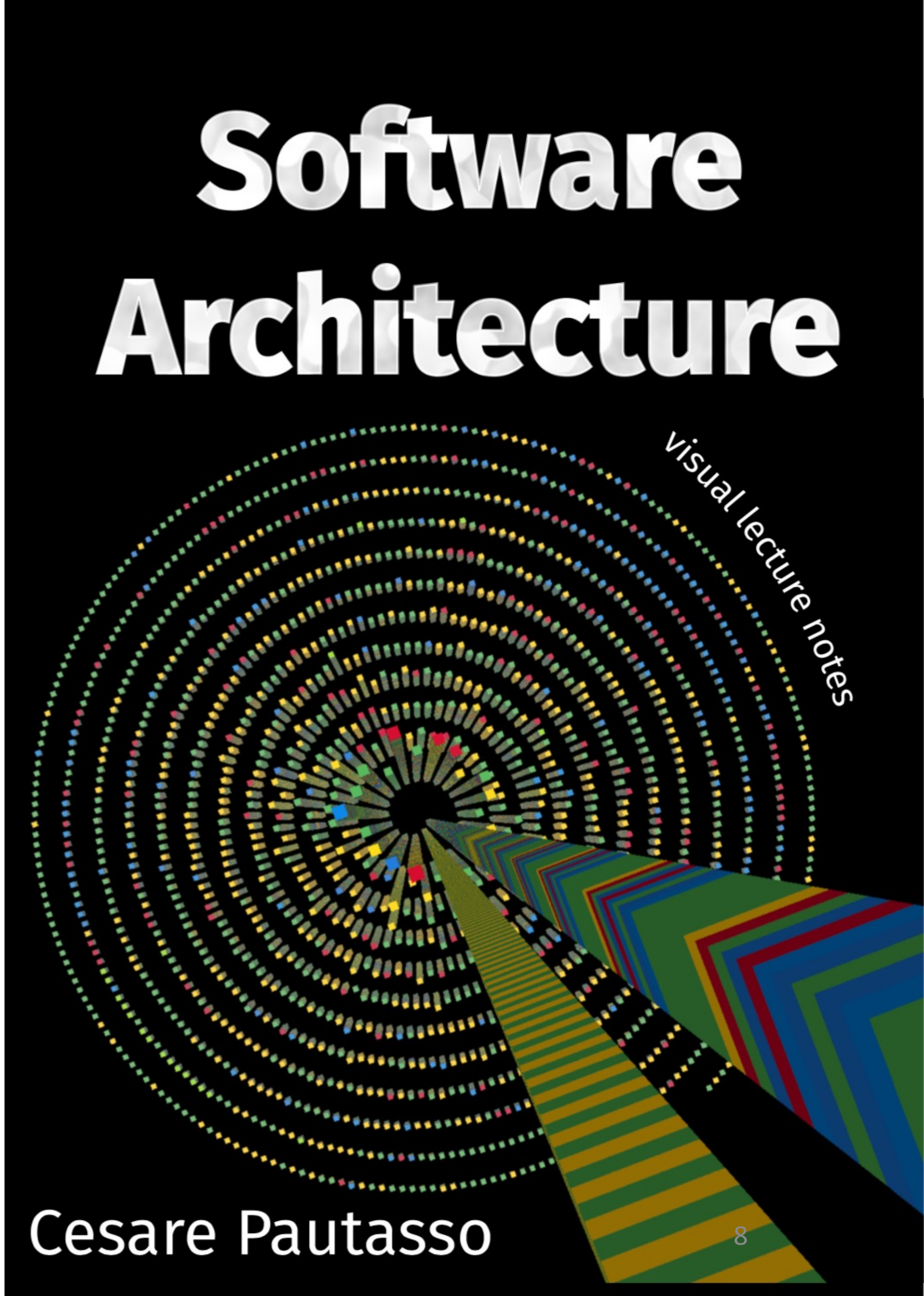

Technology Strategy Patterns
The Phoenix Project
The Software Architect Elevator
Context
Just Enough Software Architecture
Software Systems Architecture
Thinking in Systems
The Model Thinker
Entire System
Building Microservices
Fundamentals of Software Architecture
Enterprise Integration Patterns
Pattern-Oriented Software Architecture
Connected Components
Clean Code
Refactoring
Design Patterns
Pattern-Oriented Software Architecture
Single Component

A Software Architecture Body of Knowledge

- A very broad topic
- Mix of people skills, technical skills, and domain sensitivity
- Reusable architectural knowledge often *abstract*
- Architects need ability to make such knowledge *concrete* in their own context
- The architect is never finished learning

https://se.ewi.tudelft.nl/delftswa/suggested-reading

1. Introduction
2. Quality Attributes
3. Definitions
4. Modeling Software Architecture
5. Modularity and Components
6. Reusability and Interfaces
7. Composability and Connectors
8. Compatibility and Coupling
9. Deployability, Portability and Containers
10. Scalability
11. Availability and Services
12. Flexibility and Microservices



What do Software Architects do?

1. What are the key responsibilities of the software architect?

2. What makes a great software architect?

3. Can you name examples of (well-known) software architects?

Please enter your thoughts in the chat!

Software Architects in Software History

- Margaret Hamilton – Apollo moon lander
- Steve Jobs – Apple
- Erich Gamma – Visual Studio Code
- Adele Goldberg – Smalltalk
- Ken Thomson & Dennis Ritchie – Unix
- Fred Brooks – IBM OS360
- Grace Hopper – Flow-Matic / Cobol
- Ada Lovelace – The first?

For more, see https://computingthehumanexperience.com/people/

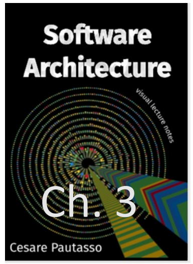
Key Responsibilities: *1. Make decisions*

- Architects carry overall responsibility for all technical decisions

- Lead an organization that takes the right decisions
- Willing and able to take them where needed

- High level decisions (”styles”) governing overall architecture
- Ability to *defer* decisions when safe to do so
  - Keep options open in light of future developments

Thus: The Great Software Architect …
… must be Technical Authority



Software Engineering

- Excellent Software Engineering skills
- Promote good development practices
- Solve the hard problems
- Lead technical development team by example
- Understand impact of decisions
- Defend architectural design decisions
- Plan and manage software releases

Technology

- Know and understand relevant technology
- Evaluate and influence choice of 3rd party frameworks, components and platforms
- Track technology evolution
- Know what you do not know

"Coding Architect": Join team, contribute code, program in pair

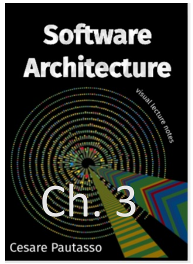
Key Responsibilities: *2. Talk to Business*

- Architects can explain business impact of technical decisions taken

- Traditionally: Map “problem domain” to “solution domain”
- Modern: Turn technical capabilities into new business opportunities

- Translate technical risk into business risk
- Willing and able to easily switch technical and business perspectives

Thus, the Great Architect ...
... must be Great Communicator

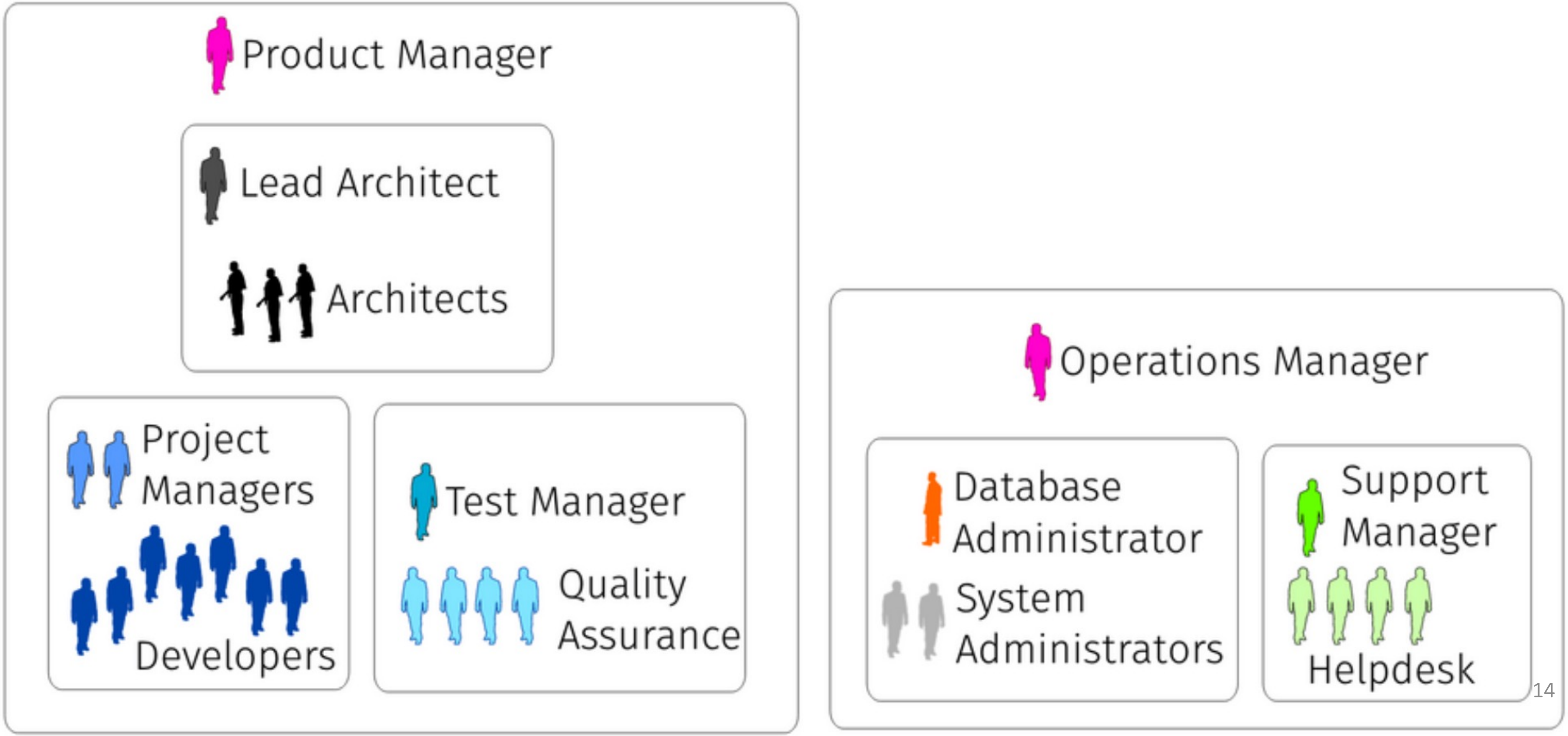


Software Architecture

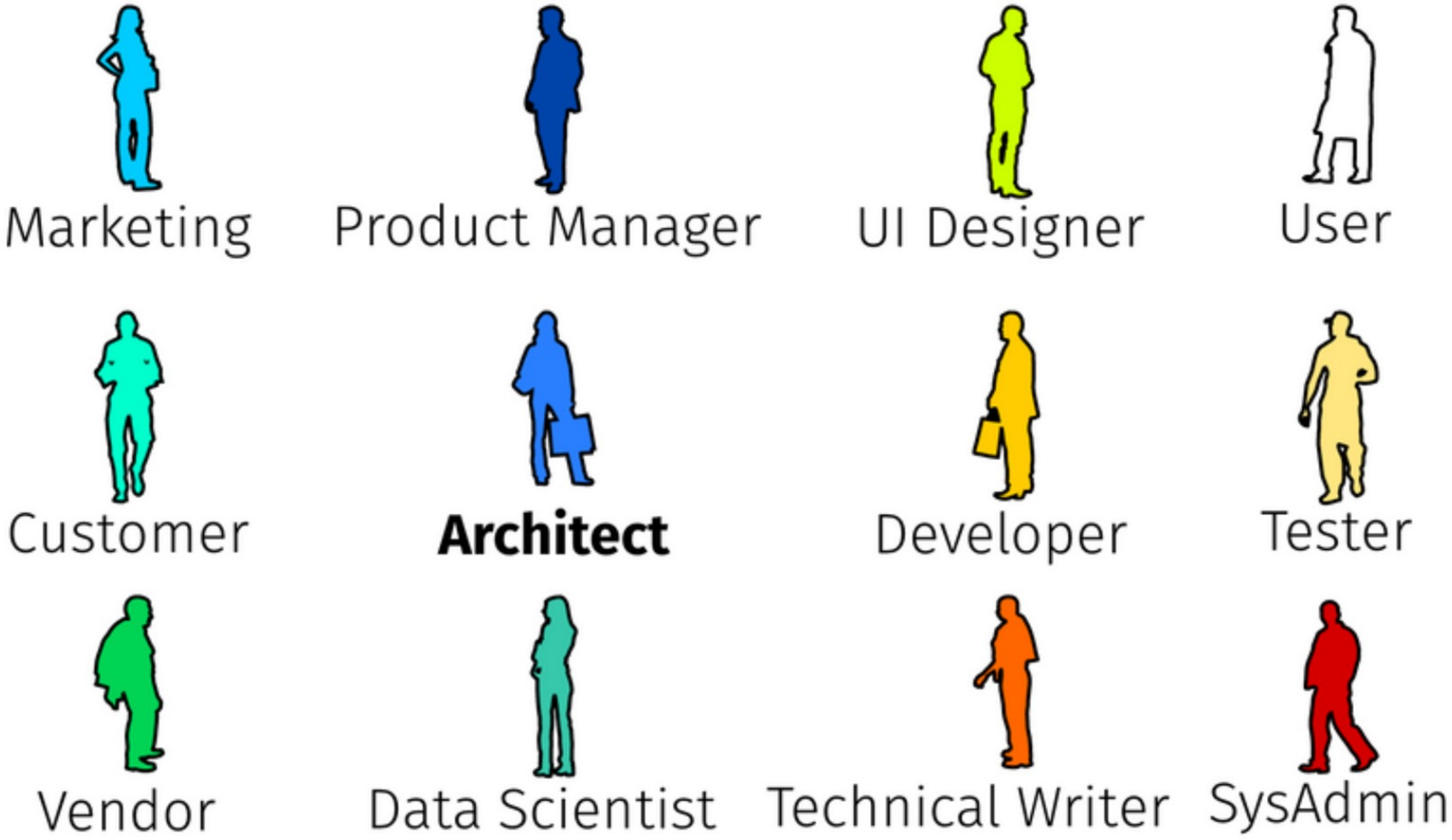
visual lecture notes

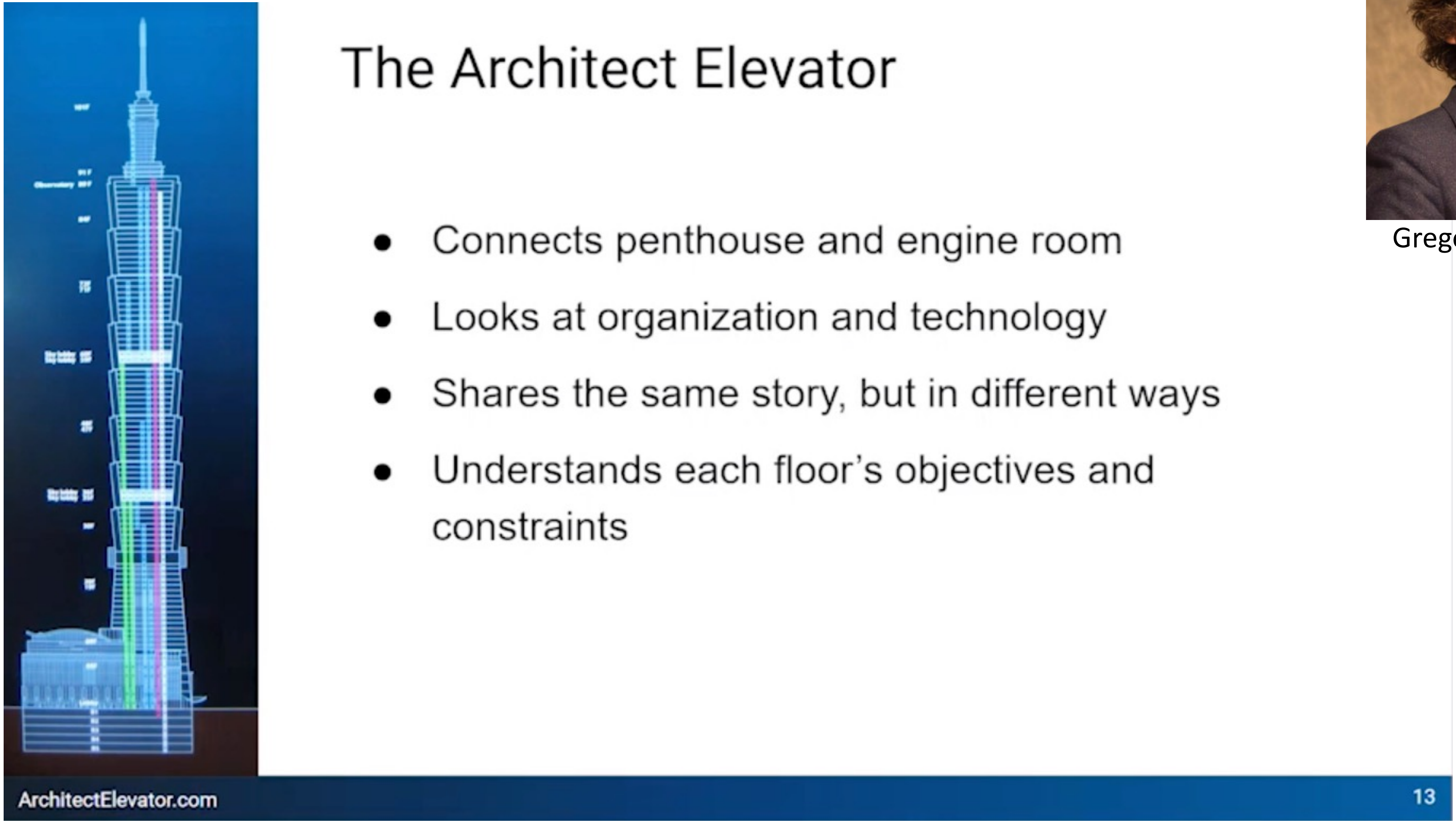
Ch. 3

Cesare Pautasso



People to Talk to





The Architect Elevator

- Connects penthouse and engine room
- Looks at organization and technology
- Shares the same story, but in different ways
- Understands each floor's objectives and constraints

ArchitectElevator.com

13

Gregor Hohpe

https://www.youtube.com/watch?v=Zq2VcRZmz78

Shared Story = Product Vision

- Clear vision of what the product is and will do
- Simple, compelling, articulated, shared
- Comes with a credible roadmap towards this vision.
- Expressible in terms that are understandable to end users
- Driven / enabled by sound architectural foundations

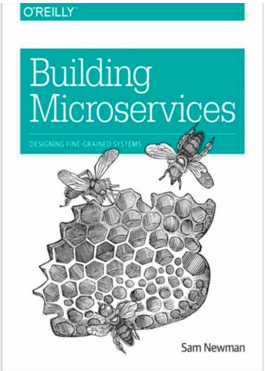
- Co-production of product manager and architect

Key Responsibilities: *3. Embrace Change*

- Architects enable (embrace!) change

- Architects are “living in the first derivative”
- Manage change-induced risk
- Anticipate change
- Defer decisions that would block change
- Safeguard successful rate of change
- Optimize processes to accelerate rate of change
- Work with incomplete information

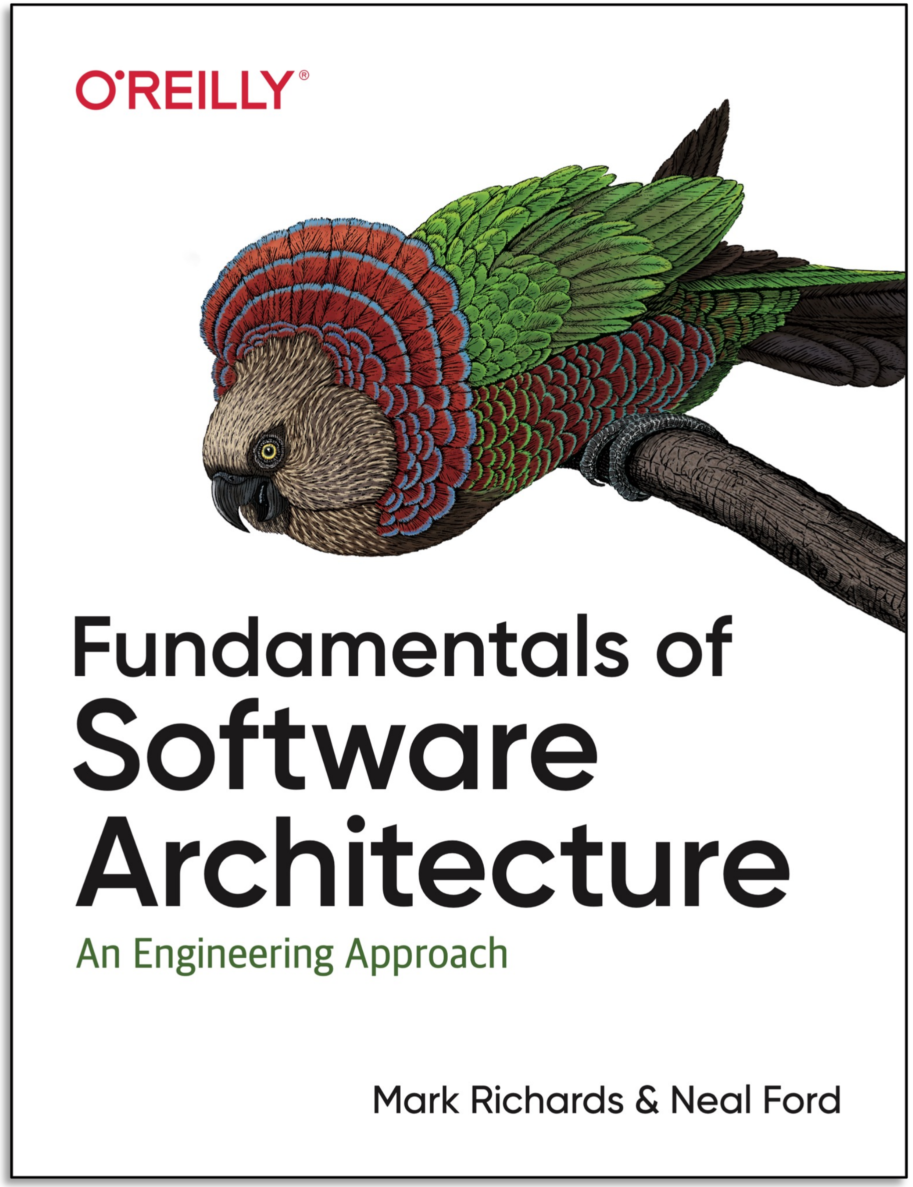
Sam Newman: Core Responsibilities of the "Evolutionary Architect"



- **Vision**: Ensure there is *a clearly communicated technical vision* for the system that will help your system *meet the requirements of your customers and organization*
- **Empathy:** Understand impact of your decisions on end users and team
- **Collaboration:** Engage with as many people as possible to realize vision
- **Adaptability:** Adjust vision when needed
- **Autonomy:** Balance autonomy and overall consistency
- **Governance:** Ensure system built meets vision

“Expectations of an Architect”

- Make architecture decisions
- Continually analyze the architecture
- Keep current with latest ~~trends~~ technological developments
- Ensure compliance with decisions
- Diverse exposure and experience
- Have business domain knowledge
- Possess interpersonal skills
- Understand and navigate politics



O'REILLY®

Fundamentals of Software Architecture

An Engineering Approach

Mark Richards & Neal Ford

Exercises

Reflect on a software development project, or even better an organization you are familiar with:

1. Who were the architects?
2. How did they fulfill their three key responsibilities?
3. Who were the architects mostly talking to? How many floors did they span?
4. How explicit was the (technical) vision? What was this vision?
5. What did the project do to optimize the rate of change?
6. What do you see as architectural do’s and don’ts in this project?

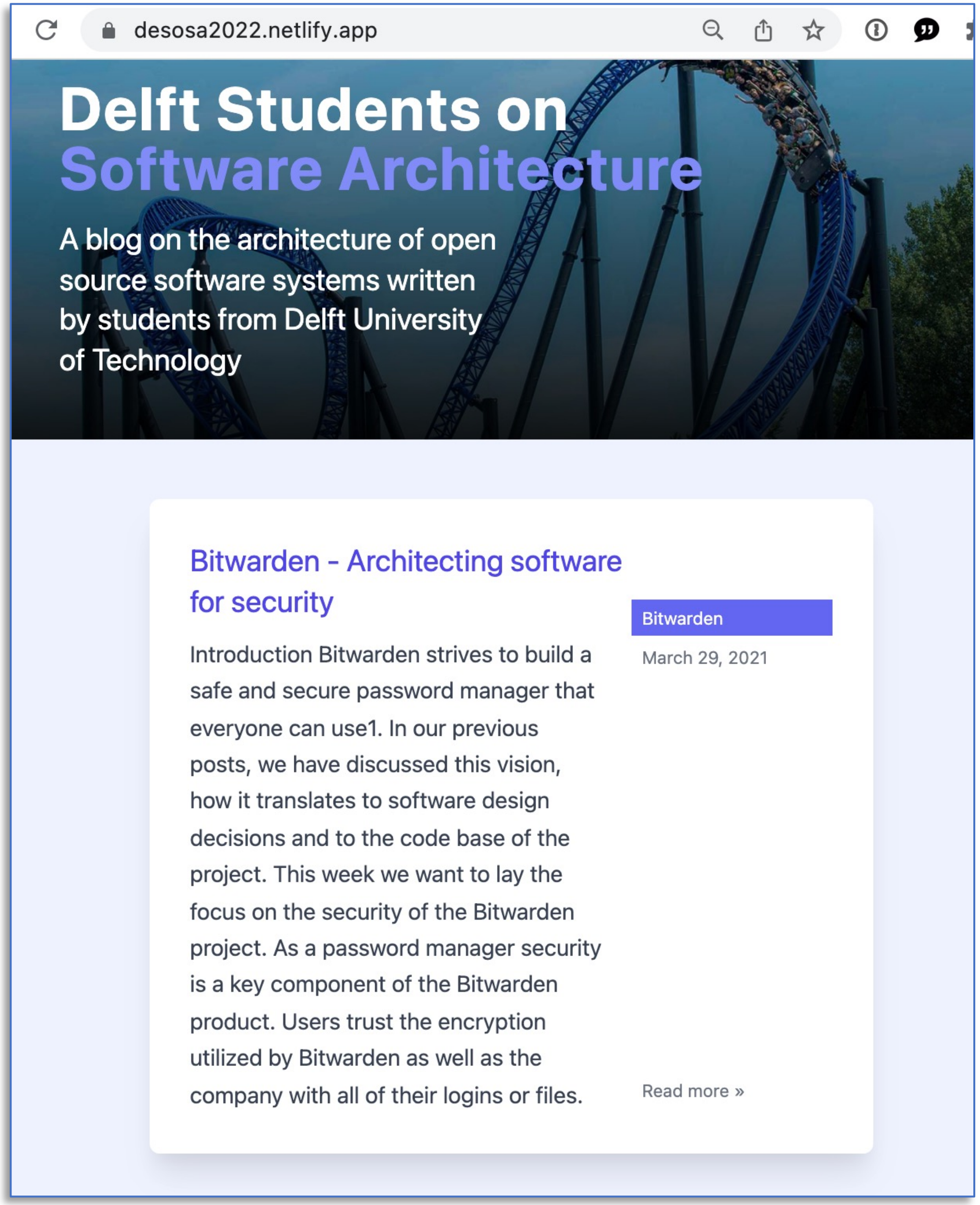
Further Reading

- Martin Fowler. Who needs an architect? IEEE Software, 2003 https://martinfowler.com/ieeeSoftware/whoNeedsArchitect.pdf
- Gregor Hohpe. The Architect Elevator — Visiting the upper floors. https://martinfowler.com/articles/architect-elevator.html
- Gregor Hophe. The Software Architect Elevator. O'Reilly, 2020. Chapters 1-5
- Sam Newman. Building Microservices. O'Reilly, 2015. Chapter 2.
- Mark Richards and Neal Ford. Fundamentals of Software Architecture. O'Reilly, 2020. Chapter 1.
- Cesare Pautasso. Software Architecture. Leanpub, 2020. Chapter 3 https://leanpub.com/software-architecture/

IN4315 Labwork:

Software architecture is about

- **People**:
  - You work in teams of four
- **Real systems**:
  - You adopt an open source system
  - To analyze, describe, and improve
  - And interact with its developers
- **Communication:**
  - You will write and present



Delft Students on Software Architecture

A blog on the architecture of open source software systems written by students from Delft University of Technology

Bitwarden - Architecting software for security

Bitwarden

March 29, 2021

Introduction Bitwarden strives to build a safe and secure password manager that everyone can use1. In our previous posts, we have discussed this vision, how it translates to software design decisions and to the code base of the project. This week we want to lay the focus on the security of the Bitwarden project. As a password manager security is a key component of the Bitwarden product. Users trust the encryption utilized by Bitwarden as well as the company with all of their logins or files.

Read more »

Team Formation

- 4 != 3, 4 != 5
- Aim for a diverse team:
  - Git knowledge, programming skills, writing, presentation, domain knowledge
  - Bachelor background, master track, geography, ...

- Brightspace discussion forum “Partners Wanted”
- Form your group on Brightspace (Collaboration / Groups)

- **DEADLINE: Monday February 14, 17:00**

Team *Coach*

- One of the teachers / TAs / PhD students
- One per team
- 30m meetings in week 2, 4, 6 of the course
  - Offer short presentation of progress
  - Ask-them-anything
- Think how to map theory to your system
- Think about interesting aspects of your system to study
- Online or in person

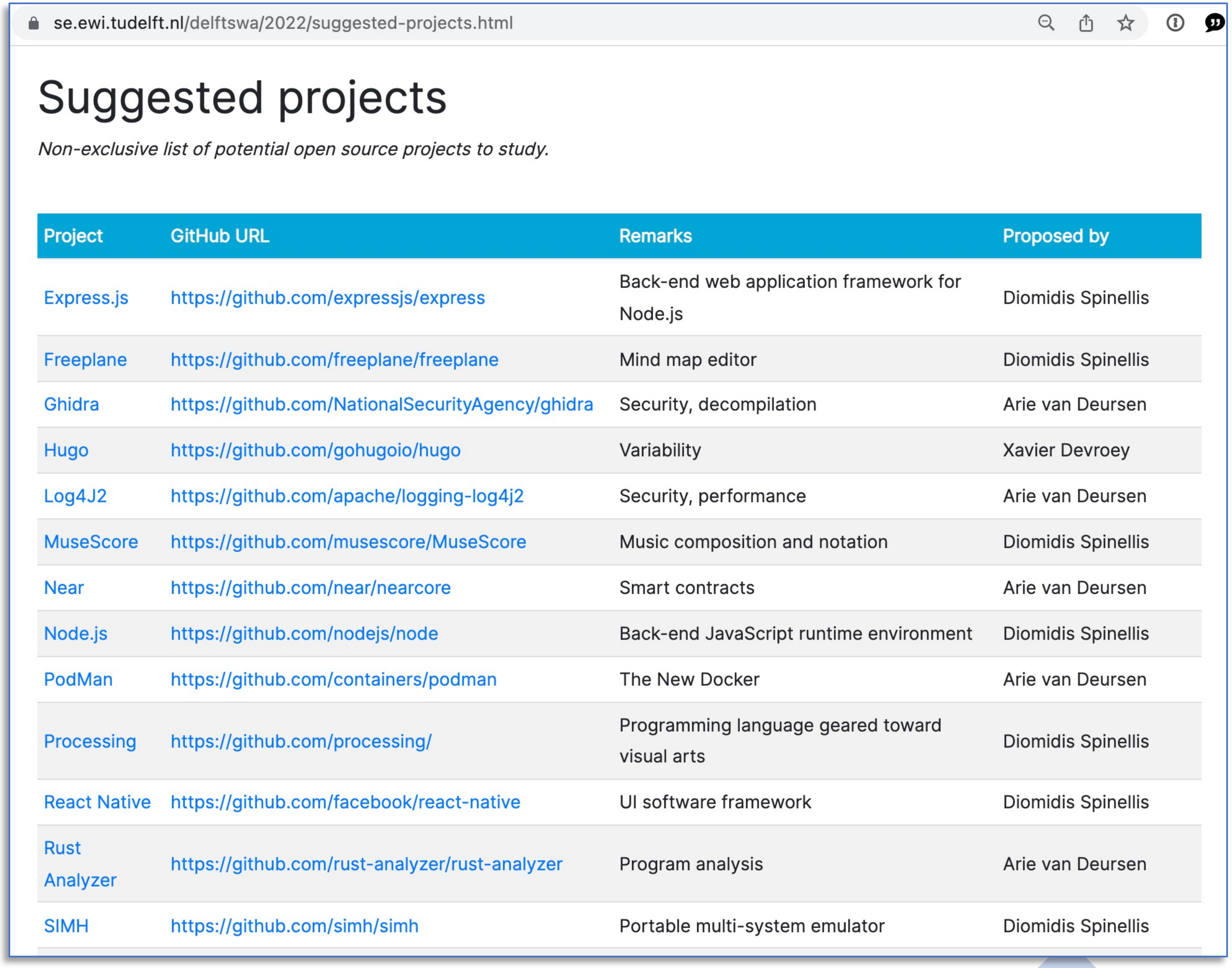
Coaches:
Leonhard Applis
Arie van Deursen
Raoul Kalisvaart
Zoe Kotti
Lorena Poenaru-Olaru
Erik Sennema
Thodoris Sotiropoulos
Diomidis Spinellis

System Selection

- A system your team is passionate about
- A system that is sufficiently active:
  - Open to external contributions
  - At least one accepted pull request per day
- A system that's not too simple
- A system that may be very complex, but then possibly with meaningful sub-system to focus on.
- Written in any programming language you master
- Selection must be approved by TAs

Use Brightspace
"Claim your project" forum.

DEADLINE:
Monday February 14, 17:00



Suggested projects

Non-exclusive list of potential open source projects to study.

Project	GitHub URL	Remarks	Proposed by
Express.js	https://github.com/expressjs/express	Back-end web application framework for Node.js	Diomidis Spinellis
Freeplane	https://github.com/freeplane/freeplane	Mind map editor	Diomidis Spinellis
Ghidra	https://github.com/NationalSecurityAgency/ghidra	Security, decompilation	Arie van Deursen
Hugo	https://github.com/gohugoio/hugo	Variability	Xavier Devroey
Log4J2	https://github.com/apache/logging-log4j2	Security, performance	Arie van Deursen
MuseScore	https://github.com/musescore/MuseScore	Music composition and notation	Diomidis Spinellis
Near	https://github.com/near/nearcore	Smart contracts	Arie van Deursen
Node.js	https://github.com/nodejs/node	Back-end JavaScript runtime environment	Diomidis Spinellis
PodMan	https://github.com/containers/podman	The New Docker	Arie van Deursen
Processing	https://github.com/processing/	Programming language geared toward visual arts	Diomidis Spinellis
React Native	https://github.com/facebook/react-native	UI software framework	Diomidis Spinellis
Rust Analyzer	https://github.com/rust-analyzer/rust-analyzer	Program analysis	Arie van Deursen
SIMH	https://github.com/simh/simh	Portable multi-system emulator	Diomidis Spinellis

Learn from Open Source Architects: *Offer them a Contribution*

- Make a useful contribution to the system you study

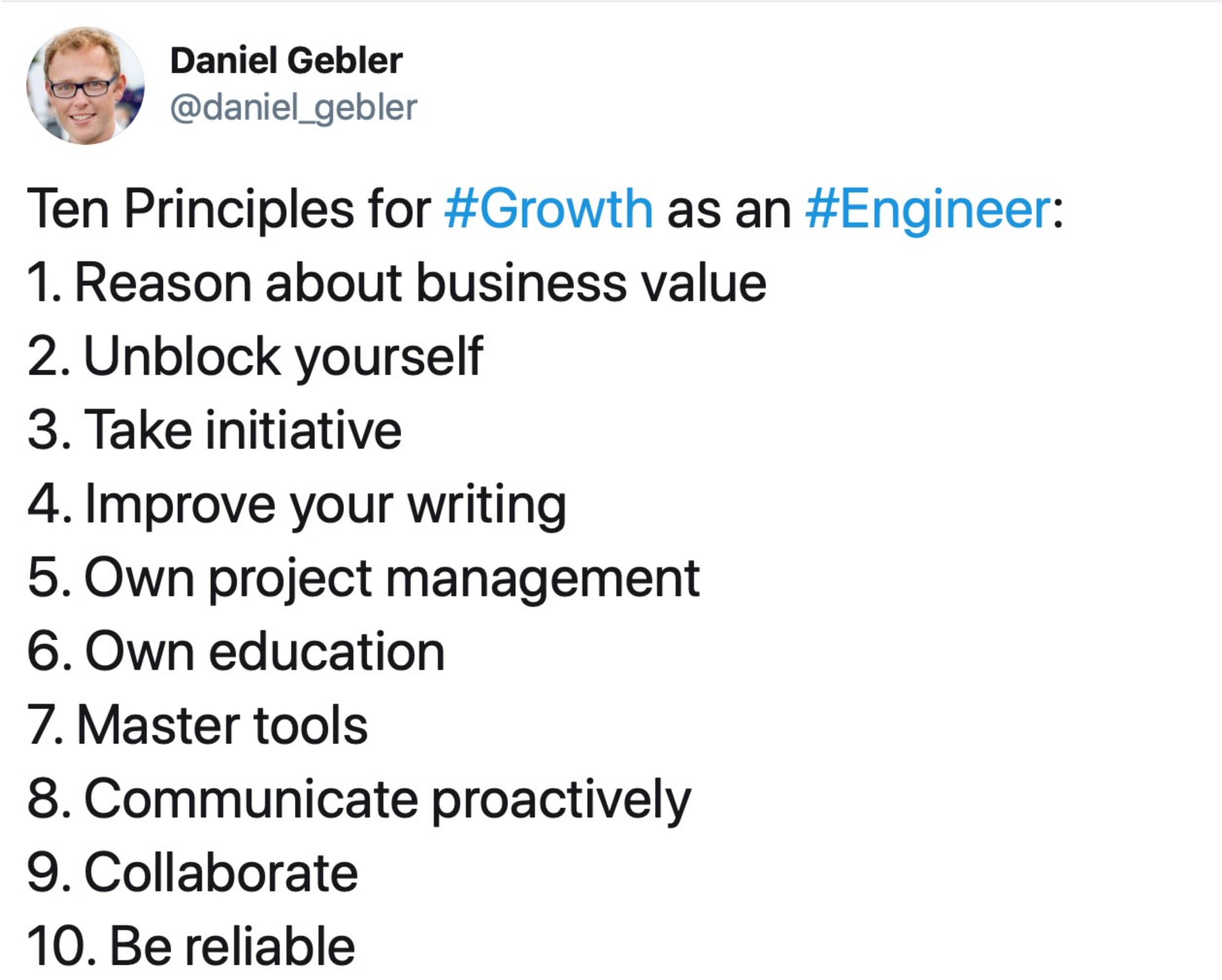
- Offer it to the system's architects as a *pull request*

- They will discuss it with you,
  … and hopefully *merge* it.

Get in touch with the architects!

Make them read your work

Interview them for your blog?!

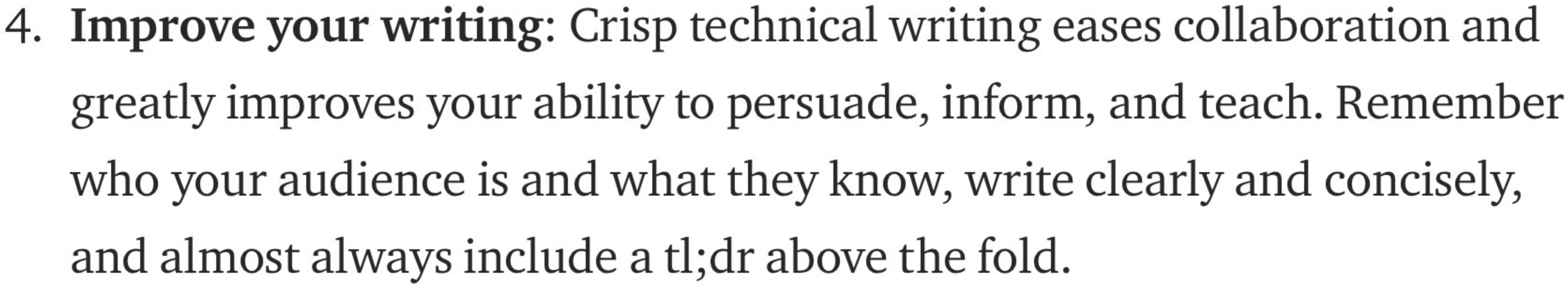


Daniel Gebler
@daniel_gebler

Ten Principles for #Growth as an #Engineer:

1. Reason about business value
2. Unblock yourself
3. Take initiative
4. Improve your writing
5. Own project management
6. Own education
7. Master tools
8. Communicate proactively
9. Collaborate
10. Be reliable

https://medium.com/@daniel.heller/ten-principles-for-growth-69015e08c35b



4. **Improve your writing**: Crisp technical writing eases collaboration and greatly improves your ability to persuade, inform, and teach. Remember who your audience is and what they know, write clearly and concisely, and almost always include a tl;dr above the fold.

Assignments E1-E4: (Technical) Essay Writing

Each team writes four essays (1500-2000 words):

1. the product vision, including required capabilities, roadmap, product context, domain model, and stakeholder analysis.
2. architectural decisions made, including system decomposition, tradeoff points, as well as architectural styles and patterns adopted.
3. an assessment of quality and (potential) technical debt; and
4. a scalability study identifying possible scalability issues and proposing architectural changes to address them.

Peer Review

- Learn one project very well – your own
- Learn about other projects by studying other team's essays

- Each student writes four reviews, one for essays E1-E4 each
- Each group receives feedback in 16 reviews
  - Four reviews for each essay E1-E4

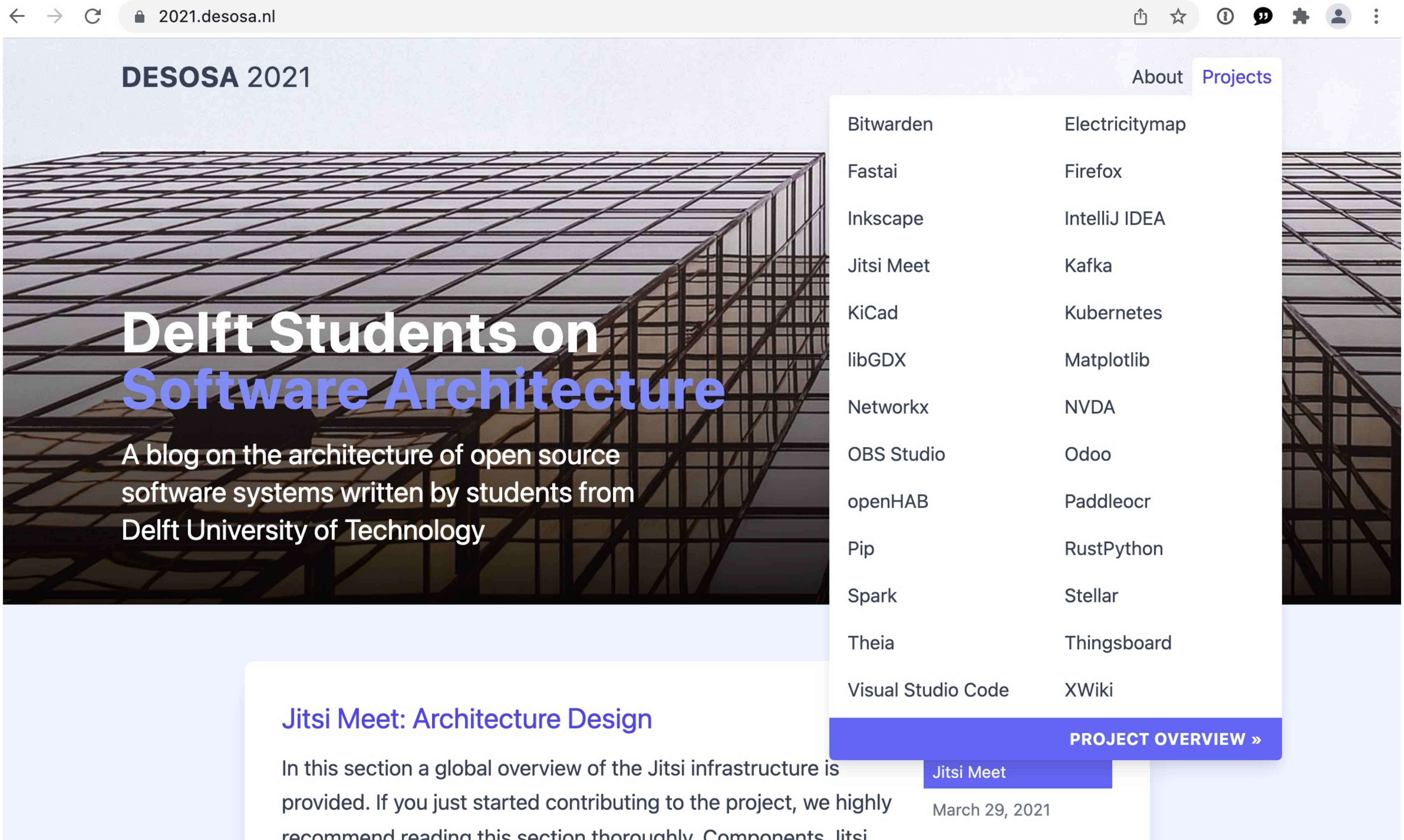
- We'll use peer.ewi.tudelft.nl

Public Writing makes Better Writers

- Objective 1: Write for the course
- Objective 2: *Write for the world*

- Throughout the course, your team can make your work available
  - This is *optional:* if you prefer privacy that's OK too
  - Simply flip flag in blog's meta-data

- *Delft Students on Software Architecture* (DESOSA)



Delft Students on Software Architecture

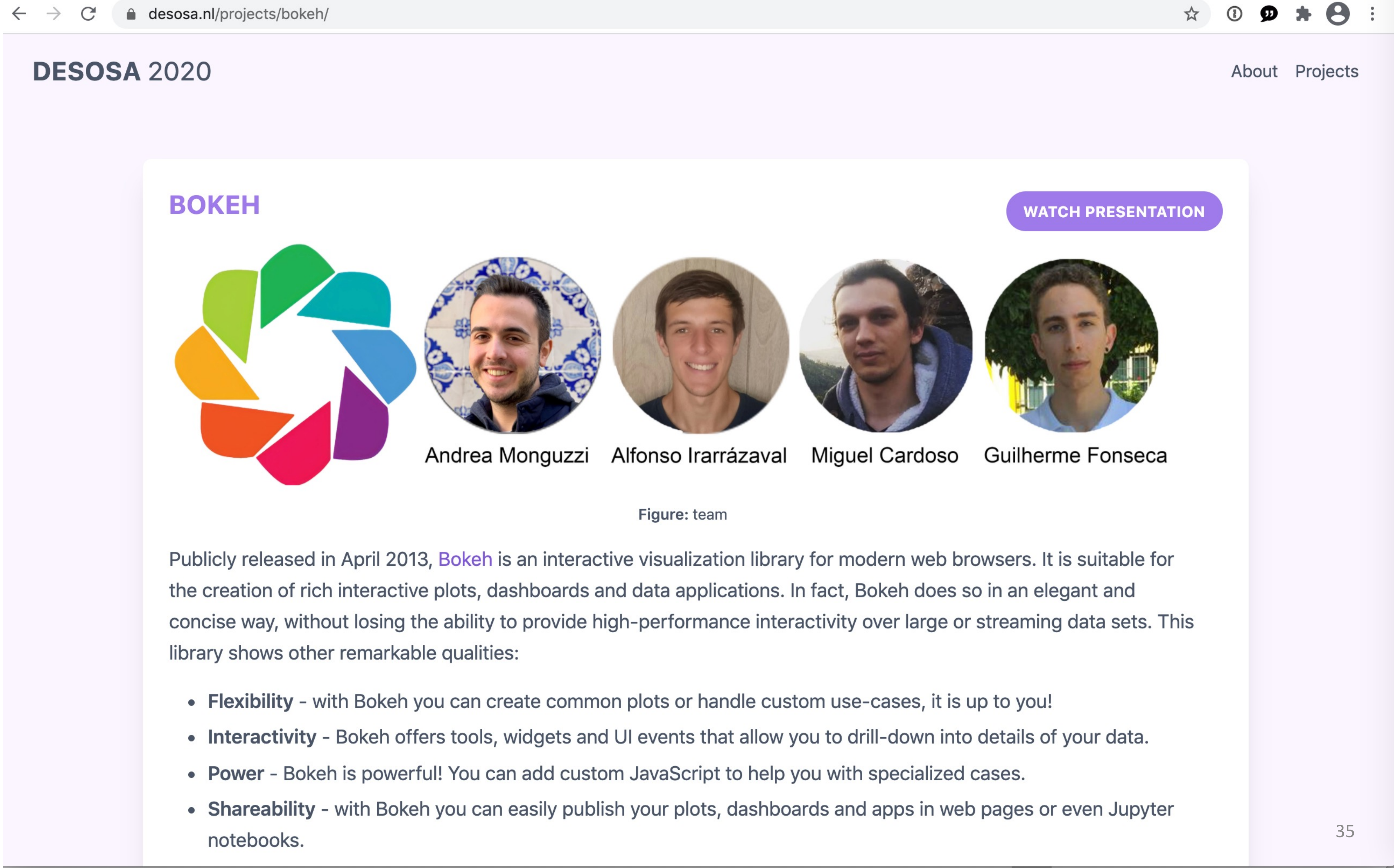
A blog on the architecture of open source software systems written by students from Delft University of Technology

Jitsi Meet: Architecture Design

In this section a global overview of the Jitsi infrastructure is provided. If you just started contributing to the project, we highly recommend reading this section thoroughly. Components. Jitsi

Jitsi Meet

March 29, 2021



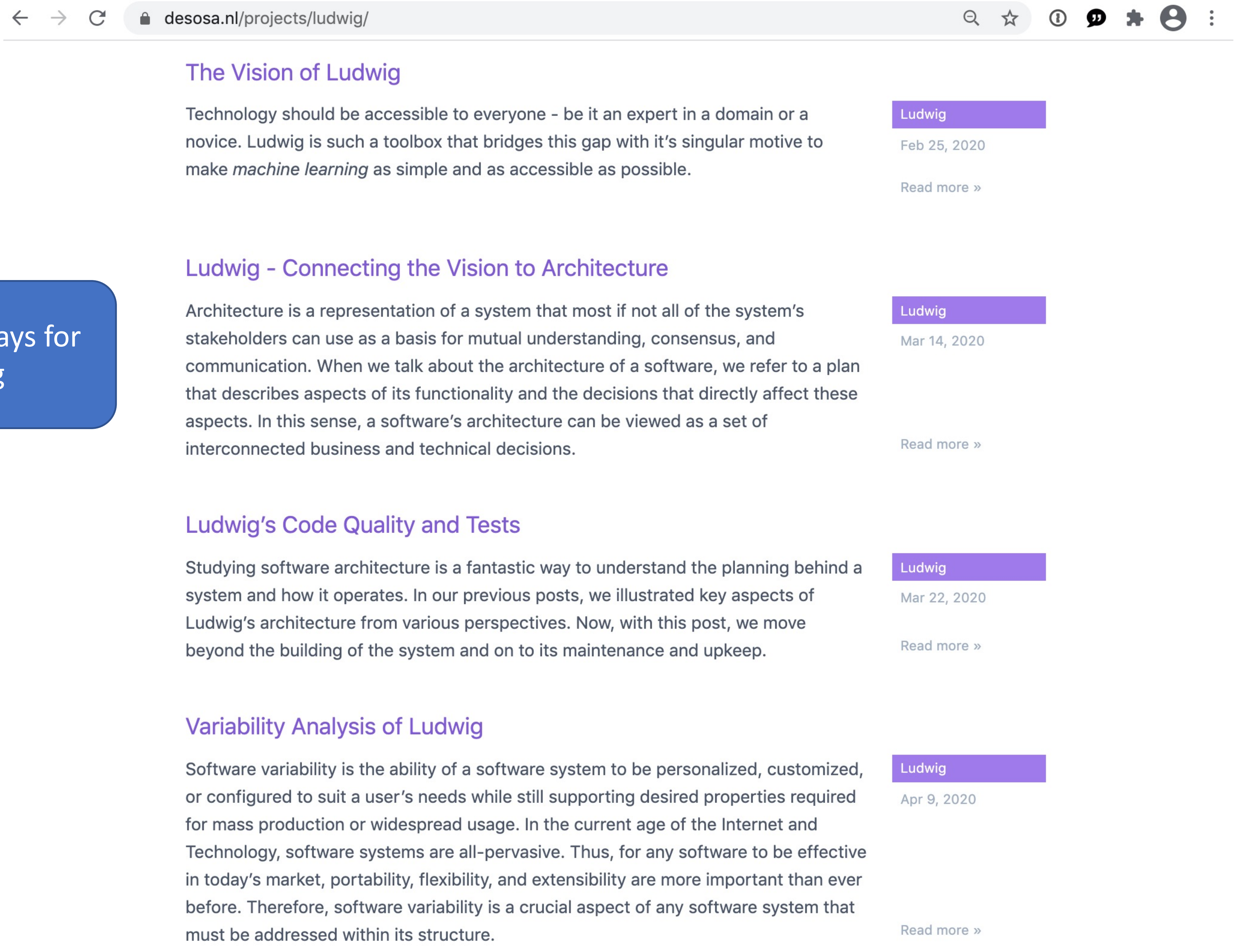
BOKEH

Figure: team

Publicly released in April 2013, Bokeh is an interactive visualization library for modern web browsers. It is suitable for the creation of rich interactive plots, dashboards and data applications. In fact, Bokeh does so in an elegant and concise way, without losing the ability to provide high-performance interactivity over large or streaming data sets. This library shows other remarkable qualities:

- **Flexibility** - with Bokeh you can create common plots or handle custom use-cases, it is up to you!
- **Interactivity** - Bokeh offers tools, widgets and UI events that allow you to drill-down into details of your data.
- **Power** - Bokeh is powerful! You can add custom JavaScript to help you with specialized cases.
- **Shareability** - with Bokeh you can easily publish your plots, dashboards and apps in web pages or even Jupyter notebooks.

The four essays for Ludwig



The Vision of Ludwig

Technology should be accessible to everyone - be it an expert in a domain or a novice. Ludwig is such a toolbox that bridges this gap with it's singular motive to make *machine learning* as simple and as accessible as possible.

Ludwig

Feb 25, 2020

Read more »

Ludwig - Connecting the Vision to Architecture

Architecture is a representation of a system that most if not all of the system's stakeholders can use as a basis for mutual understanding, consensus, and communication. When we talk about the architecture of a software, we refer to a plan that describes aspects of its functionality and the decisions that directly affect these aspects. In this sense, a software's architecture can be viewed as a set of interconnected business and technical decisions.

Ludwig

Mar 14, 2020

Read more »

Ludwig's Code Quality and Tests

Studying software architecture is a fantastic way to understand the planning behind a system and how it operates. In our previous posts, we illustrated key aspects of Ludwig's architecture from various perspectives. Now, with this post, we move beyond the building of the system and on to its maintenance and upkeep.

Ludwig

Mar 22, 2020

Read more »

Variability Analysis of Ludwig

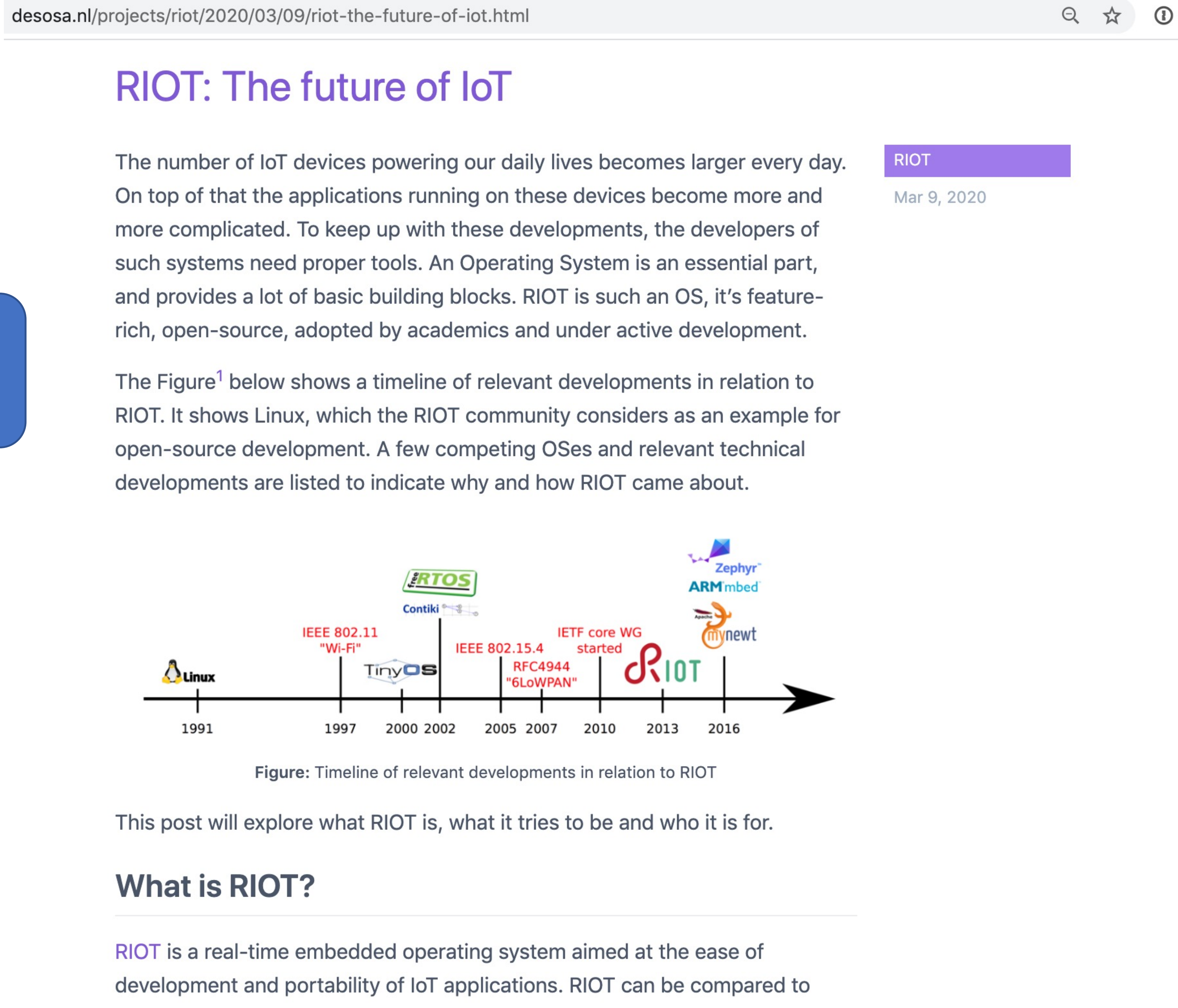
Software variability is the ability of a software system to be personalized, customized, or configured to suit a user's needs while still supporting desired properties required for mass production or widespread usage. In the current age of the Internet and Technology, software systems are all-pervasive. Thus, for any software to be effective in today's market, portability, flexibility, and extensibility are more important than ever before. Therefore, software variability is a crucial aspect of any software system that must be addressed within its structure.

Ludwig

Apr 9, 2020

Read more »

The first essay for RIOT



desosa.nl/projects/riot/2020/03/09/riot-the-future-of-iot.html

RIOT: The future of IoT

RIOT

Mar 9, 2020

The number of IoT devices powering our daily lives becomes larger every day. On top of that the applications running on these devices become more and more complicated. To keep up with these developments, the developers of such systems need proper tools. An Operating System is an essential part, and provides a lot of basic building blocks. RIOT is such an OS, it's feature-rich, open-source, adopted by academics and under active development.

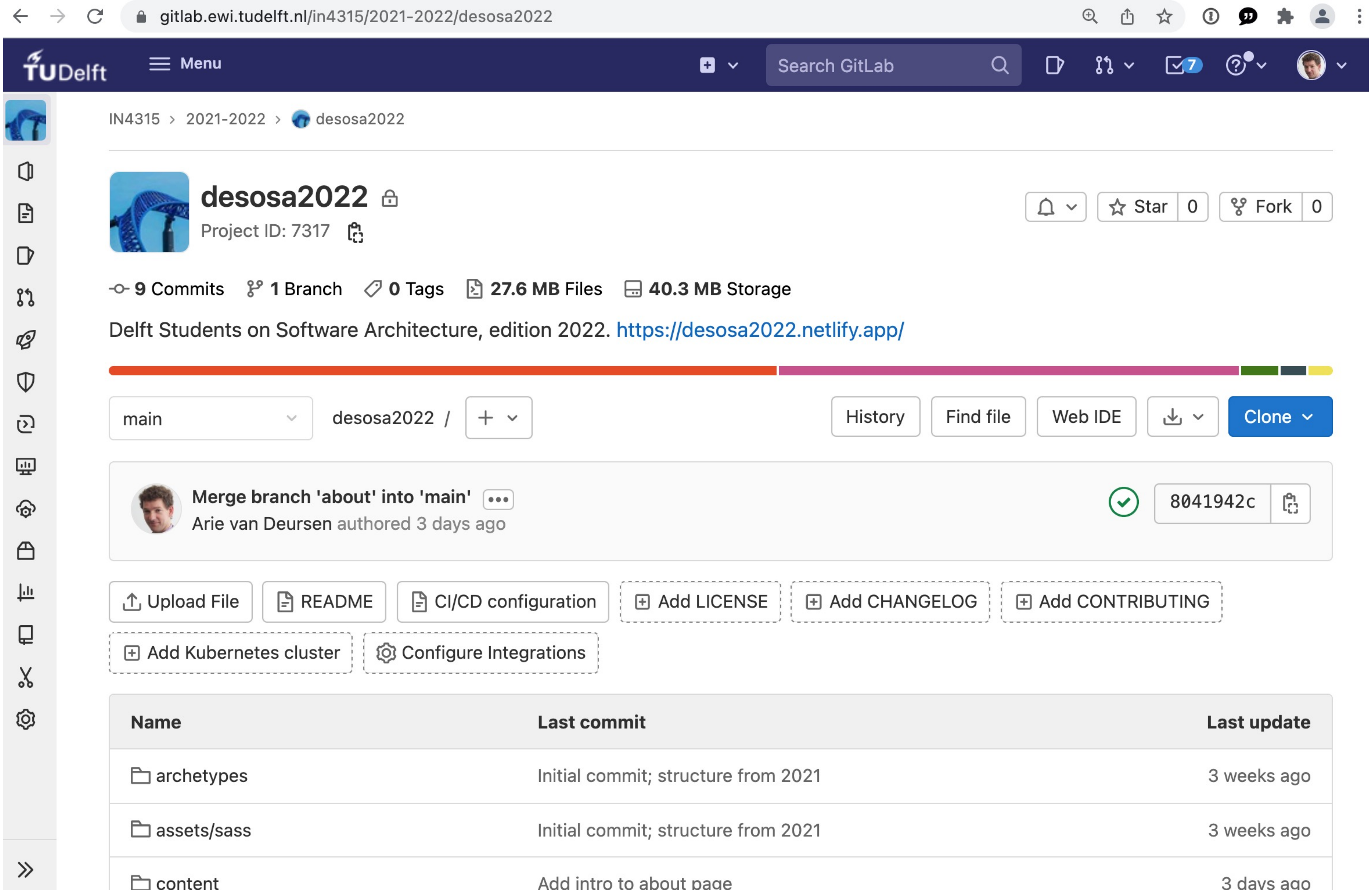
The Figure[1] below shows a timeline of relevant developments in relation to RIOT. It shows Linux, which the RIOT community considers as an example for open-source development. A few competing OSes and relevant technical developments are listed to indicate why and how RIOT came about.

Figure: Timeline of relevant developments in relation to RIOT

This post will explore what RIOT is, what it tries to be and who it is for.

What is RIOT?

RIOT is a real-time embedded operating system aimed at the ease of development and portability of IoT applications. RIOT can be compared to



gitlab.ewi.tudelft.nl/in4315/2021-2022/desosa2022

IN4315 › 2021-2022 › desosa2022

desosa2022

Project ID: 7317

9 Commits 1 Branch 0 Tags 27.6 MB Files 40.3 MB Storage

Delft Students on Software Architecture, edition 2022. https://desosa2022.netlify.app/

Merge branch 'about' into 'main'

Arie van Deursen authored 3 days ago

8041942c

Name	Last commit	Last update
archetypes	Initial commit; structure from 2021	3 weeks ago
assets/sass	Initial commit; structure from 2021	3 weeks ago
content	Add intro to about page	3 days ago

Blogging in Hugo

- All text in markdown
- YAML meta data about posts

- Full site rendered by Hugo
- Can be previewed locally

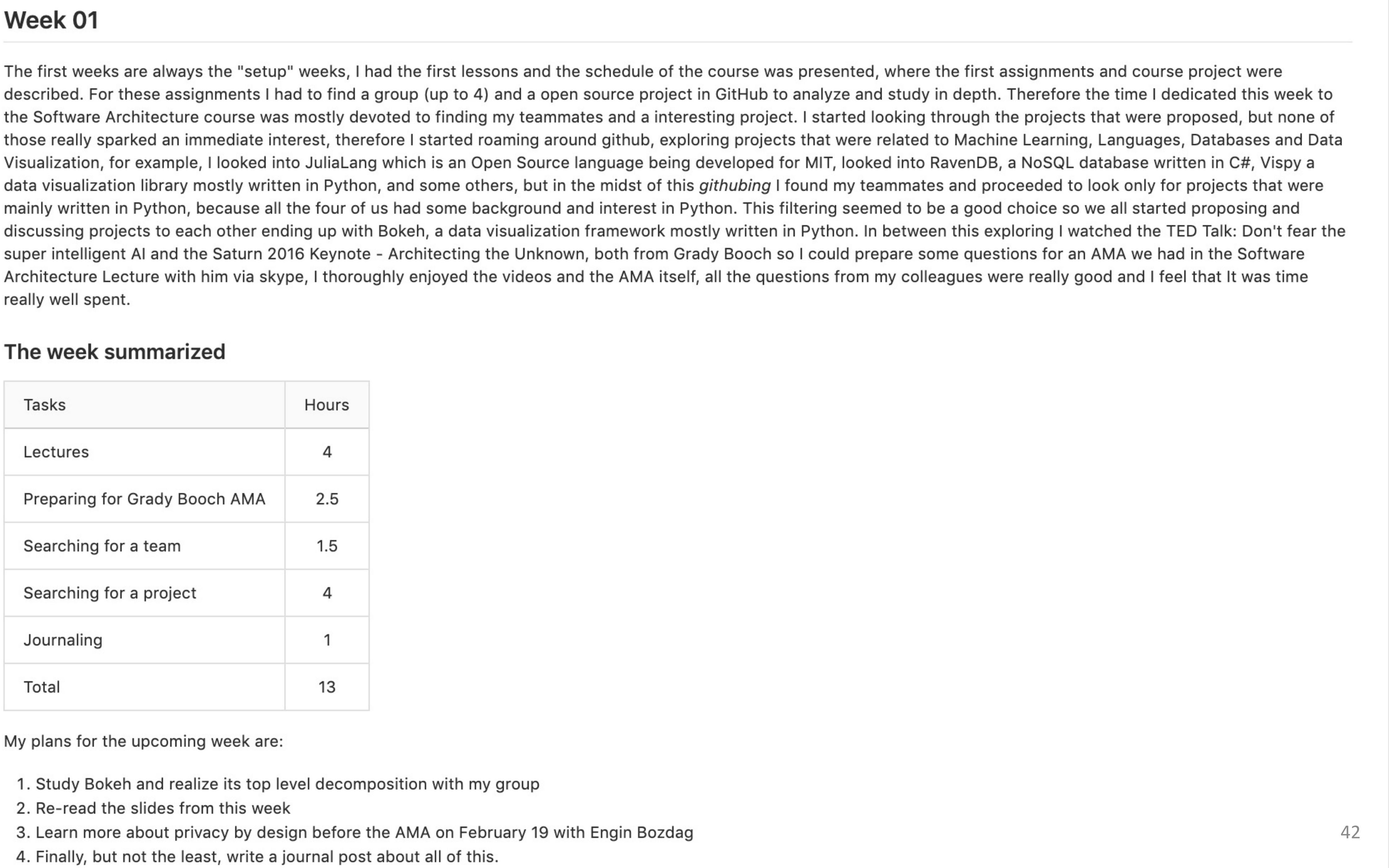
- Textual version control and diffing in git

The DESOSA 2022 GitLab Repo

- 100 students in a single git repo – worked very well previous years
- content/projects/<your project name>
  - /posts
  - /images
  - /contributions
  - /journals
- You can push branches and merge pull requests
- Merge is team decision: Full team is responsible
- Only make changes to your folder!

Manage your Time!

- Considerable freedom (own initiative) in *what* you do
- Not everything you do may be visible in essays
- Therefore, you need to *explain* how you spent your time
- 5 EC = 140 hours; In 9 week course = 16 hours per week!
- Per student: short, reflective journal, commit one entry per week
  - Track how many hours *you* spent
  - Main activities conducted
  - Main output produced
  - Summary of key things learned



Week 01

The first weeks are always the "setup" weeks, I had the first lessons and the schedule of the course was presented, where the first assignments and course project were described. For these assignments I had to find a group (up to 4) and a open source project in GitHub to analyze and study in depth. Therefore the time I dedicated this week to the Software Architecture course was mostly devoted to finding my teammates and a interesting project. I started looking through the projects that were proposed, but none of those really sparked an immediate interest, therefore I started roaming around github, exploring projects that were related to Machine Learning, Languages, Databases and Data Visualization, for example, I looked into JuliaLang which is an Open Source language being developed for MIT, looked into RavenDB, a NoSQL database written in C#, Vispy a data visualization library mostly written in Python, and some others, but in the midst of this *githubing* I found my teammates and proceeded to look only for projects that were mainly written in Python, because all the four of us had some background and interest in Python. This filtering seemed to be a good choice so we all started proposing and discussing projects to each other ending up with Bokeh, a data visualization framework mostly written in Python. In between this exploring I watched the TED Talk: Don't fear the super intelligent AI and the Saturn 2016 Keynote - Architecting the Unknown, both from Grady Booch so I could prepare some questions for an AMA we had in the Software Architecture Lecture with him via skype, I thoroughly enjoyed the videos and the AMA itself, all the questions from my colleagues were really good and I feel that It was time really well spent.

The week summarized

Tasks	Hours
Lectures	4
Preparing for Grady Booch AMA	2.5
Searching for a team	1.5
Searching for a project	4
Journaling	1
Total	13

My plans for the upcoming week are:

1. Study Bokeh and realize its top level decomposition with my group
2. Re-read the slides from this week
3. Learn more about privacy by design before the AMA on February 19 with Engin Bozdag
4. Finally, but not the least, write a journal post about all of this.

All Communication: Mattermost

See registration link on BrightSpace

- Announcements – main channel, low traffic, essentials only!
- Questions – ask (answer!) questions here
  - We may create some more sub-channels here
- Team-XYZ (public): Main communication hub for your team
  - Accessible to all; others can help / learn
  - Use to leave an evidence trail of you work.
  - Use to integrate with (learn from / help) other teams
  - All communication in English
  - *MINIMIZE USE OF WHATSAPP, EMAIL, TELEGRAM, ... (and not even Signal)*
  - In person / video call? Post short summary on Mattermost
- Off-Topic – your random noise

Personal (Pandemic) Complications

- Make sure you stay safe and healthy
- When all goes well:
  - Make your hours, and keep your journal up to date on weekly basis
- In case of serious issues: Always contact EEMCS student counsellor
  - We'll find a solution
  - Your up to date journal will be the starting point
- For minor disturbances:
  - Use your journal to explain temporary lack of progress
  - Indicate in journal how you and your team will handle it
- Feel free to contact TAs or teacher(s) at any time (email, mattermost)

Teaching Philosophy: Open Learning

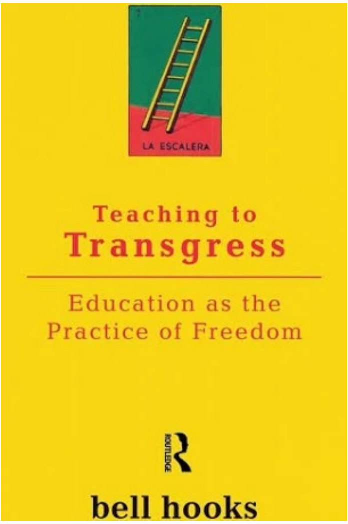
- This course is open by design
  - You learn from what others are doing
  - You share your work with others

- Interaction with open source systems is public
  - You can use an anonymous github account if you wish

- Team decision to make their writings public

Erik Duval

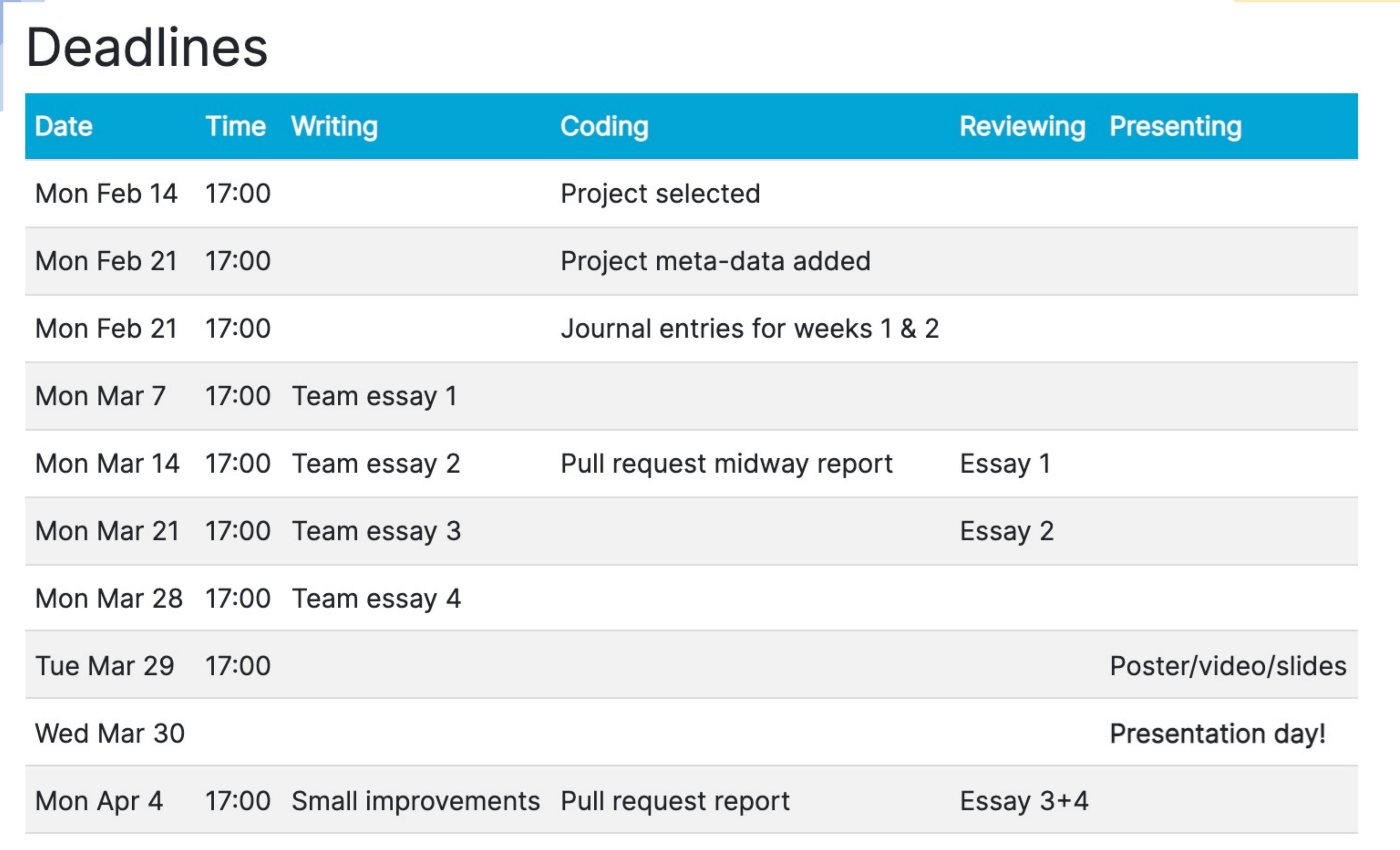
bell hooks



Closing Day: Wednesday March 30, full day

- Details will follow and will depend on:
  - nr of students participating
  - and pandemic developments
- To prepare
  - A ten minute video from each group
  - A poster from each group
  - Both optionally made public
- To participate:
  - Watch a selection of other teams
  - Ask questions and give feedback

Ambition:
An on campus celebration,
where you learn
and share knowledge,
concluded with drinks



Deadlines

Date	Time	Writing	Coding	Reviewing	Presenting
Mon Feb 14	17:00		Project selected		
Mon Feb 21	17:00		Project meta-data added		
Mon Feb 21	17:00		Journal entries for weeks 1 & 2		
Mon Mar 7	17:00	Team essay 1			
Mon Mar 14	17:00	Team essay 2	Pull request midway report	Essay 1	
Mon Mar 21	17:00	Team essay 3		Essay 2	
Mon Mar 28	17:00	Team essay 4			
Tue Mar 29	17:00				Poster/video/slides
Wed Mar 30					Presentation day!
Mon Apr 4	17:00	Small improvements	Pull request report	Essay 3+4	

Grading

Students will receive grades based on the following:

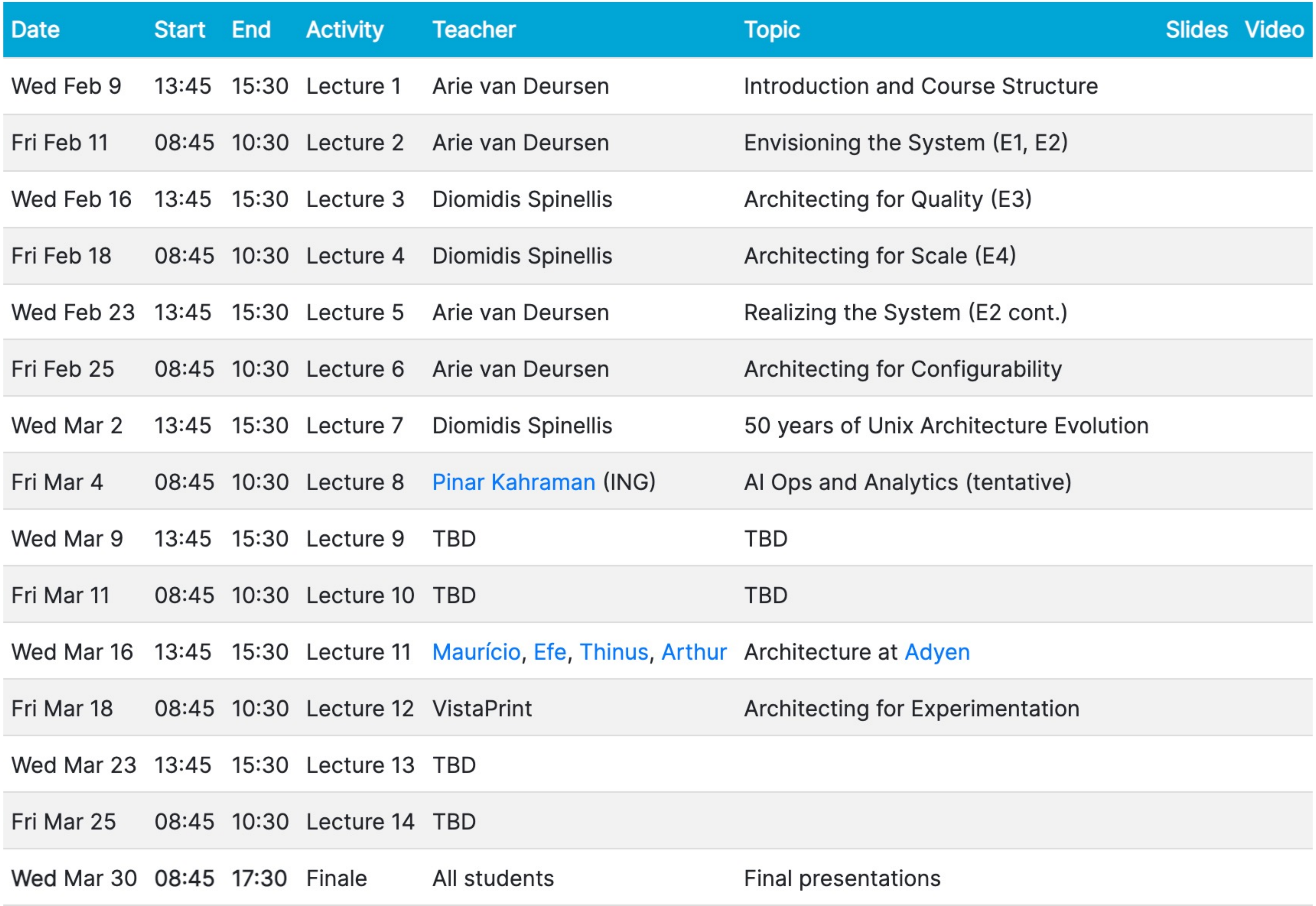
- `E`: Team performance for each of the four essays (1-10), composed form the average of the four essays E1..E4.
- `C`: Team performance for code contributions (1-10)
- `P`: Team performance for video presentation (1-10)
- `R`: Individual performance in peer reviews (-1, 0, 1): zero by default
- `A`: Individual performance in participation (-1, 0, 1): zero by default

The *team grade* is the weighted average of the team activities:

```
T = (3*E + C + P)/5
```

The *individual grade* then is the team grade to which a bonus can be added (or subtracted) for exceptional (top/bottom X%) results.

```
I = T + 0.5 * (R + A)
```



Date	Start	End	Activity	Teacher	Topic	Slides	Video
Wed Feb 9	13:45	15:30	Lecture 1	Arie van Deursen	Introduction and Course Structure		
Fri Feb 11	08:45	10:30	Lecture 2	Arie van Deursen	Envisioning the System (E1, E2)		
Wed Feb 16	13:45	15:30	Lecture 3	Diomidis Spinellis	Architecting for Quality (E3)		
Fri Feb 18	08:45	10:30	Lecture 4	Diomidis Spinellis	Architecting for Scale (E4)		
Wed Feb 23	13:45	15:30	Lecture 5	Arie van Deursen	Realizing the System (E2 cont.)		
Fri Feb 25	08:45	10:30	Lecture 6	Arie van Deursen	Architecting for Configurability		
Wed Mar 2	13:45	15:30	Lecture 7	Diomidis Spinellis	50 years of Unix Architecture Evolution		
Fri Mar 4	08:45	10:30	Lecture 8	Pinar Kahraman (ING)	AI Ops and Analytics (tentative)		
Wed Mar 9	13:45	15:30	Lecture 9	TBD	TBD		
Fri Mar 11	08:45	10:30	Lecture 10	TBD	TBD		
Wed Mar 16	13:45	15:30	Lecture 11	Maurício, Efe, Thinus, Arthur	Architecture at Adyen		
Fri Mar 18	08:45	10:30	Lecture 12	VistaPrint	Architecting for Experimentation		
Wed Mar 23	13:45	15:30	Lecture 13	TBD			
Fri Mar 25	08:45	10:30	Lecture 14	TBD			
Wed Mar 30	08:45	17:30	Finale	All students	Final presentations		

How to Spend Week 1?

- Find a team
- Find a system
- Get onto gitlab
- Make your first DESOSA commit

- Watch Gregor Hohpe’s “Architect Elevator” video
- Study Pautasso’s Chapters 1-3
- Explore https://docs.arc42.org/, sections 1-3 (requirements, stakeholders, constraints, context, external interfaces)