Automate Everything

Guest lecture at FAU Erlangen-Nürnberg

22.06.2022 | AMOS

Engineering und Software-Entwicklung

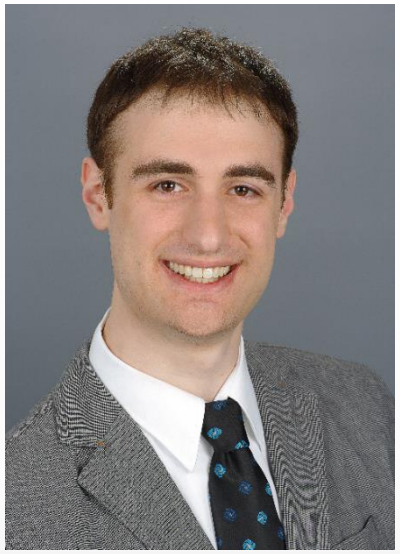
Agenda

Engineering und Software-Entwicklung

- About ESE & me
- Is there anything outside pipelines?
- A journey from the requirements engineering to the customer and back
  - Start station: Requirements Engineering
  - Stopover: Developer environment
  - Gaining speed: Continuously driving through pipelines
  - Destination Station: At the customer
  - The return: Maintenance
- Time to talk

About ESE & Myself

Engineering und Software-Entwicklung



- My name is Ralf Spengler.
- Studied at FAU
  - in the beginning molecular science for 4 semesters
  - switched to computer science in 2010 (one of the best decisions of my life)
  - finished my master thesis in 10/2015
- First worked at HEITEC AG while studying and afterwards
- Since 2018: ESE – Engineering and Software-Entwicklung GmbH in Erlangen
- I‘m
  - Software Engineer with high experience in agile development of safety critical products
  - Have broad knowledge (ISTQB® CT-ALTM, iSAQB® CPSA-FL, Scrum.org™ PSM I, IREB® CPRE-FL)

Why am I here?

BSI Lagebericht Cybersicherheit 2021 (BSI Report on Cybersecurity 2021)

Engineering und Software-Entwicklung

- 144 million new versions of malicious software
- Critical infrastructure such as institutions of the public healthcare have been attacked (e.g. a university clinic in Germany could not treat new emergencies for 13 days)
- Many companies were blackmailed by attackers using ransomware (e.g. Colonial Pipeline Company)
- Coordinated-Vulnerability-Disclosure: Many companies have troubles to determine, whether they are affected by a security vulnerability and if so, to patch it. (e.g. Microsoft Exchange: after about 2 months still 9% were vulnerable)
- Heavy Supply-Chain-Attacks (SolarWinds)

We cannot prevent all vulnerabilities – especially in 3rd party software – but we can be prepared to rollout bug fixes ASAP.

Data and Facts

Engineering und Software-Entwicklung

founded	Headquarters	Locations	Employees	Revenue	Certificates
1997	Braunschweig	Berlin / Hennigsdorf Braunschweig Erlangen Frankfurt am Main Hannover Kiel München Wolfsburg	2021: 387	2021: 33 Mio. Euro 2020: 31 Mio. Euro 2019: 29,5 Mio. Euro 2018: 27 Mio. Euro	CPPM according to iSQI®/ PMI® DIN EN ISO/IEC 17020 FRA-Consultants ISO 9001:2015 IRIS (ISO TS 22163:2017) iSTQB® -Tester TISAX Certificate

Our Competences

Engineering und Software-Entwicklung

Services	Sectors	Projects
Software-Engineering	Rail	Project realization
Testing & Verification	Automotive	Consulting
Assessment Services	IT /Industry	Project support

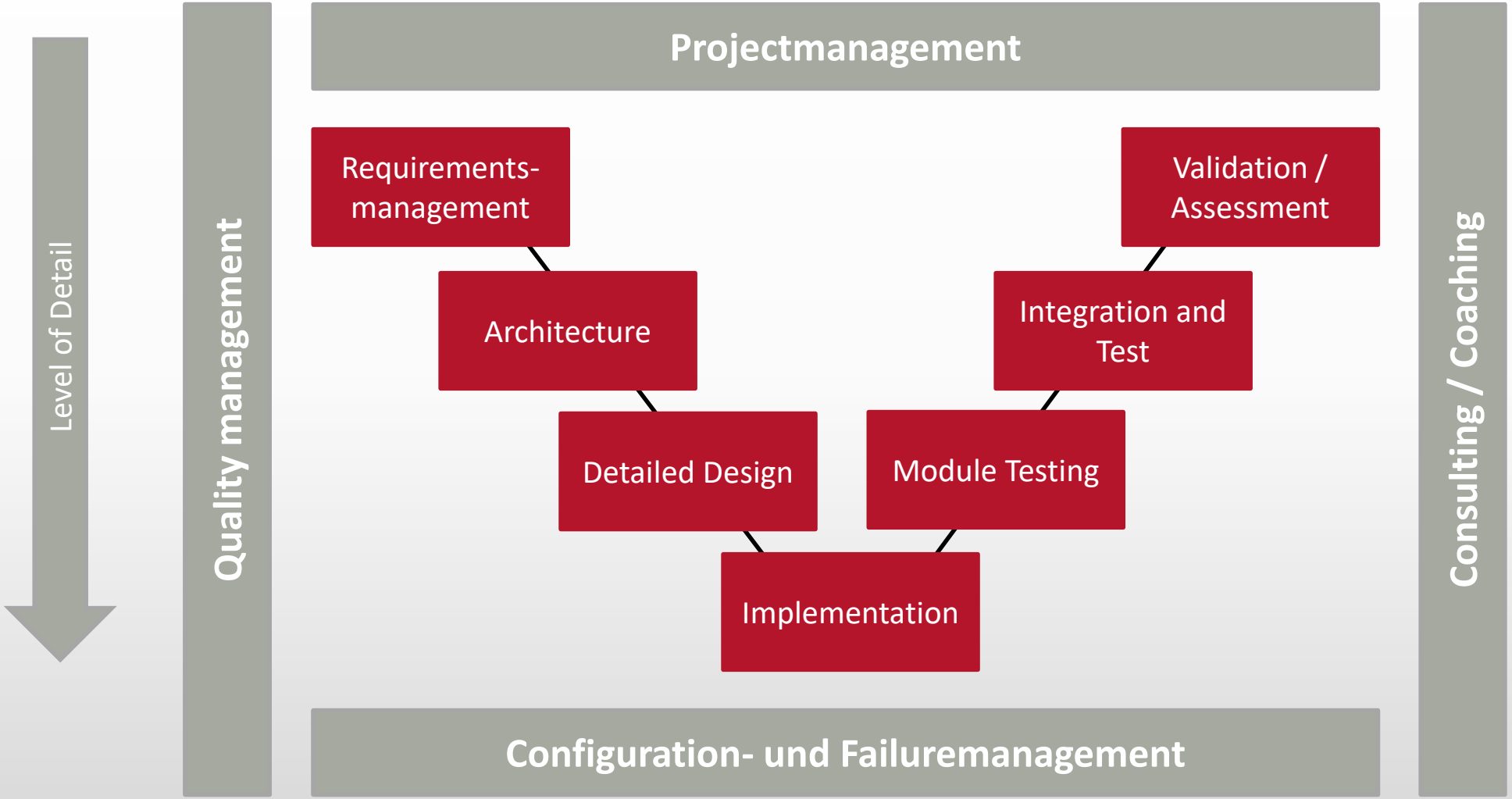
Is there anything outside pipelines?

Engineering und Software-Entwicklung

Let‘s look at the V-Model!

Although it might look strange from an agile point of view.

Engineering und Software-Entwicklung



Keep in mind …

Engineering und Software-Entwicklung

- It‘s the **team** that must **decide** what to automate and how to realize it.
- If you are insecure what to do:
  - **Do not** ask what fits to a process framework.
  - **Read the agile manifesto** and **discuss** it inside the team.
  - **Automation must bring value**: Get trusted with the product quality model defined by ISO/IEC 25010.
  - **Use empiricism**
- Most important drivers for the slides you will see:
  - **Feedback culture**: Agile development is about collecting all kind of feedback in short cycles.
  - „Simplicity – the art of maximizing the amount of work not done – is essential.“ : **Be smart – be lazy**

A Journey from the Requirements Engineering to the Customer and Back

Engineering und Software-Entwicklung

Start Station: Requirements Engineering

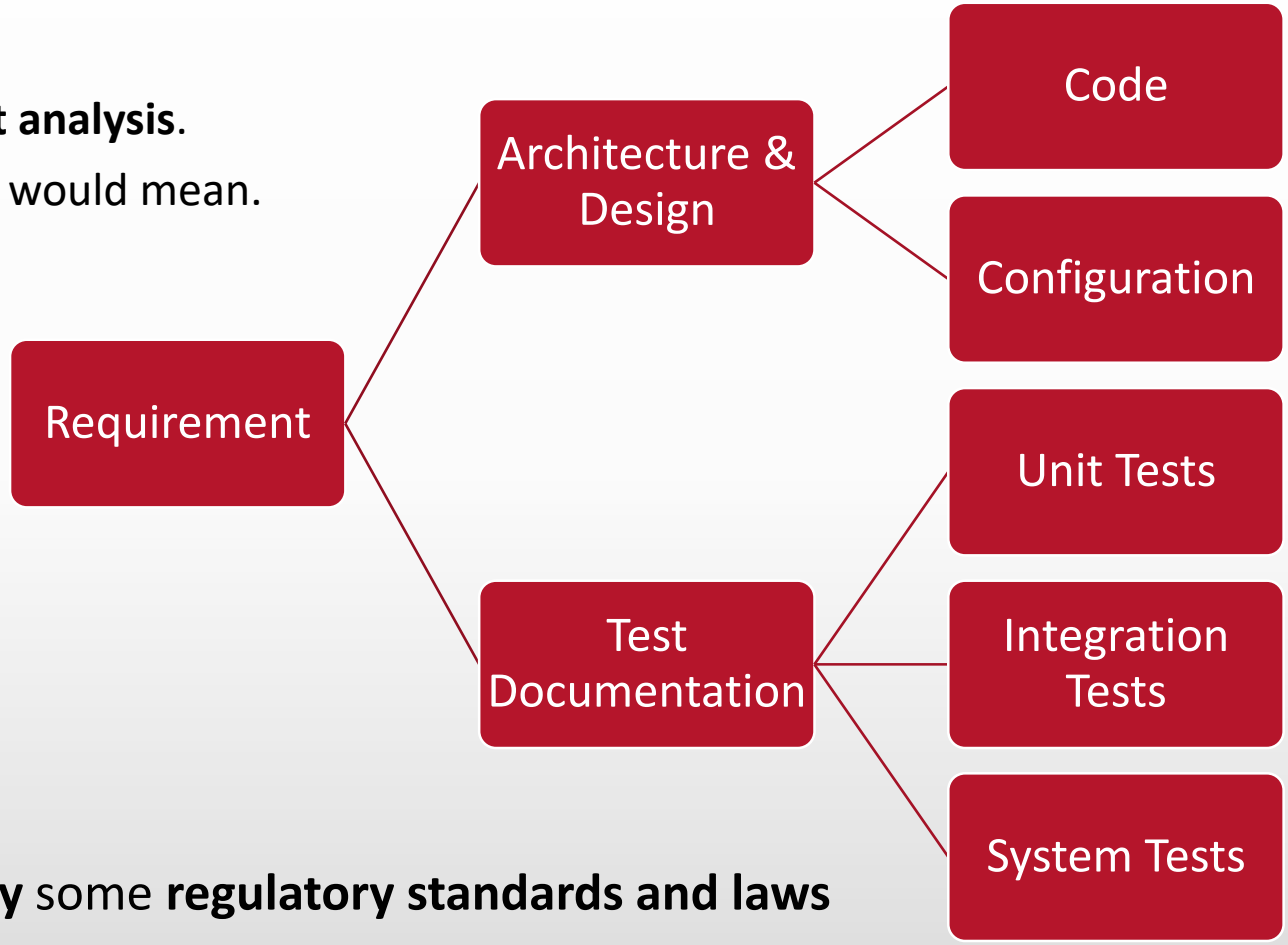
Engineering und Software-Entwicklung

Traceability

Covers the whole lifecycle but starts right here

Engineering und Software-Entwicklung

- Requirement Management
  - Use tools to **automate the impact analysis**.
  - Fast first **feedback** what a **change** would mean.
  - Less manual work.



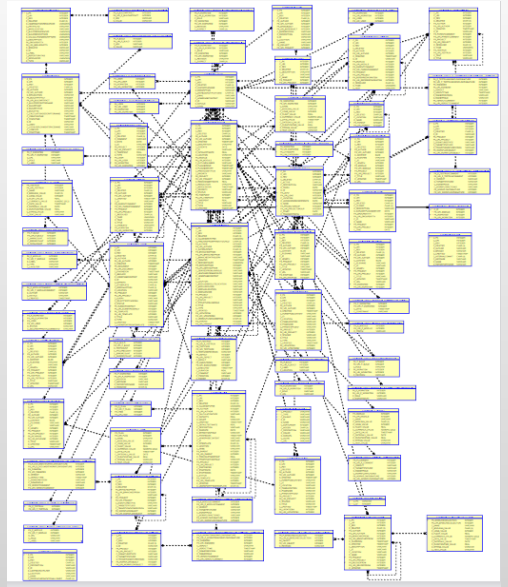
- Very helpful if you need to **satisfy** some **regulatory standards and laws**

Modeling

Engineering und Software-Entwicklung

- Design usable **GUI together with the relevant stakeholders**, e.g. using Qt Design Studio or QtCreator
  - Get immediate **feedback** from stakeholders.
  - Helps to gain a **common understanding** of the **workflow**.
  - Helps to gain a common understanding of the **expected design**.
  - Can be directly **integrated** into your **application**.

- **Generate code** from models, e.g. database schema, e.g. Enterprise Architect
  - Models help to get **feedback** whether you understood all attributes and relationships you must be aware of.
  - Generate a script to setup your database: Makes **switching between databases** easier.
  - Understand bad documented 3rd-party systems ----------> (Screenshot of https://almdemo.polarion.com/polarion/sdk/doc/database/FullDBSchema.pdf)



Testability

Engineering und Software-Entwicklung

- Use **Behavior-Driven-Development** to get requirements in an **ubiquitous language** that can be used to automate testing (e.g. **Gherkin-Syntax** and Cucumber)
  - **Sentence templates** help to reduce the ambiguity of natural language.
  - It is still natural language that **can be understood by all stakeholders**. Scenarios enable them to give you and each other **feedback**.
  - Defining the initial context, an event and the expected outcome:
    That‘s what you formally do when writing **test cases**

  - Just to give you two small examples:

Given that we write a scenario together **When** stakeholders read it **Then** they will be able to understand it	**Given** a user enters 1 divided by 0 into a calculator **When** the user presses the „=“-button **Then** „ERR“ will be shown on the display

Stopover: Developer Environment

Engineering und Software-Entwicklung

Unify the Development Environment

„But it works at my machine!“

Engineering und Software-Entwicklung

- **Virtual environments**, e.g. python venv, nodeenv, …
- **Dependency management**, e.g. npm, pip, nuget, …
- **Virtual machines**
- **Containers** (e.g. docker, podman, k8s)
- **Scripts** to setup developer environment, e.g. bash

Expert Knowledge

Cross-functional team – bridging the knowledge gap

Engineering und Software-Entwicklung

- Ideal world: All members of a team have the same level of knowledge.

- Real world: People come and go, sometimes you need rare, expensive expert knowledge of different categories
  - **T-Shaped people**
  - **nobody knows everything**

- **Automation** of complex tasks that need expert knowledge **as living documentation of how** some things needs to be done, e.g. using containers, invoke, scripts
  - Must be documented => Knowledge written down
  - Can be peer-reviewed => Knowledge being spread

Gaining Speed: Continuously driving through Pipelines

Engineering und Software-Entwicklung

Building

Engineering und Software-Entwicklung

- Build your artifacts
  - **They must work.** There is no value in testing something that does not fulfill the minimum requirement to be executable. **Don’t waste time on running time-consuming test cases.**
  - Which artifacts must be **retained** how long? (Memory = Money)

- Bundle what you need for rollout & debugging:
  - SBOM (Software bill of materials)
  - Changelog
  - Configs
  - Documentation
  - Logfiles from test case execution
  - …

Compliance Checking

Engineering und Software-Entwicklung

- Is your license compatible with those of your dependencies? Are you hurting e.g. export restrictions?
  - **Not** a **trivial** task
  - Can change with sub dependencies or by activating plugins
  - Collect **feedback** about your **license compliance** right **from the start**. Exchanging libraries at a late point in development costs much time and money.
  - **Use tools**, e.g. FOSSology, Black Duck, Apache Rat, Barista, liccheck, Eclipse SW360, …
  - Get trusted with OSS-licenses: (e.g. using https://tldrlegal.com/)
    - GPL
    - LGPL
    - MIT
    - BSD
    - CC
    - MPL
    - Apache
    - …

Versioning

Problem

Engineering und Software-Entwicklung

- Problem: How do I **assign version numbers** to my software?
  - **Time**: (e.g. by month/year) can be misleading if 04/2022 is the first release of year 2022.
  - **Sequentially**: 1, 2, 3, 4, … Are there important changes to interfaces? Is it worth to dig into Changelogs?
  - **Arbitrarily**: Throw a coin and decide whether the result makes you happy. Seriously? All interfaces may have changed and some manager might decide to increase a version number 1.0.0 to 1.0.1, just because no new features have been implemented.
  - …
- To efficiently **communicate the contents of a version**, they should be more than a number.

Versioning

Solution – Part 1

Engineering und Software-Entwicklung

- **Semantic Versioning**: MAJOR.MINOR.PATCH
  - MAJOR = BREAKING CHANGE
  - MINOR = Downward compatible new features
  - PATCH = Bug fixes, internal refactoring, non-functional changes

- **Conventional Commits**: Structured, human, and machine readable commit message
  - ***refactor**: switch version request to REST API from POST to GET*

    The REST-API now enables users to fetch the current API-version via a simple GET-call instead of the previously used POST-call.

    *Implements **#1234***

    ***BREAKING CHANGE**: The program will no longer respond to a call POST to https://myprogram.abc/api/version. Use GET instead.*

Versioning

Solution – Part 2

Engineering und Software-Entwicklung

- Tools such as semantic-release will
  - **automatically calculate a version number** for you based on the commit messages in the repository
  - add **version tags** to your repository
  - support you in writing the version number to files and committing it to the repository
  - generate a **changelog** for the version

Testing

General

Engineering und Software-Entwicklung

- **Shorten pipeline runtimes** by reusing e.g. the same **container images** used by developers. You **do not** have to **spend** much **time on installing dependencies**.

- Enforce usage of **local tools** by using them **inside the pipeline**.

- **Testing built artifacts** is more important than single lines of code.
  - Example: Unit tests executed on code may succeed because in the context of the repository everything is available, but when testing it „outside“, you can detect e.g. resources such as images that have not been bundled.

Testing
Static testing

Engineering und Software-Entwicklung

- Run static tests to speed up reviews and make them more efficient:
  - **Teach yourself**:
    - How can things be done in a better way using new language features?
    - Learn about common mistakes.
  - **Prevent bike shedding**: code formatting, name of variables, order of imports
  - **Detect careless mistakes**: if-statements followed by „;“, unused variables, … this list is endless…
  - **Expert knowledge**: Vulnerabilities, Anti-Patterns

Testing

Dynamic testing

Engineering und Software-Entwicklung

- Unit tests:
  - Easy to write and quick to execute.
  - Good for **testing algorithms** and **error handling**.
- Integration tests:
  - If integration tests are hard to write, you should **consider whether your architecture fits your needs**.
- System tests: although they are formally not the same, see acceptance tests
- Acceptance tests:
  - Acceptance test-driven development (e.g. Cucumber)
  - Very powerful, especially when doing automated deployment.
  - Depending on the system, automation can be complex (e.g. when hardware components are involved).

Triggering

Engineering und Software-Entwicklung

- Do you have projects depending on the one your pipeline is currently building?
  - If you release a new version, try to **auto-upgrade**. (e.g. depending on how strict you pin versions by just running the affected pipeline or using something like npm update, pip-tools, …)

- Run pipelines regularly – even if you do not work on a project – to
  - **early detect** updates of dependencies that make **maintenance** necessary
  - ensure that those projects still build while the **build environment maybe changed**

Destination Station: At the Customer

Engineering und Software-Entwicklung

Collecting User Feedback

“Dangerous” (GDPR, regulated by laws)

Engineering und Software-Entwicklung

- Built-In **Interview**:
  - **Ask** the user **questions** (e.g. video/audio quality)?
  - Find out whether users like **new functionality**.
  - Do they have **troubles**?
  - Ask users for **suggestions** on how **to improve the program**.
- **Monitoring** user behavior:
  - Which functions are used how often?
  - How does a user navigate through a GUI?
- Collecting **device** information:
  - Average consumption of **resources** (e.g. CPU, RAM)
  - **Information about the OS** the system is running on

Collecting Notifications

Engineering und Software-Entwicklung

- Collect notifications about **critical events**:
  - How often per day and installation is a version of an application going down?
  - How often and how long are important connections lost?
  - …
- Input for **maintaining multiple versions**:
  - Which version is used how often?
  - Under which license is the application running?
- Provide the possibility to **report errors** and **provide log files** (GDPR, laws, anonymization, passwords).
  - Logging guidelines can help to reduce risks (e.g. logging passwords, reviews)
  - Test with controlled personal data (e.g. IP, username) and search logs automatically to prevent being not compliant to laws.

The Return: Maintenance

Engineering und Software-Entwicklung

Monitoring

Engineering und Software-Entwicklung

- **Automate evaluation of user-feedback** to get alerted if something strange happens.
  - Is a new version going down more often than the last one?
  - Are users uninstalling the latest version and switch back to an older one?
  - **Keep it simple**: Trend Charts often are enough. You don’t need AI for everything!

- Know your **dependencies (SBOM)**:
  - **Automonitoring** of available **updates** of 3rd-party-software
  - **Automonitoring** of **vulnerabilities** in and **patches** for 3rd-party-software

Impact Analysis

Here we go again

Engineering und Software-Entwicklung

Remember the first thing we talked about in the beginning: **Traceability**

Summary

Engineering und Software-Entwicklung

Summary

Engineering und Software-Entwicklung

- Automate to gain fast and valuable feedback.
- You can automate much more than “just” building, testing and deploying.
- Automation that‘s not part of CI/CD-pipelines can bring a high return on invest.
- Use automation for proactive maintenance.

ESE GmbH – Technology to trust!

Engineering und Software-Entwicklung

Part of DB E.C.O. Group

Wer sind wir?

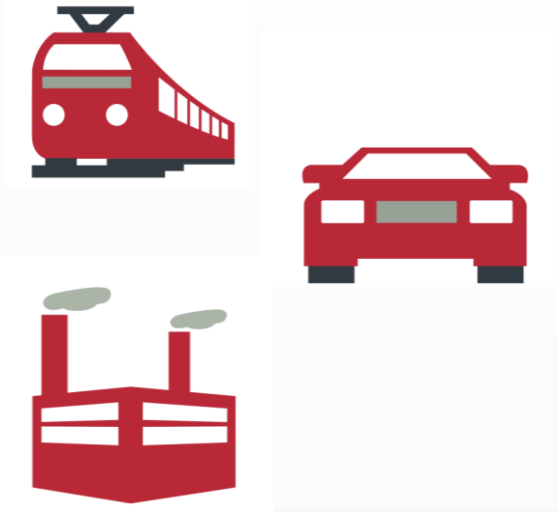
420
MITARBEITENDE

25 Jahre Erfahrung

8 Standorte

7 Geschäftsfelder

33 MIO. EURO UMSATZ



Was erwartet Dich bei uns?

- Teamgeist
- Mentoring
- Weiterbildung
- Events
- Projektvielfalt

Was bringst Du idealerweise mit?

- MINT-Studium
- Technisches Interesse
- Deutsch & Englisch
- Programmier-kenntnisse
- Interesse an Safety oder Security

Wie kannst Du starten?

- Direkteinstieg
- Werkstudenten-tätigkeit

Saskia Wieting
0174 6217336
jobs@ese.de
www.ese.de

Werkstudent Softwareentwicklung (m/w/d)

Engineering und Software-Entwicklung

Fachbereich
SW-/System-Test,
SW-Entwicklung

Standort(e)
Erlangen

Karrierelevel
Student/Praktikant

Vertragsart
Werkstudent

Deine Aufgaben	Dein Profil	Wohlbefinden bei der ESE
Unterstützung bei Konzeption, Implementierung und Verifizierung von Softwaremodulen	Du beherrschst mindestens eine Programmiersprache	Mentoring
Mithilfe in verschiedenen Softwareentwicklungs- und -Testteams	Interesse an strukturierter Softwareentwicklung und -architektur	Bezahlter Urlaub + flexible Arbeitszeiten
Erstellung von Machbarkeitsstudien, Analysen etc.	Teamfähigkeit, Eigenmotivation, fließende Deutsch- und sehr gute Englischkenntnisse	Offene Kommunikation und Kooperation
Und viele weitere spannende Themen …	Interesse oder Vorwissen über Safety oder Security	Mobile Working

Get in Contact with us

Karsten Raddatz
Branch Manager Erlangen
Phone: +49 9131 6102-984
Mobile: +49 172 41 200 80
Karsten.Raddatz@ese.de

Ralf Spengler
Lead Specialist - Software Engineer
Phone: +49 9131 6102-984
Ralf.Spengler@ese.de

Saskia Wieting
Recruiting
Phone: +49 531 23880 52
jobs@ese.de

Engineering und Software-Entwicklung

Thank you for Listening

Time to Talk

Engineering und Software-Entwicklung

References

Engineering und Software-Entwicklung

- „Die Lage der IT-Sicherheit in Deutschland 2021“ - Bundesamt für Sicherheit in der Informationstechnik (BSI), Artikelnummer: BSI-LB21/51
- Agile Manifesto: https://agilemanifesto.org/
- Starke, G. (2020). *Effektive Softwarearchitekturen: Ein praktischer Leitfaden*. Carl Hanser Verlag GmbH & Co. KG.
- Pohl, K., & Rupp, C. (2015). *Basiswissen Requirements Engineering: Aus- und Weiterbildung nach IREB-Standard zum Certified Professional for Requirements Engineering Foundation Level* (4th ed.). dpunkt.
- Spillner, A., & Linz, T. (2012). *Basiswissen Softwaretest: Aus- Und Weiterbildung Zum Certified Tester Foundation Level Nach Istqb-Standard*. dpunkt.
- Spillner, A., Roßner, T., Winter, M., & Linz, T. (2014). *Praxiswissen Softwaretest - Testmanagement: Aus- und Weiterbildung zum Certified Tester - Advanced Level nach ISTQB-Standard*. dpunkt.
- https://iso25000.com/index.php/en/iso-25000-standards/iso-25010 retrieved on March 3, 2022
- https://www.aquasec.com/cloud-native-academy/vulnerability-management/open-source-vulnerability-scanning/ retrieved on March 3, 2022
- https://oss-compliance-tooling.org/Tooling-Landscape/OSS-Based-License-Compliance-Tools/ retrieved on March 3, 2022
- https://tldrlegal.com/ retrieved on March 3, 2022
- https://semver.org/ retrieved on March 3, 2022
- https://www.conventionalcommits.org/ retrieved on March 3, 2022

Picture Credits

Engineering und Software-Entwicklung

All images that are used without providing additional information have been created, or are licensed by the *ESE Engineering und Software-Entwicklung GmbH*.