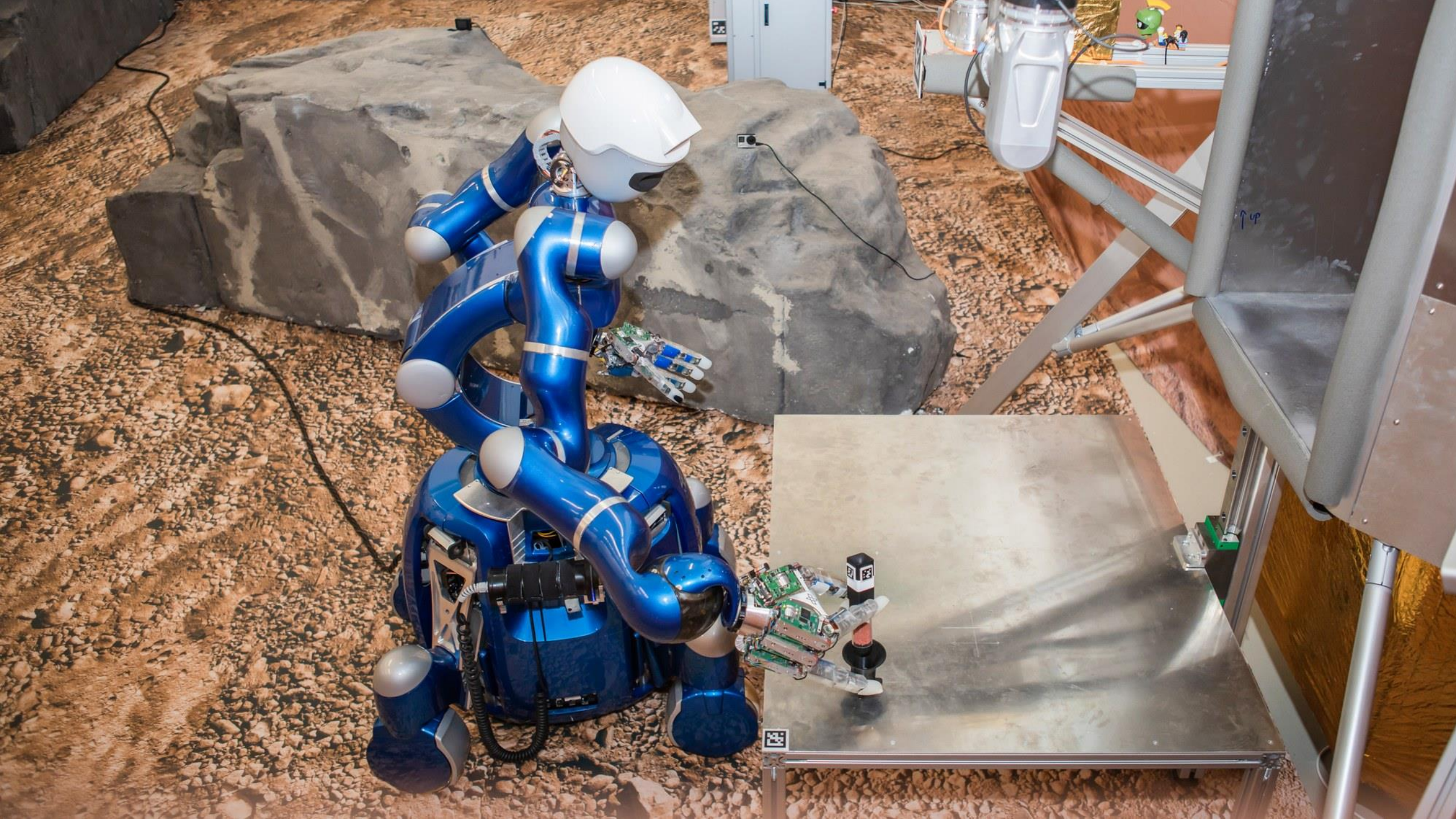
RESEARCH SOFTWARE: AN OVERLOOKED REALM IN SOFTWARE ENGINEERING RESEARCH

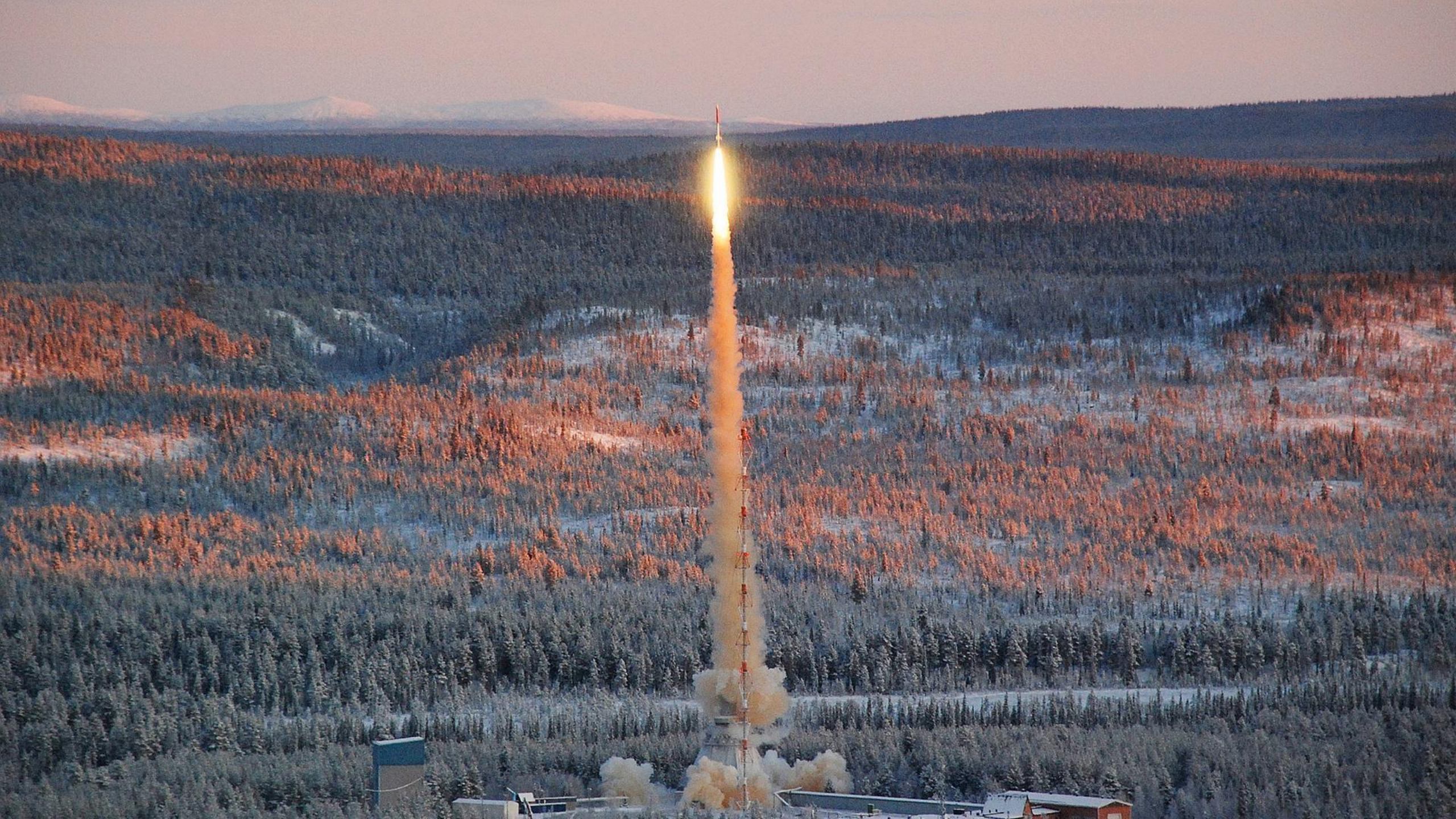
Prof. Dr. Michael Felderer

UNIVERSITY OF COLOGNE

DLR




MAIUS 2

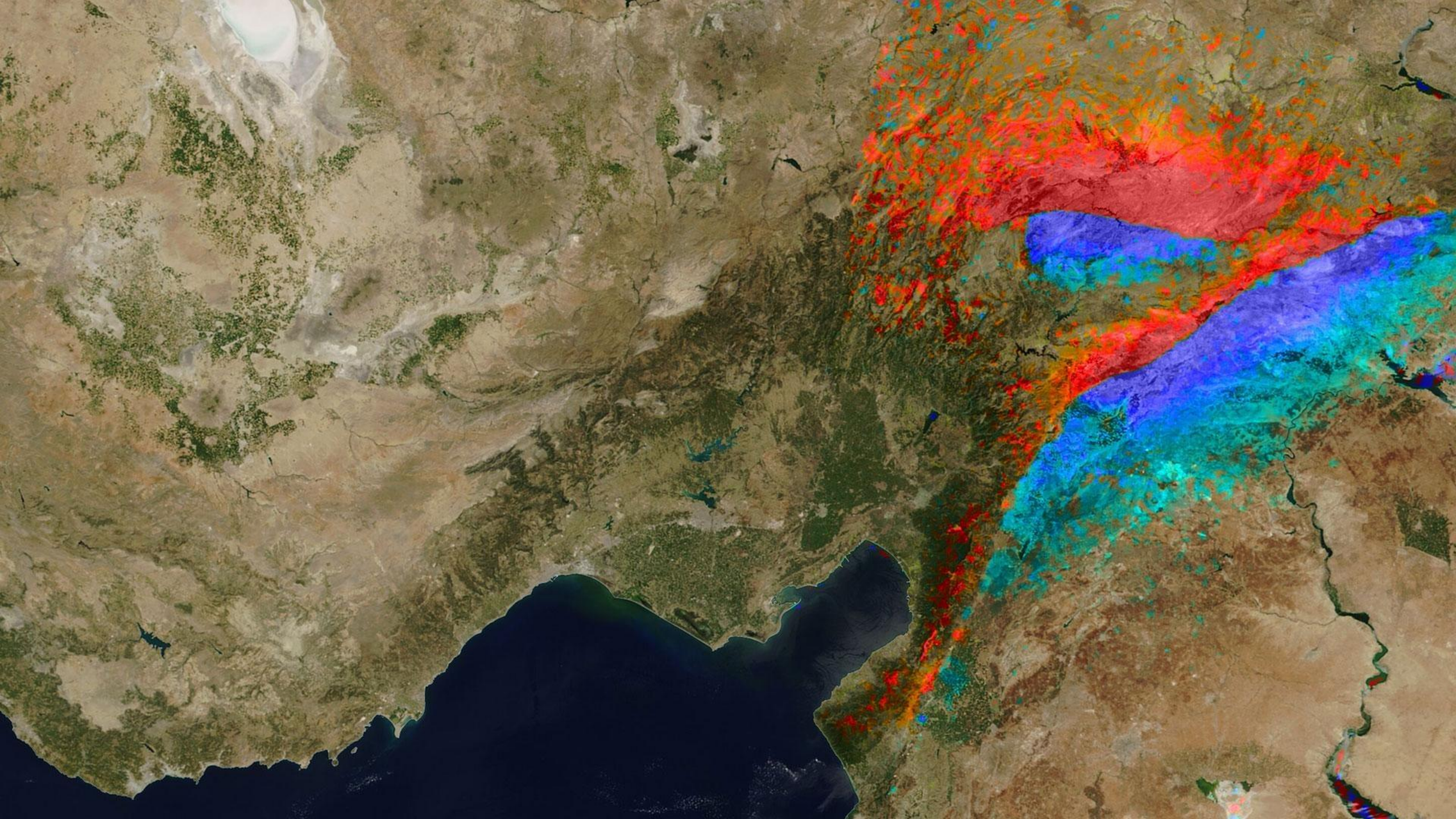


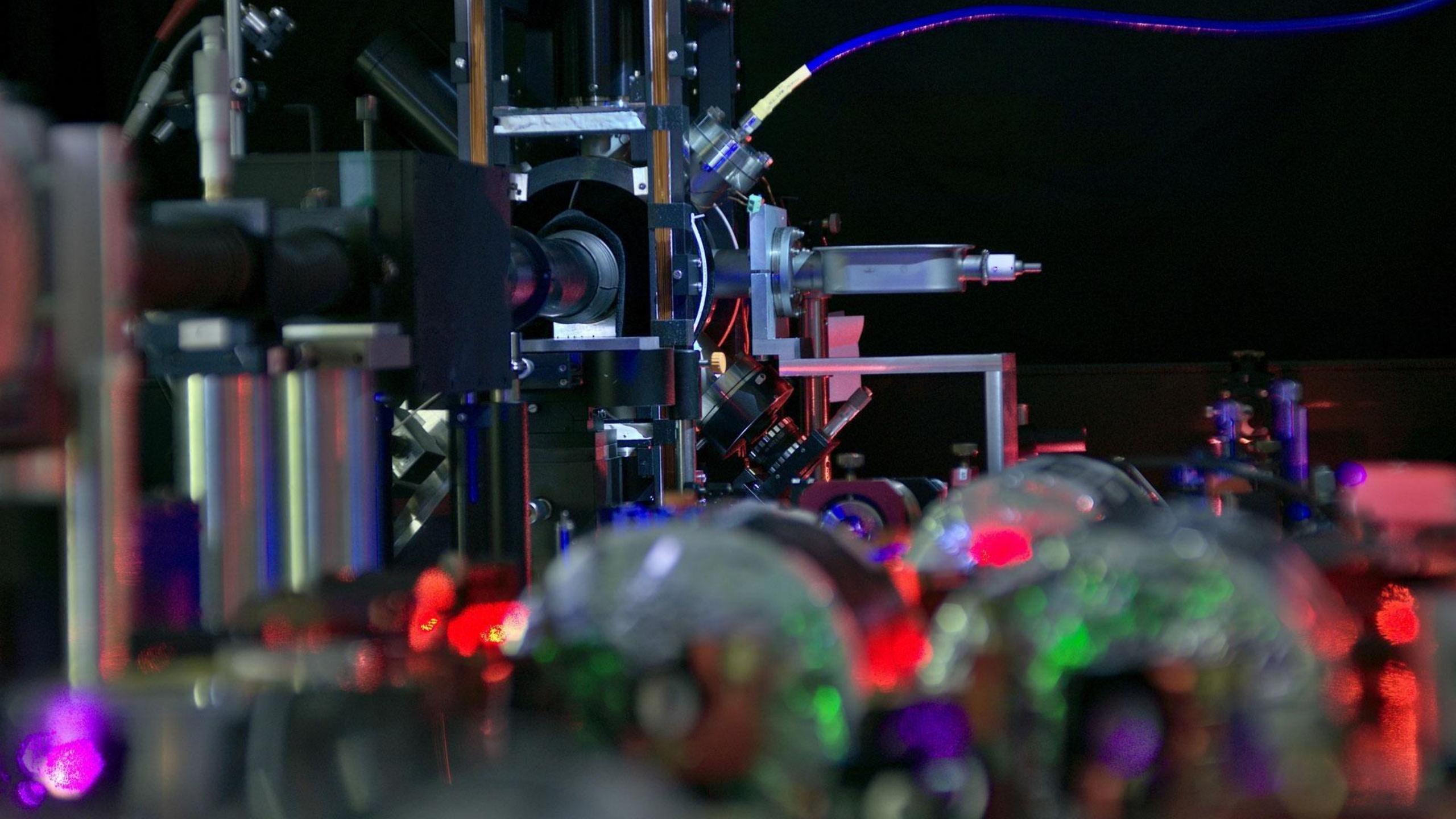

SOFIA
STRATOSPHERIC OBSERVATORY
FOR INFRARED ASTRONOMY
NASA
DLR
N747NA


699M
H2
HYDROGEN
renfe
adif
CAF
TOYOTA


SU
Deutsches Zentrum
für Luft- und Raumfahrt


GT
96





DLR in Numbers

10,000 Employees

20% develop software

55 Institutes and Facilities

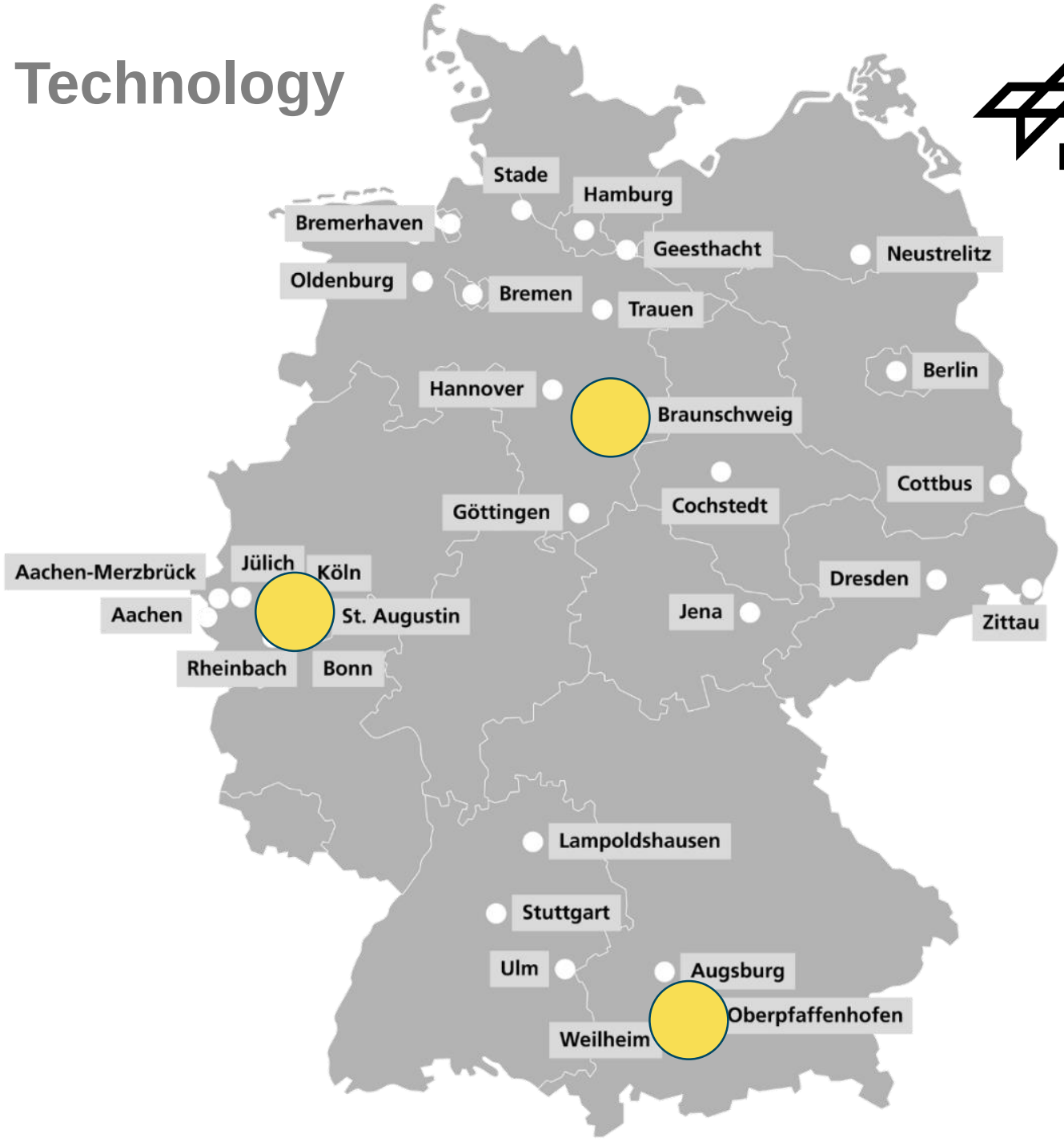
35 Locations and Offices

DLR Institute for Software Technology

200 Employees

3 Departments

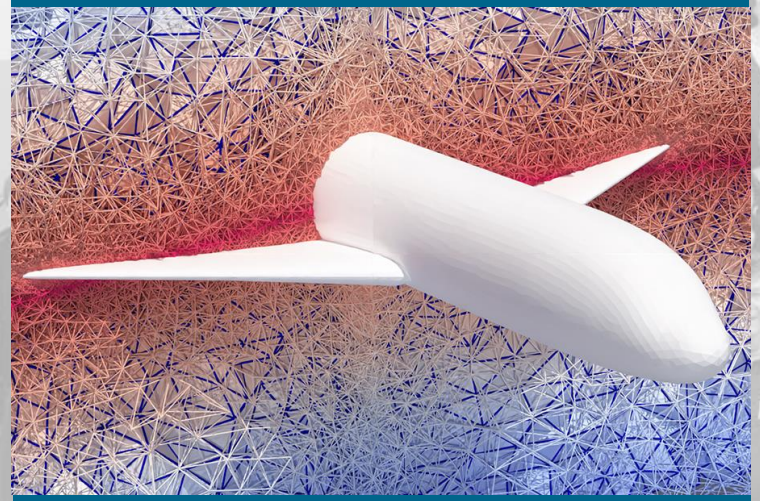
3 Main Locations



https://www.dlr.de/sc/

DLR Institute for Software Technology

Software for Aeronautics and Space



Visualisation in VR and AR

Quantum Software and Algorithms

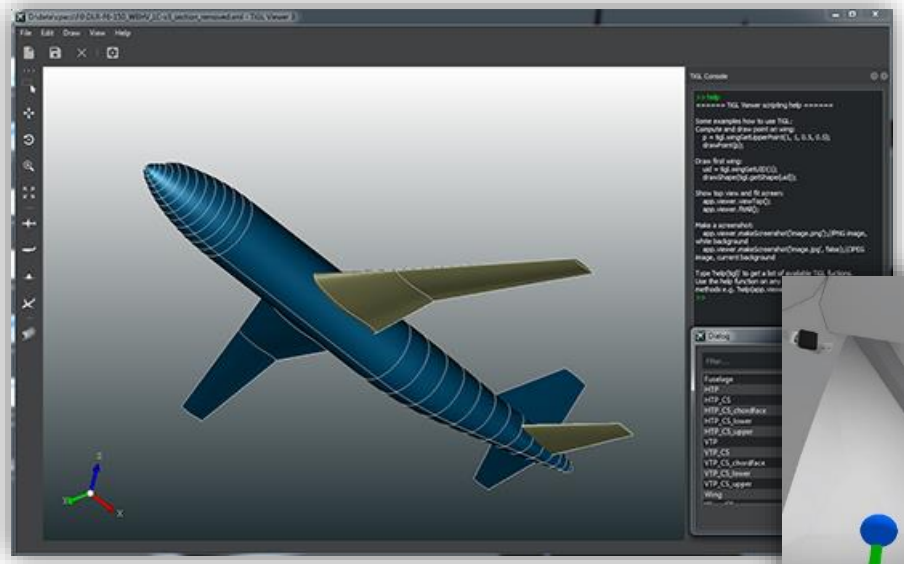
Software and Systems Engineering

Artificial Intelligence

High-Performance Computing

- Research on dependable software systems and algorithms with a focus on aeronautics, space, energy, transport and security
- Designing and transferring efficient development processes and sustainable digital solutions through the use of state-of-the-art software technologies

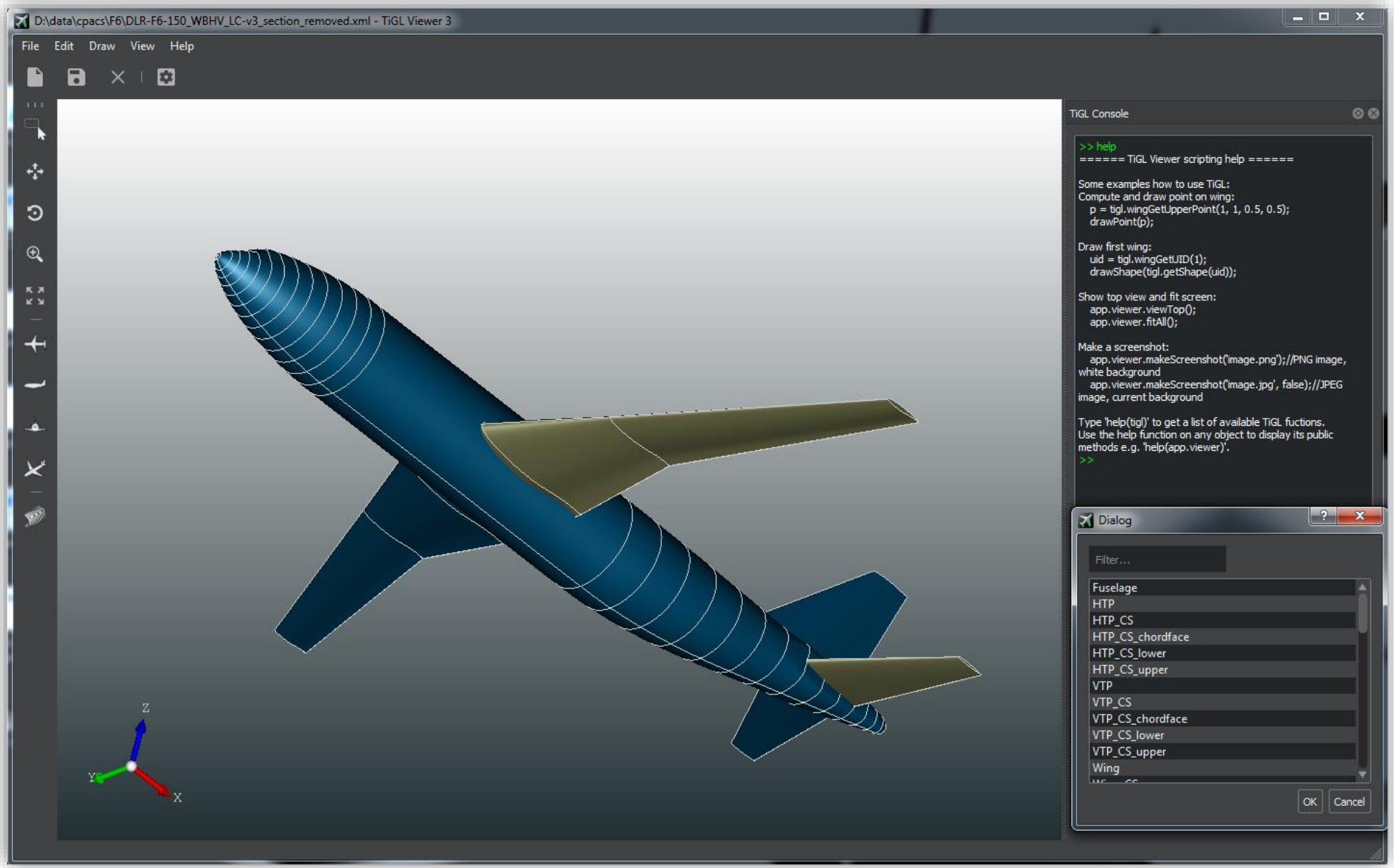
Research software is created during the research process or for a research purpose



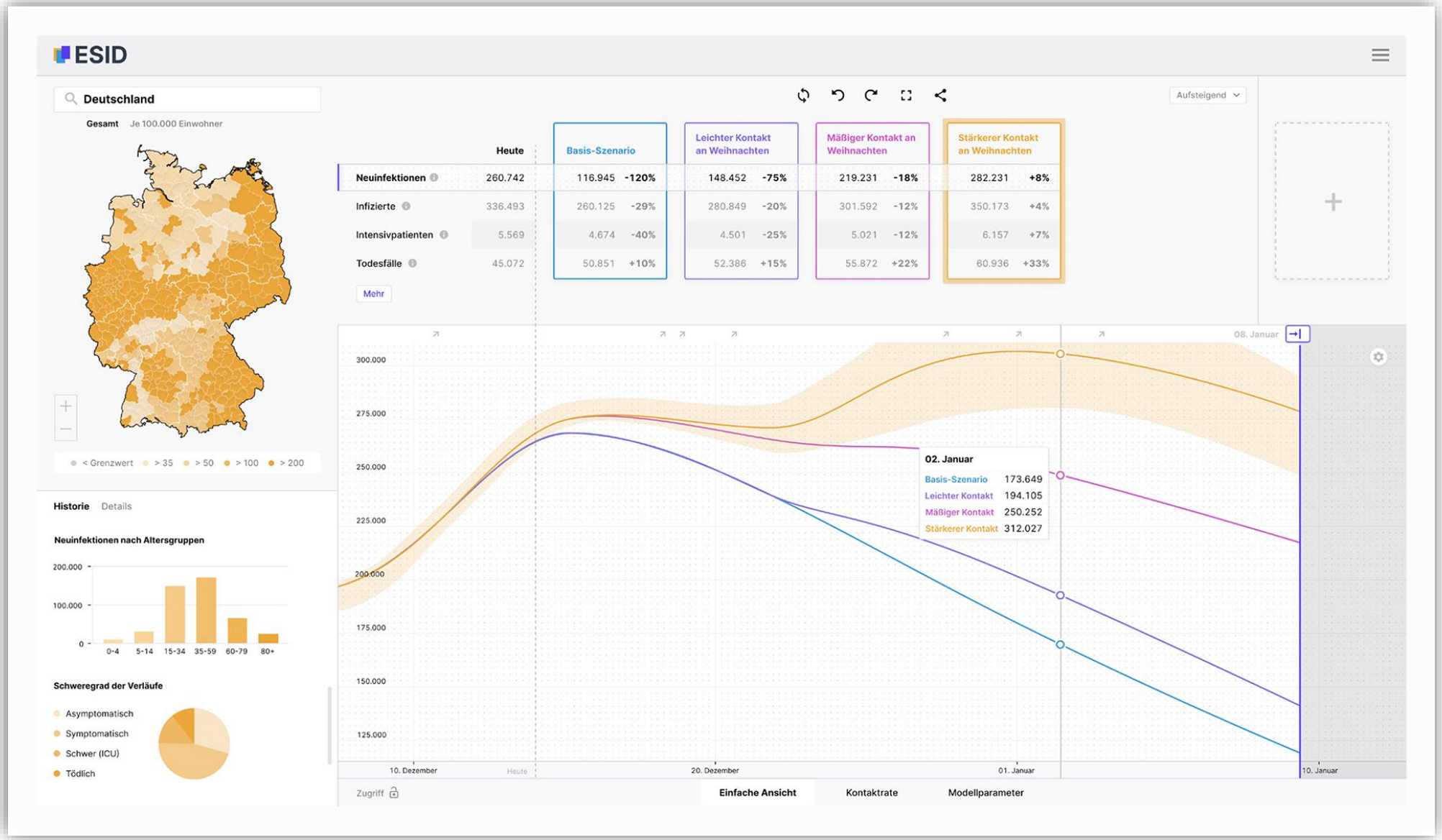
```
LaserCurrentDri   laserStack.stac   laserStackWithC   testSequence.se
device LaserCurrentDriver1_0
  parameter currentA float 0.0 to 0.999984741 default = 0.0
  parameter currentB float 0.0 to 0.999984741 default = 0.0
  parameter globalLaserEnable boolean default = false
  parameter laserADisable boolean default = true
  parameter laserBDisable boolean default = true
  parameter powerDownModeA integer 0 to 3 default=0
  parameter powerDownModeB integer 0 to 3 default=0
  parameter signalSourceA integer 0 to 15 default=13
  parameter signalSourceB integer 0 to 15 default=13
  parameter tempSens boolean default=false
  parameter readA float read only default=0.0
  parameter readB float read only default=0.0
  parameter readTemp float read only default=0.0

  process getLaserCurrent
     ut readTemp
     ut tempSens
     ut readA
     ut readB
     attern w 1 1 0 1 0 0 0 1
     attern w 1 0 0 1 0 0 0 1
     attern w 1 0 1 1 0 0 0 0
     attern w 1 0 1 1 0 0 0 1
     epeat from -1 to  -3
      pattern r 1 0 0 1 readTemp[repetition] 0 0 0
```

Central element of digitalisation in academic and industrial research

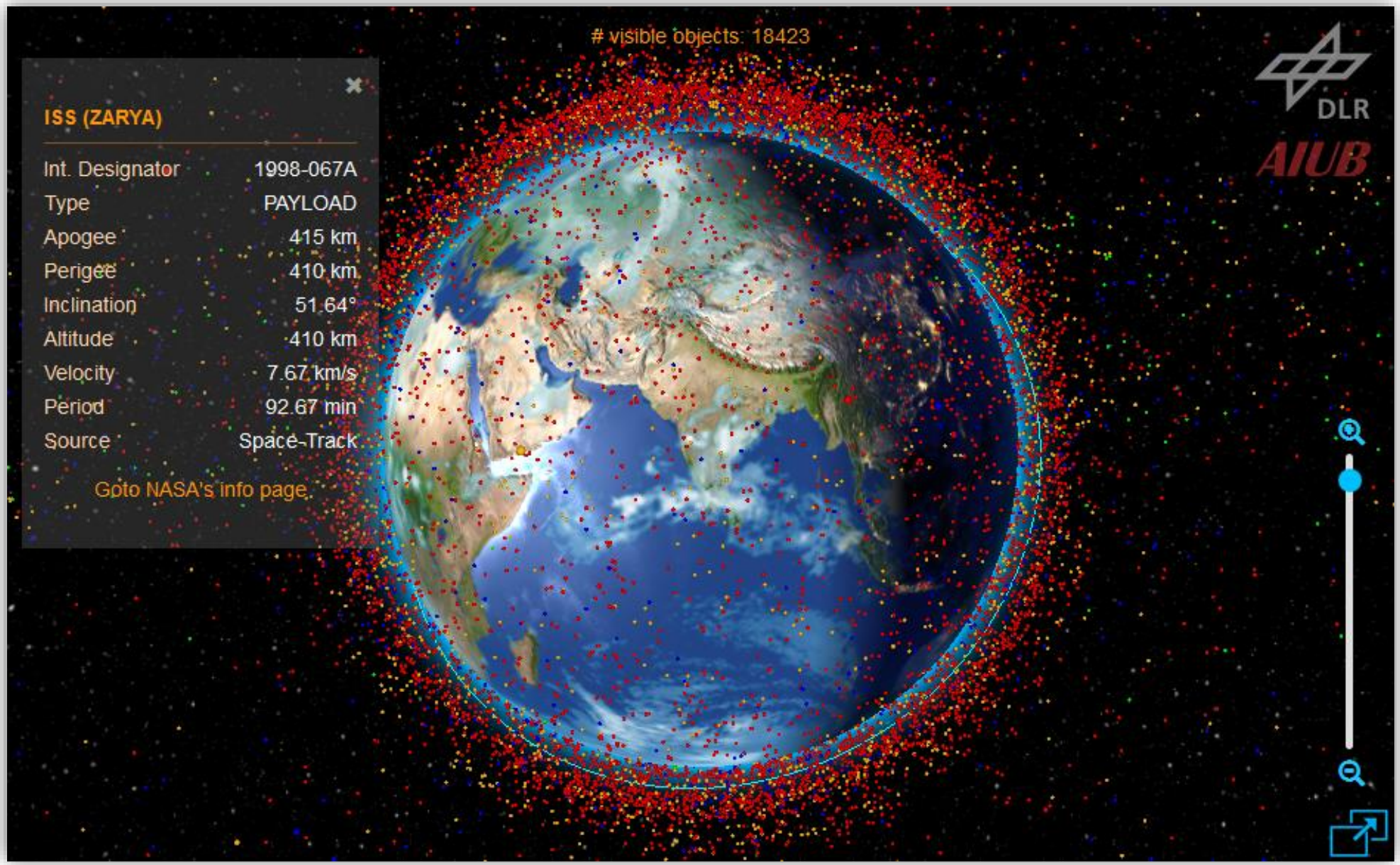
Modeling, Simulation, and Data Analytics Software



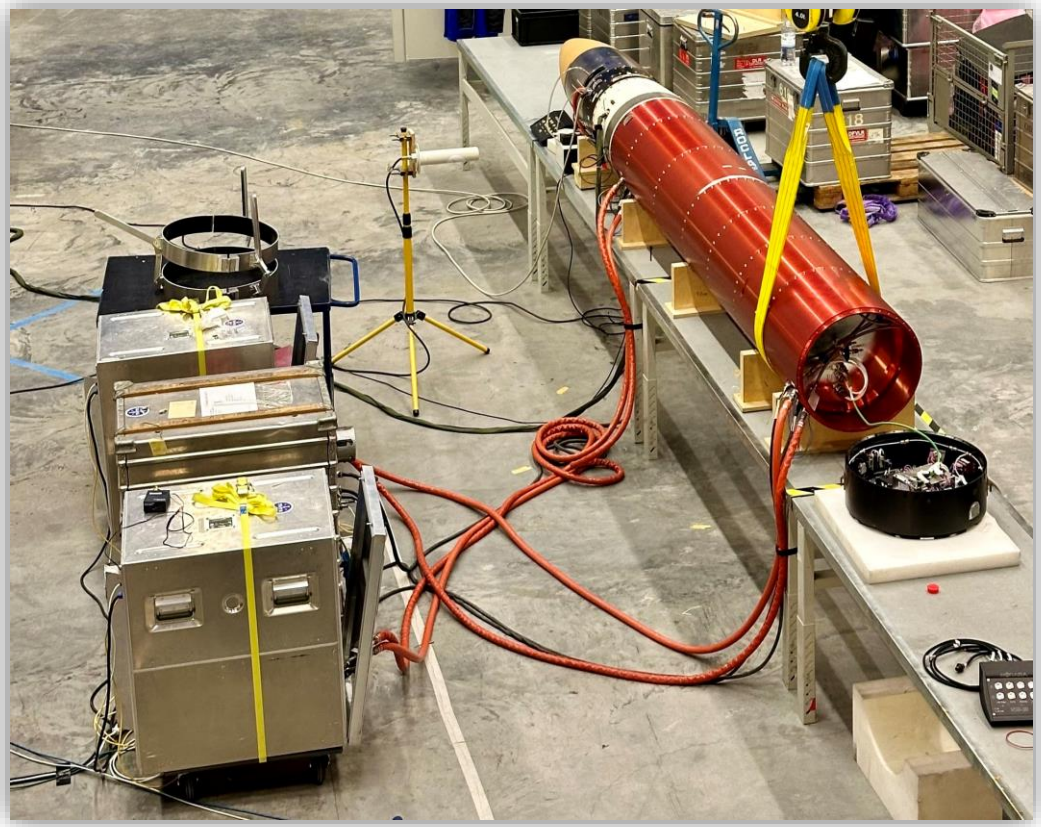
Modeling, Simulation, and Data Analytics Software



Modeling, Simulation, and Data Analytics Software



Embedded Control Software

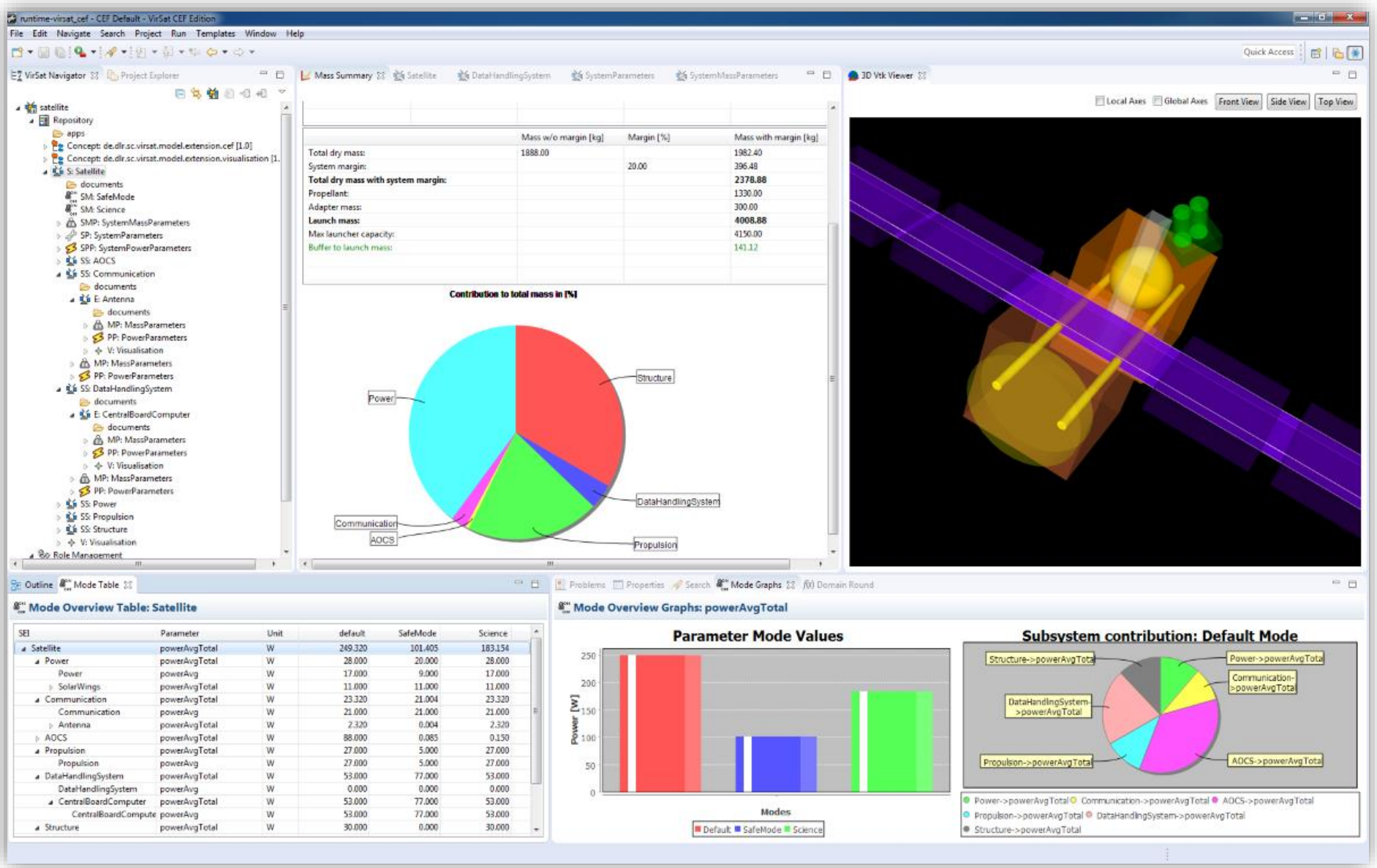


LaserCurrentDri | laserStack.stac | laserStackWithC | testSequence.se

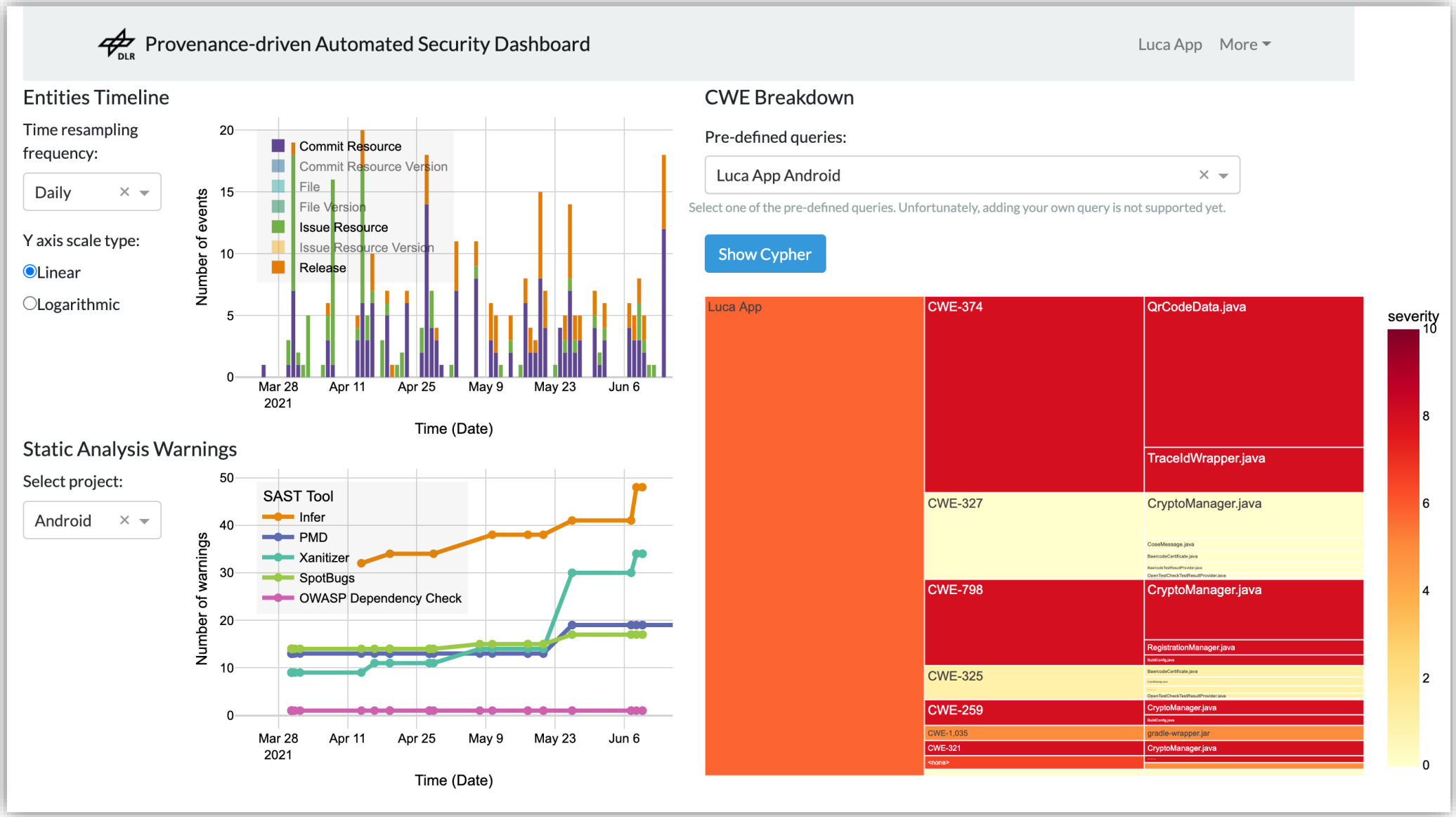
```
device LaserCurrentDriver1_0
  parameter currentA float 0.0 to 0.999984741 default = 0.0
  parameter currentB float 0.0 to 0.999984741 default = 0.0
  parameter globalLaserEnable boolean default = false
  parameter laserADisable boolean default = true
  parameter laserBDisable boolean default = true
  parameter powerDownModeA integer 0 to 3 default=0
  parameter powerDownModeB integer 0 to 3 default=0
  parameter signalSourceA integer 0 to 15 default=13
  parameter signalSourceB integer 0 to 15 default=13
  parameter tempSens boolean default=false
  parameter readA float read only default=0.0
  parameter readB float read only default=0.0
  parameter readTemp float read only default=0.0

  process getLaserCurrent
    out readTemp
    out tempSens
    out readA
    out readB
    pattern w 1 1 0 1 0 0 0 1
    pattern w 1 0 0 1 0 0 0 1
    pattern w 1 0 1 1 0 0 0 0
    pattern w 1 0 1 1 0 0 0 1
    repeat from -1 to  -3
      pattern r 1 0 0 1 readTemp[repetition] 0 0 0
```

Software Prototypes in Engineering Research

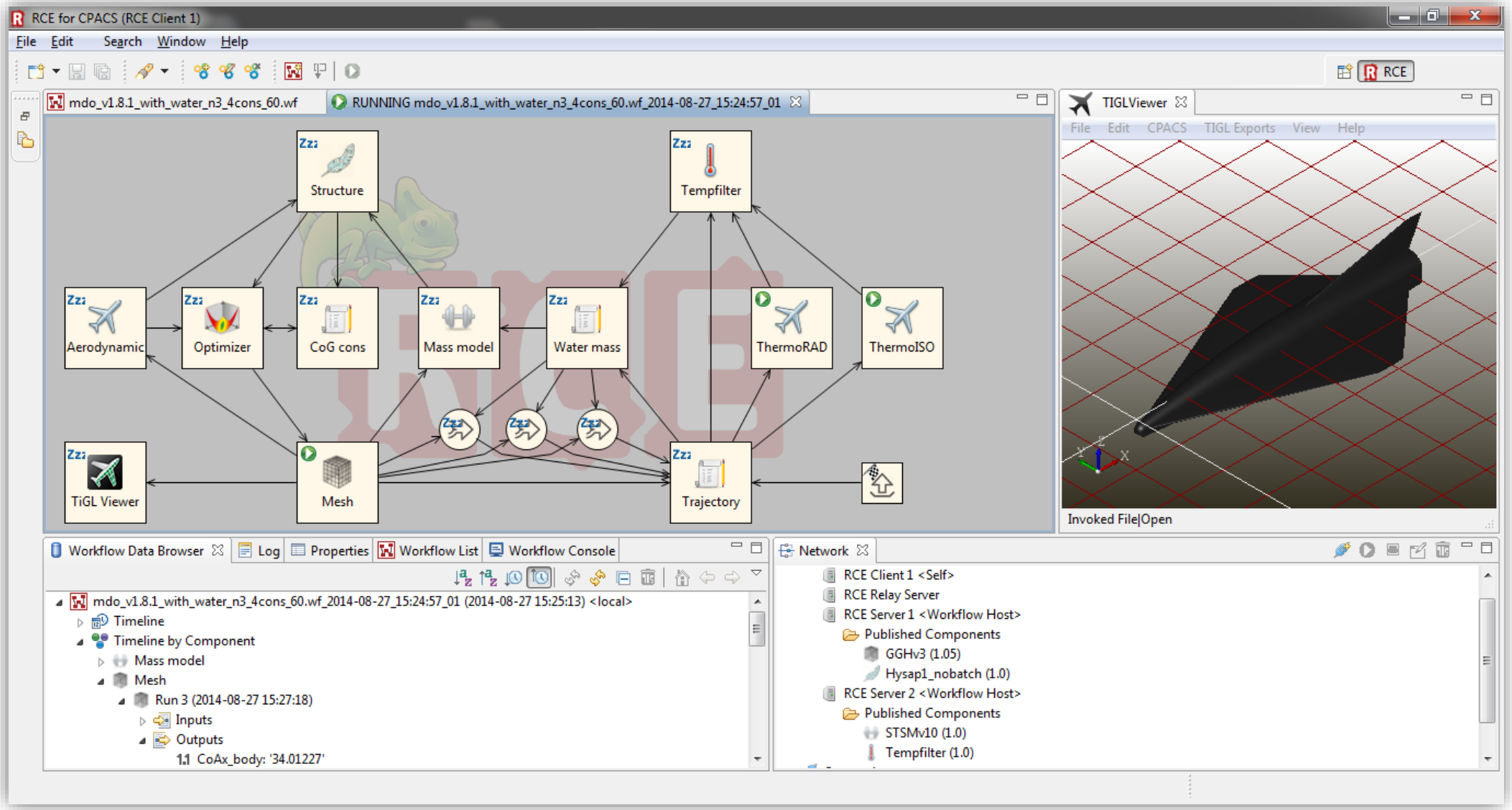
Software Prototypes in Engineering Research



Software Prototypes in Engineering Research



Infrastructure and Platform Software



Empirical Investigation on Research Software

Software Development at the German Aerospace Center: Role and Status in Practice

Lynn von Kurnatowski
German Aerospace Center (DLR)
Oberpfaffenhofen, Germany
lynn.kurnatowski@dlr.de

Tobias Schlauch
German Aerospace Center (DLR)
Braunschweig, Germany
tobias.schlauch@dlr.de

Carina Haupt
German Aerospace Center (DLR)
Berlin, Germany
carina.haupt@dlr.de

The diversity of research focuses is also reflected in the programming languages that are used. The most frequently used programming language is Python with about 23%, followed by C++ with about 14%, MATLAB with about 12% and C with about 11%.

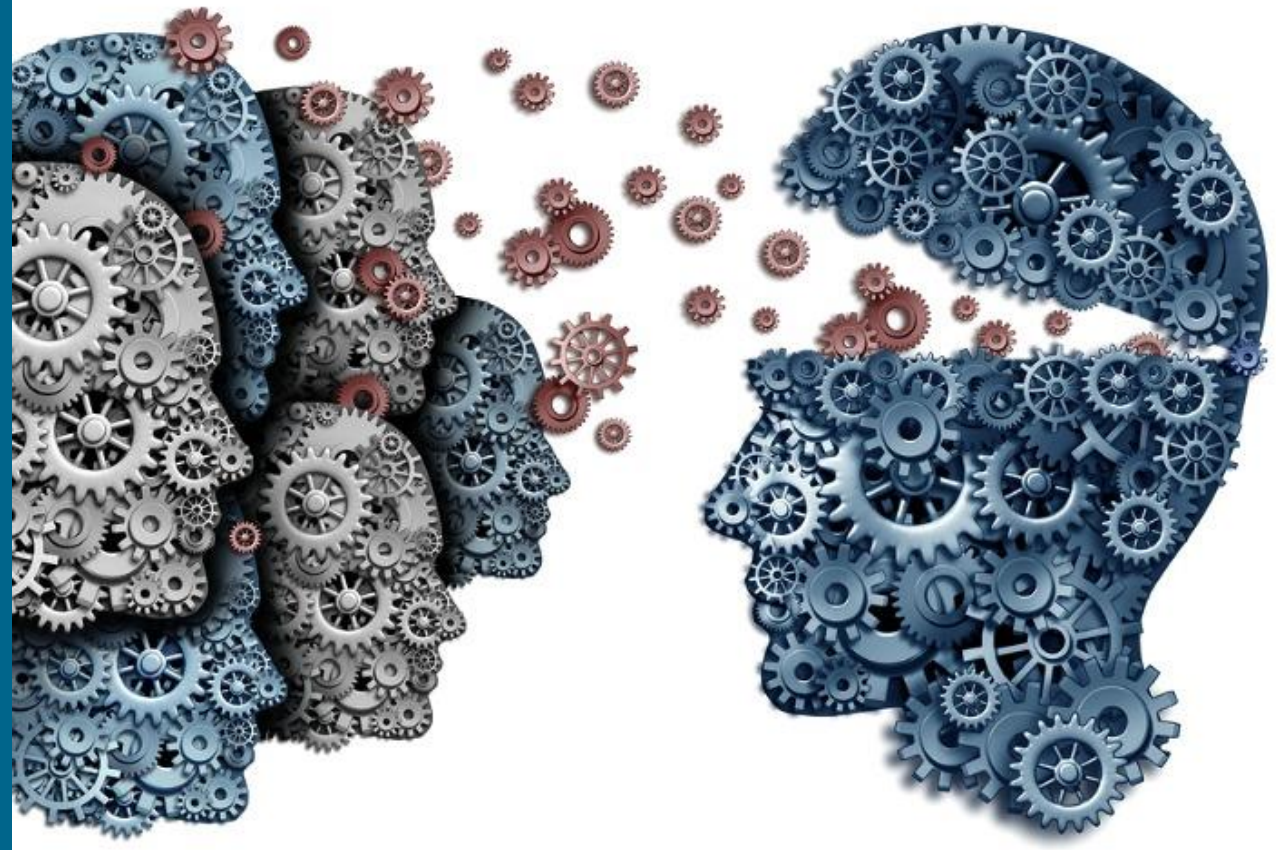
Year: 2018
N: 773

[1] Kurnatowski, L., Schlauch, T., Haupt, C. (2020) Software Development at the German Aerospace Center: Role and Status in Practice. ICSE (Workshops) 2020.

Research Software can also be successful outside research …

Research Software - Goal

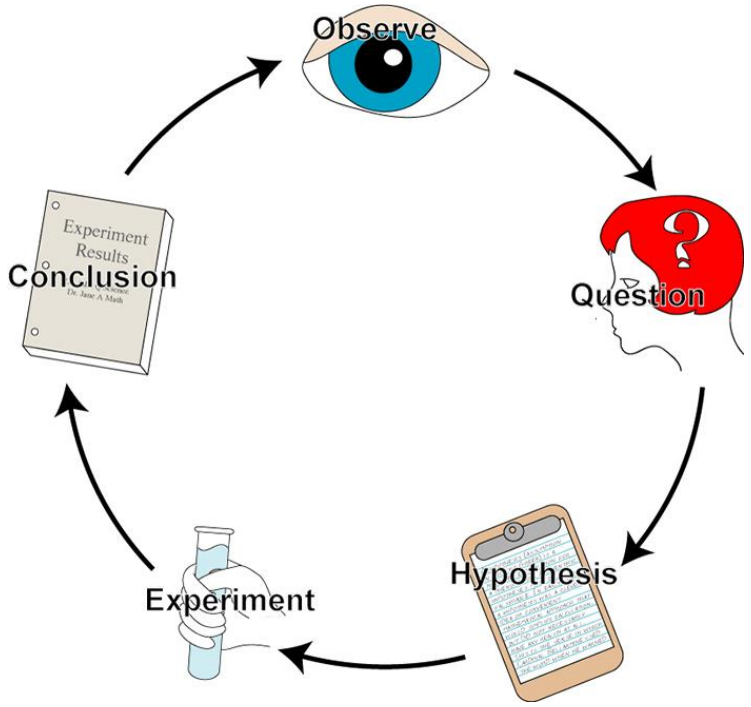
Enables or supports scientific investigation with the aim to create and validate knowledge



The goal of business software is helping to perform business functions with the aim to support running a successful business

Research Software – Functional Requirements

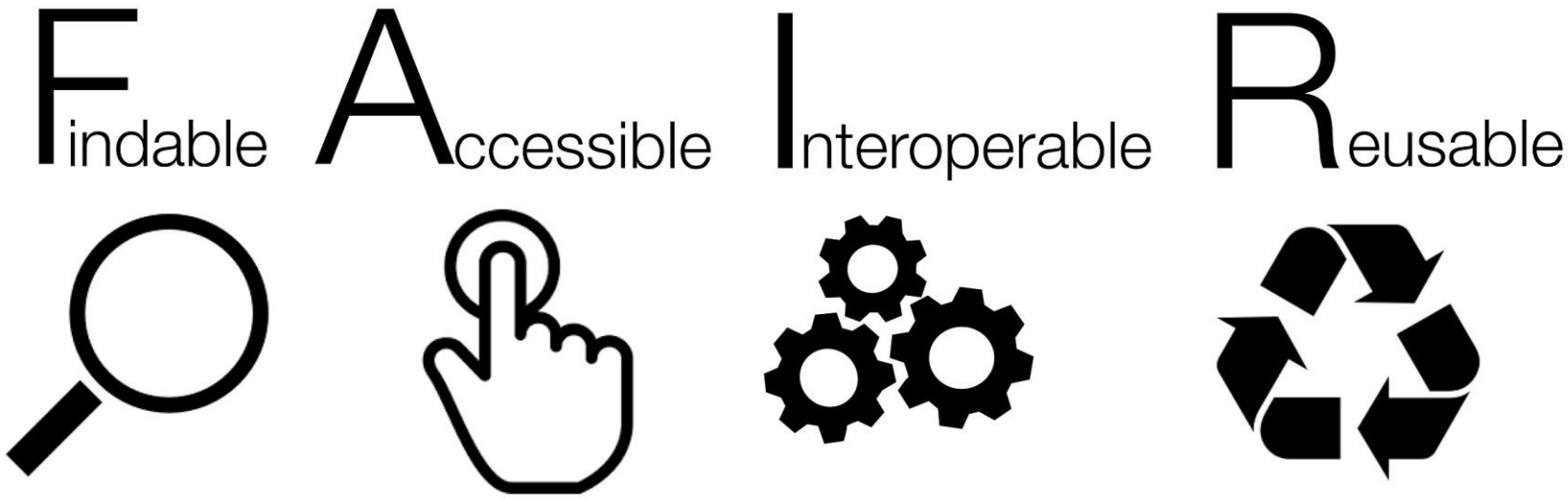
Functional requirements are often unknown up front



Verification and Validation are difficult

Research Software – Non-Functional Requirements

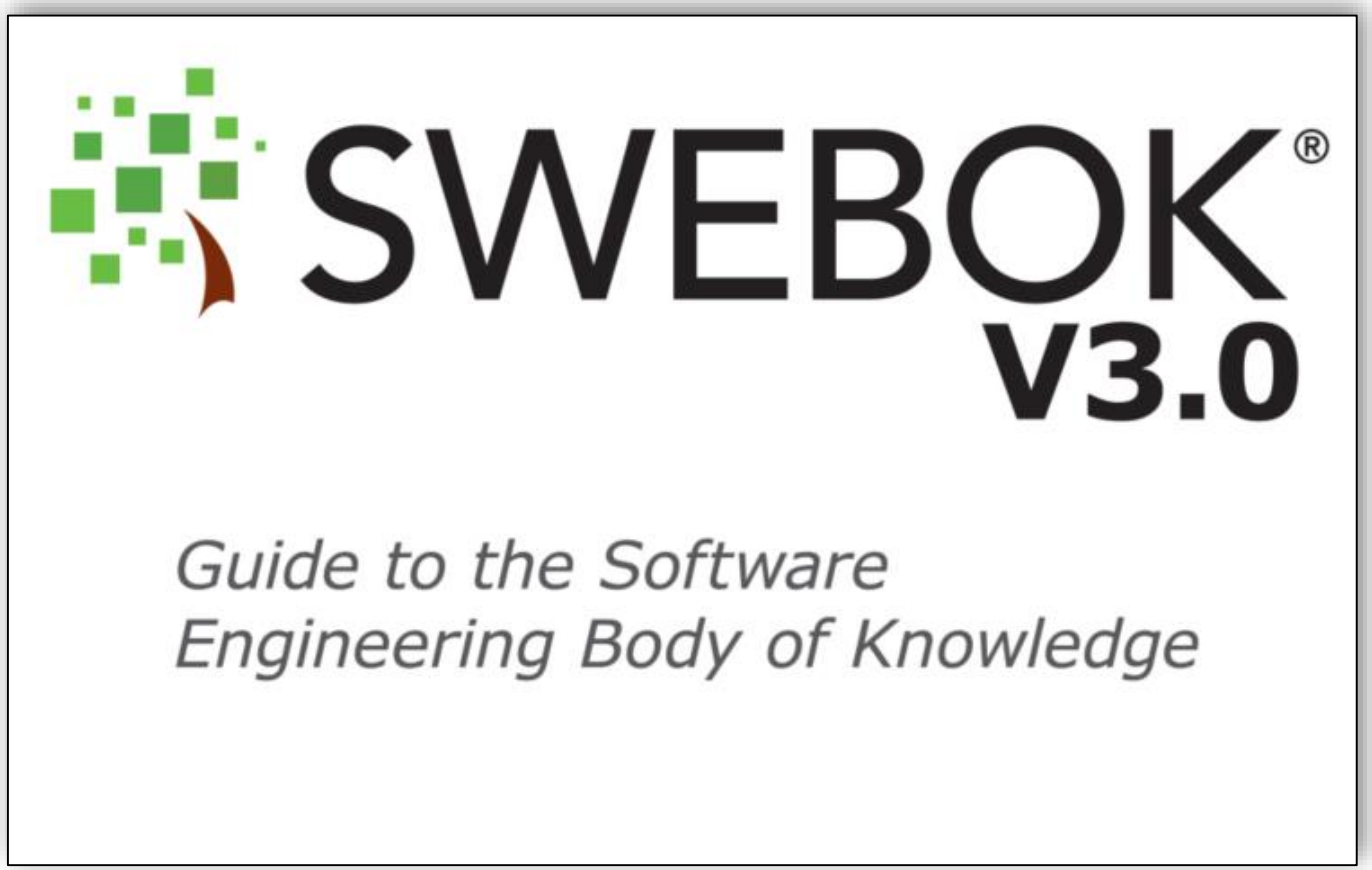
Research reproducibility is key



Productivity & Credibility Crisis

Research software is a critical artifact that requires software engineering

Well, just apply software engineering techniques …



NATO Software Engineering Conference (1968)

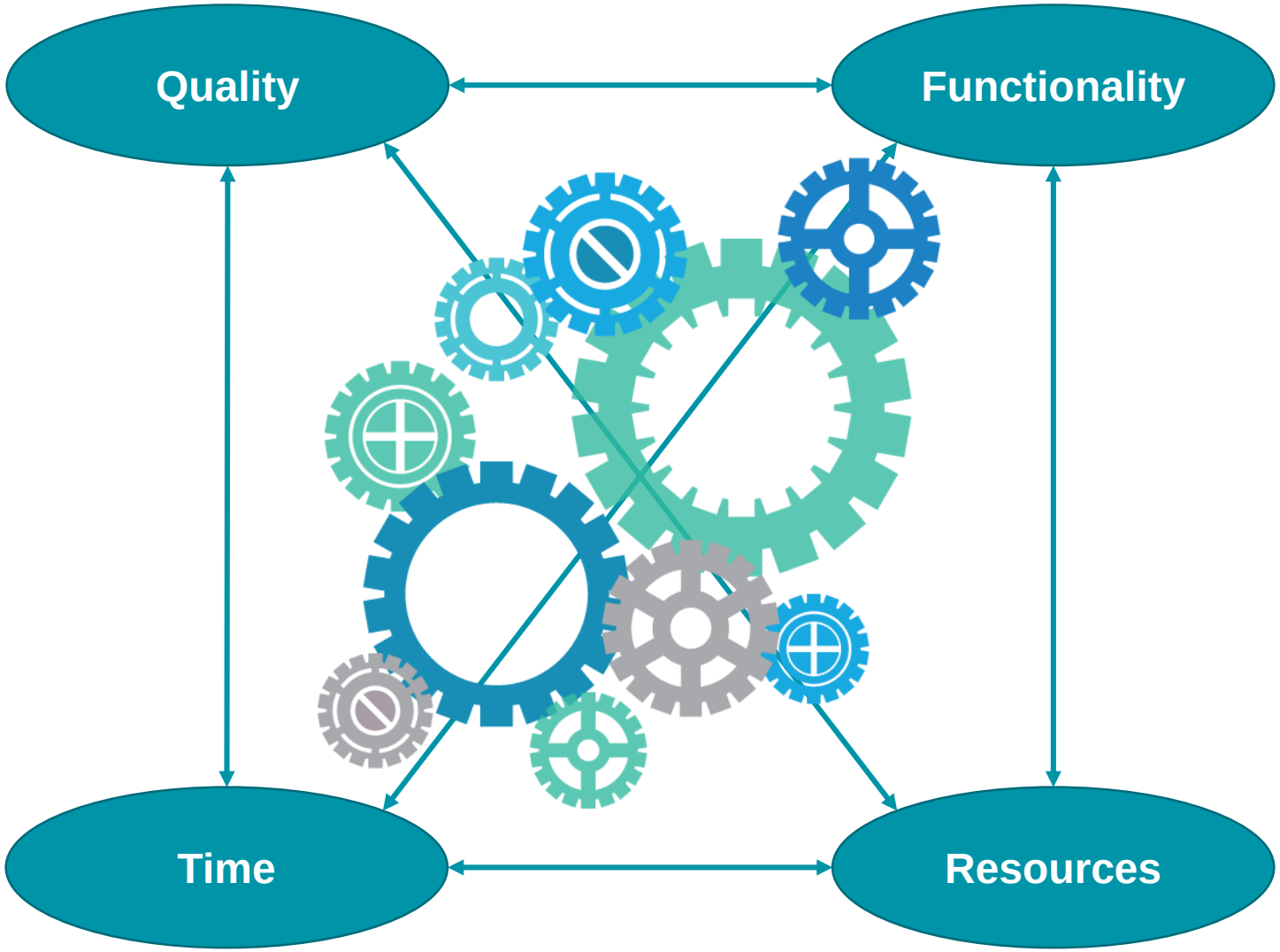
Software Engineering

Mission-critical business and embedded software

Computational Science

Scientific data processing applications

Trade Off in Software Engineering



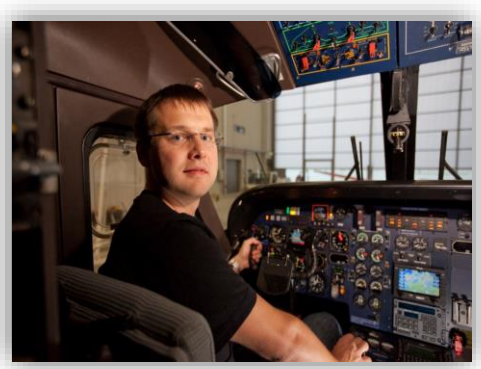
Research software engineering is the use of software engineering practices in research applications

Is it different?

Difference: Software Developers

Formally trained software engineers are seldom the developers of research software (also in industry)



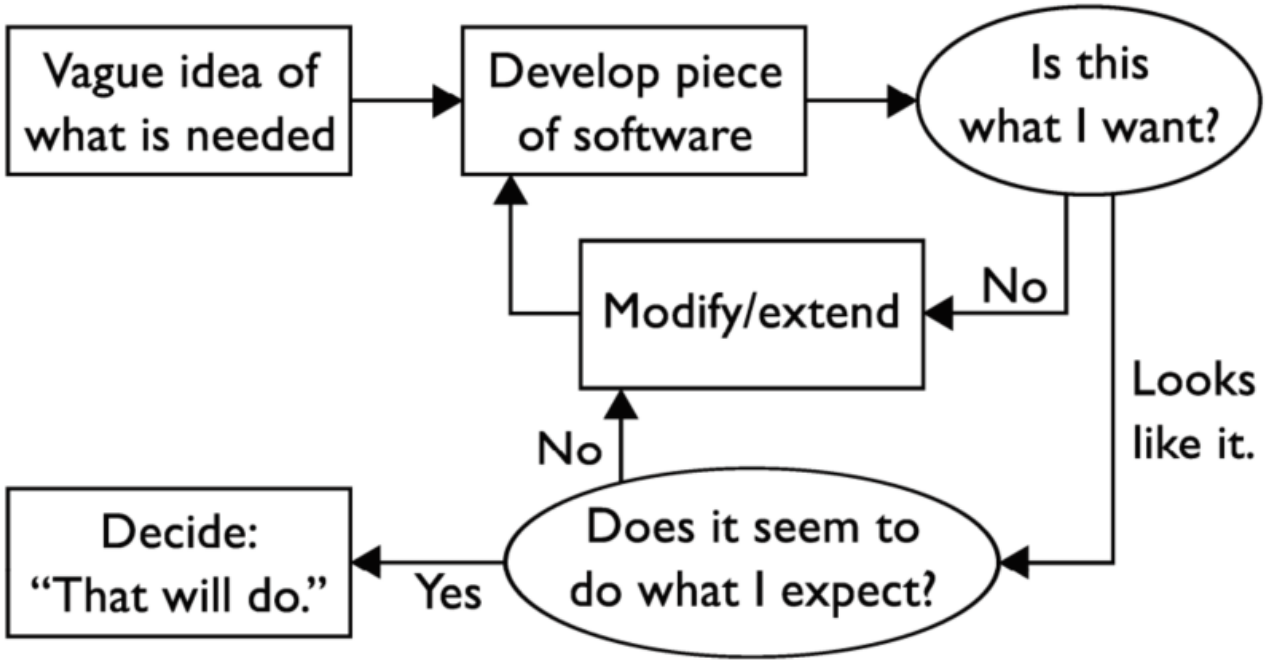



Difference: Time Scale

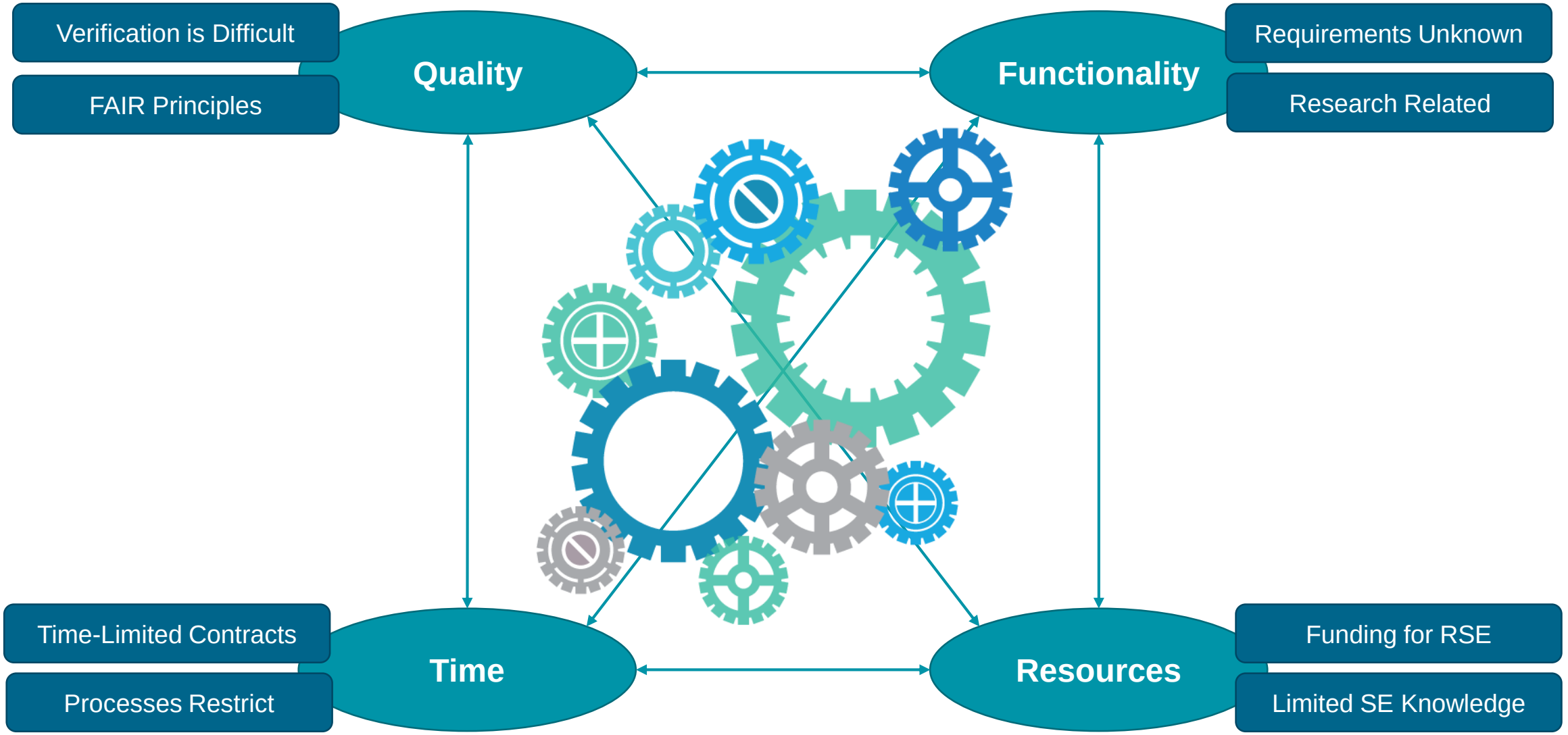
Long-lived software primarily developed by individuals with time limited contracts

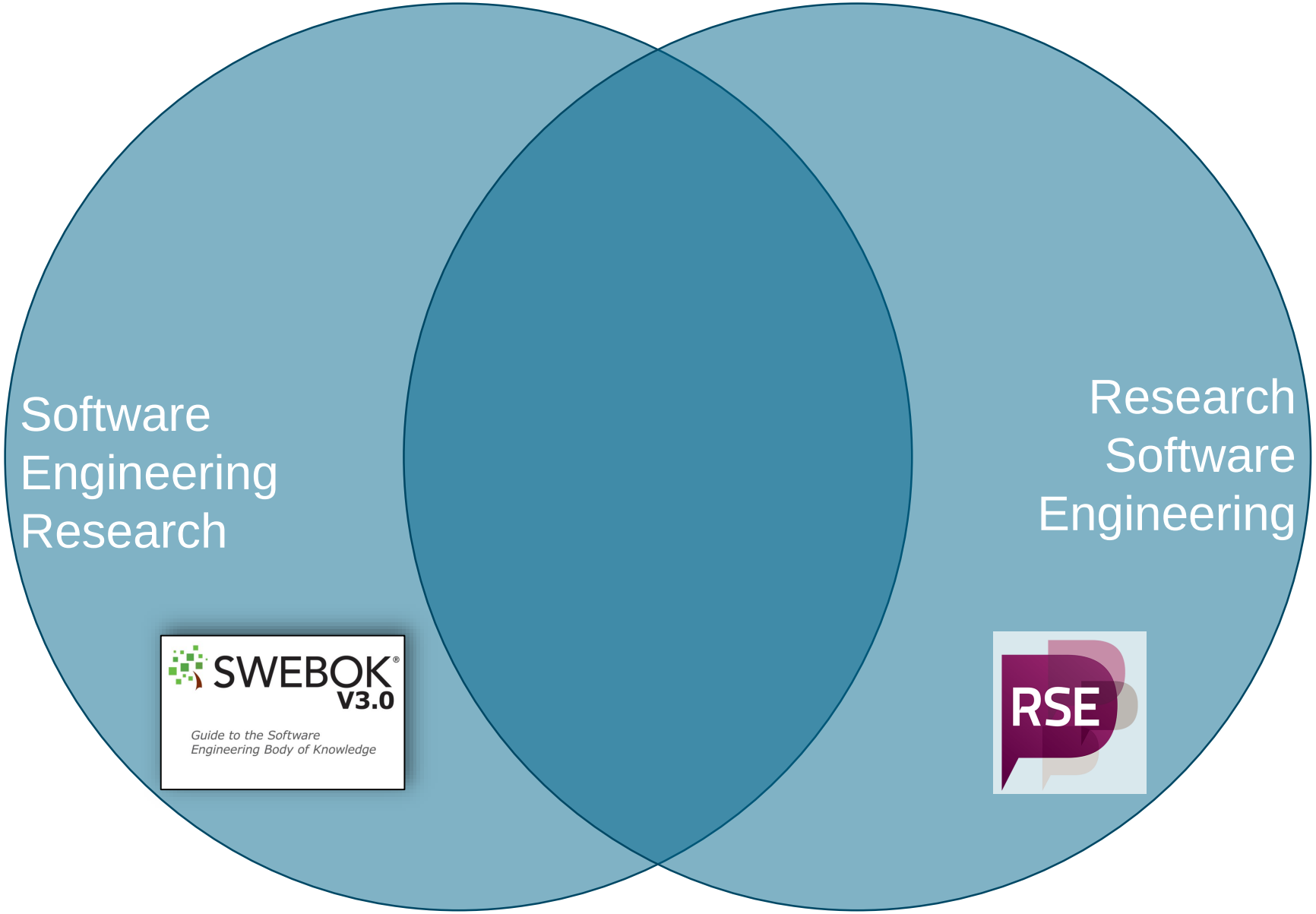
Difference: Process

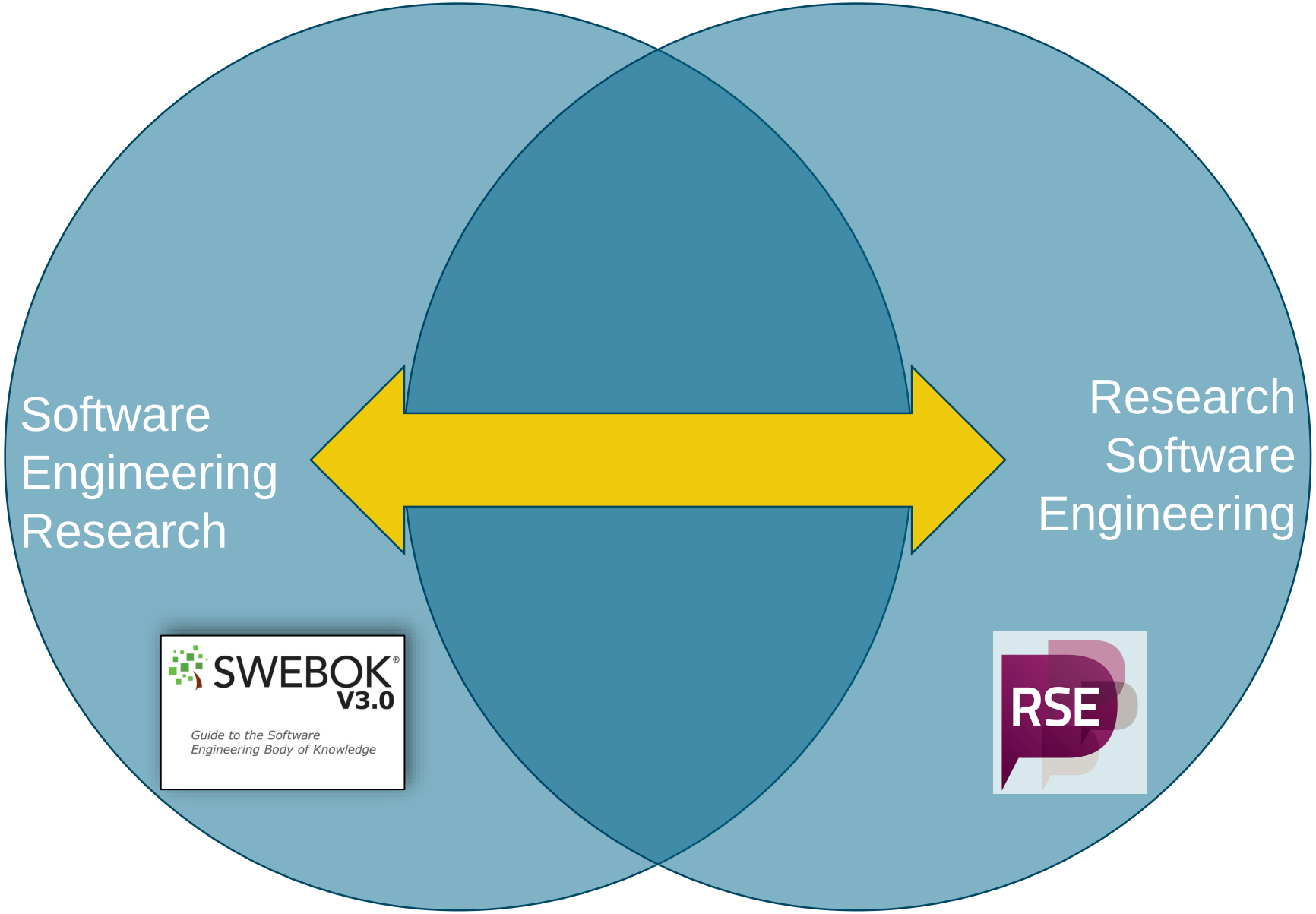
Overly structured software processes restrict research

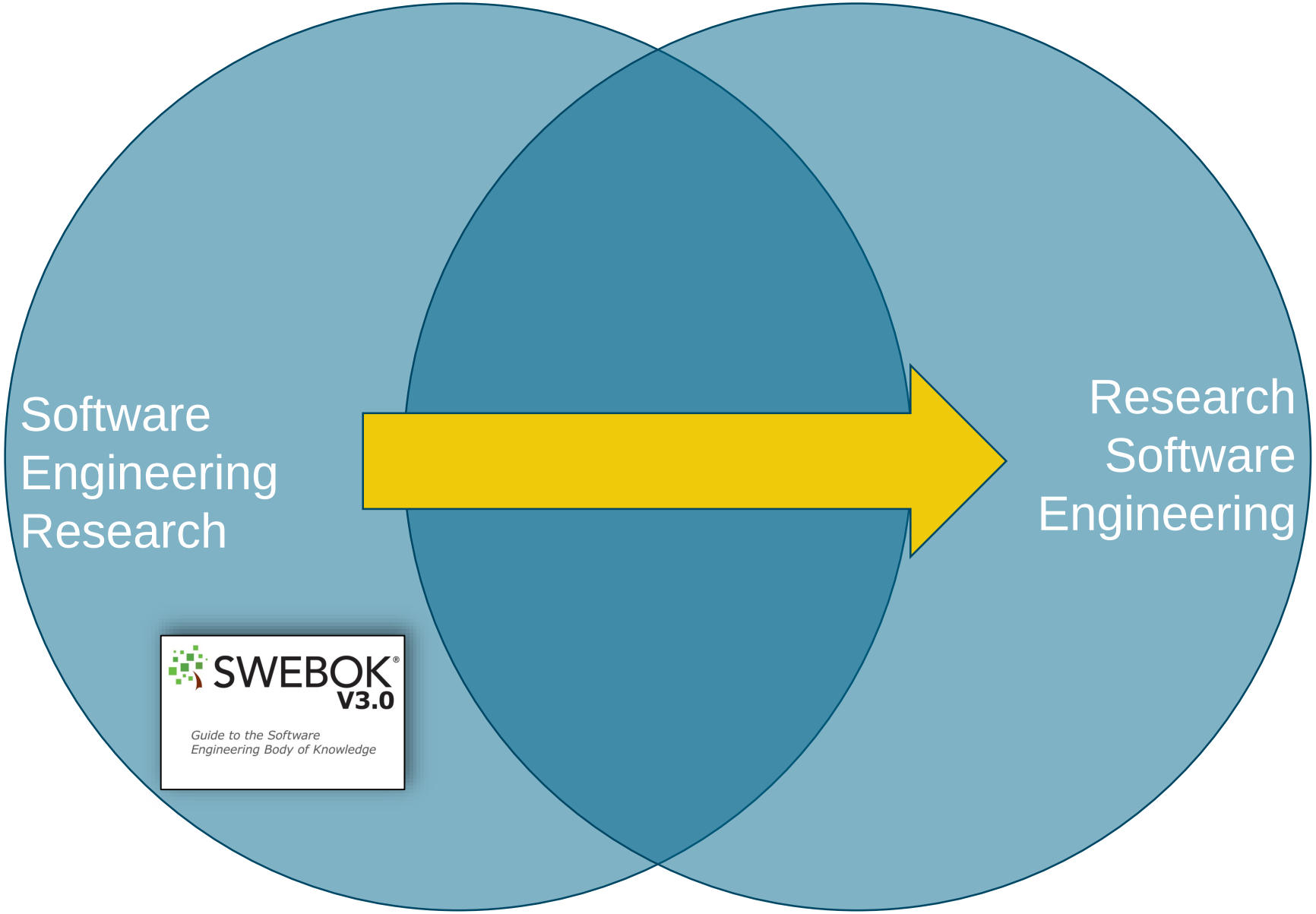


Trade Off in Software Engineering: Specifics of RSE

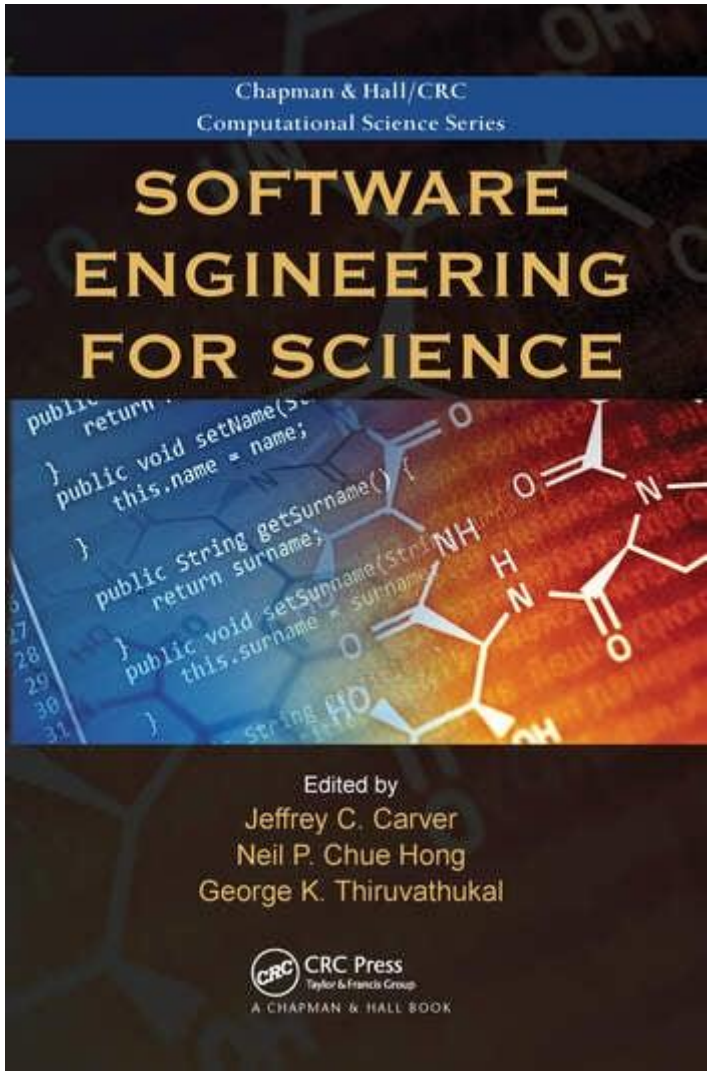



Software
Engineering
Research
Research
Software
Engineering
SWEBOK
V3.0
Guide to the Software
Engineering Body of Knowledge
RSE


Software
Engineering
Research
Research
Software
Engineering
SWEBOK
V3.0
Guide to the Software
Engineering Body of Knowledge
RSE


Software
Engineering
Research
Research
Software
Engineering
SWEBOK
V3.0
Guide to the Software
Engineering Body of Knowledge

SE Practices for Research Software Engineering



Automated Metamorphic Testing of Scientific Software (Kanewala et al.)

Evaluating Hierachical Domain-Specific Languages for Computational Science (Johanson et al.)

[2] Carver, J. C., Hong, N. P. C., Thiruvathukal, G. K. (2016) Software engineering for science. CRC Press.

Metamorphic Testing

```
public static double findGyrationRadius(double[] atomsXCoords,
                                        double[] atomsYCoords, double[] atomsZCoords){
    double fSum=0;
    for(int i=0;i<atomsXCoords.length;i++){
        for(int j=0;j<atomsXCoords.length;j++){
            double dx=atomsXCoords[j]-atomsXCoords[i];
            double dy=atomsYCoords[j]-atomsYCoords[i];
            double dz=atomsZCoords[j]-atomsZCoords[i];
            fSum+=(dx*dx)+(dy*dy)+(dz*dz);
        }
    }
    double fRadius=1.0/(2*atomsXCoords.length*atomsXCoords.length)*fSum;
    return Math.sqrt(fRadius);
}
```

Figure 7.1: Function from the SAXS project described in Section 7.5.1 used for calculating the radius of gyration of a molecule.

$$R_g^2 \stackrel{\text{def}}{=} \frac{1}{2N^2} \sum_{i \neq j} |\mathbf{r}_i - \mathbf{r}_j|^2$$

Describes the distribution of atoms of a molecule around its axis

Metamorphic Testing

```
@Test
public void findGyrationRadiusRandTest() {
 Random rand=new Random();
 int arrLen=rand.nextInt(MAXSIZE)+1;

 //initial test cases
   double[] iX=new double[arrLen];
   double[] iY=new double[arrLen];
   double[] iZ=new double[arrLen];

   for(int k=0;k<arrLen;k++){
       iX[k]=rand.nextDouble();
       iX[k]=rand.nextDouble();
       iX[k]=rand.nextDouble();
       }

   //Executing the initial test case on the function under test
   double intialOutput=SAXSFunctions.findGyrationRadius(iX, iY, iZ);

   //create follow-up test cases by randomly permuting the array
     elements
   double[] fX=permuteElements(iX);
   double[] fY=permuteElements(iY);
   double[] fZ=permuteElements(iZ);

   //Executing the follow-up test case on the function under test
   double followUpOutput=SAXSFunctions.findGyrationRadius(fX, fY, fZ);

   assertEquals(intialOutput,followUpOutput,eps);}
```

Figure 7.2: JUnit test case that uses the permutative MR to test the function in Figure 7.1.

Domain-Specific Languages in Computational Sciences

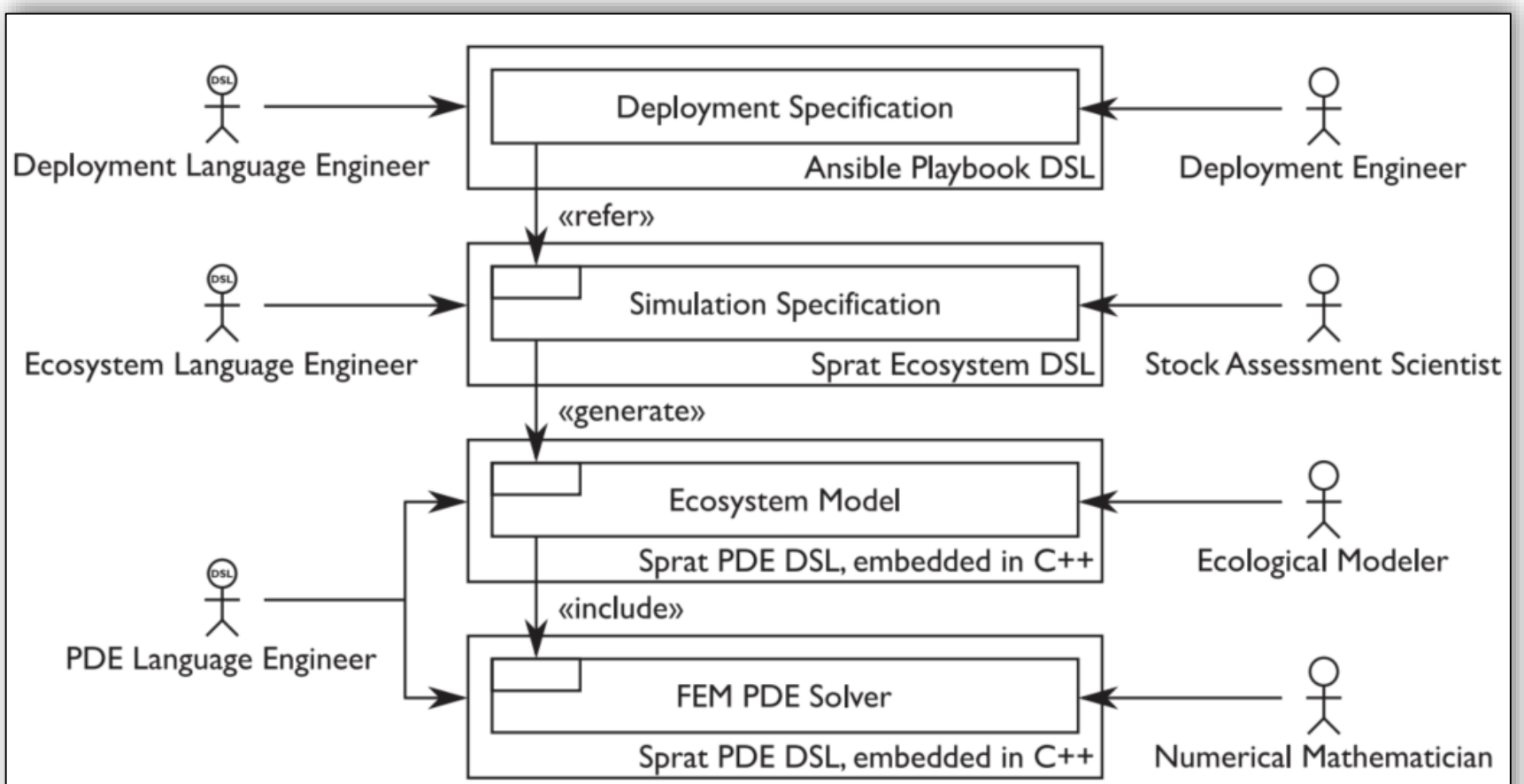


Figure 8.4: DSL hierarchy for the Sprat Marine Ecosystem Model.

Domain-Specific Languages in Computational Sciences

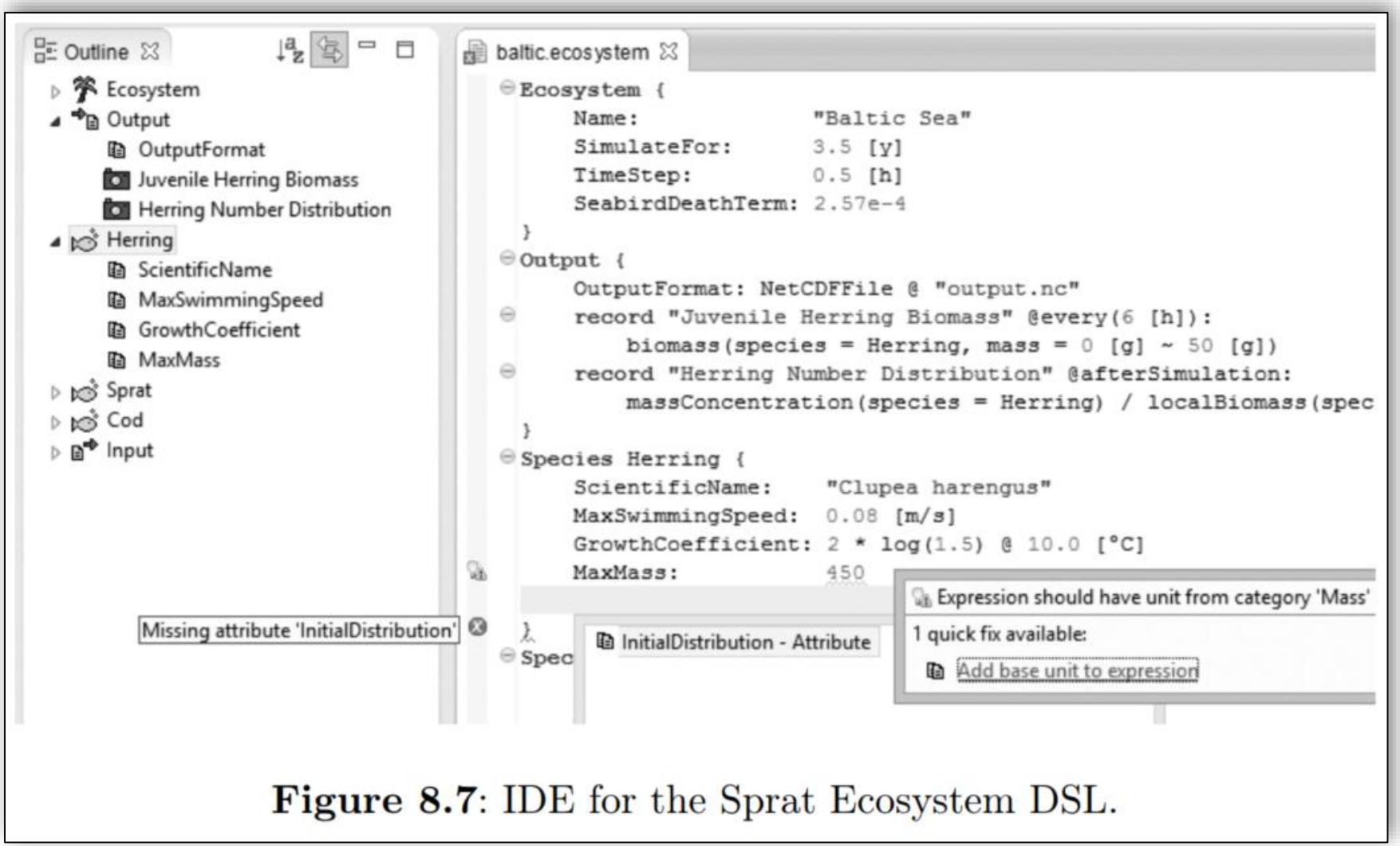
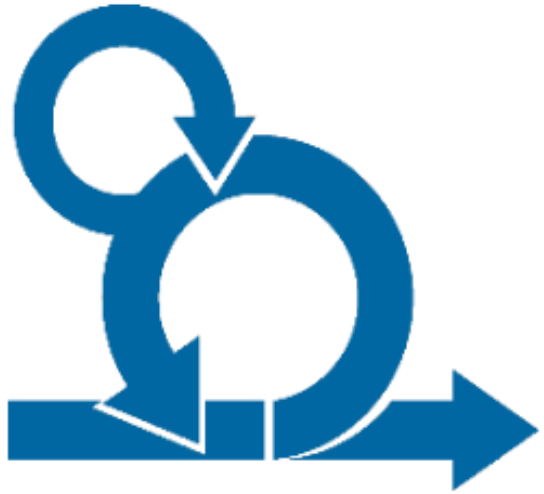


Figure 8.7: IDE for the Sprat Ecosystem DSL.

Many other relevant topics …

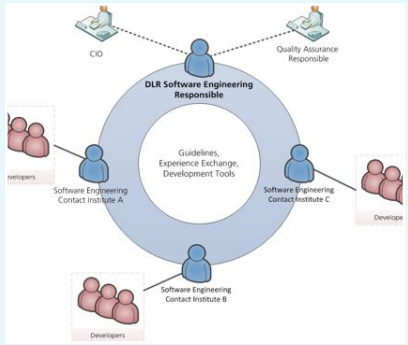
Agile software development

Continuous software development



[3] Schönborn, M. T. (2023) Adopting Software Engineering Concepts in Scientific Research: Insights from Physicists and Mathematicians Turned Consultants. Computing in Science & Engineering.

Software Engineering Initiative at DLR

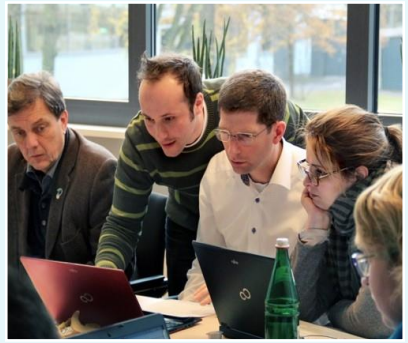


Community Building & Experience Exchange

Change Management

Recommendation	Comment
EÄM.2: The most important information describing how to contribute to development are stored in a central location. *(from application class 1)*	Build steps ar missing
EÄM.5: Known bugs, important unresolved tasks and ideas are at least noted in bullet point form and stored centrally. *(from application class 1)*	
EÄM.7: A repository is set up in a version control system. The repository is adequately structured and ideally contains all artifacts for building a usable software version and for testing it. *(from application class 1)*	
EÄM.8: Every change of the repository ideally serves a specific purpose, contains an understandable description and leaves the software in a consistent, working state. *(from application class 1)*	

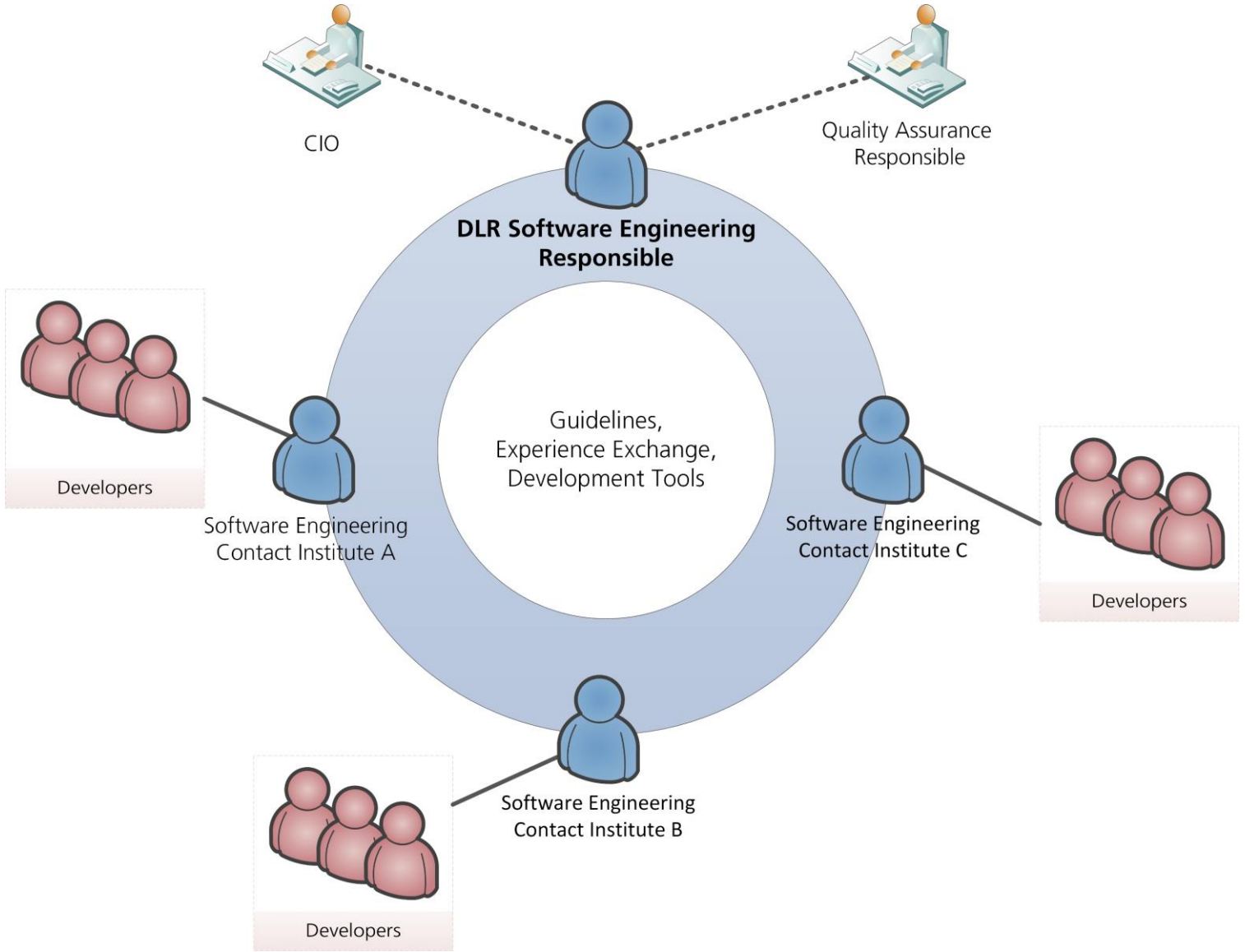
Guidelines & Tools



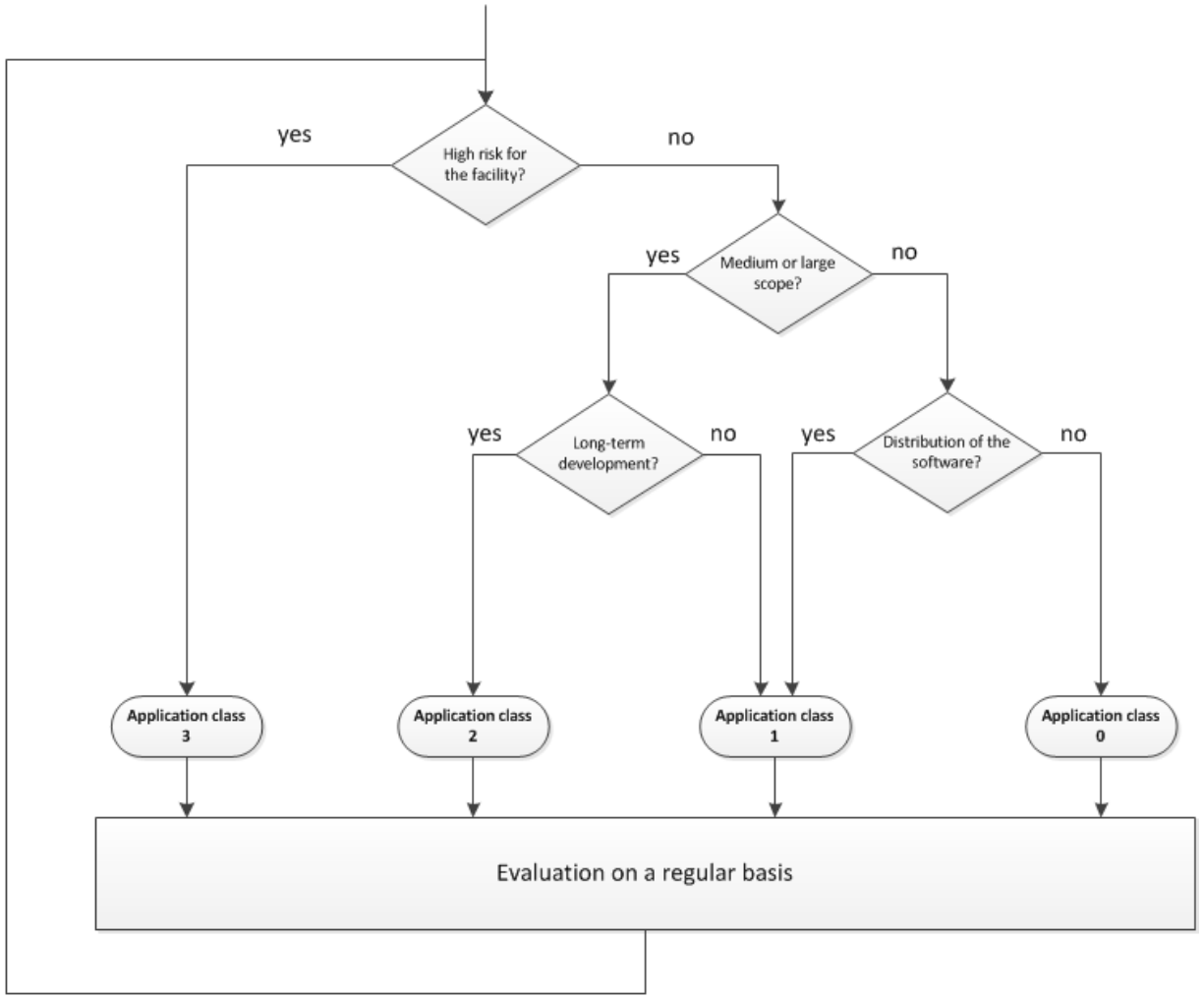
Trainings

Consulting

DLR Software Engineering Network

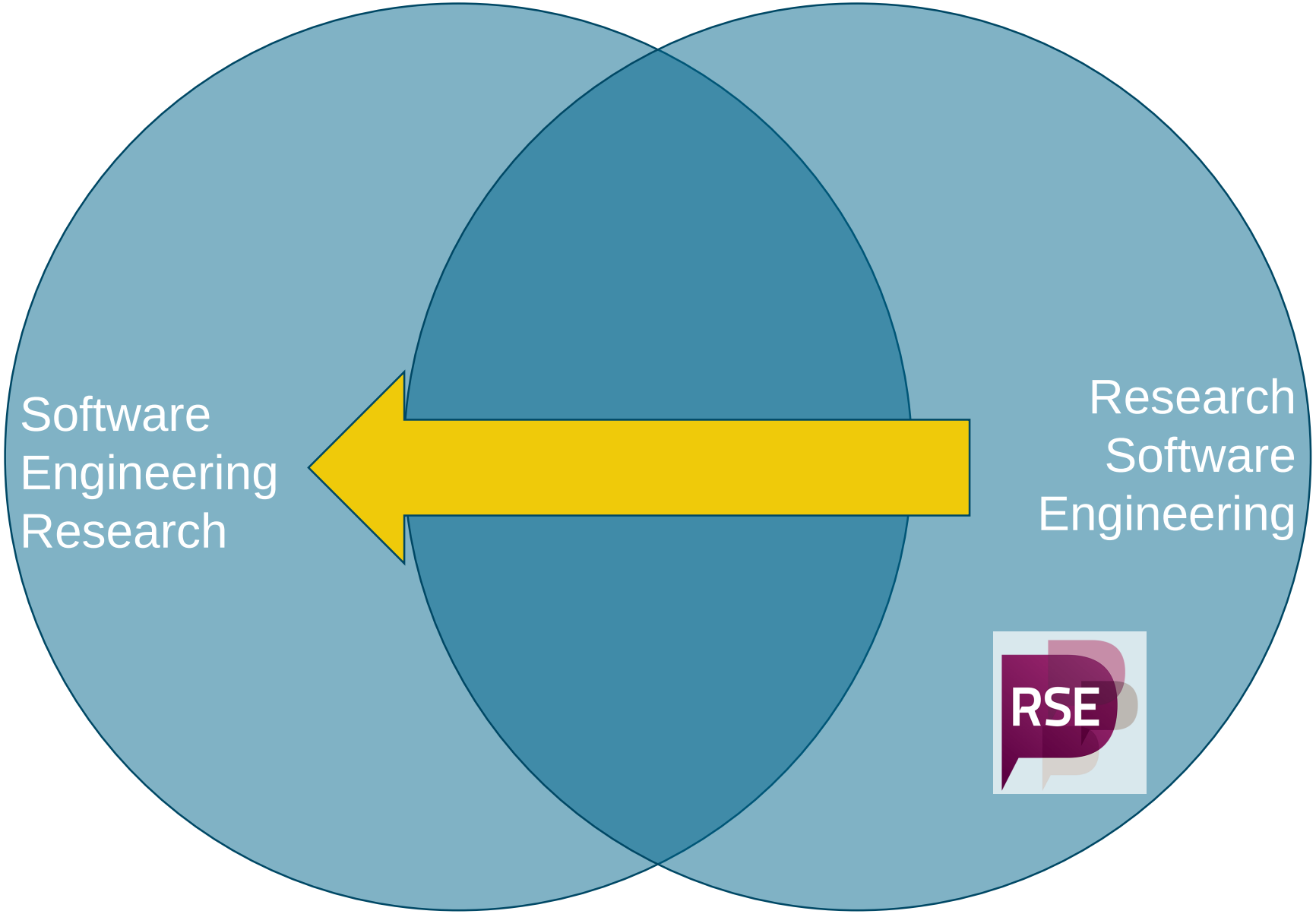


DLR Software Engineering Guidelines



Recommendation	from AC	Explanation
EAM.1: The problem definition is coordinated with all parties involved and documented. It describes the objectives, the purpose of the software, the essential requirements and the desired application class in a concise, understandable way.	1	It is important that the problem definition is early coordinated between the parties involved to prevent misunderstandings and incorrect developments. The problem definition also provides important hints for later use and further development.
EAM.2: Functional requirements are documented at least including a unique identifier, a description, the priority, the origin and the contact person.	2	Requirements must be clearly identifiable to refer to them during development and to trace them back to software changes (see the Change Management section). In addition, prioritisation helps to determine the order of implementation. Finally, information about the contact person and the origin is essential in case of questions.
EAM.3: The constraints are documented.	1	The relevant constraints (e.g., mandatory programming languages and frameworks, the opera-

https://zenodo.org/records/1344612

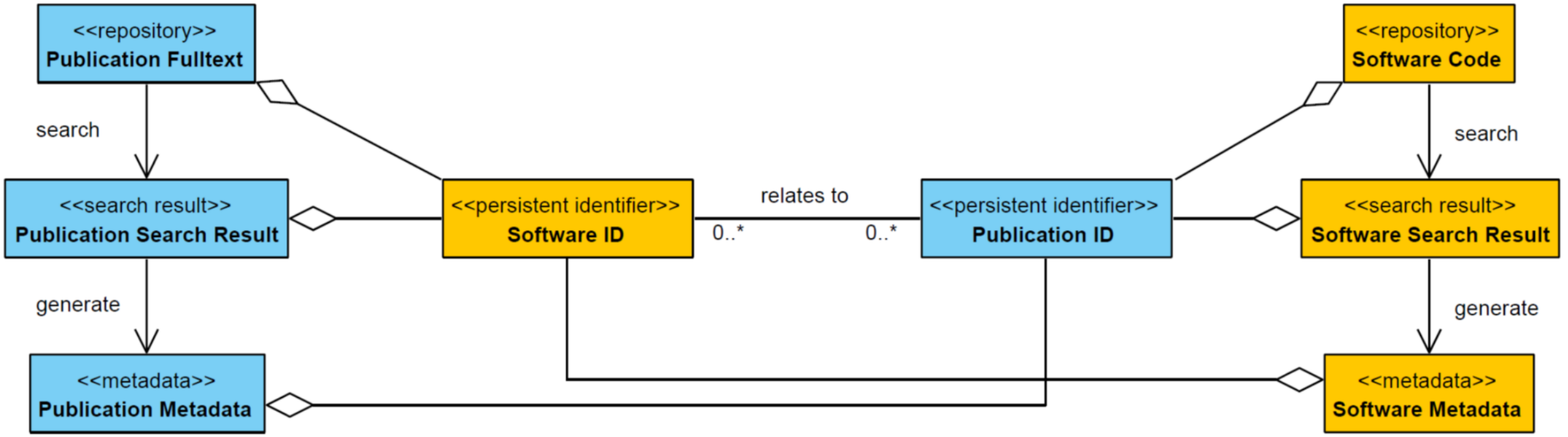

Software
Engineering
Research
Research
Software
Engineering
RSE

Open Science in Software Engineering

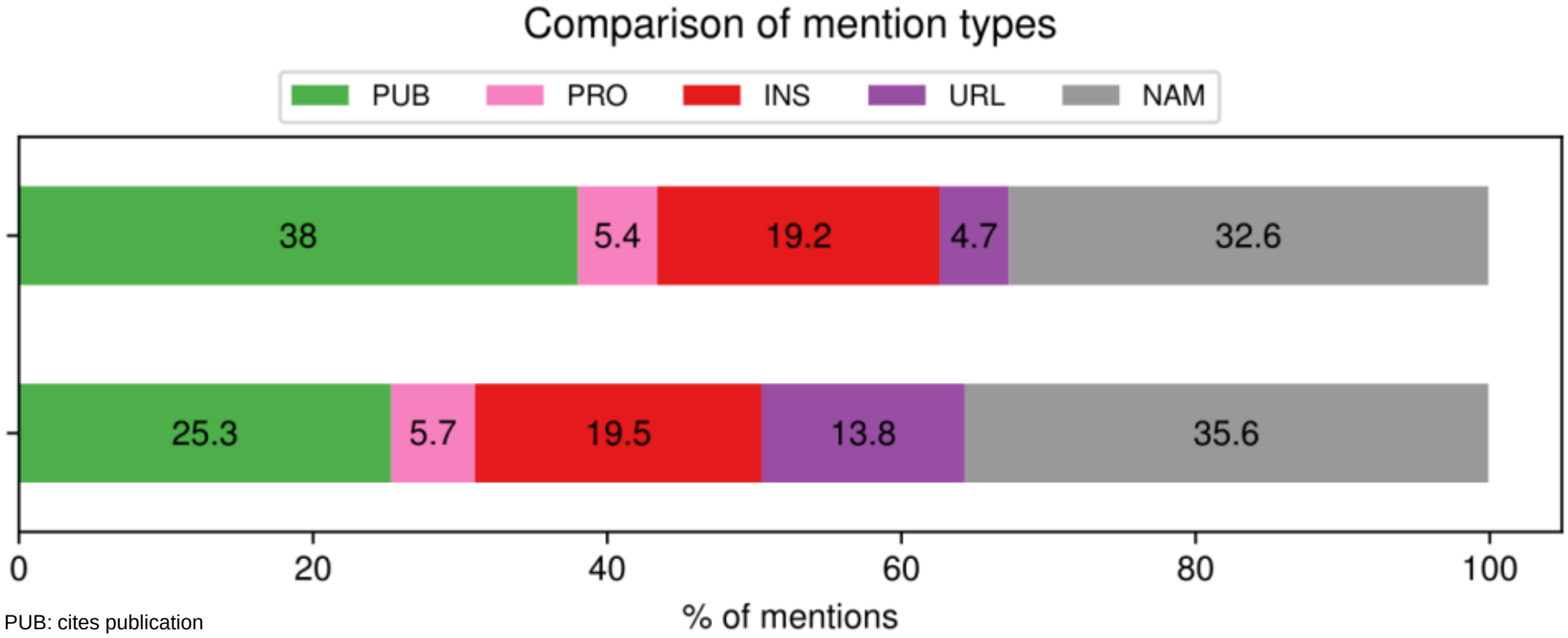
Daniel Mendez, Daniel Graziotin, Stefan Wagner, and Heidi Seibold

Abstract Open science describes the movement of making any research artifact available to the public and includes, but is not limited to, open access, open data, and open source. While open science is becoming generally accepted as a norm in other scientific disciplines, in software engineering, we are still struggling in adapting open science to the particularities of our discipline, rendering progress in our scientific community cumbersome. In this chapter, we reflect upon the essentials in open science for software engineering including what open science is, why we should engage in it, and how we should do it. We particularly draw from our

Research Software Publishing



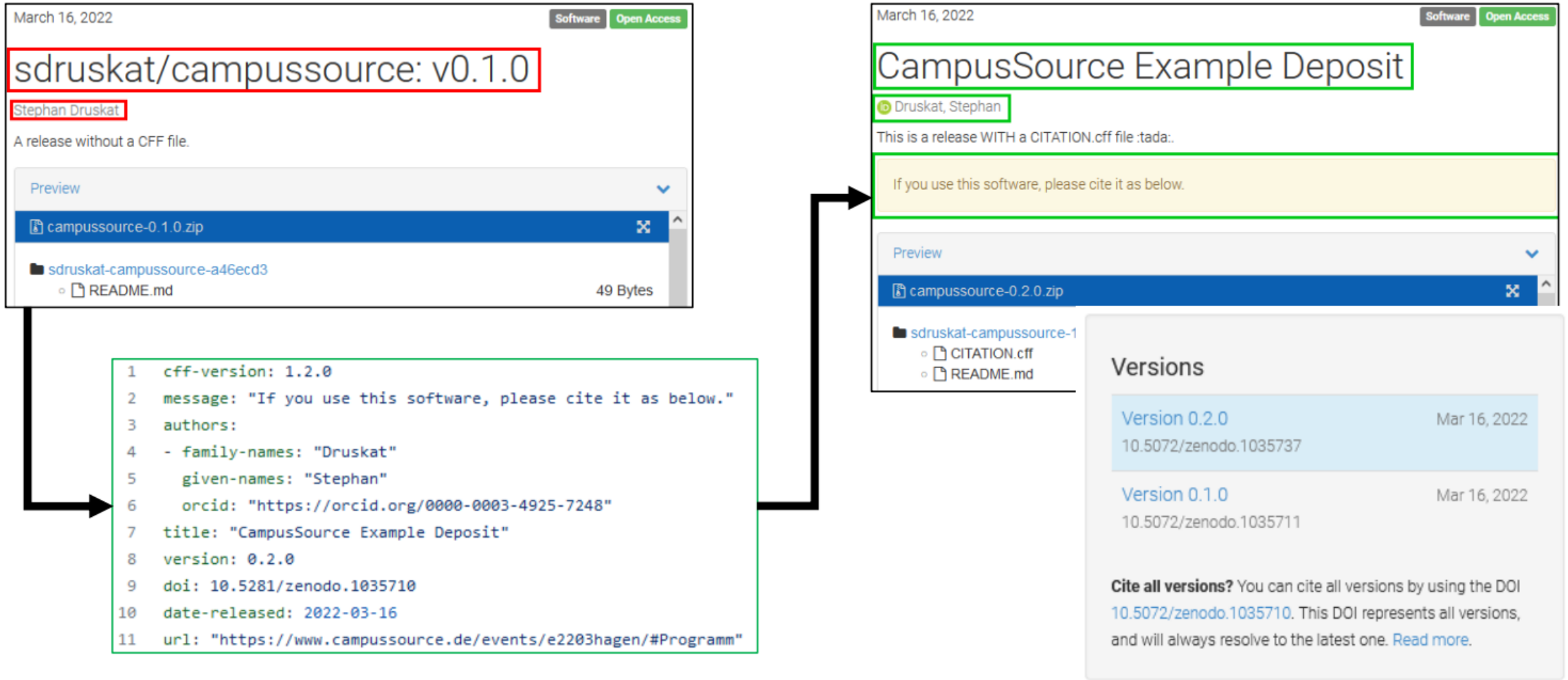
Software is currently mentioned, not cited …



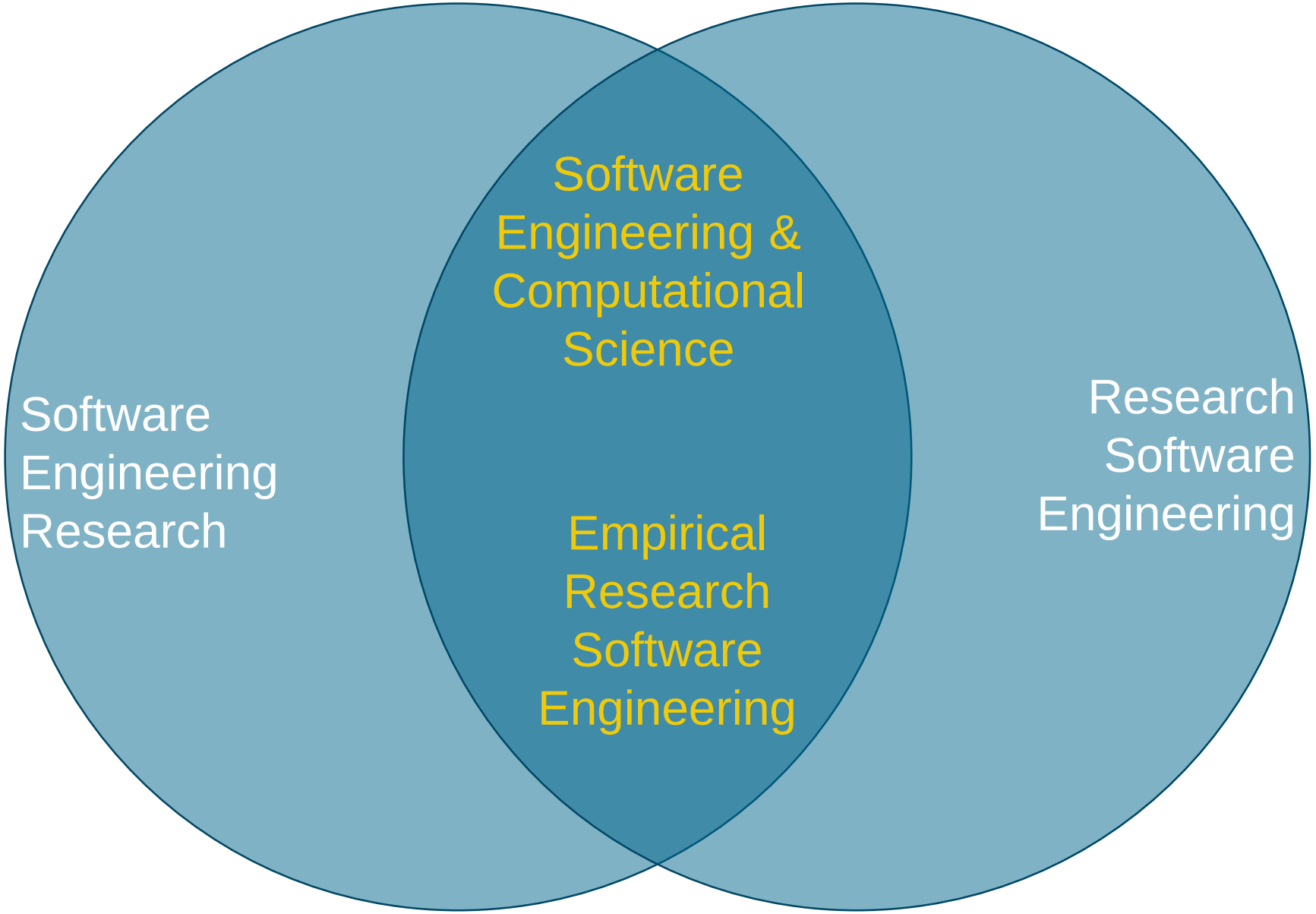
PUB: cites publication
PRO: cites project name/website
INS: instrument-like
URL: URL in text
NAM: in-text name only

[4] Druskat, S., Chue Hong, N. P., Kornek, P., Buzzard, S., Konovalov, A. (2023) Don't mention it: challenges to using software mentions to investigate citation and discoverability, PeerJ Computer Science.

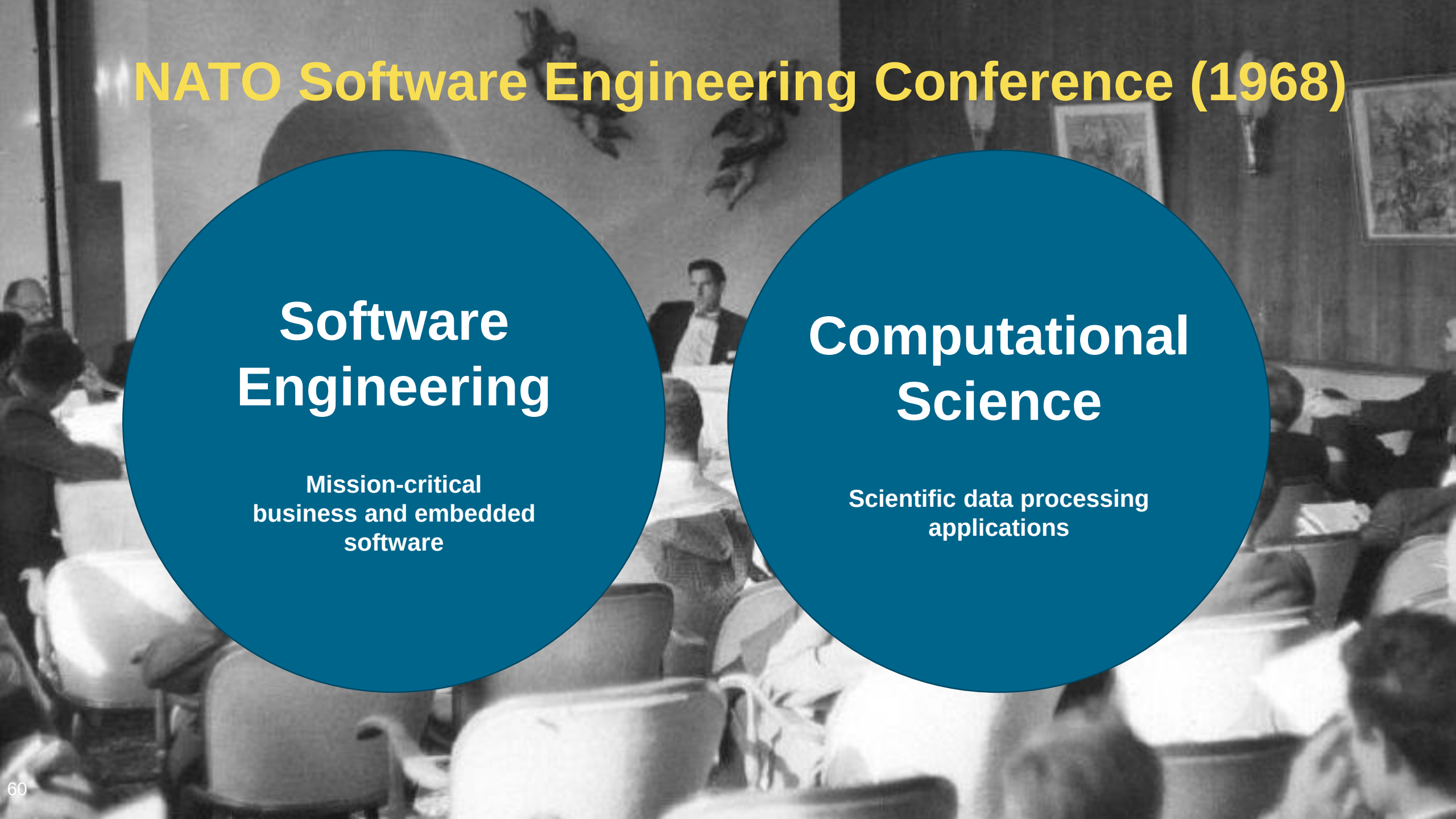
Citation File Format (CFF)



https://citation-file-format.github.io/


Software Engineering Research
Software Engineering & Computational Science
Empirical Research Software Engineering
Research Software Engineering

NATO Software Engineering Conference (1968)



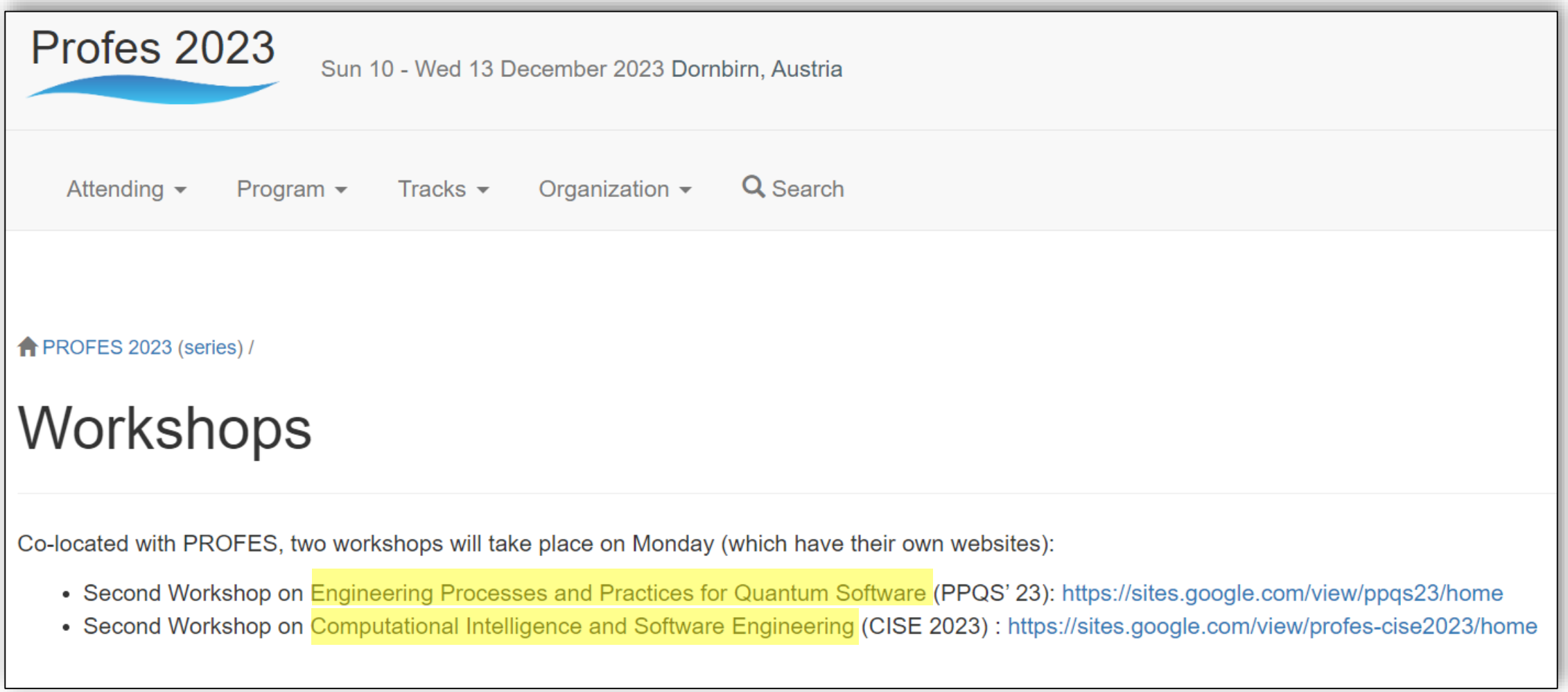
Software Engineering

Mission-critical business and embedded software

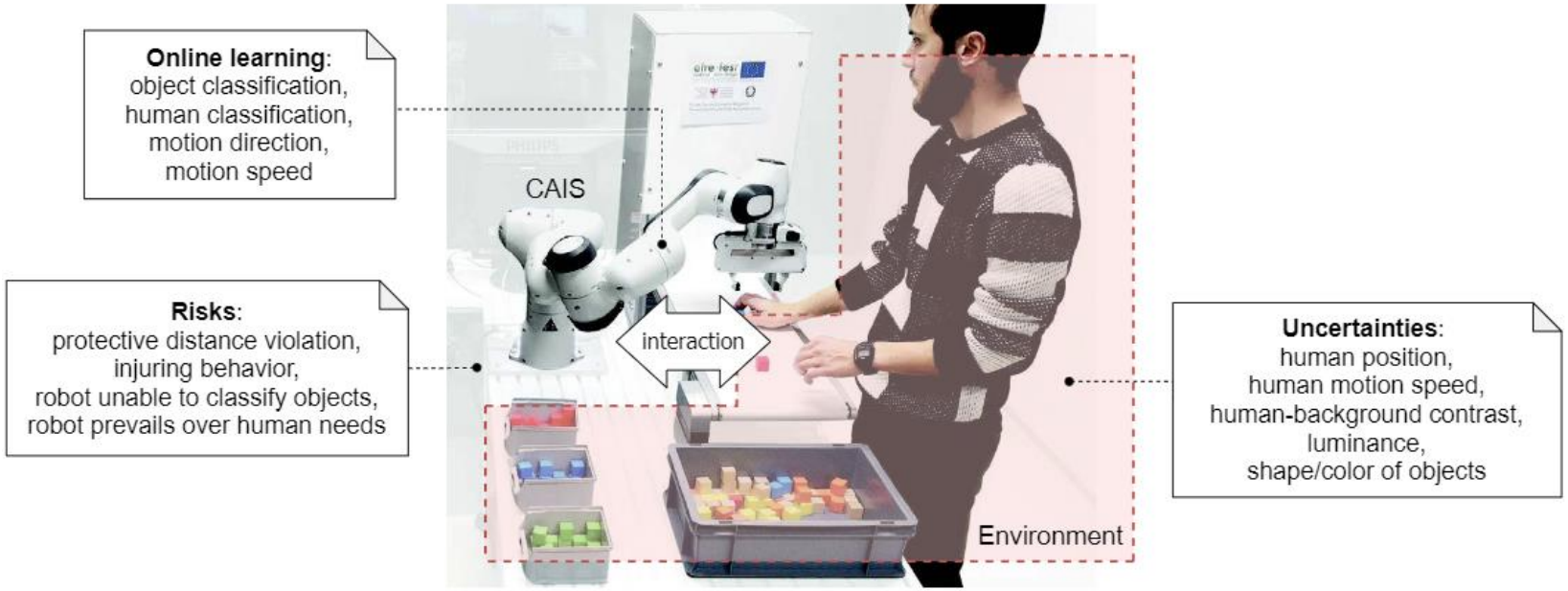
Computational Science

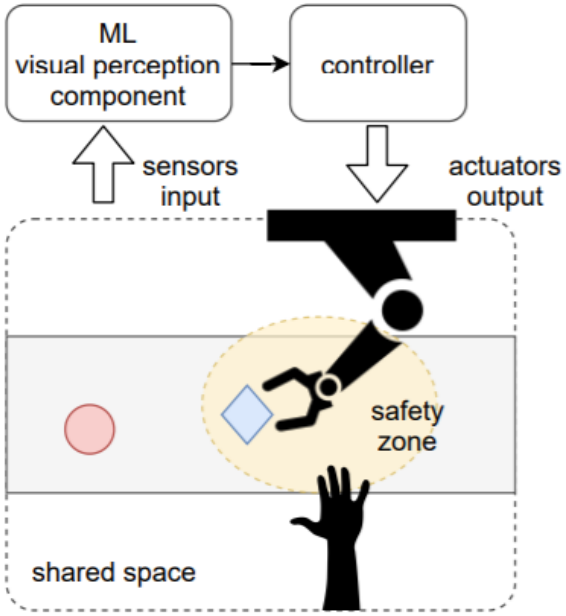
Scientific data processing applications

PROFES 2023


Profes 2023
Sun 10 - Wed 13 December 2023 Dornbirn, Austria
Attending
Program
Tracks
Organization
Search
PROFES 2023 (series) /
Workshops
Co-located with PROFES, two workshops will take place on Monday (which have their own websites):
Second Workshop on Engineering Processes and Practices for Quantum Software (PPQS' 23): https://sites.google.com/view/ppqs23/home
Second Workshop on Computational Intelligence and Software Engineering (CISE 2023) : https://sites.google.com/view/profes-cise2023/home

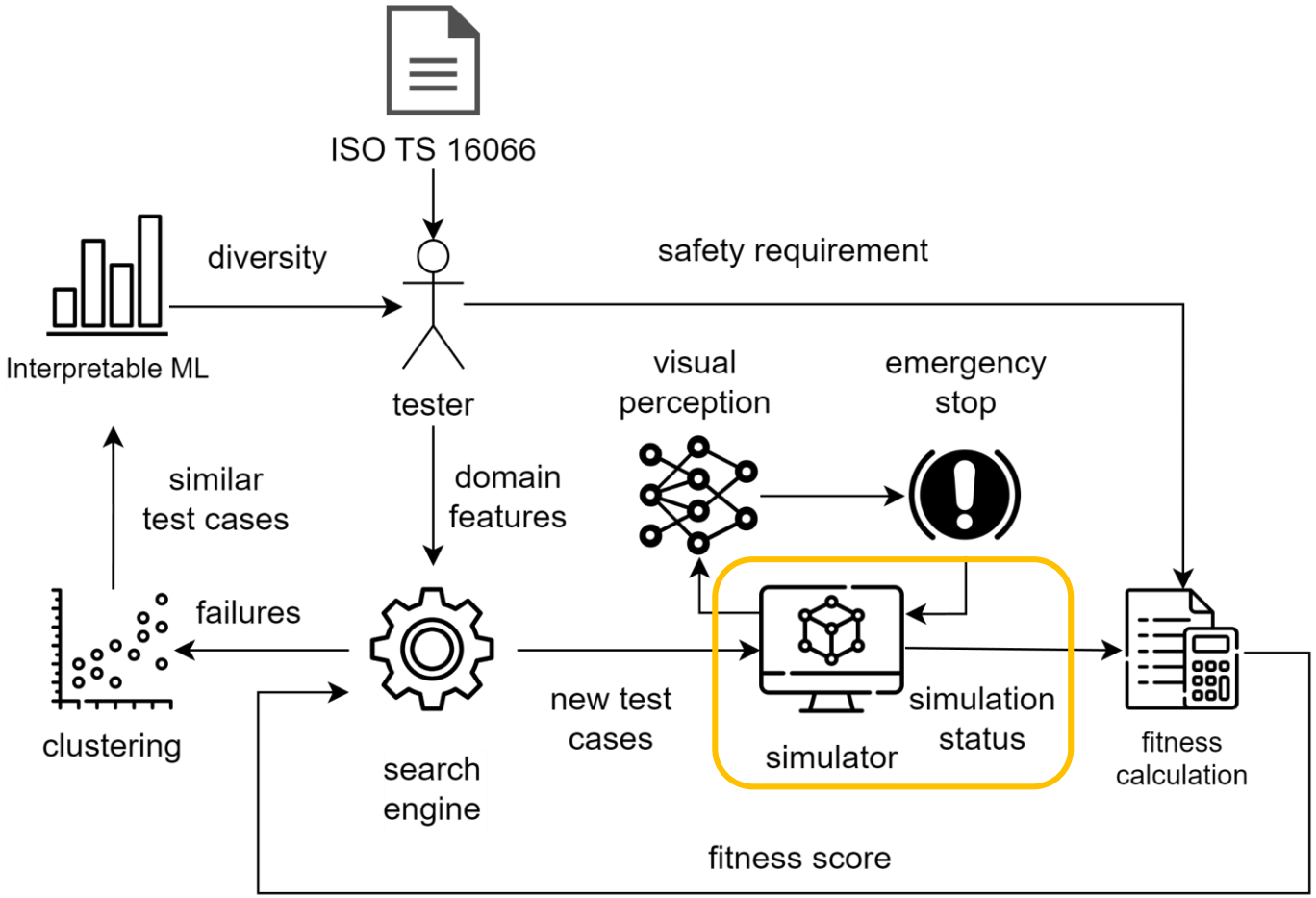
Online Testing of Collaborative AI Systems



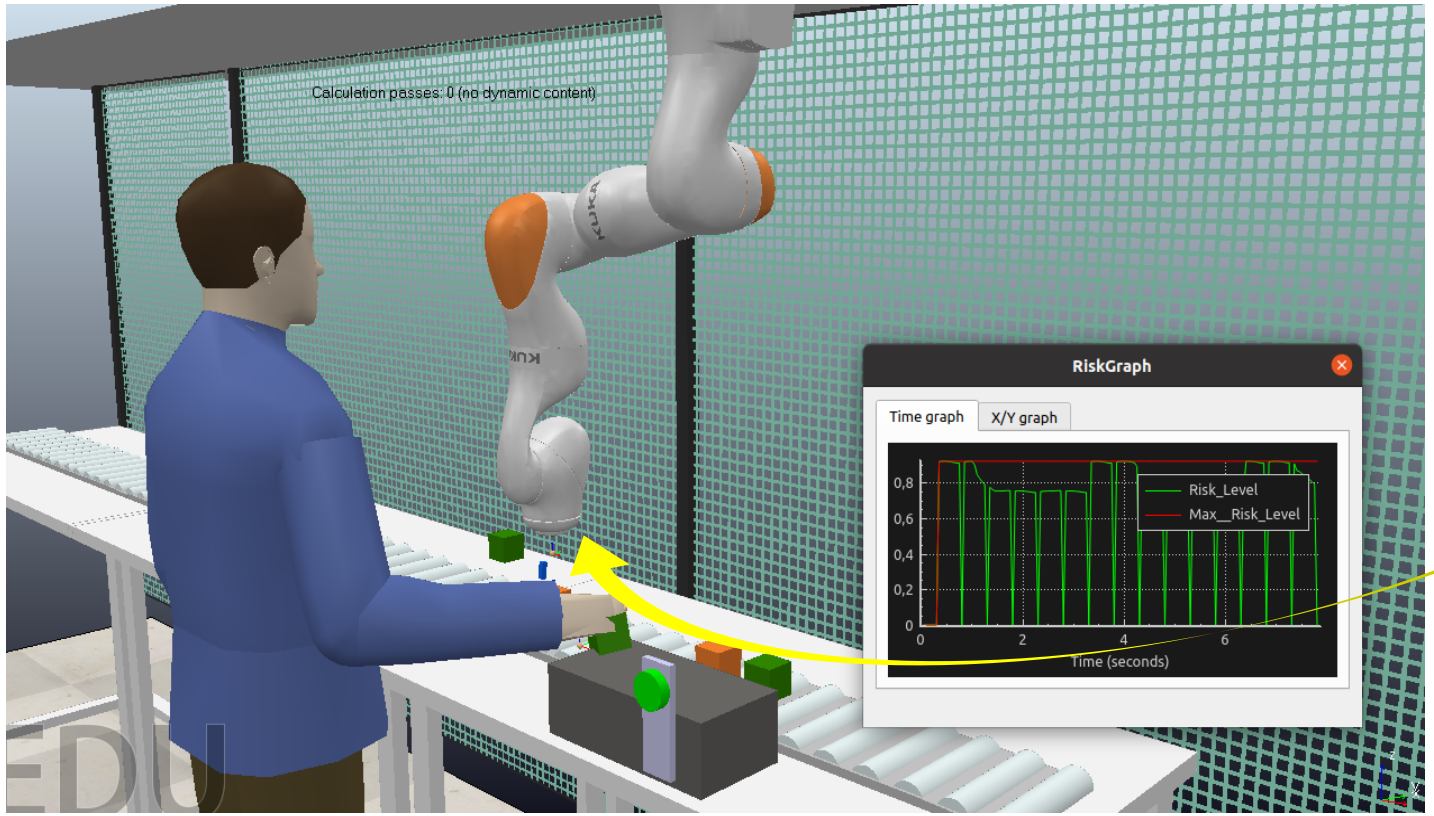


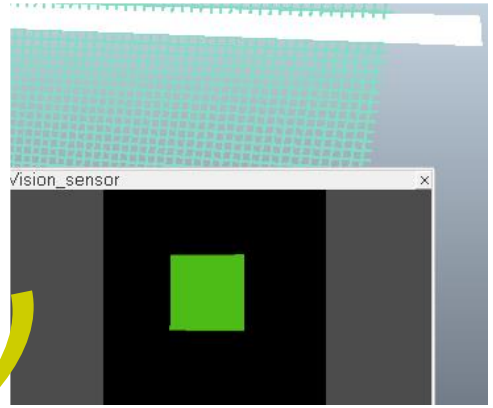
[5] Adigun, J., Huck, T., Camilli, M., Felderer, M. (2023) Risk-driven Online Testing and Test Case Diversity Analysis for ML-enabled Critical Systems. The 34th IEEE International Symposium on Software Reliability Engineering.

Online Testing Approach



domain feature	type	lower bound	upper bound
diffuse light (R)	float	0.0	1.0
diffuse light (G)	float	0.0	1.0
diffuse light (B)	float	0.0	1.0
human speed (m/s)	float	0.1	0.5
robot speed (m/s)	float	0.05	0.5
wait time human (s)	integer	1	50
wait time robot (s)	integer	1	50



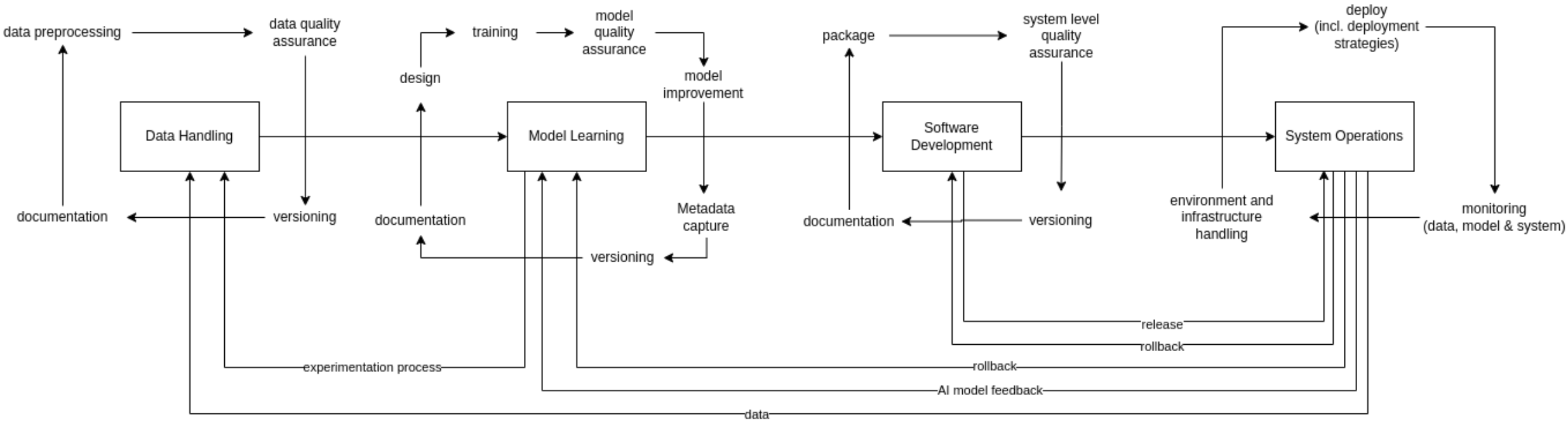


OpenCV

CoppeliaSim
from the creators of V-REP

Non-trivial implementation of an industrial collaborative AI system simulation

Pipeline for Continuous Development of ML models

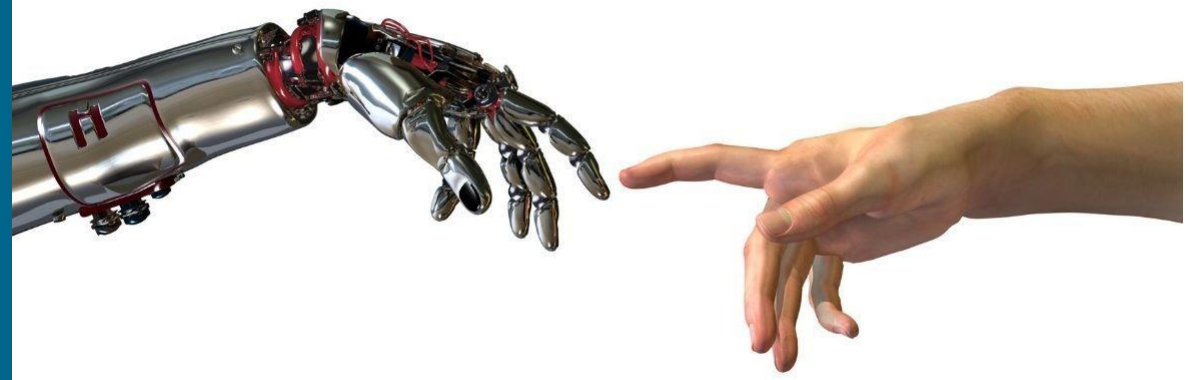


[6] Steidl, M., Felderer, M., Ramler, R. (2023) The pipeline for the continuous development of artificial intelligence models – Current state of research and practice. Journal of Systems and Software, 199, 111615. Elsevier.

Synergies in AI Engineering

Computational Science offers FAIR simulations, mathematical models, heterogenous large-scale data and use cases for AI Engineering

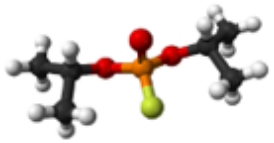
Software engineering offers SE processes and techniques for AI Engineering



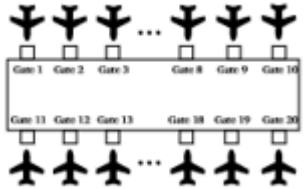
Quantum Computing

Quantum computing is a multidisciplinary field comprising aspects of computer science, physics, and mathematics that utilizes quantum mechanics to solve complex problems faster than on classical computers

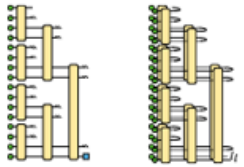
Quantum Simulation



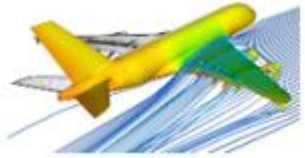
Combinatorial Optimisation



Quantum Enhanced Machine Learning



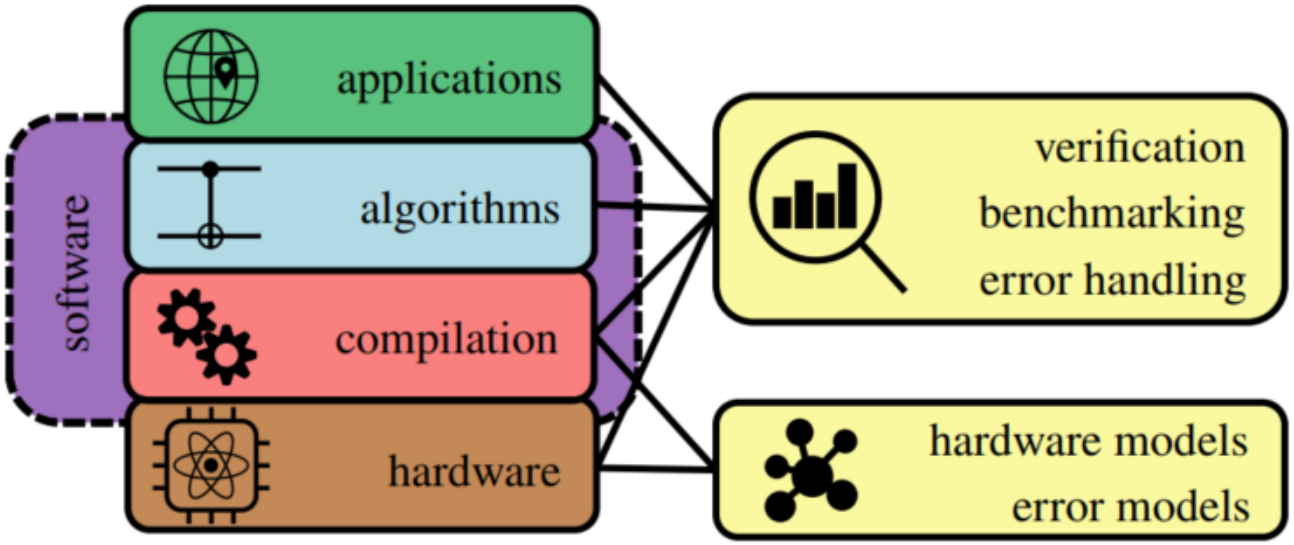
Classical Simulation



Required Error Correction

Quantum Computing Software Stack

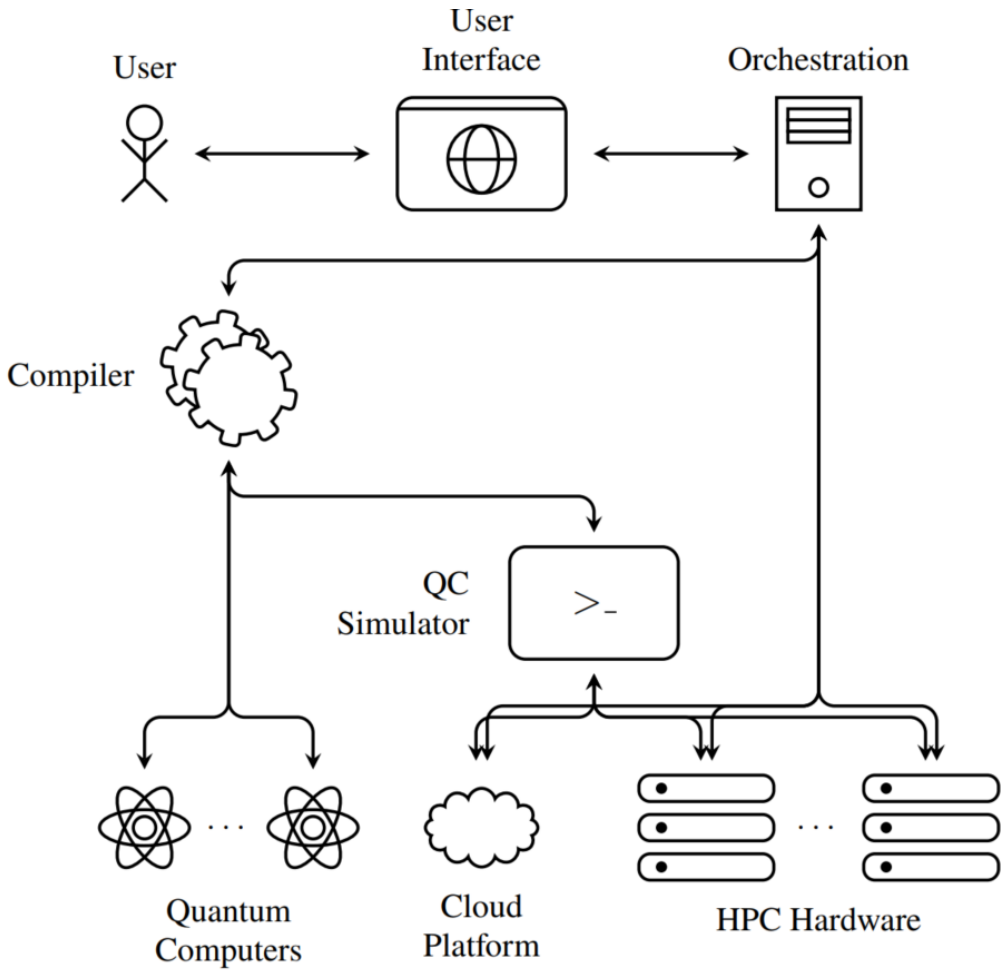
In particular, techniques from computer science, programming languages and software engineering are needed to realize a quantum computing software stack



[7] Basermann, A., et al. (2024) Quantum Software Ecosystem Design. Springer (to appear)

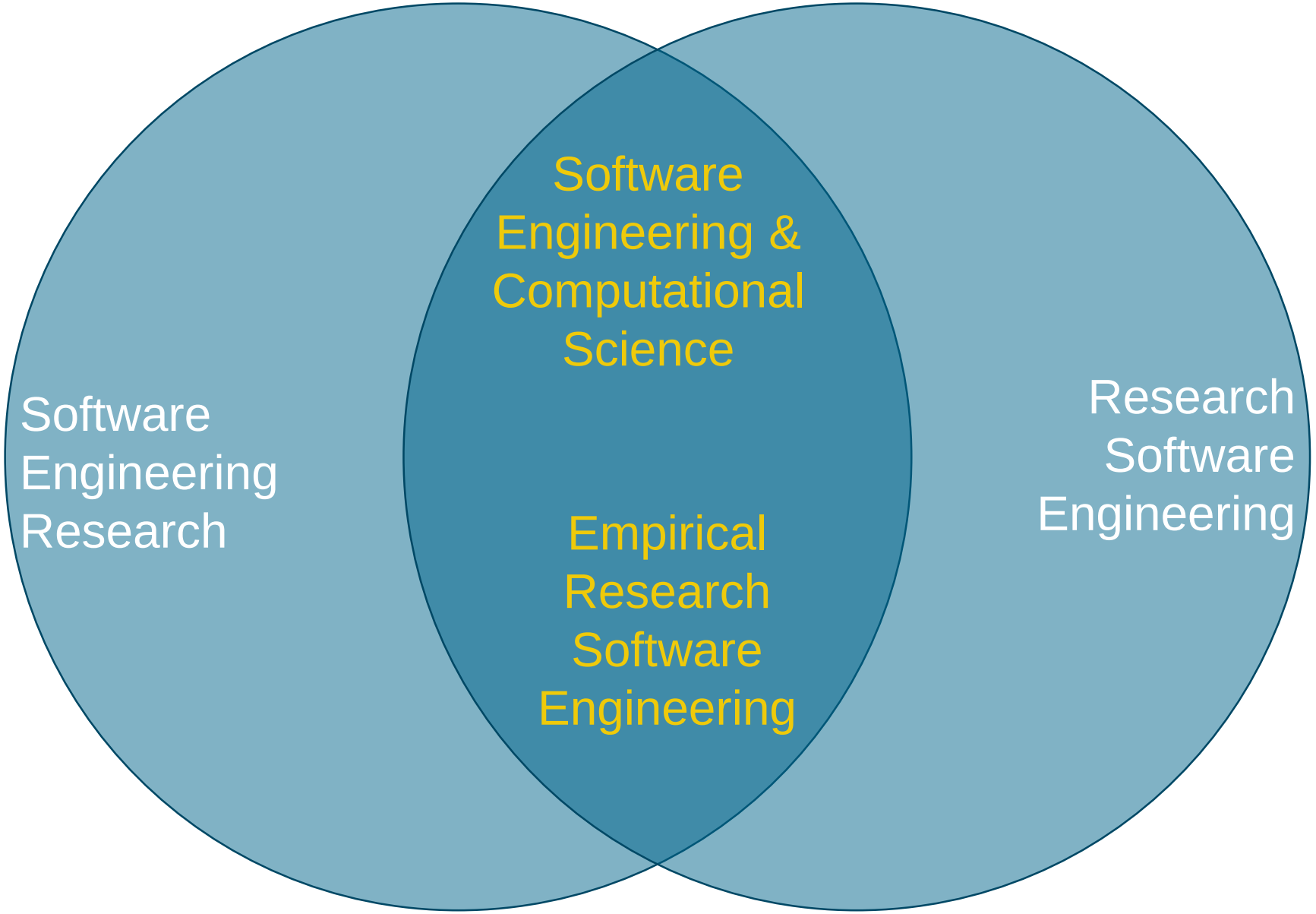
Quantum Computing Platform

… a Quantum Computing Platform supporting hybrid computing
requires even more software engineering



The recent disruptive technology Quantum Computing relies on physics and computational science and is based on research software

Software engineering is required to develop a quantum software stack and a quantum computing platform


Software Engineering Research
Software Engineering & Computational Science
Empirical Research Software Engineering
Research Software Engineering

Research software engineering provides an interesting application context for empirical software engineering

- Dealing with development under resource constraints
- Dealing with highly dynamic requirements and complex domains
- Dealing with long-lived software artifacts and reuse
- Dealing with vast configuration spaces
- Dealing with software development by domain experts

Research Questions for RSE

There is little knowledge about the relation between Software Engineering and Research Software Engineering

How to organize software-centric scientific processes?

What are suitable business models for research software?

What is specific about RSE compared to other SE specializations?

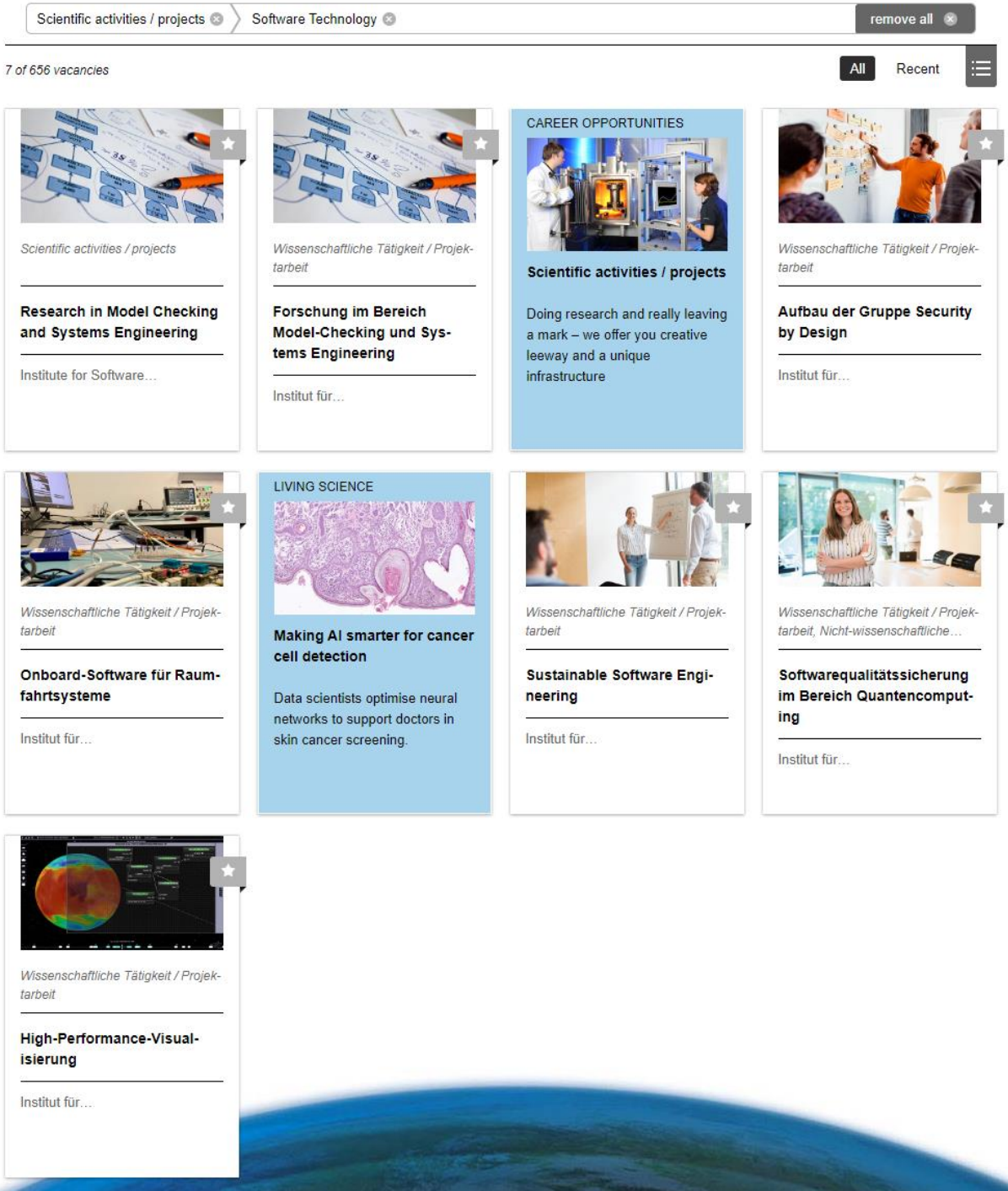
What are the types and maturity levels of research software?

Which skills and educational formats are required for RSE practitioners?

How to integrate SE techniques into research software development?

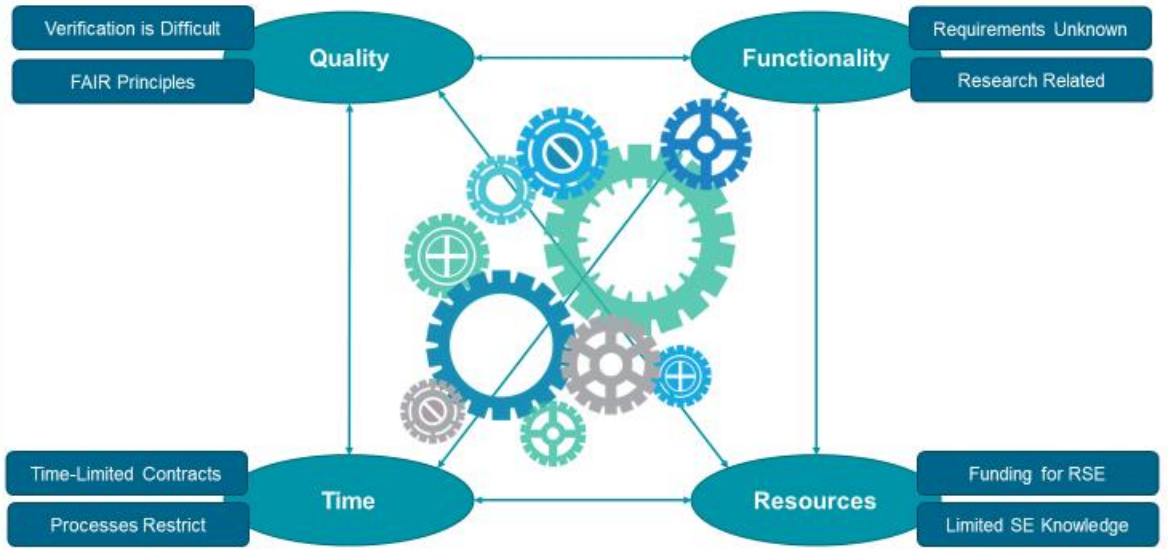
Is research software of poorer quality than industry software?

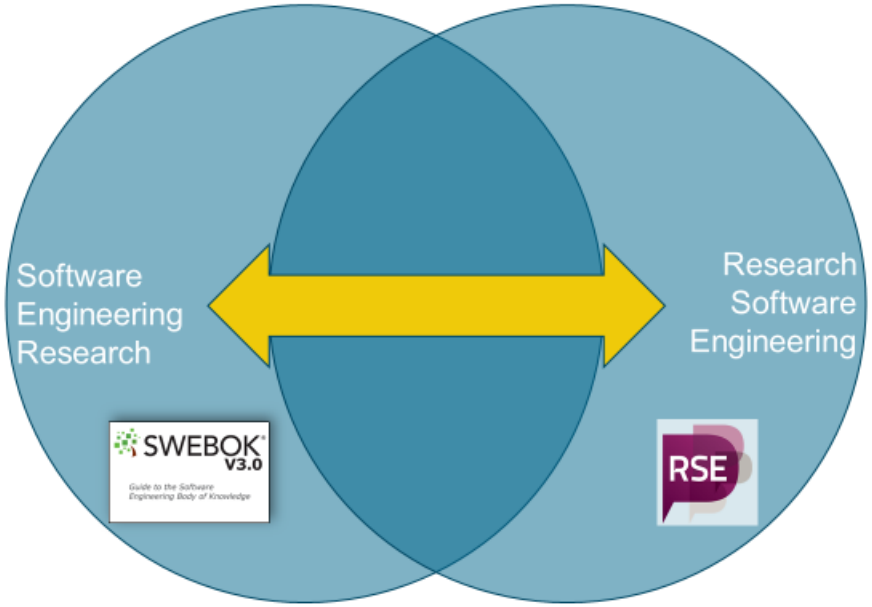
https://dlr.de/sc

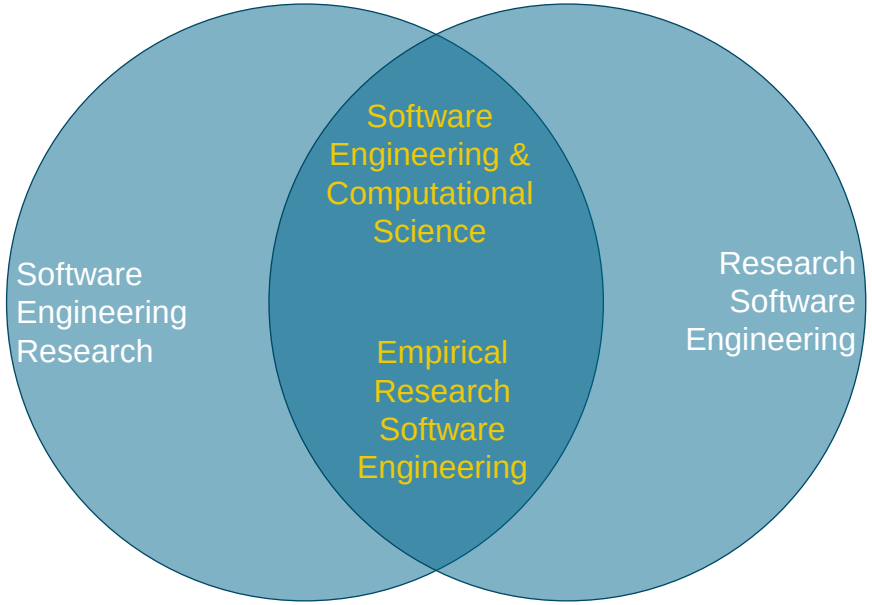


Research software is created
during the research process
or for a research purpose

Trade Off in Software Engineering: Specifics of RSE







References

[1] Kurnatowski, L., Schlauch, T., Haupt, C. (2020) Software Development at the German Aerospace Center: Role and Status in Practice. ICSE (Workshops) 2020.

[2] Carver, J. C., Hong, N. P. C., Thiruvathukal, G. K. (2016) Software engineering for science. CRC Press.

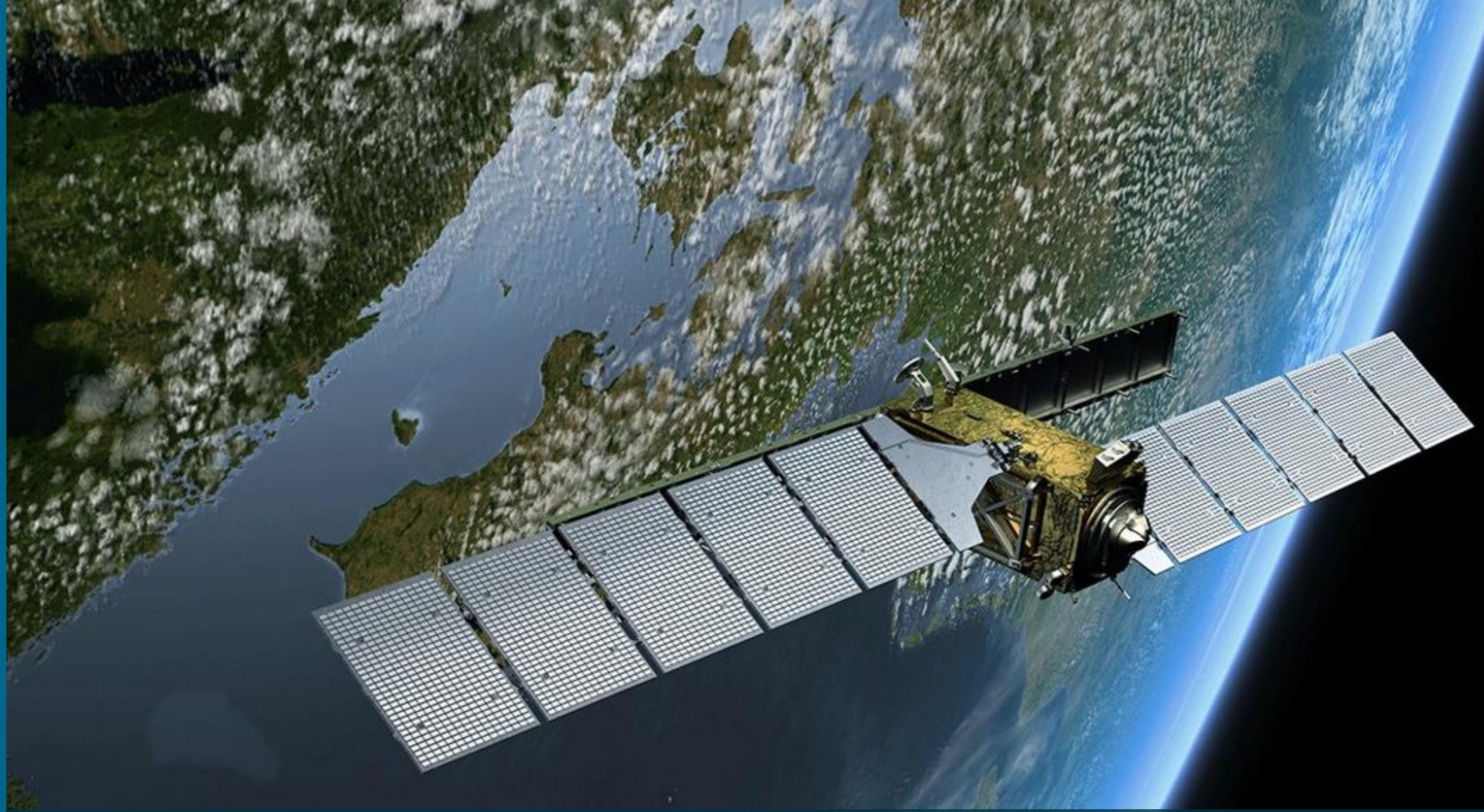
[3] Schönborn, M. T. (2023) Adopting Software Engineering Concepts in Scientific Research: Insights from Physicists and Mathematicians Turned Consultants. Computing in Science & Engineering.

[4] Druskat, S., Chue Hong, N. P., Kornek, P., Buzzard, S., Konovalov, A. (2023) Don’t mention it: challenges to using software mentions to investigate citation and discoverability, PeerJ Computer Science.

[5] Adigun, J., Huck, T., Camilli, M., Felderer, M. (2023) Risk-driven Online Testing and Test Case Diversity Analysis for ML-enabled Critical Systems. The 34th IEEE International Symposium on Software Reliability Engineering.

[6] Steidl, M., Felderer, M., Ramler, R. (2023) The pipeline for the continuous development of artificial intelligence models – Current state of research and practice. Journal of Systems and Software, 199, 111615. Elsevier.

[7] Basermann, A., et al. (2024) Quantum Software Ecosystem Design. Springer (to appear)



Prof. Dr. Michael Felderer
Institute for Software Technology

michael.felderer@dlr.de
https://www.dlr.de/sc/