Open Source Entwicklung als Teil der Hochschullehre

Dr. István Koren, Process and Data Science, RWTH Aachen University
koren@pads.rwth-aachen.de

Arbeitskreis Open Source der GI – Themenabend Lehre & Forschung, 2022-10-05

CC BY SA

Chair of Process and Data Science
RWTH AACHEN UNIVERSITY

Agenda

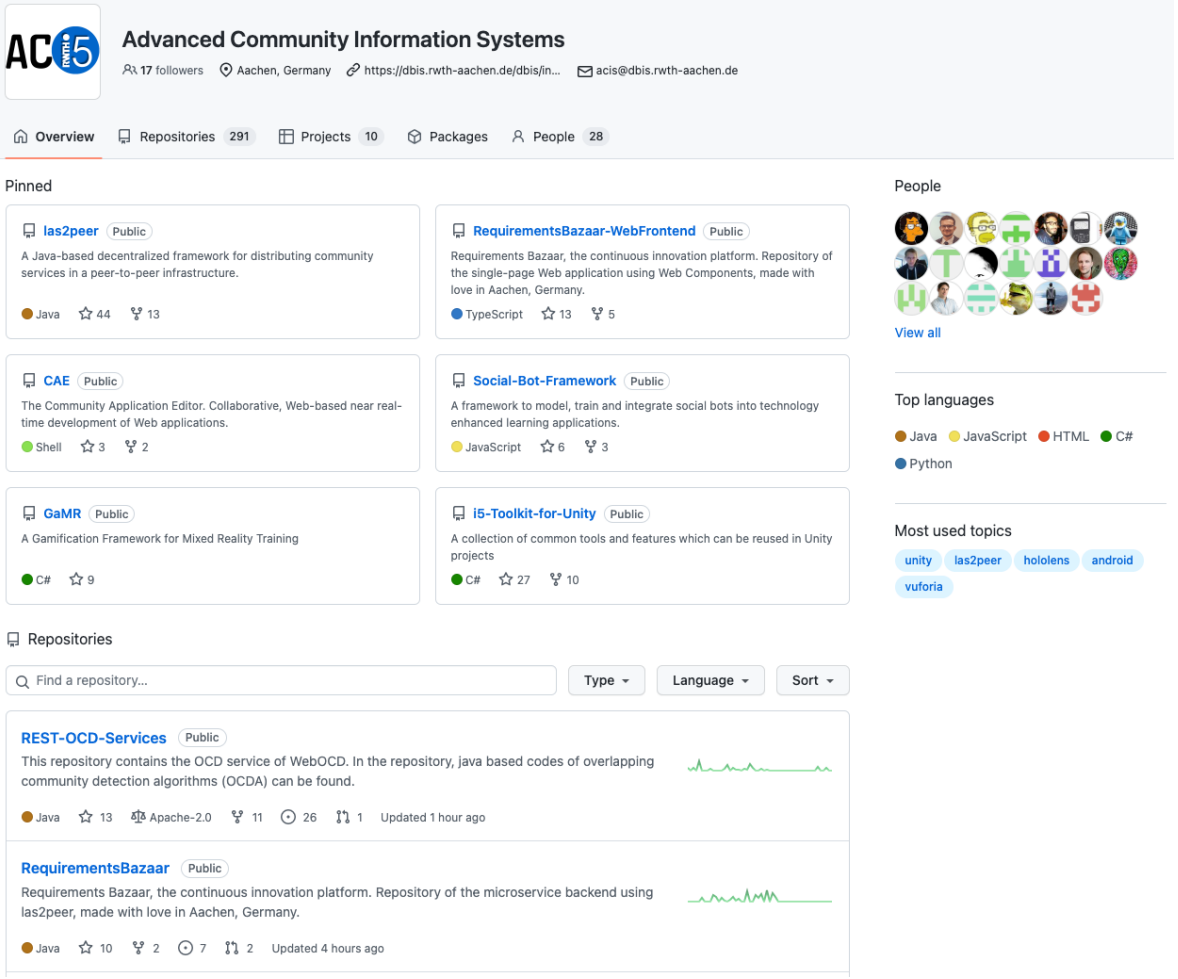
- Open-Source-Prinzipien am Institut
- Fallbeispiel: Softwarepraktikum
- Challenges
- Verstetigung

Open-Source-Prinzipien

- Alles open source!
  - Ergebnisse aus Forschungsprojekten
  - studentische Projektarbeiten
  - Code von wissenschaftlichen Hilfskräften
  - Bachelor- & Masterarbeit (den Code!)
  - permissive Lizenzen (aka Apache 2.0 / MIT)

- Eigene Vorbildfunktion
  - selbst (als Person/Lehrstuhl) alles open source anbieten
  - eigene GitHub/GitLab-Profile
  - „eat your own dog food“

- Street Credibility
  - Code Style & Git Best Practices befolgen
  - selbst Pull Requests bei anderen Projekten erstellen
  - Teilnahme bei Open-Source-Events



https://github.com/rwth-acis

Praktikum „High-tech Entrepreneurship and New Media“

Im Praktikum „High-tech Entrepreneurship and New Media“ bearbeiten Studierenden-Teams klar definierte Fragestellungen aus der lokalen Startupszene. Dabei lernen sie agile Methoden, Open-Source-Praktiken und die örtliche Startup-Szene kennen.

Ablauf Kickoff-Veranstaltung des Praktikums

- Einführung „Communities of Practice“
- Vortrag „State-of-the-Art Software Development Methods“
- Tutorial „Source Code Management“
- Demoprojekt „Dating Website“
- Agile Development Workshop (nach zwei Wochen)

Vortrag „State-of-the-Art Software Development Methods“

- Ziele
  - Vorstellung Agile Manifesto
  - Scrum im Detail
- Scrum
  - Roles
  - Sprint Artifacts
  - User Stories
  - Taskboards
- Code Style Guides
  - naming conventions
  - indentation
  - documentation
  - encoding
- Code Reviews

Scrum Roles 1/2

- Product Owner
  - Responsible for features and success of product
  - Creates and prioritizes user stories
  - Has last say on *Product Backlog*

- Development Team
  - Self-organized
  - Estimates time complexity of tasks
  - Derives smaller tasks from user stories
  - Independently decides *how* to fulfill tasks

Scrum Roles 2/2

- Scrum Master
  - Responsible for successful Scrum
  - Moderation of meetings
  - Makes sure developers are fine

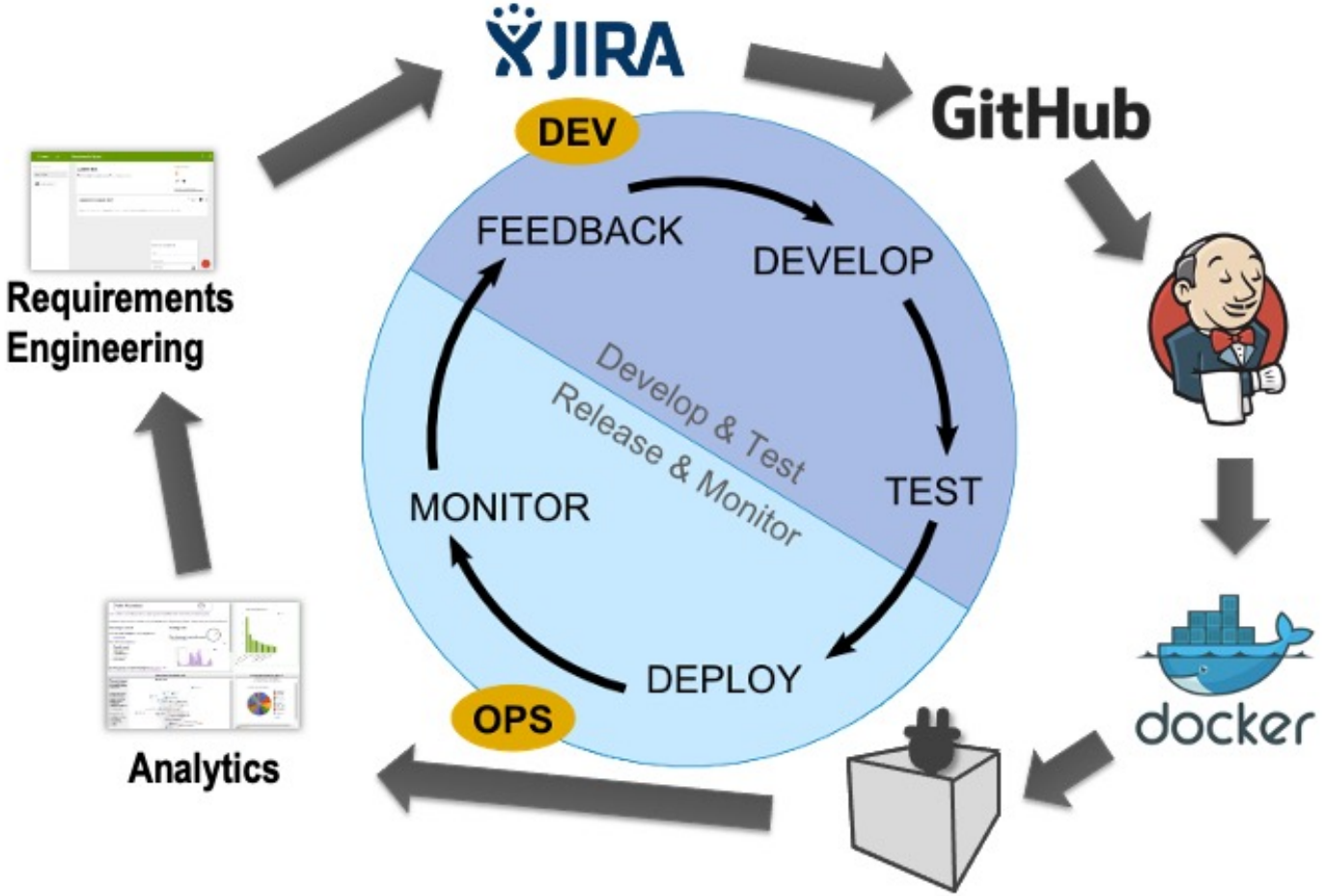
- In Agile development, (good or bad) results are attributed to the team not individuals
- Collective Code Ownership
  - Everybody is responsible for all the code
  - Everyone may change all parts of the code

Tutorial „Source Code Management“

- Source Control
  - welche Dateien haben sich geändert?
  - wer hat Änderungen gemacht?
  - wann sind die Änderungen erfolgt?
  - wie unterscheiden sich die Dateien?
  - warum wurden sie verändert?

- Branching & Merging Modell
  - 2 Branches: *main* und *develop*
  - 1 branch pro Issue aus Backlog
    - *feature/login*
    - *bugfix/memberpage*
  - sobald fertig, Pull Request gegen *develop*
  - schließlich Review von Teammember

- Continuous Integration
- Continuous Delivery
- Continuous Deployment



DevOps Lebenszyklus und Tools

Agile Development Workshop (nach zwei Wochen)

- Ziele
  - herausfinden, was an agilen Methoden (insb. Scrum) bereits bekannt ist
  - spielerisch agile Prinzipien erarbeiten
  - Antipatterns erleben
  - Best Practices verstetigen

- „Scrum from Hell“
  - eignet sich für Teams à 6-8 Teilnehmenden
  - simuliert Scrum-Meeting
  - Teilnehmende bekommen (geheim) ein „Impediment“ aka Torpedo-Ziel
  - restlichen Anwesenden sollen herausbekommen, was schief gegangen ist

- „99 Testballons“
  - Teams müssen in mehreren Sprints so viele Ballons wie möglich produziert
  - anfangs sehr vage Kundenanforderungen, später immer präziser

Challenges

- „Code-Scham“
  - „Mein Code ist noch nicht schön genug!“
  - „Da möchte ich noch Test-Cases reinmachen!“
  - „Darf ich das überhaupt open source stellen?“
- Licensing
- Firmenpolicies & NDAs
- Hochschulpolicies…
- Was oft hilft: Aufklärung
  - Lizenzen erklären
  - GitHub-Profil zur Selbstvermarktung nutzen
  - Firmen schauen immer öfter auf öffentliche GitHub-Profile
  - Software-Einsatz tracken
  - realistisch bleiben

Weitere Challenges der Softwareentwicklung in großen Forschungsprojekten…

TABLE I
COMMON SOFTWARE ENGINEERING CHALLENGES IN RESEARCH PROJECTS

Early Decisions	Architectural decisions with considerable scope need to be made early in the project, as it is well known that early mistakes in a project are the costliest [21]. These decisions should also include reflections on work package and task structure.
Short Cycles	In the time frame of an R&D project, development cycles must be kept short, and initial architectures and prototypes must be provided early in the project to allow for frequent refinement loops driven by research and end-user involvement.
Sustained Impact	Deployed solutions have to scale both horizontally and vertically, as project outcomes should be exploited and sustained beyond the project's funding period. Typically, exploitability and impact of project results are key performance indicators for funding agencies, although market-readiness of ICT research project outcomes is often limited [22]).
Distributed Community	The initial community is a union of individuals affiliated to different partners. It needs to act as seed for a growing and flourishing wider community involving external members in a limited amount of time.
Support Infrastructure	The development process and environment, including procedures and tools for source code management, issue tracking, software branding, and similar, need to be defined, implemented and maintained. Eventual licensing and maintenance costs and efforts may not be underestimated.
Licensing	An OSS strategy usually helps sustain and transfer project results into practice and is thus interesting for funding agencies [8]. By definition, any OSS must ship with an OSS license. In large project consortia, agreement on one license for all developed components is hard to reach. The use of third-party OSS and differing exploitation goals among partners impose the use of different, however compatible licenses.
Stakeholder Engagement	Societal software projects inherently require instruments to achieve wide coverage of stakeholder engagement. Such instruments thus need to be as inclusive and supportive as possible even for non-technical audiences.
Baseline	In research projects, the state of the art builds the baseline for all activities. "Quick & dirty" approaches ignoring existing knowledge may thwart the cutting-edge research ambitions of the project. However, software engineering processes need to be flexible to understand and adapt to emergent change, resulting from the complex and never ending interplay of people influencing technology and vice versa, as we find it in socio-technical systems and as it is described by ICT-related adaptations of structuration theory [23].
Unknown Territory	Research outcomes inherently remain unspecified beforehand, while the research methodology leading to those expected outcomes must be well-defined. This poses a concrete challenge, as software engineering activities need to synchronize with primary research activities.
Success Awareness	Any software artifact developed within research projects requires rigorous evaluation with respect to success as complex construct combining quality and impact as they are perceived by different relevant stakeholder groups. With respect to sustainability and given that success itself is a highly dynamic and context-dependent construct, success awareness must be conceptualized as result of ongoing long-term developmental evaluation even beyond project lifetime [9], [19].
Decision Making	Although research institutions collaborate in projects with legally binding contracts and agreed work plans, all of them have different motivations, internal agendas and working processes to be aligned with the overall project agenda. This diversity often poses an obstacle to efficient decision making.

Mehr dazu:

Preparing Research Projects for Sustainable Software Engineering in Society
Renzel, Dominik, Koren, István, Klamma, Ralf, and Jarke, Matthias
In Proceedings of the 2017 IEEE/ACM 39th IEEE International Conference on Software Engineering (ICSE), Buenos Aires, Argentina, May 20-28, 2017.
doi:10.1109/ICSE-SEIS.2017.4

Verstetigung

- Hackathons: teilnehmen (lassen) & selbst welche anbieten

- Open Source Treffen: Studierende mitnehmen & vorstellen lassen
  - FOSDEM: jährliches Treffen in Brüssel, z.B. Lightning Talks (Januar/Februar)
  - Chemnitzer Linux-Tage (März)
  - FrOSCon: jährliches Treffen an Hochschule Bonn-Rhein-Sieg (August)

- GitHub ist immer noch öffentlichkeitswirksamer als ein self-hosted GitLab

- Eat your own dog food!
  - Projekte über mehrere Jahre laufen lassen
  - Maintainer bestimmen
  - inkrementell Features updaten/hinzufügen

- immer noch & wieder Aufklärungsarbeit nötig, gerade bei Kolleg:innen ohne Informatik-Hintergrund

Zusammenfassung

- Voraussetzungen
  - Infrastruktur mit Code Hosting, Issue Tracking, CI, CD-Pipelines
  - genug Wissen über gängige Praktiken
  - Lizenzen abklären: was möchte/darf ich?

- Open Source als Teil von studentischen Arbeiten
  - einfach mal verpflichtend für Bachelor- & Masterarbeiten einführen…

- Verstetigung der Open-Source-Aktivitäten erfordert ein wenig Aufwand…
  - klare Governance-Strukturen
  - aktive Kommunikation
  - hinreichende Dokumentation
  - Praktiken challengen

Vielen Dank
für Ihre Aufmerksamkeit

Dr. István Koren
Chair of Process & Data Science
RWTH Aachen University

koren@pads.rwth-aachen.de