Technische Universität Braunschweig

SURESOFT:

Ein Ansatz für nachhaltige Softwareent
in der Wissenschaft

Suresoft
SUSTAINABLE RESEARCH SOFTWARE

Robert Strötgen (et. al.), Technische Universität Braunschweig

DINI-Workshop “Forschungssoftware managen” 15.09.2022

Who are we?

18 People from 7 Institutes and Facilities

INSTITUT FÜR AKUSTIK

Institute of Operating Systems and Computer Networks

IfN Institut für Nachrichtentechnik

iRMB

Institut für Physikalische und Theoretische Chemie

Technische Universität Braunschweig

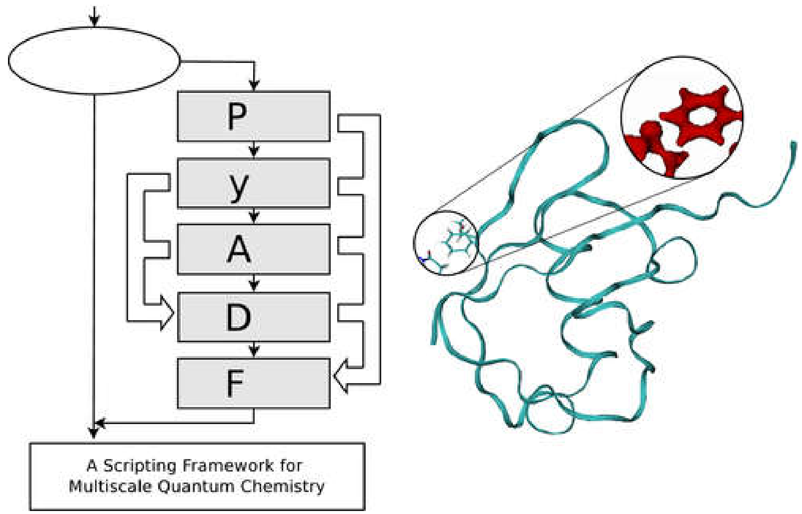
University Library & Gauß-IT-Zentrum

SURESOFT Projects

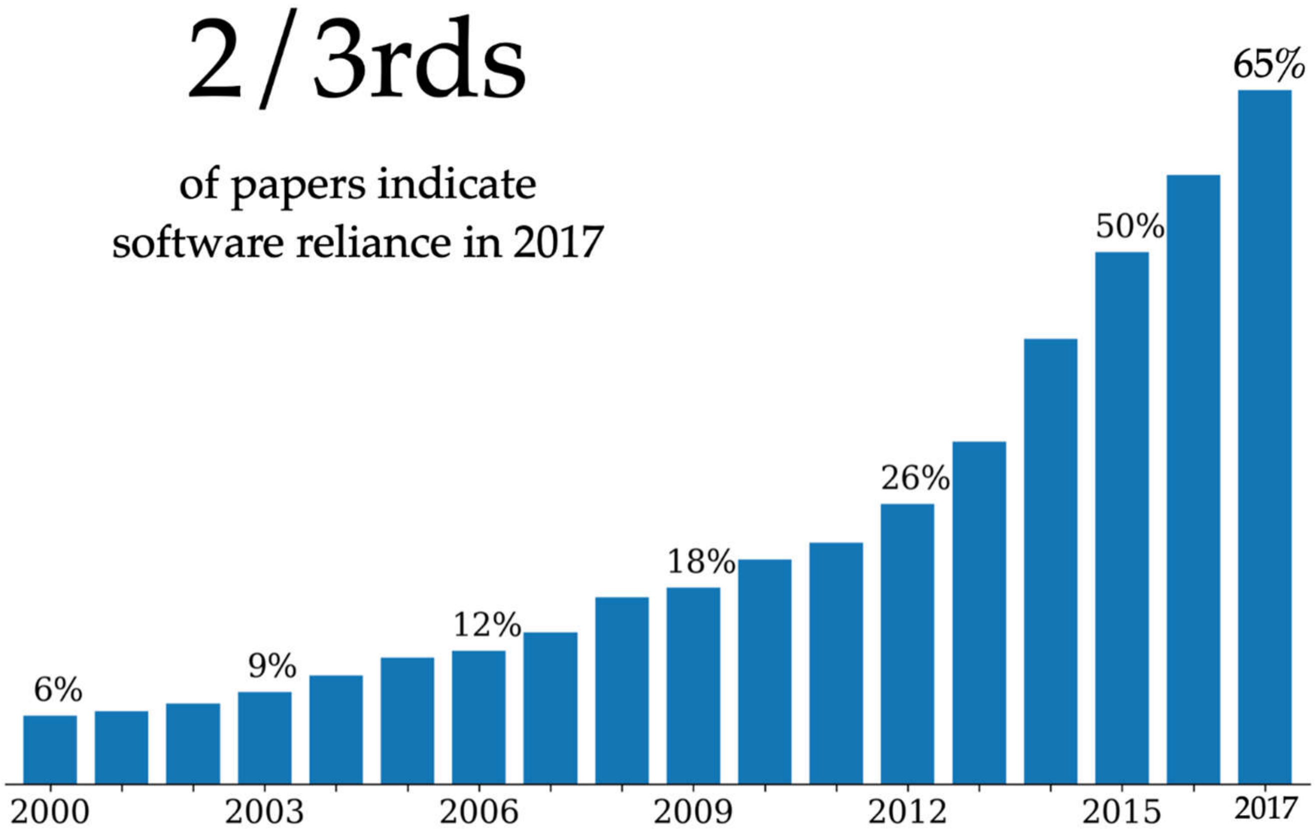
INSTITUT FÜR AKUSTIK
TECH elPaSo

VIRTUAL FLUIDS

Themis



Publications relying on software



https://bit.ly/37XEJ2u

Use of research software

- 92% of academics use research software
- 69% say that their research would not be practical without it
- 56% develop their own software

https://bit.ly/2zZPhSa

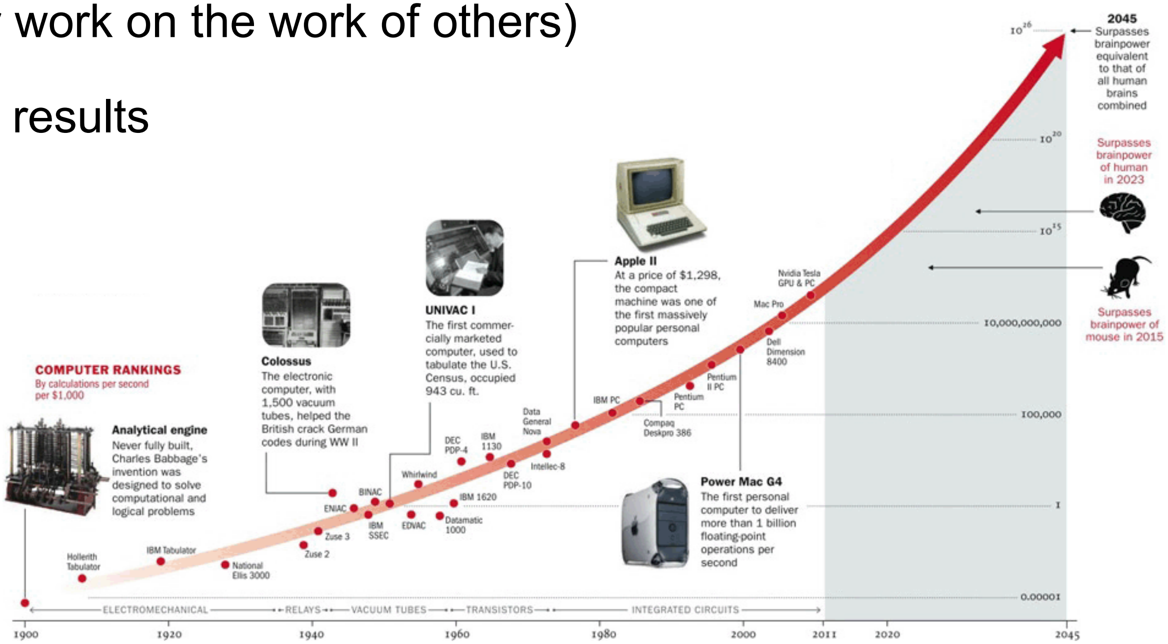
https://bit.ly/2BAvzwQ

The typical scientific developer

- highly educated
- finds it easy to learn programming languages (university courses and/or self educated)
- no formal education in software engineering
- prefer writing software themself
- no reputation for developing software
- software itself has no value - only a tool for domain specific research
- time pressure: publish or perish
- doesn’t sees her- or himself as a software developer

Growing demands on scientific software

- increasing complexity (e.g. multi physics, multiple groups)
- longer life span (base your work on the work of others)
- reproducible and verifiable results

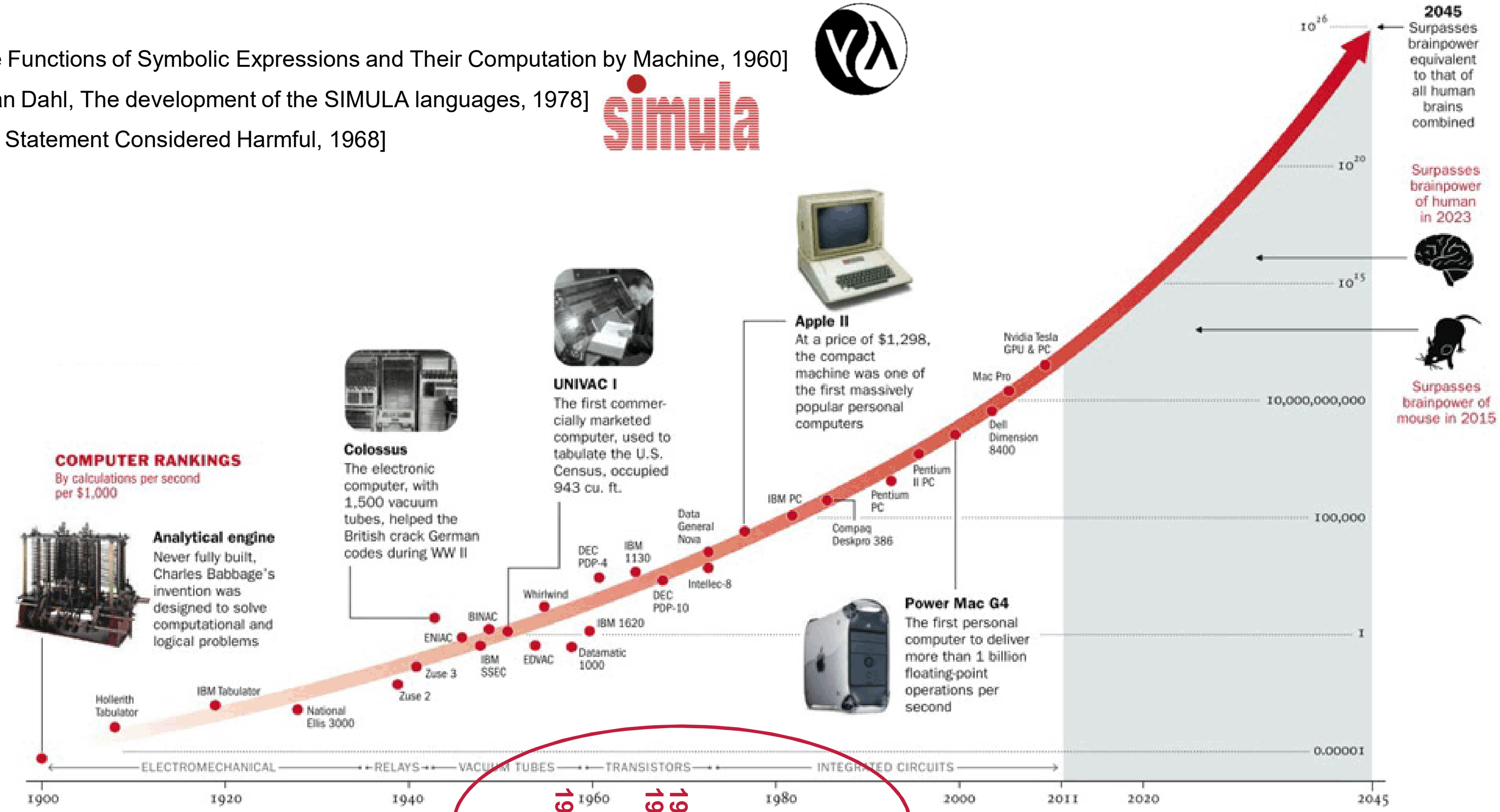


Growth in technology vs. software development paradigms

[1] [John McCarthy, Recursive Functions of Symbolic Expressions and Their Computation by Machine, 1960]

[2] [Kristen Nygaard, Ole-Johan Dahl, The development of the SIMULA languages, 1978]

[3] [Edsger W. Dijkstra, Go To Statement Considered Harmful, 1968]



1958 1966 1968

LISP[1] OOP[2] Structured Programming[3]

Technische Universität Braunschweig

Suresoft
Sustainable Research Software

Take Home Messages

In accordance to Wirth's law one can argue:

"Software systems grow faster in size and complexity than methods to handle complexity are invented."

[Niklaus Wirth, "A Plea for Lean Software", 1995]

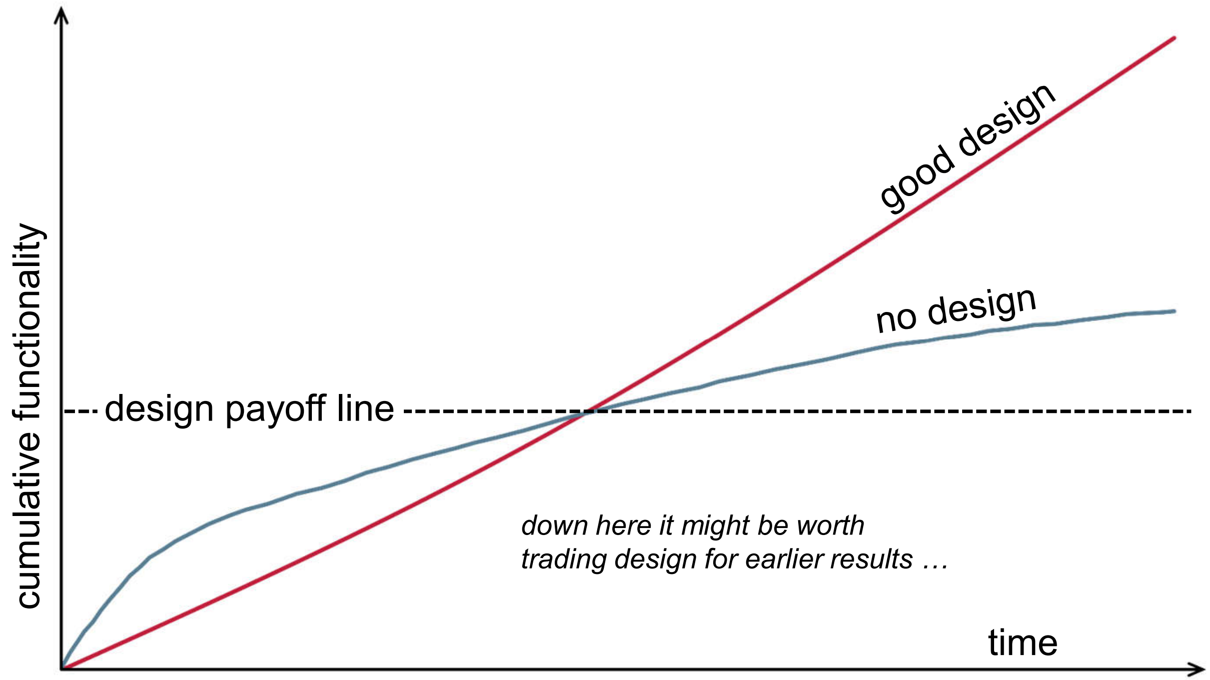
» We need to **make the best possible use of the software development techniques available** to cope with the growth in complexity.

"The gap between the best software engineering practice and the average practice is very wide — perhaps wider than in any other engineering discipline. [...] **The difference between the the great and the average approach an order of magnitude.**"

[Frederick P. Brooks, No Silver Bullet: Essence and Accidents of Software Engineering, 1987]

Productivity Crisis

- floating point performance is constantly rising
- time-to-solution is inceasing
- scientists spend 50% of the time finding bugs
  [P. Prabhu, A Survey of the Practice of Computational Science, 2011]



Design Stamina Hypothesis
https://bit.ly/2A64CAR

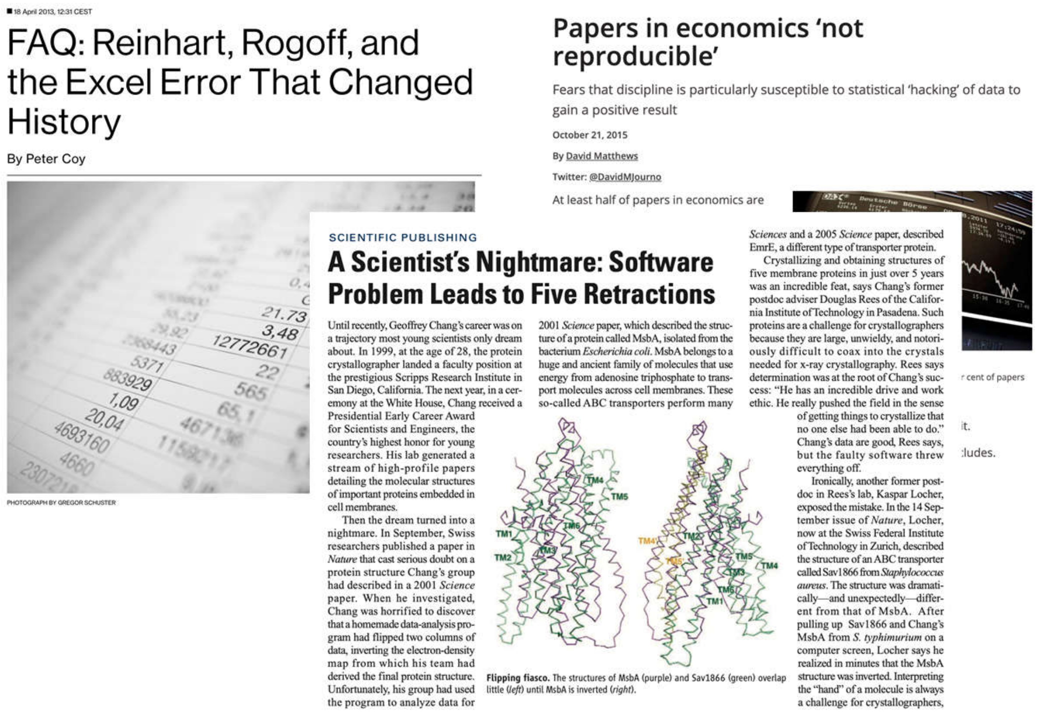
Productivity Crisis

"The only way to go **fast** is to go **well**."

Robert C. Martin

Credibility Crisis

Questionable reliability, accuracy, reproducibility and verifiability of the results …



18 April 2013, 12:31 CEST

FAQ: Reinhart, Rogoff, and the Excel Error That Changed History

By Peter Coy

PHOTOGRAPH BY GREGOR SCHUSTER

Papers in economics 'not reproducible'

Fears that discipline is particularly susceptible to statistical 'hacking' of data to gain a positive result

October 21, 2015

By David Matthews

Twitter: @DavidMJourno

At least half of papers in economics are

SCIENTIFIC PUBLISHING

A Scientist's Nightmare: Software Problem Leads to Five Retractions

Until recently, Geoffrey Chang's career was on a trajectory most young scientists only dream about. In 1999, at the age of 28, the protein crystallographer landed a faculty position at the prestigious Scripps Research Institute in San Diego, California. The next year, in a ceremony at the White House, Chang received a Presidential Early Career Award for Scientists and Engineers, the country's highest honor for young researchers. His lab generated a stream of high-profile papers detailing the molecular structures of important proteins embedded in cell membranes.

Then the dream turned into a nightmare. In September, Swiss researchers published a paper in *Nature* that cast serious doubt on a protein structure Chang's group had described in a 2001 *Science* paper. When he investigated, Chang was horrified to discover that a homemade data-analysis program had flipped two columns of data, inverting the electron-density map from which his team had derived the final protein structure. Unfortunately, his group had used the program to analyze data for

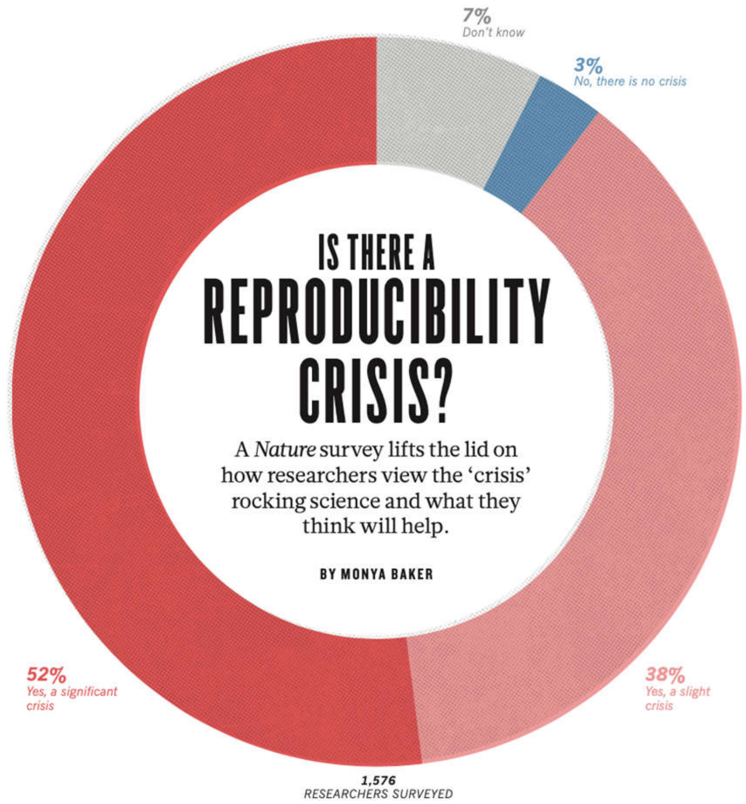
2001 *Science* paper, which described the structure of a protein called MsbA, isolated from the bacterium *Escherichia coli*. MsbA belongs to a huge and ancient family of molecules that use energy from adenosine triphosphate to transport molecules across cell membranes. These so-called ABC transporters perform many

Flipping fiasco. The structures of MsbA (purple) and Sav1866 (green) overlap little (*left*) until MsbA is inverted (*right*).

Sciences and a 2005 *Science* paper, described EmrE, a different type of transporter protein.

Crystallizing and obtaining structures of five membrane proteins in just over 5 years was an incredible feat, says Chang's former postdoc adviser Douglas Rees of the California Institute of Technology in Pasadena. Such proteins are a challenge for crystallographers because they are large, unwieldy, and notoriously difficult to coax into the crystals needed for x-ray crystallography. Rees says determination was at the root of Chang's success: "He has an incredible drive and work ethic. He really pushed the field in the sense of getting things to crystallize that no one else had been able to do." Chang's data are good, Rees says, but the faulty software threw everything off.

Ironically, another former postdoc in Rees's lab, Kaspar Locher, exposed the mistake. In the 14 September issue of *Nature*, Locher, now at the Swiss Federal Institute of Technology in Zurich, described the structure of an ABC transporter called Sav1866 from *Staphylococcus aureus*. The structure was dramatically—and unexpectedly—different from that of MsbA. After pulling up Sav1866 and Chang's MsbA from *S. typhimurium* on a computer screen, Locher says he realized in minutes that the MsbA structure was inverted. Interpreting the "hand" of a molecule is always a challenge for crystallographers,



https://go.nature.com/2DgtDKR

SURESOFT-Approach

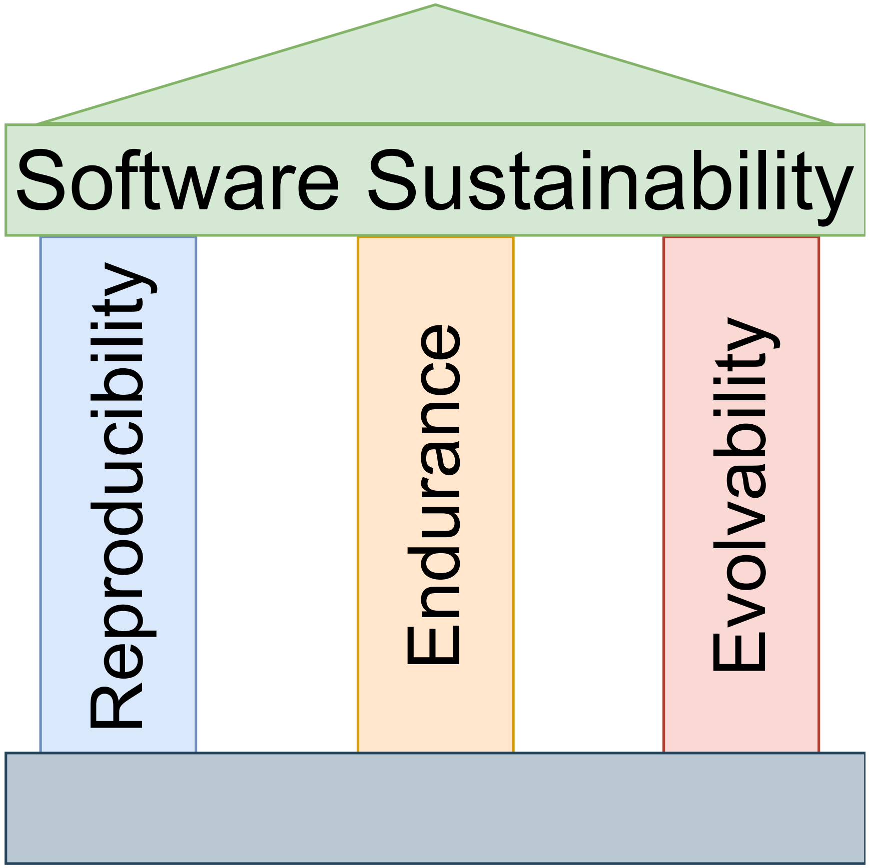
Common problems of research software

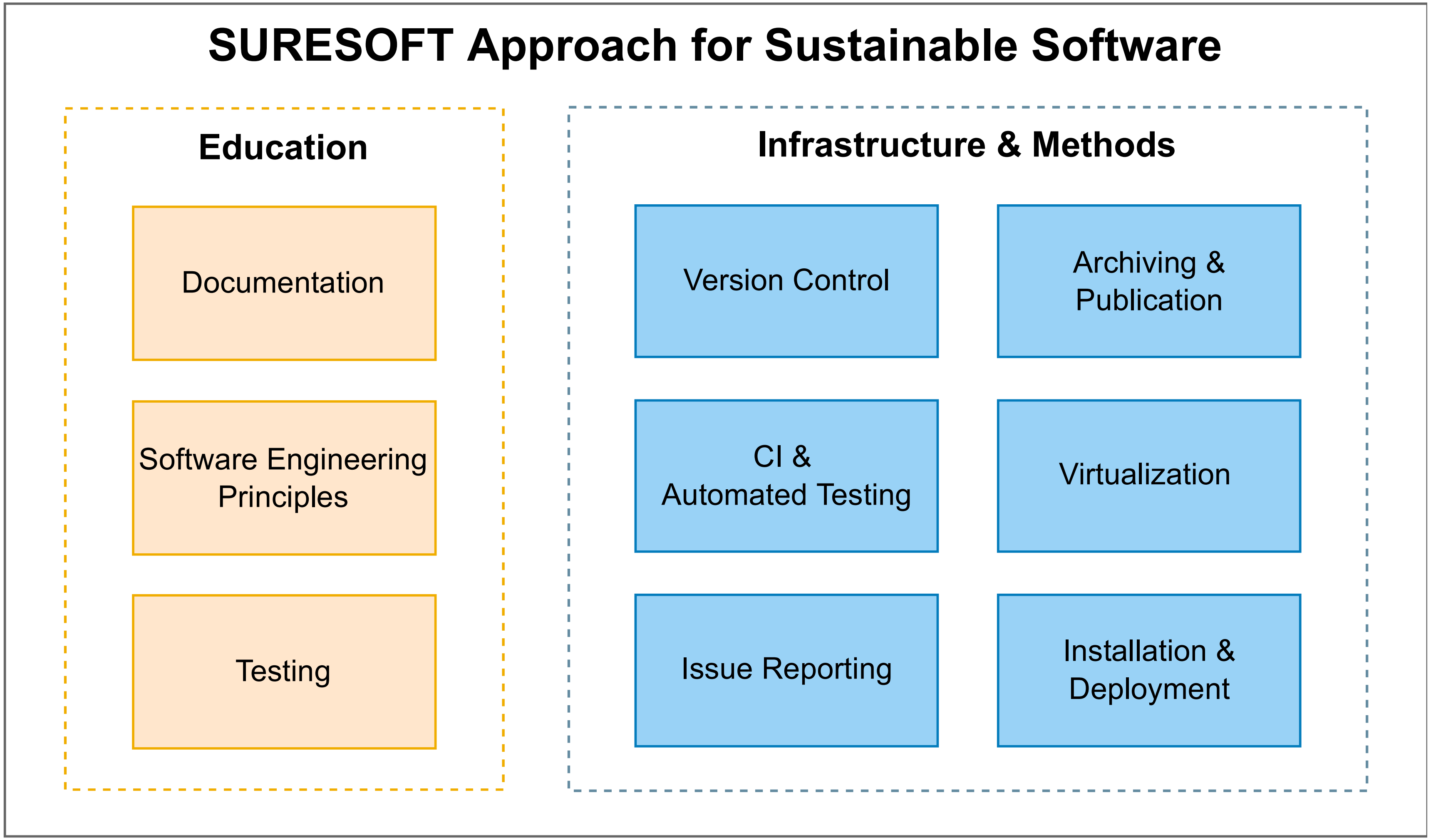
1. Software has low code quality

2. Software is neither published nor documented

3. Software depends on a specific runtime environment (e.g third party libraries), which may not be available to other researchers

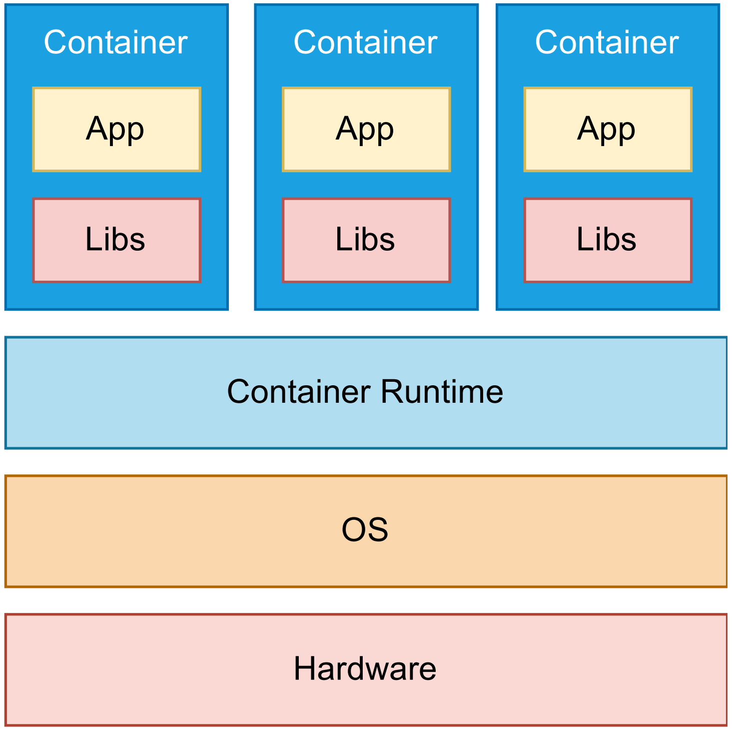
Software sustainability




SURESOFT Approach for Sustainable Software
Education
Documentation
Software Engineering Principles
Testing
Infrastructure & Methods
Version Control
Archiving & Publication
CI & Automated Testing
Virtualization
Issue Reporting
Installation & Deployment

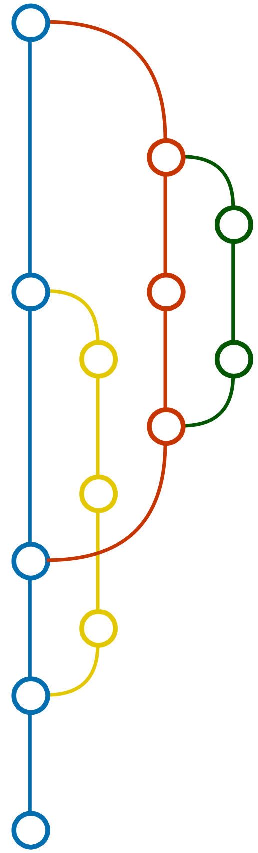
Container technologies

- Docker in CI, Singularity in HPC
- Encapsulate entire runtime environment, including dependencies
- Easy to share and use Ensures reproducibility
- Scripted environment provides basic documentation

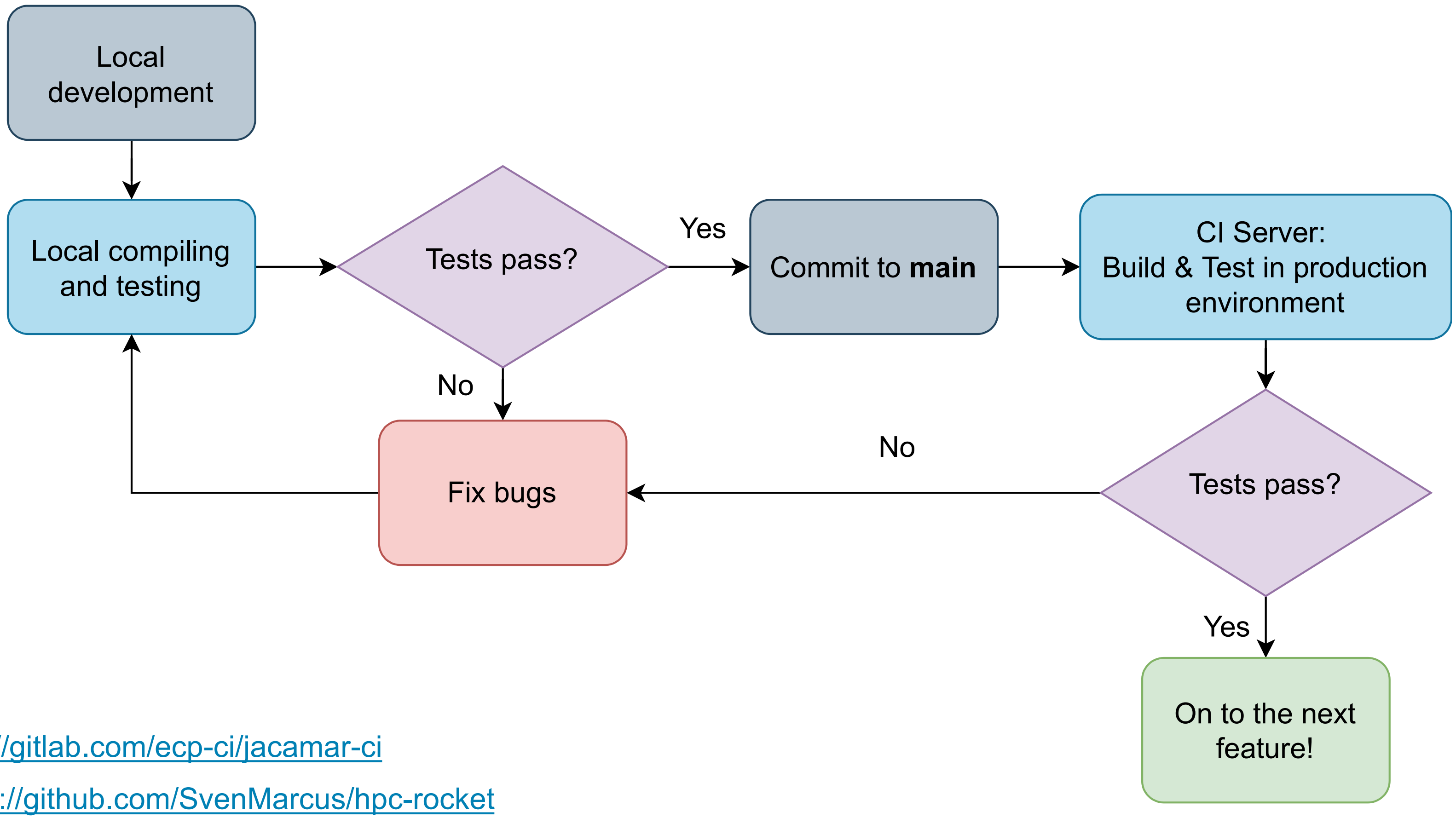


Version Control

- Track and manage changes of source code
- Commits create versions with unique identifier, documenting changes over time
- Enable collaboration through centralized repository hosting platforms (e.g. GitLab)



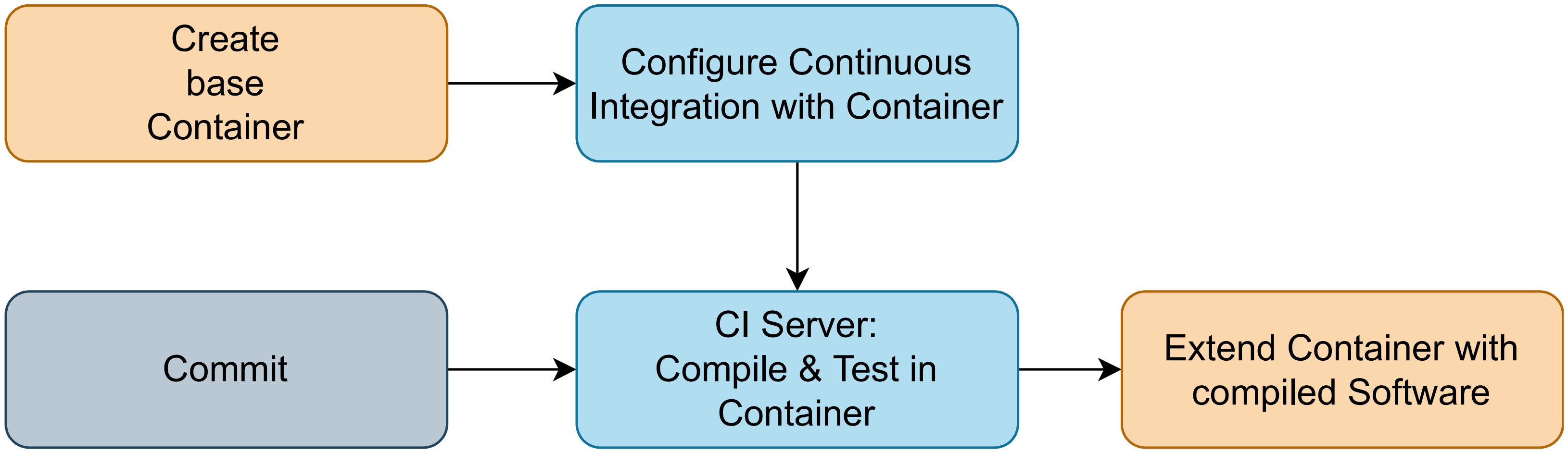
Continuous integration



Jacamar CI: https://gitlab.com/ecp-ci/jacamar-ci

HPC-Rocket: https://github.com/SvenMarcus/hpc-rocket

Continuous analysis

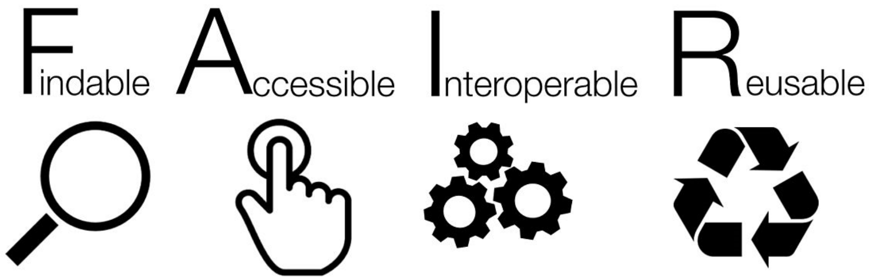


https://doi.org/10.1038/nbt.3780

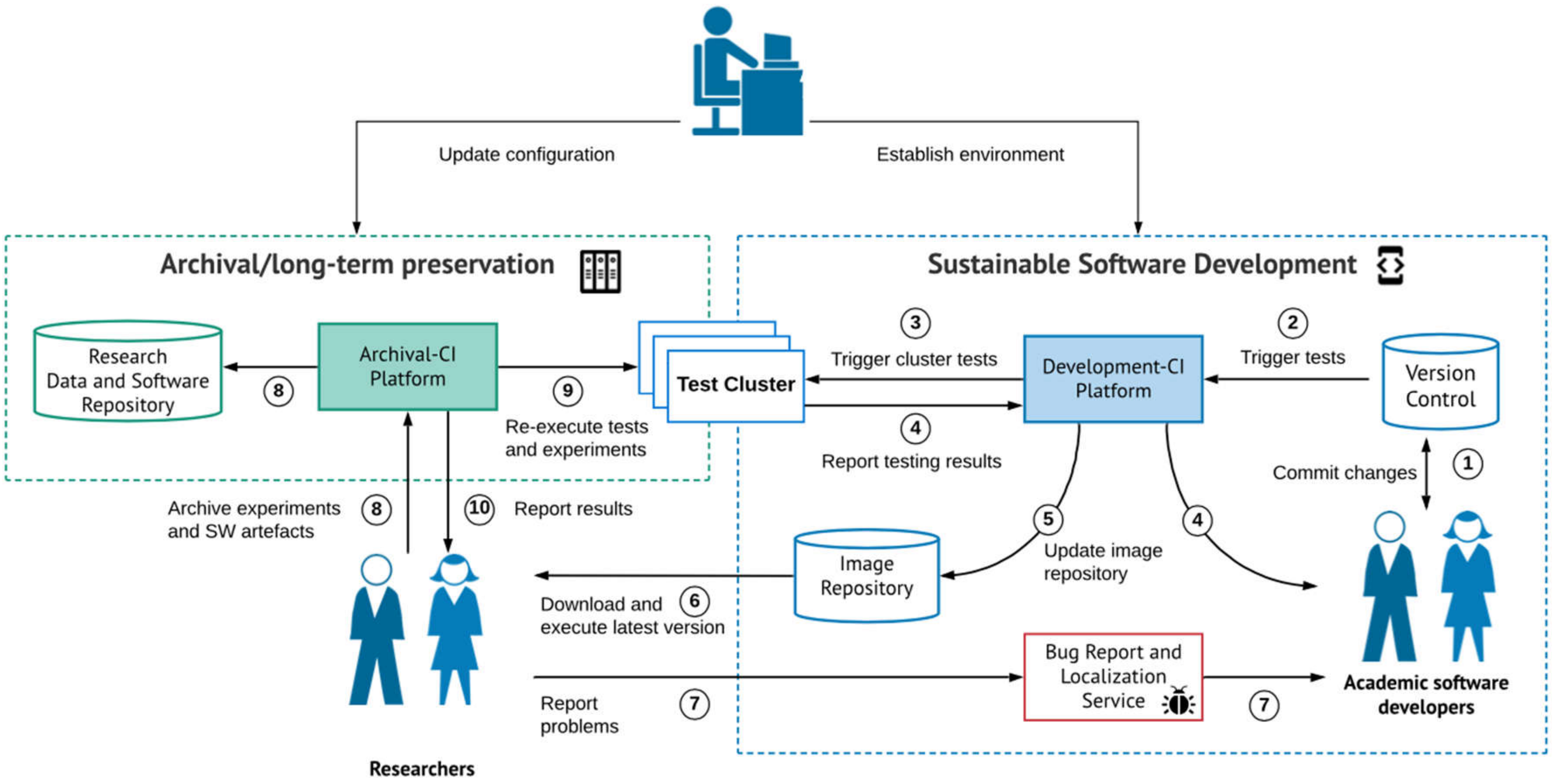
Publication & Archiving

- publish & archive the source code and the compiled executable together with a complete runtime environment in an accessible repository

- provide meaningful metadata including a unique identifier (DOI) to ensure citability, findability and reusability according the FAIR principles.



SURESOFT workflow



Education & Support

- Documentation
- Software Engineering Principles:
  - SOLID
  - Design Patterns
- Testing, TDD
- Version Control & CI
- Containerization
- Research Data Management & Long-term archiving
- Software Licensing

Workshops – Every 4 Weeks

1. Version Control using Git - June 13
2. Clean Code and Refactoring - July 11
3. Introduction to Software Testing - August 8
4. Introduction to Continuous Integration (CI) using GitLab, Github and Containers – September 5
5. Principles of Software Engineering – October 3
6. Introduction to Design Patterns – October 31
7. Working with legacy code – November 28
8. Test Driven Development – January 23
9. Documentation – February 20

Conclusion & Future Work

- The project has been running since 2020 with five software projects from different fields
- We started establishing a workflow and infrastructure and applied it to selected projects
- We are currently working on guidelines for sustainable research software to support research software developers
- Workshops about software engineering principles and modern tools within TU Braunschweig
- Eventually expand beyond TU Braunschweig to share what we have learned

ACKNOWLEDGMENTS

The SURESOFT project is funded by the German Research Foundation (DFG) as part of the “e-Research Technologies” funding programme under grants: EG 404/1-1, JA 2329/7-1, KA 3171/12-1, KU 2333/17-1, LA 1403/12-1, LI 2970/1-1 and STU 530/6-1.