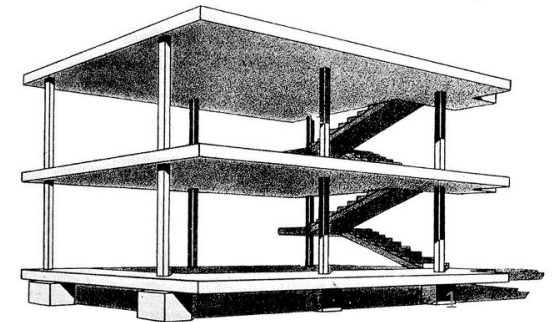


TU Delft IN4315: Software Architecture

Introduction and Labwork

Arie van Deursen



Wikipedia, Dom-ino House, Corbusier

IN4315 Software Architecture:

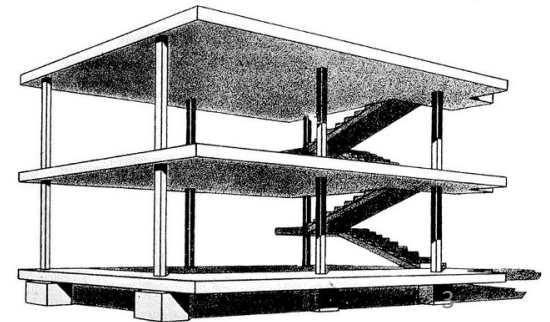
Online lectures rules of engagement

- 100+ participants?!
- Please mute – we may mute you
- Lectures will be recorded and shared on collegerama
- Switch off your camera (unless you want to be on collegerama)
- Shipra Sharma (TA) will help manage the chat
- Remarks / questions in chat always welcome!
- Please login with TU Delft credentials so we know who is chatting
- Please follow TU Delft code of conduct
- Chats will be saved
- Level of participation may be input to grade

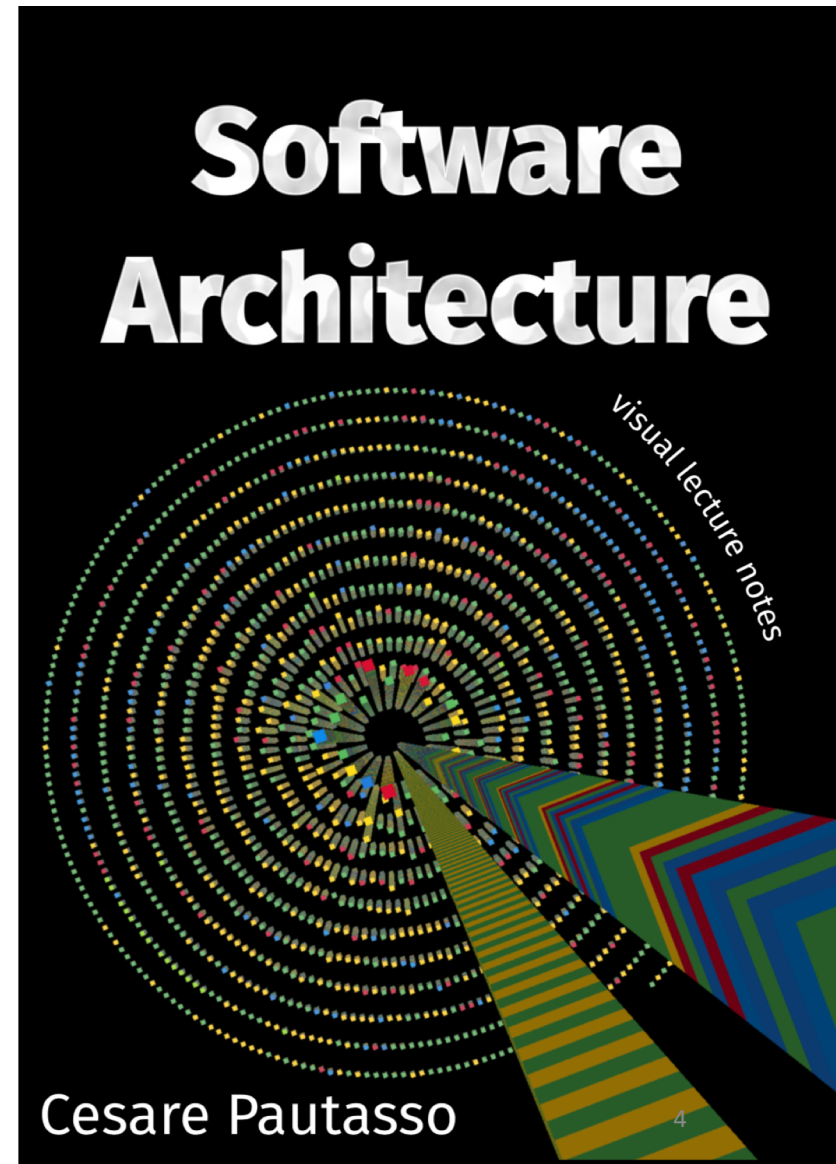


Today's Kickoff Lecture

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. [What is software architecture?]



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 good software architect?
3. Can you name examples of well known, great 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)

- Architects carry overall responsibility for all technical decisions
- Lead an organization that takes the right decisions
- Willing and able to take them where needed
- Understand which technical decisions can be safely deferred

Consequence (1): 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

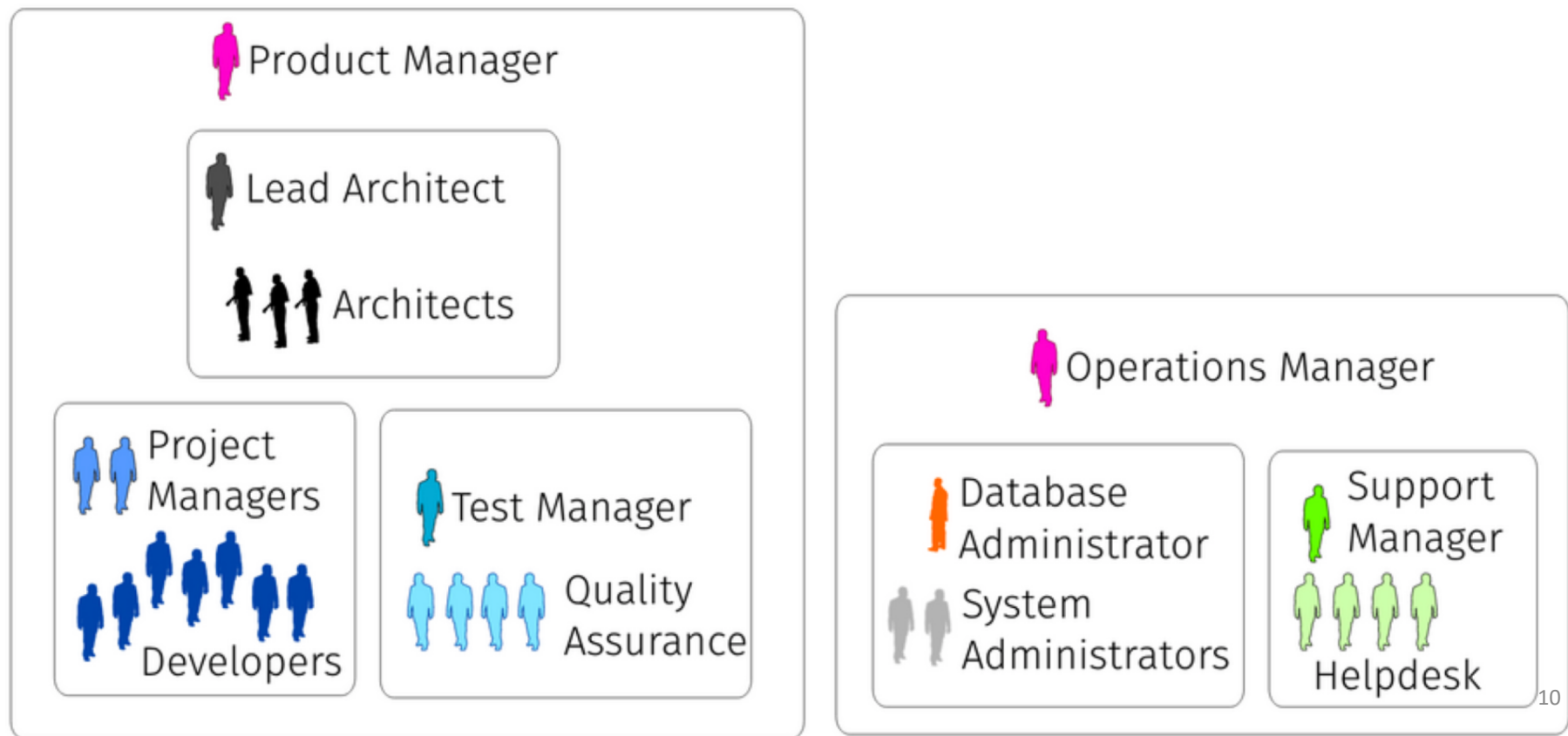
- Know and understand relevant technology
- Evaluate and influence choice of 3rd party frameworks, components and platforms
- Track technology evolution
- Know what you do not know

“Coding Architect”: Join team, contribute code, program in pair

Key Responsibilities (2)

- 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

Consequence (2): Architect must be Great Communicator



People to Talk to



Marketing



Product Manager



UI Designer



User



Customer



Architect



Developer



Tester



Vendor



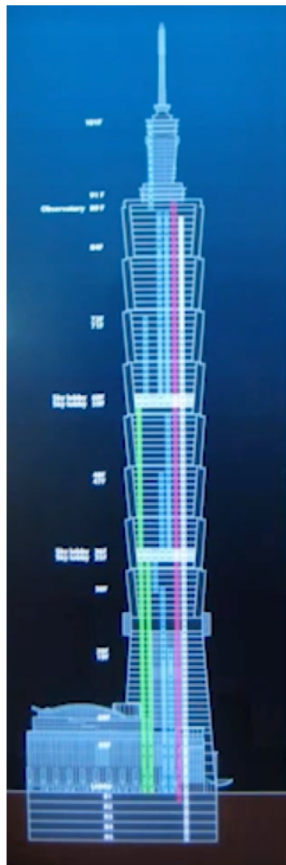
Data Scientist



Technical Writer



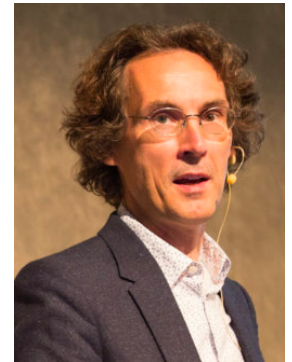
SysAdmin



ArchitectElevator.com

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



Gregor Hohpe

13

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

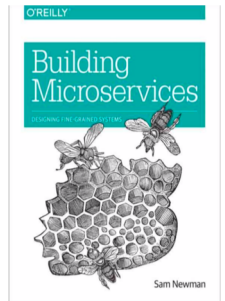
Shared Story = Product Vision

- 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)

- 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

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 Hoppe. The Software Architect Elevator. O'Reilly, 2020. Chapters 1-5
- Sam Newman. Building Microservices. O'Reilly, 2015. Chapter 2.
- Cesare Pautasso. Software Architecture. Leanpub, 2020. Chapter 3
<https://leanpub.com/software-architecture/>

IN4315 Labwork

Software architecture is about

- **people:**
 - You will work in teams of four
- **real systems:**
 - You will adopt an open source system,
 - which you will analyze, describe, and improve
 - and with whose developers you'll interact
- **communication:**
 - You will write
 - You will present

Team Formation

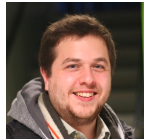
- 4 != 3, 4 != 5
- Aim for a diverse team:
 - Git knowledge, programming skills, writing, presentation, domain knowledge
 - Bachelor background, master track, geography, ...
- Brightspace discussion forum “Partners Wanted”
- Form your group on Brightspace (Collaboration / Groups)
- **DEADLINE: Monday February 15, 17:00**

Team *Coach*

- One of the teachers
- One per team
- Meetings in week 2, 4, 6 of the course
- Discuss progress
- Ask-them-anything
- Think how to map theory to your system
- Think about interesting aspects of your system to study

Coaches:

Leonhard Applis
Luís Cruz
Arie van Deursen
Xavier Devroey
Burcu Kulahcioglu Ozkan
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 15, 17:00

Suggested projects

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

Project	GitHub URL	Remarks	Proposed by
JHipster	https://github.com/jhipster/generator-jhipster	Variability	Xavier Devroey
XWiki	https://github.com/xwiki	Variability	Xavier Devroey
Theia	https://github.com/eclipse-theia/theia	Variability	Xavier Devroey
Hugo	https://github.com/gohugoio/hugo	Variability	Xavier Devroey
Drupal	https://git.drupalcode.org/project/drupal	Variability	Xavier Devroey
Odoo	https://github.com/odoo/odoo	Variability	Xavier Devroey
Homebrew	https://github.com/Homebrew/brew	Variability	Xavier Devroey
Scaphandre	https://github.com/hubblo-org/scaphandre	Energy	Luís Cruz
Sitespeed	https://github.com/sitespeedio/sitespeed.io		Luís Cruz
Electriciy Map	https://github.com/tmrowco/electricitymap-contrib	Energy	Luís Cruz
Firecracker	https://github.com/firecracker-microvm/firecracker		Luís Cruz
Timescale DB	https://github.com/timescale/timescaledb		Luís Cruz
CleverHans	https://github.com/cleverhans-lab/cleverhans	Adversarial Attacks on NNs	Leonhard Applis
HLS	https://github.com/haskell/haskell-language-server	Haskell IDE	Leonhard Applis

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

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. quality and evolution; and
4. a [deeper analysis](#) based on the lectures or other relevant material specific to the system of choice;

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



Peer Review

Software
Architecture

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
- *Delft Students on Software Architecture* (DESOSA)

DESOSA 2020

About Projects

Delft Students on Software Architecture

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

- | | |
|----------------|--------------------|
| Ansible | ArduPilot |
| Blender | Bokeh |
| Docker Compose | ESLint |
| Gatsby | GitLab |
| Ludwig | Material UI |
| Meteor | Micronaut |
| MuseScore | Mypy |
| Next.js | NumPy |
| Open edX | openpilot |
| OpenRCT2 | RIOT |
| Ripple | Scikit-learn |
| Sentry | Signal for Android |
| Solidity | spaCy |
| Spyder | TensorFlow |

Putting Sentry into Context

Imagine you created a nice application for other people to use. Your product is gaining traction, new users start flowing in and you start earning some money. Life is great and you decide to roll out a big update you have been working on. But after a few days of the

DESOSA 2020

About Projects

BOKEH

WATCH PRESENTATION



Andrea Monguzzi



Alfonso Irarrázaval



Miguel Cardoso



Guilherme Fonseca

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 »](#)

RIOT: The future of IoT

The first essay for RIOT

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 Figure¹ 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.

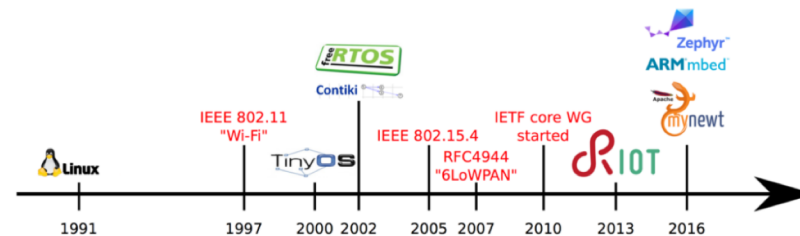


Figure: Timeline of relevant developments in relation to RIOT

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

RIOT

Mar 9, 2020

DESOSA: Past / Present

Past (pre 2019)

- Book with chapters
- One chapter per team
- Each team own git repo
- Book published *after* course
- Graded by teachers
- Teams can opt-out
- All documents in markdown

Present (2020, 2021)

- Collection of essays (blogs)
- Four essays per team
- All teams in one shared git repo
- Blogs shared *during* course
- Peer review
- Teams can opt-out
- All documents in markdown

DESOSA 2021

About Projects

Delft Students on Software Architecture

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

Micronaut - A perfect Microservice framework?

Whereas the previous blogposts focused on the internal
architecture view of Micronaut, we decided to change now the
perspective in our last blogpost and look at Micronaut through
the eyes of a software architect who wants to build a
microservice application. Micronaut claims to be "a modern,

Micronaut

April 9, 2020

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 8 week course = 17.5 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 – essentials only!
- Off-Topic – your random noise
- Questions-{contributions, essays, lectures, tech}
- 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
 - *DO NOT USE WHATSAPP, EMAIL, TELEGRAM, ... (and not even Signal)*
 - In person / video call? Post short summary on Mattermost

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
- Work-in-progress writing is visible within course only
- Your interaction with open source systems is public
- You can decide if you want to make your writings public

← → ↺

gitlab.ewi.tudelft.nl/in4315/2020-2021/desosa2021

🔍 ☆ ⓘ 🗨 ⚙ 👤 ⋮

TU Delft

Projects ▾ Groups ▾ More ▾

🔍 Search or jump to...

📄 🔄 📄 📄 📄 📄

DESOSA 2021

desosa2021

Project overview

Details

Activity

Releases

Repository

Issues 1

Merge Requests 0

Requirements

CI / CD

Security & Compliance

Operations

Packages & Registries

Analytics

Wiki

Snippets

Members

Settings

IN4315 > 2020-2021 > desosa2021

DESOSA 2021

desosa2021

Project ID: 4112

🔔

☆ Star 0

🍴 Fork 0

🔗 40 Commits

🌿 1 Branch

🏷 0 Tags

📄 4.1 MB Files

💾 122.8 MB Storage

Delft Students on Software Architecture, Edition 2021

<https://2021.desosa.nl/>

main

desosa2021

+


History

Find file

Web IDE

📄

Clone



Add information on the CI pipelines

Casper Boone authored 1 day ago

Verified ✓ 2b38d061

📄 README

📄 CI/CD configuration

📄 Add LICENSE

📄 Add CHANGELOG

📄 Add CONTRIBUTING

📄 Add Kubernetes cluster

Name	Last commit	Last update
📁 archetypes	Add author GitHub url support	5 days ago
📁 assets/sass	Add PDF exports through headless chrome	1 week ago
📁 content	Add single essay PDF generation	2 days ago
📁 layouts	Add epub generation	1 day ago
📁 pdf-generator	Add single essay PDF generation	2 days ago
📁 static	Add favicon	1 week ago
🔗 .gitattributes	Add gitattributes file	2 weeks ago
🔗 .gitignore	Add PDF exports through headless chrome	1 week ago
🔗 .gitlab-ci.yml	Update CI config to use new default bran...	6 days ago

40

The DESOSA 2021 GitLab Repo

- 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]

Closing Day: April 1, full day

- Each team will prepare 10min (max!) video
 - Optionally made public
- Groups of 5-6 teams will join two hour (online) session:
 - Watch video (10min) followed by Q&A (10min)
 - Learn from each other
 - Ask questions and give feedback
 - One teacher present
- Lockdown reduced? Drinks on campus!

Deadlines

Date	Time	Writing	Coding	Reviewing	Presenting
Mon Feb 15	17:00		Project selected		
Mon Feb 22	17:00		Project meta-data added		
Mon Feb 22	17:00		Journal entries for weeks 1 & 2		
Mon Mar 8	17:00	Team essay 1			
Mon Mar 15	17:00	Team essay 2	Pull request midway report	Essay 1	
Mon Mar 22	17:00	Team essay 3		Essay 2	
Mon Mar 29	17:00	Team essay 4		Essay 3	
Wed Mar 31	17:00				Presentation Video
Tue Apr 6	17:00		Pull request report	Essay 4	

Grading

Students will receive grades based on the following:

- **E**: Team performance for each of the four essays (1-10), composed from 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.

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