INNOVATING AT THE INTERSECTION
SOFTWARE ENGINEERING RESEARCH FOR SCIENCE AND INDUSTRY

Prof. Dr. Michael Felderer

UNIVERSITY OF COLOGNE

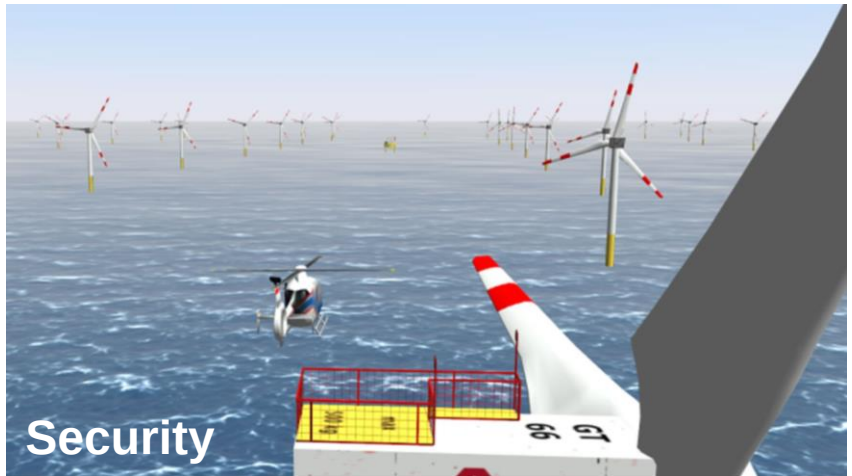
DLR

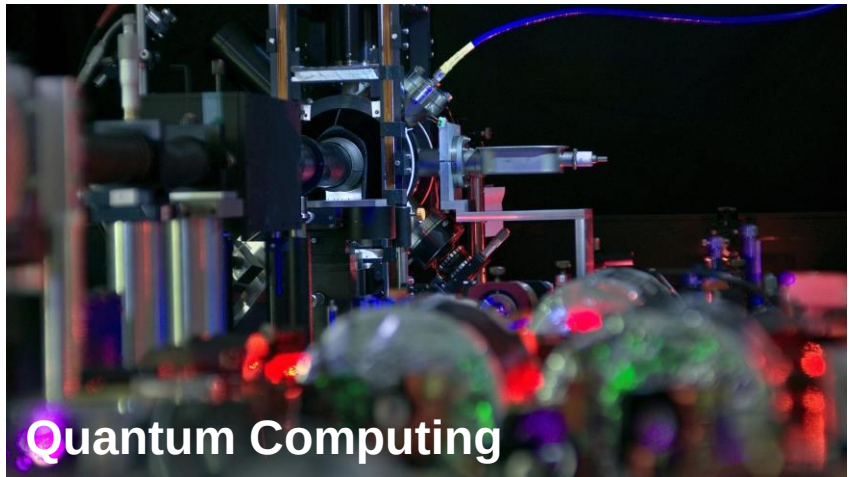
DLR Research Areas

11,000
Employees

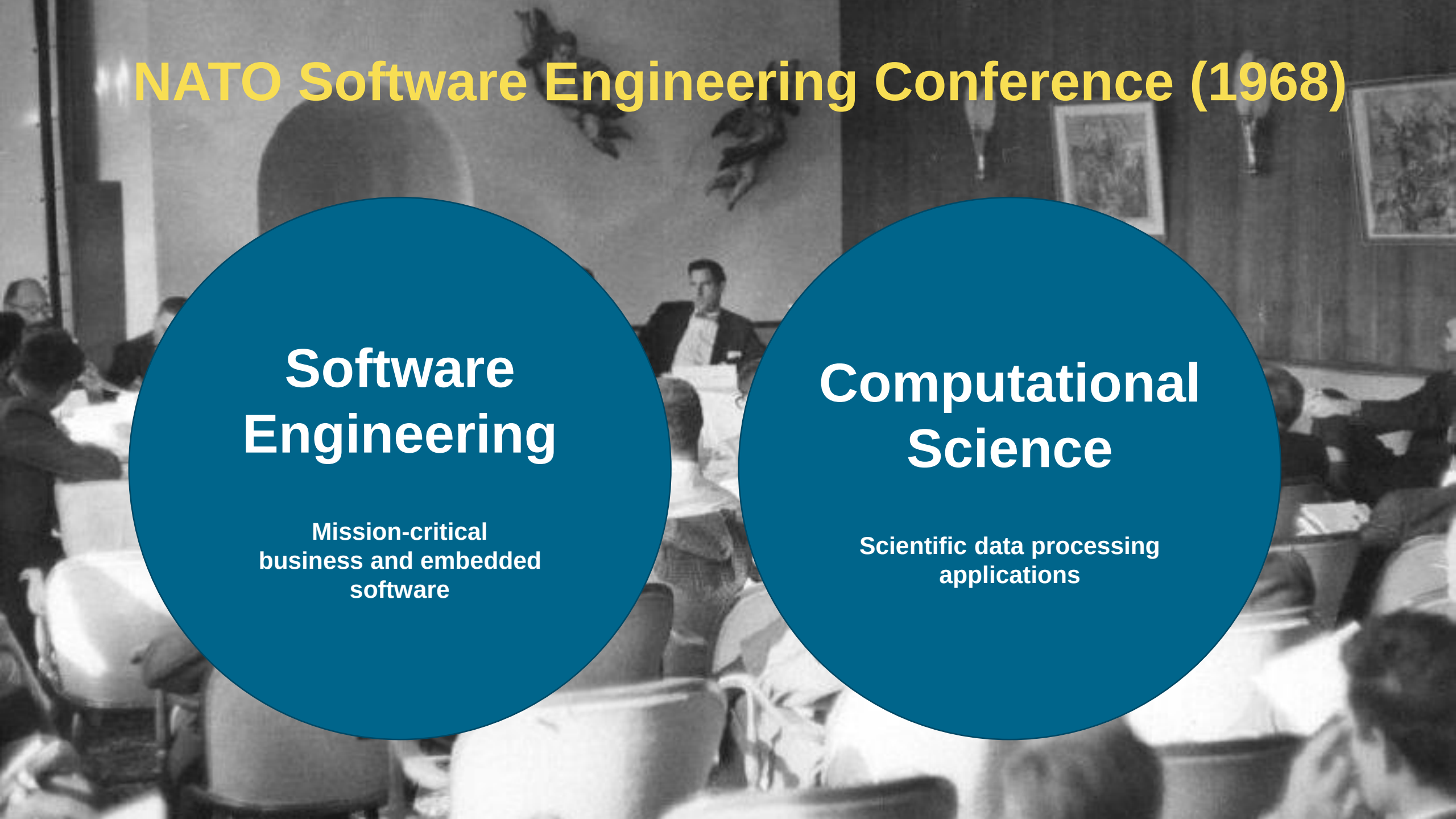
54
Institutes and Facilities

35
Locations and Offices





NATO Software Engineering Conference (1968)



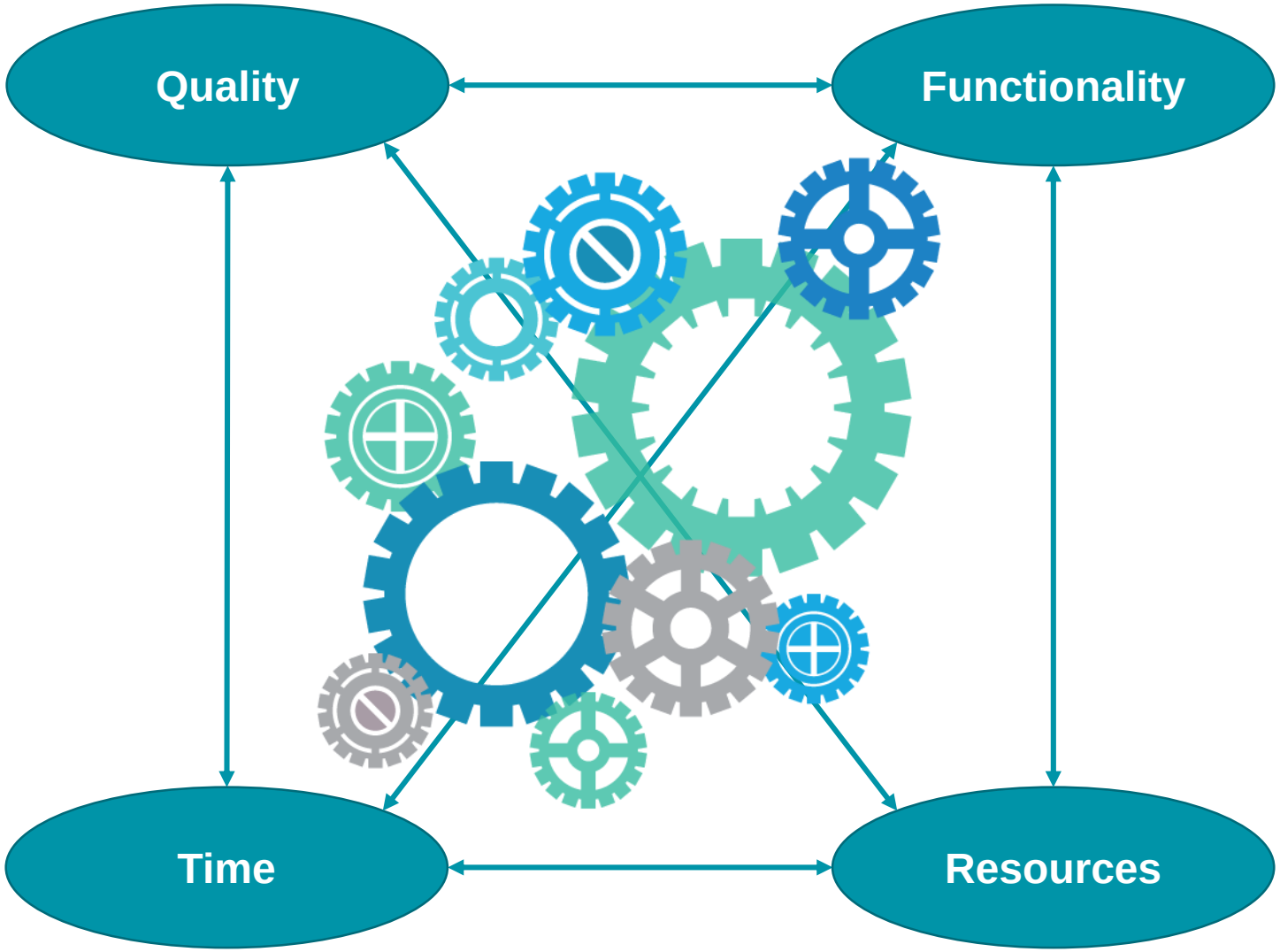
Software Engineering

Mission-critical business and embedded software

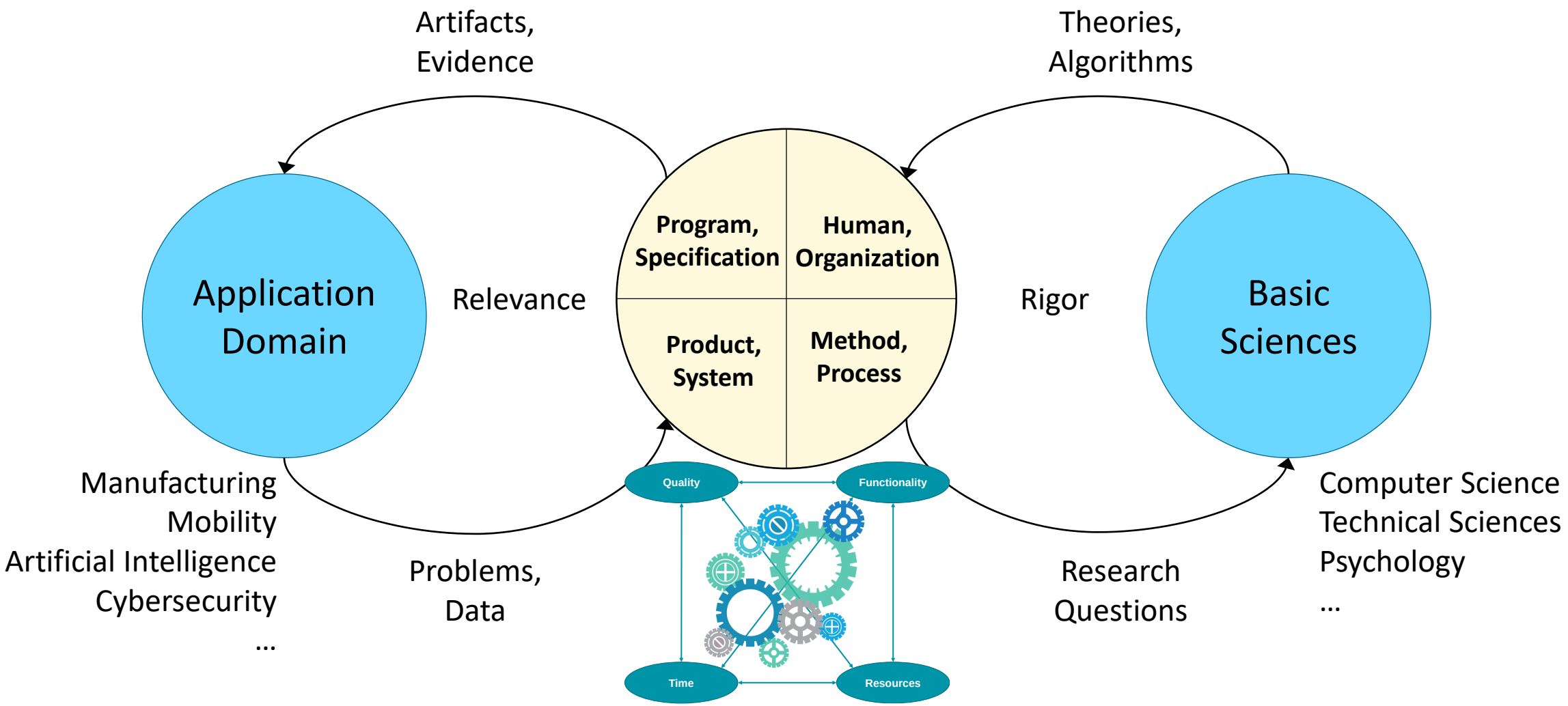
Computational Science

Scientific data processing applications

Trade Off in Software Engineering



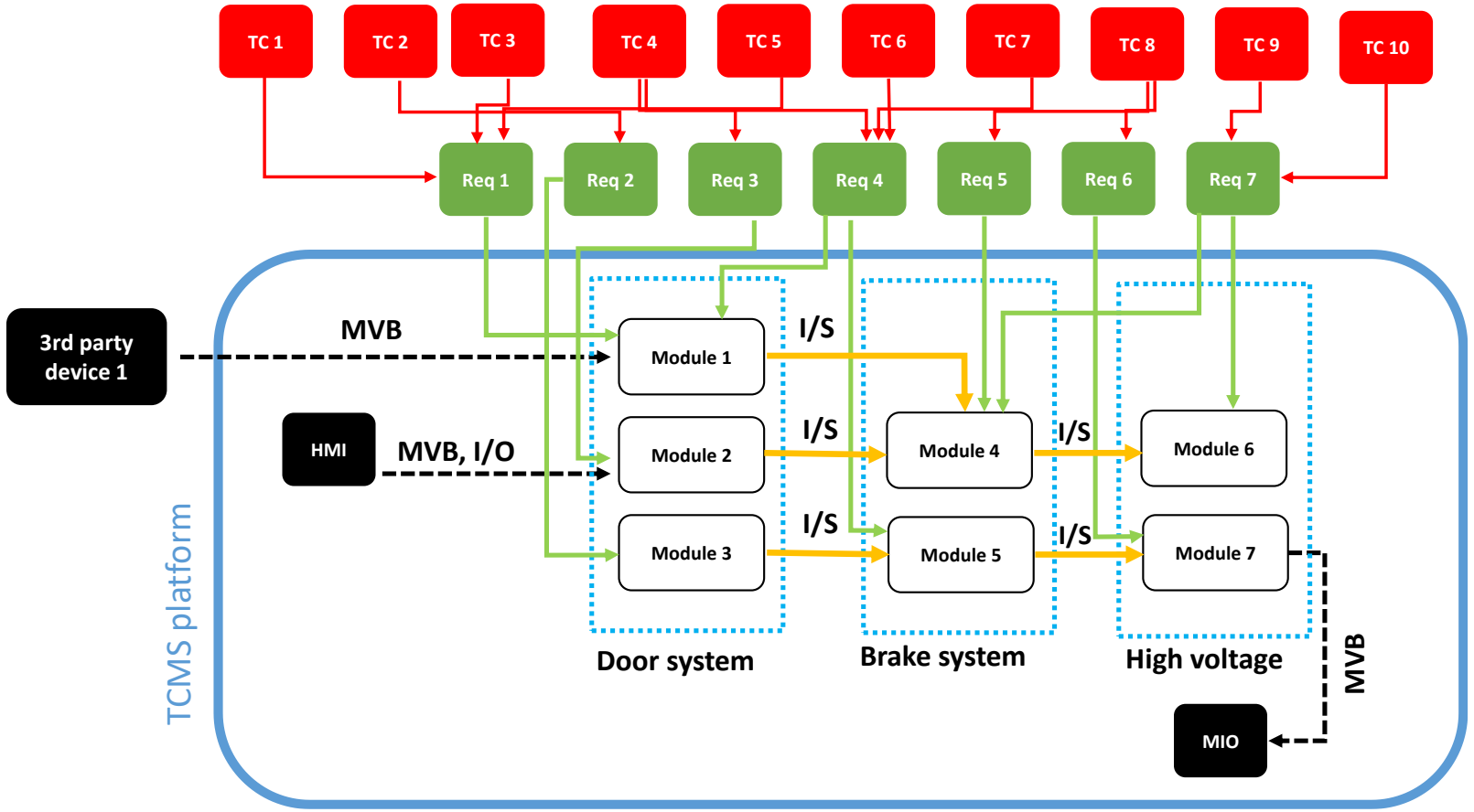
Software Engineering Research



Felderer, M., Reussner, R., Rumpe, B. (2021) Software Engineering und Software Engineering Forschung im Zeitalter der Digitalisierung. Informatik Spektrum.

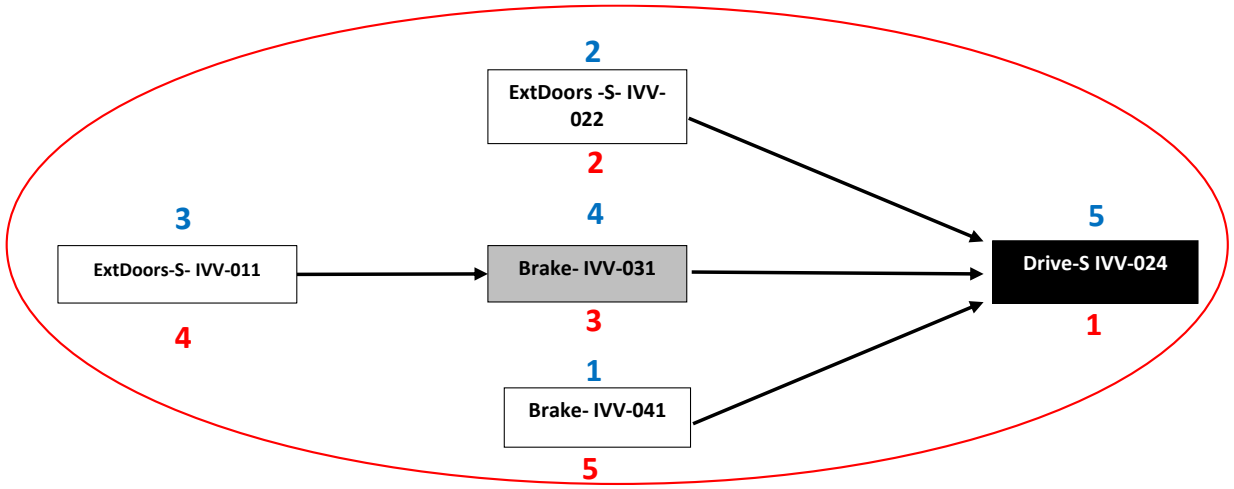
Application Domain: Mobility

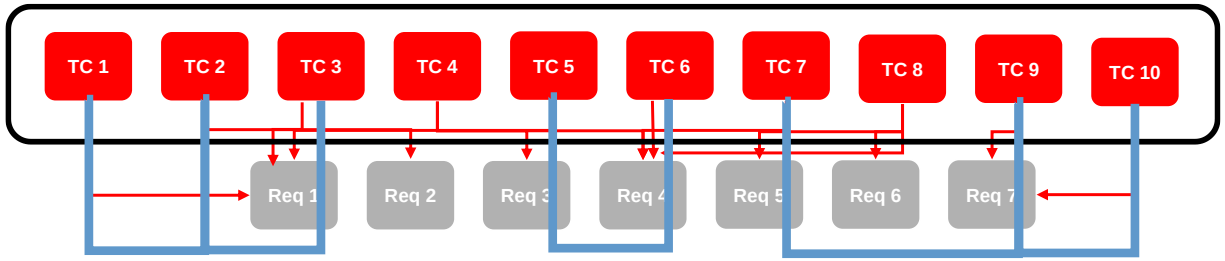
Bombardier BR 490 S-Bahn Hamburg



Established System and Software Development Processes and Artifacts

Problem: Automated Test Case Dependency Detection





Exec. order	Test case ID	Exec. 1	Exec. 2	Exec. 3	Exec. 4
1	Drive-S IVV-024	Fail	Fail	Fail	Pass
2	Brake- IVV-031	Fail	Fail	Pass	-
3	ExtDoors-S- IVV-011	Fail	Pass	-	-
4	ExtDoors -S- IVV-022	Not run	Fail	Pass	-
5	Brake- IVV-041	Fail	Pass	-	-

Estimated time with our dependency aware approach= **5 h**
Total actual time at BOMBARDIER= **45 h**

40 h saving

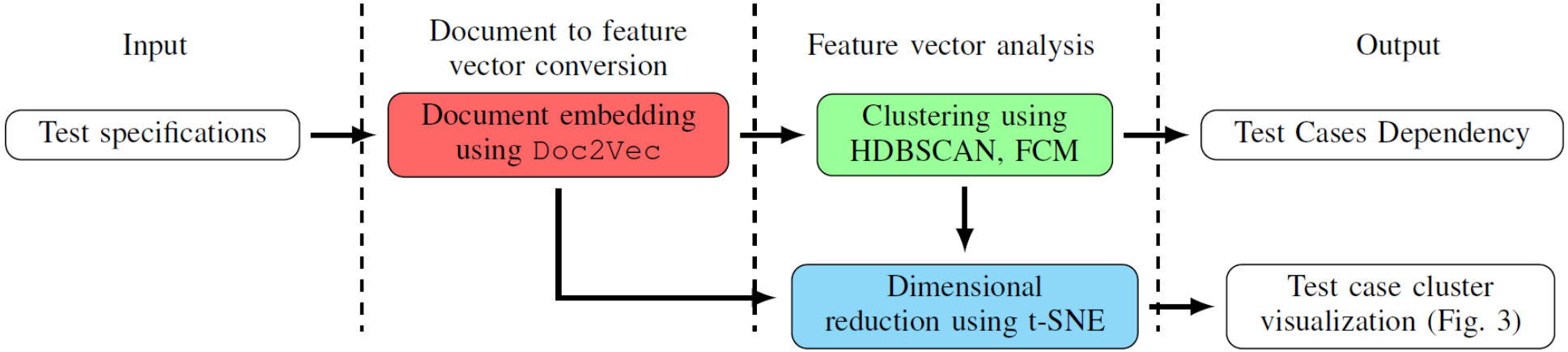
Initial State

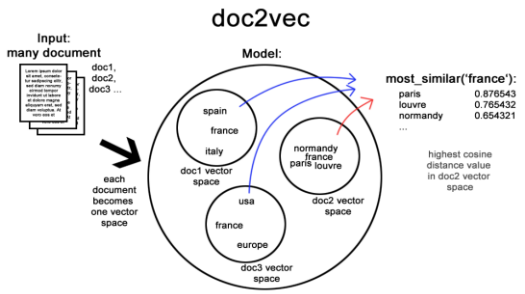
Initial state: Ready to drive DM1 cab active Ready to drive				
Step	Action	Reaction	Pass/Fail	Comments
1	Set the train at speed 36 km/h	Check that on IDU no event regarding ”undue fourth stage” is shown	Pass	
2	Lock and set from VCS Brake IO panel the signal ”relay Brake 4th stage” from DM1	Check that on IDU an event regarding ”undue fourth stage” is shown	Pass	As maintainer
3	Repeat for all cars	Check that: -the speed above are fulfilled	Pass	

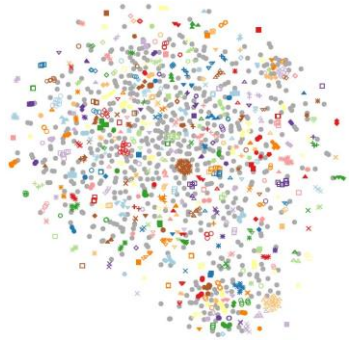
Clean-up

Improving Efficiency (Time) taking Quality and Resources into account

Approach: ML-Based Test Case Dependency Detection



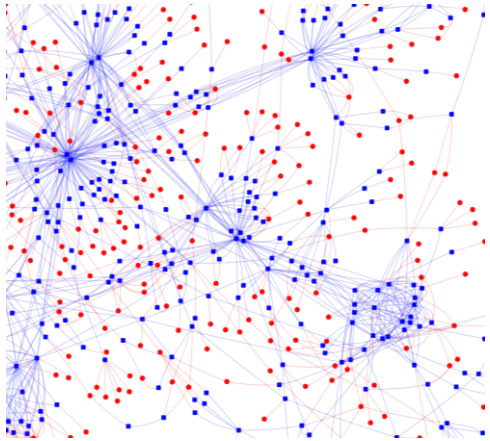




Tahvili, S., Hatvani, L., Felderer, M., Afzal, W., Bohlin, M. (2019) Automated Functional Dependency Detection Between Test Cases Using Doc2Vec and Clustering. AITest 2019.

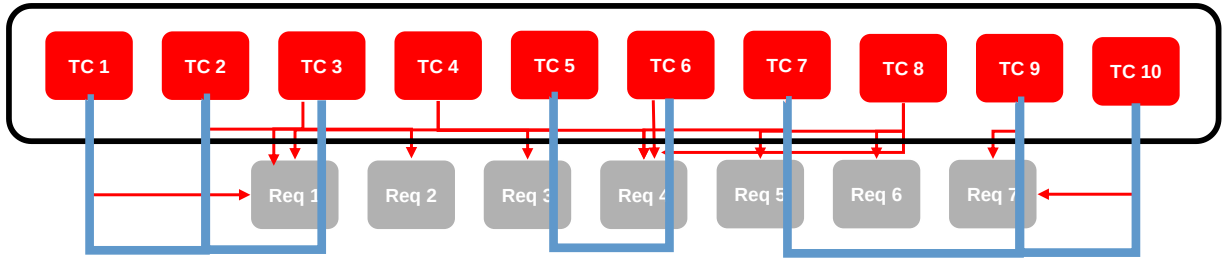
Application of timely ML approaches to develop an automated test case depencency detection method

Evaluation and Implementation



Functional dependencies between approx. 4000 reqs (blue) and approx.2000 tests (red) via artifact analysis and survey

	HDBSCAN				FCM			
D	Precision	Recall	Accuracy	F1 Score	Precision	Recall	Accuracy	F1 Score
2	**0.9981**	**0.6113**	**0.8050**	**0.7582**	0.8789	0.3871	0.6668	0.5375
3	**0.9959**	**0.2781**	**0.6385**	**0.4348**	0.8275	0.2534	0.6002	0.3879
4	**0.9891**	0.0958	0.5474	0.1747	0.8021	**0.2127**	**0.5801**	**0.3363**
5	**0.9804**	0.0535	0.5262	0.1015	0.7845	**0.1913**	**0.5694**	**0.3076**
6	**0.9628**	0.0286	0.5137	0.0555	0.7727	**0.1772**	**0.5625**	**0.2883**
7	**0.9476**	0.0203	0.5096	0.0397	0.7653	**0.1672**	**0.5580**	**0.2745**
8	**0.9287**	0.0139	0.5064	0.0273	0.7583	**0.1589**	**0.5541**	**0.2627**
9	**0.9104**	0.0111	0.5050	0.0220	0.7477	**0.1490**	**0.5494**	**0.2485**
10	**0.8913**	0.0089	0.5039	0.0176	0.7340	**0.1373**	**0.5438**	**0.2313**

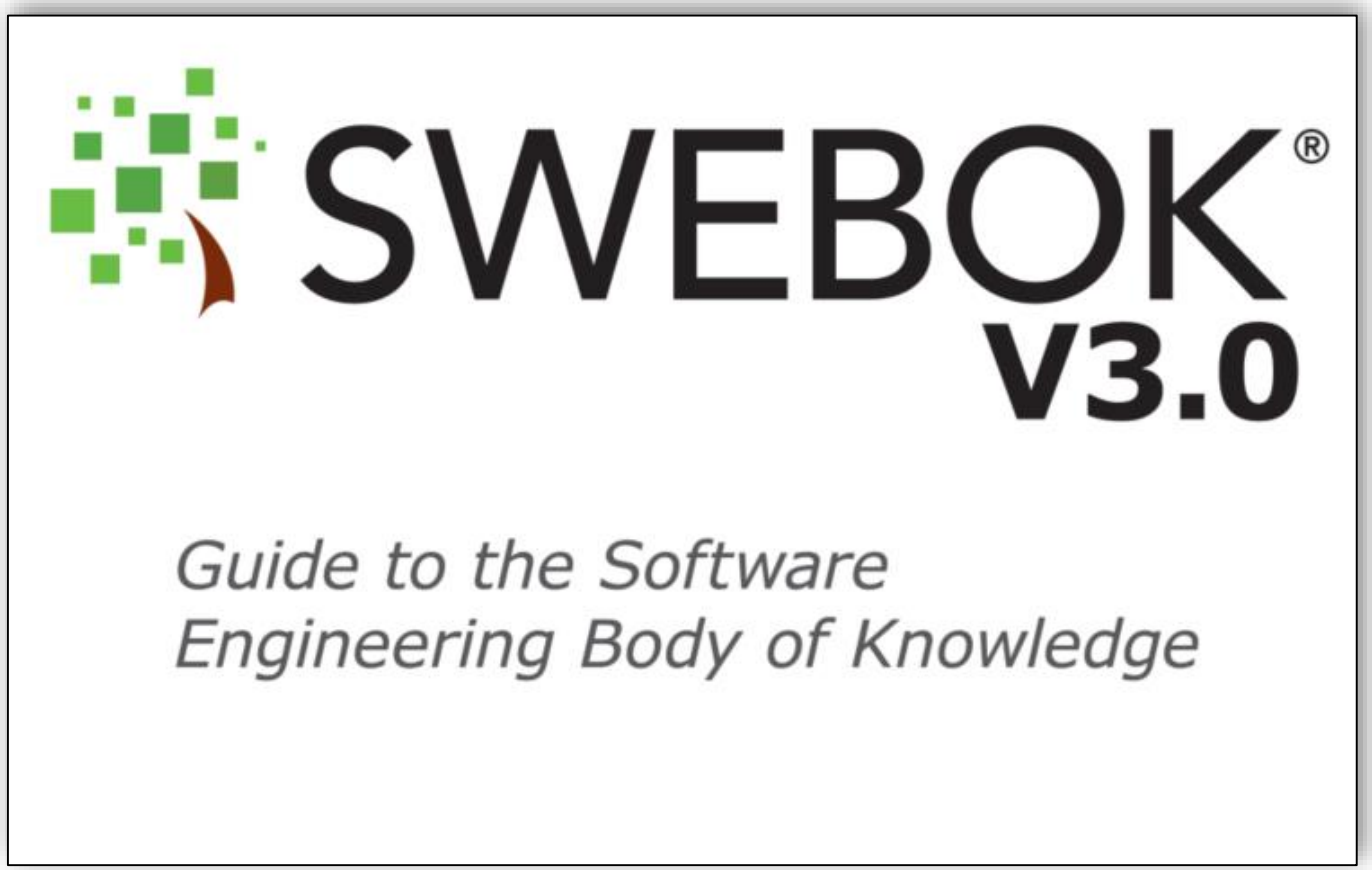


Empirically Evaluated Method is implemented in Industrial System Development Process

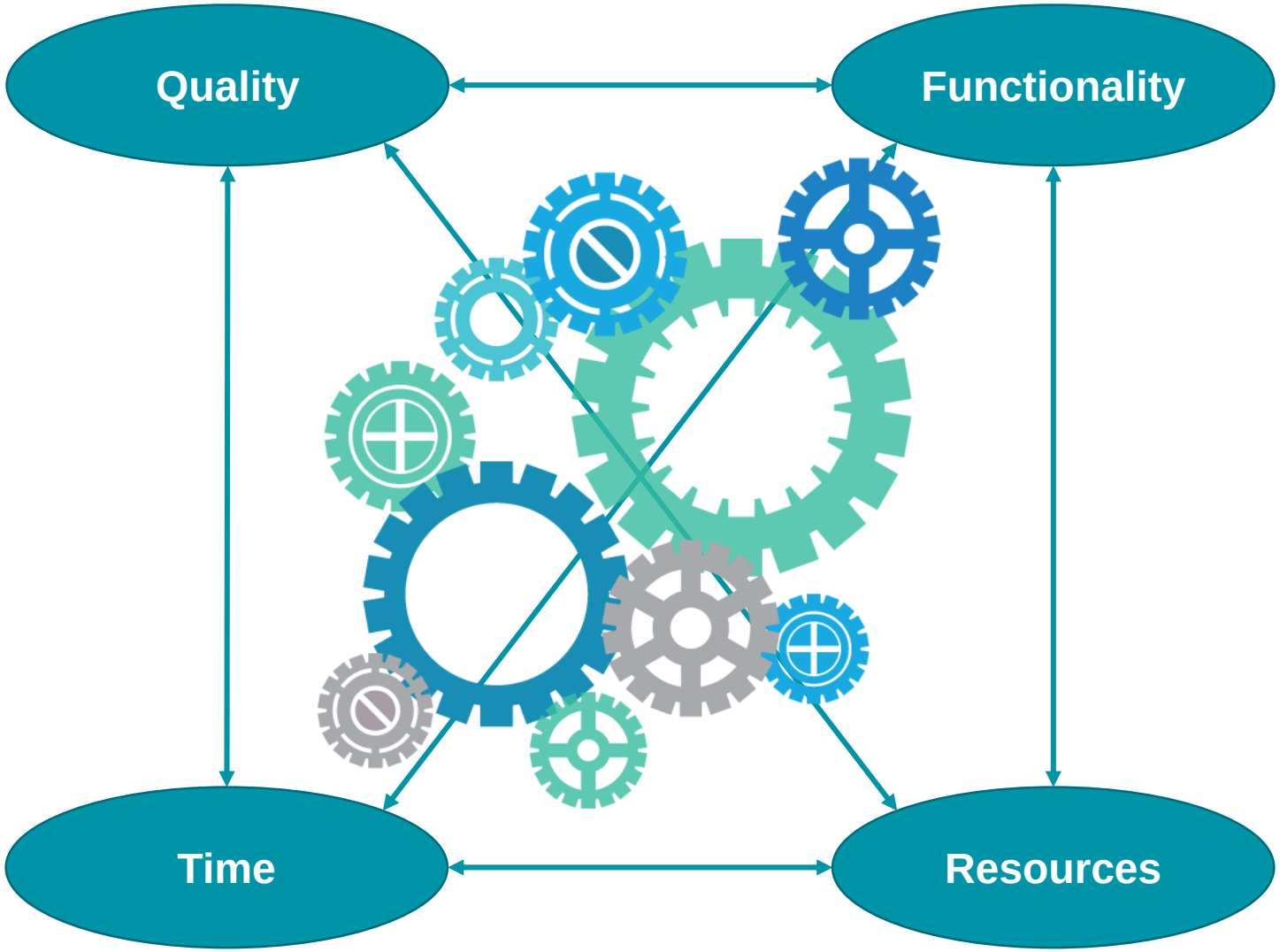
and fancy

Research software is a critical artifact that requires software engineering

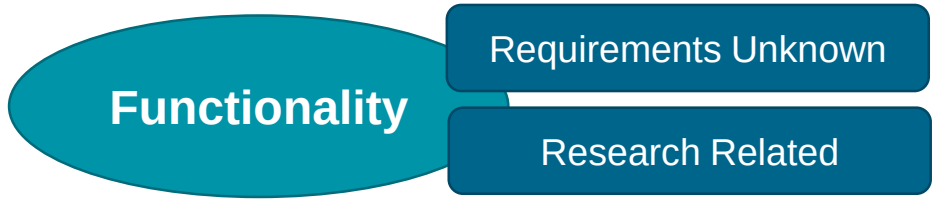
Well, just apply software engineering techniques …

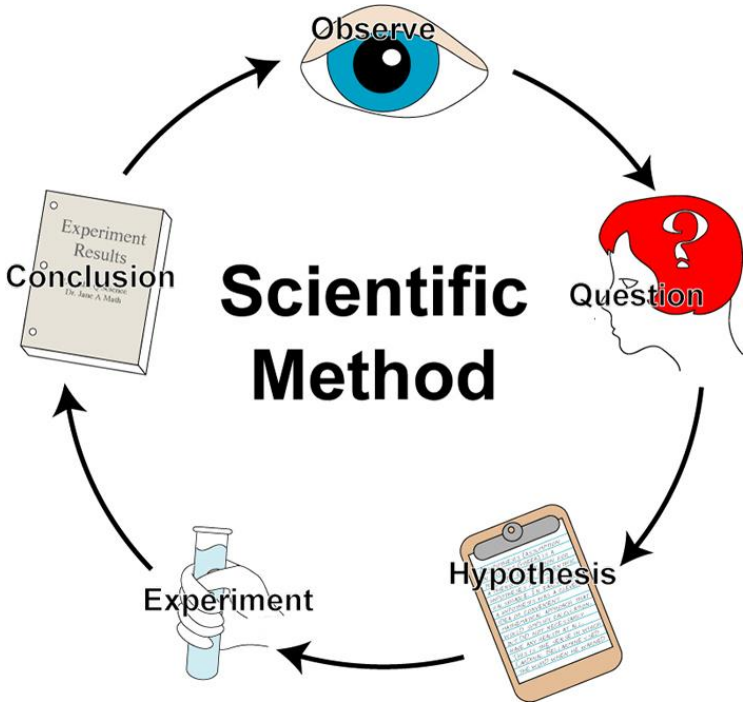


Trade Off in Software Engineering: Specifics of RSE



Trade Off in Software Engineering: Specifics of RSE





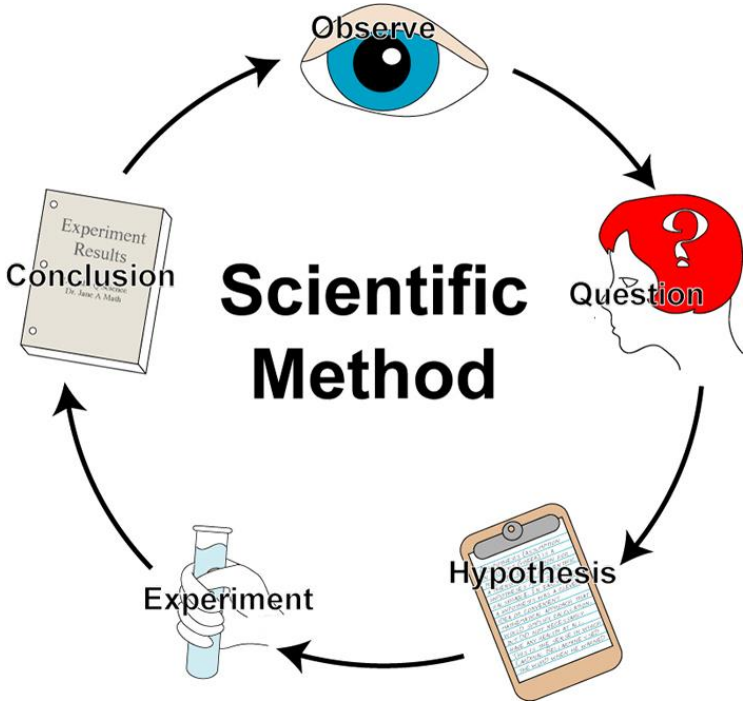
$$i\hbar\frac{\partial}{\partial t}\left|\Psi\right\rangle = \hat{H}\left|\Psi\right\rangle$$

Trade Off in Software Engineering: Specifics of RSE

Verification is Difficult

FAIR Principles

Quality



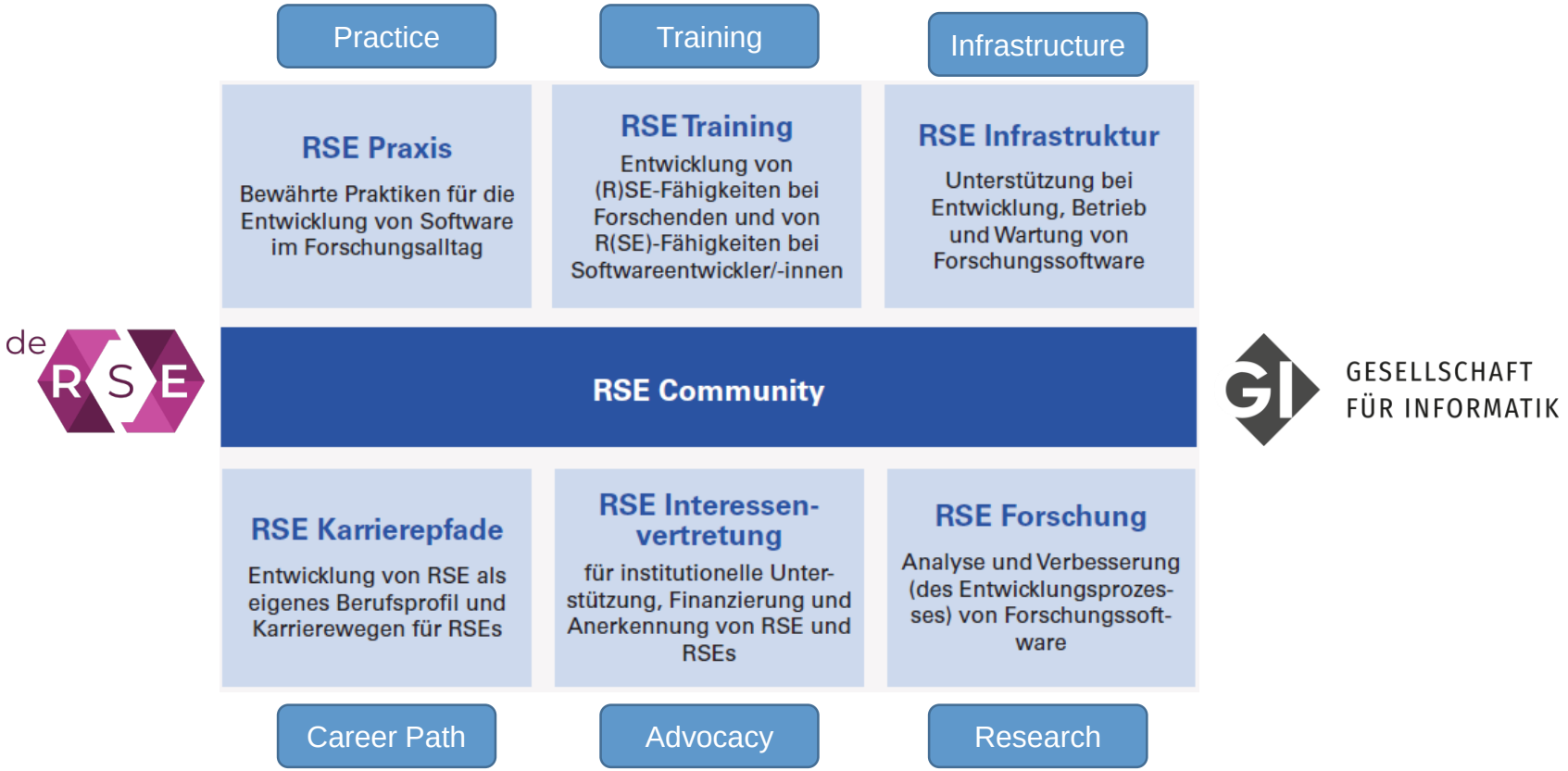
Trade Off in Software Engineering: Specifics of RSE

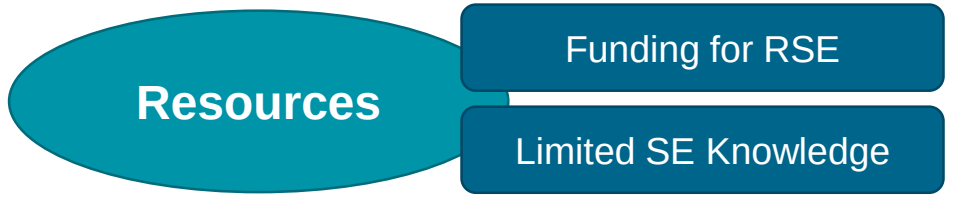
Time-Limited Contracts

Dev Processes Restrict

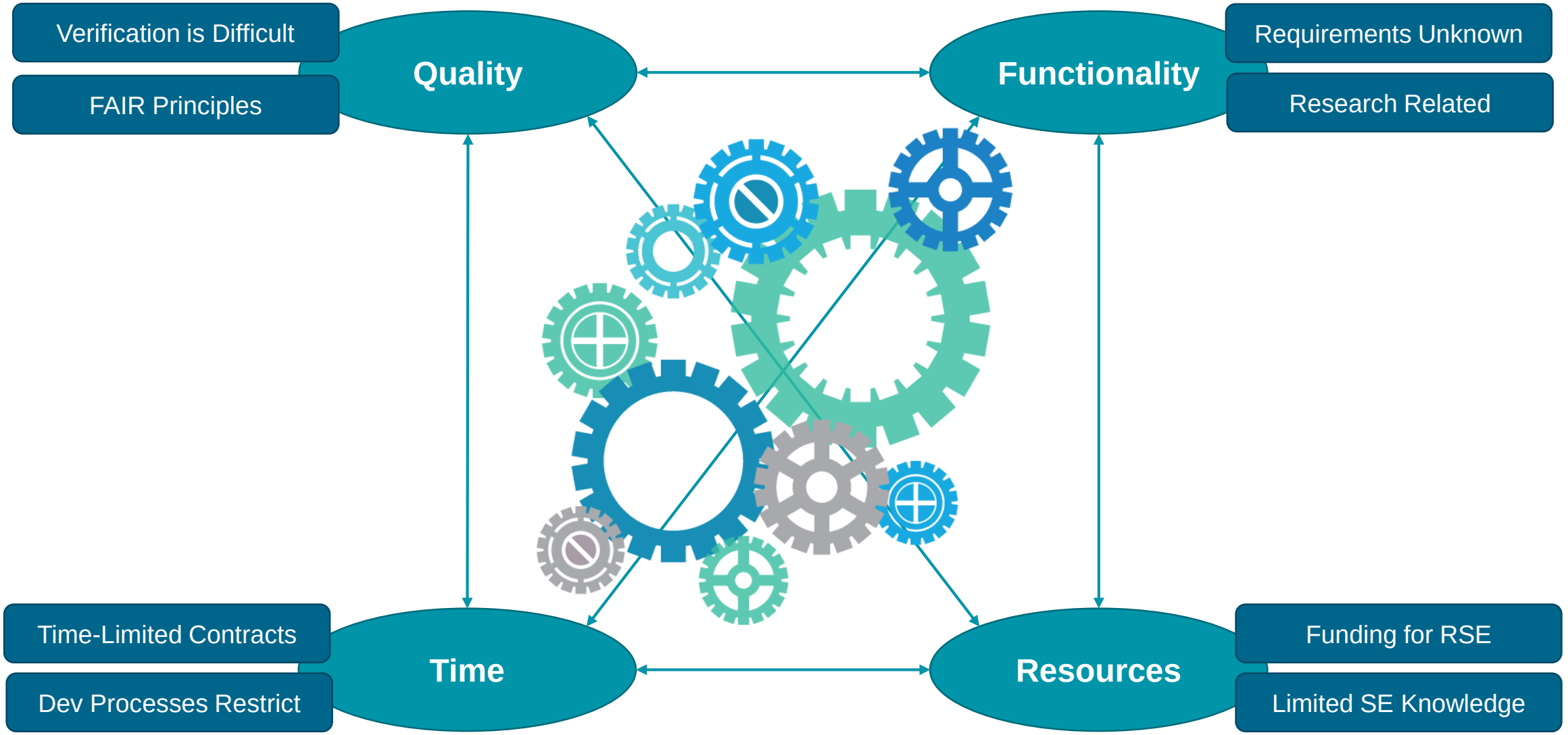
Time

Trade Off in Software Engineering: Specifics of RSE





Trade Off in Software Engineering: Specifics of RSE



Always a Problem:
Joint Goals, Knowledge Gaps, Mutual Understanding

V. Garousi, et al.: Characterizing industry-academia collaborations in software engineering: evidence from 101 projects. Empirical Software Engineering, 2019

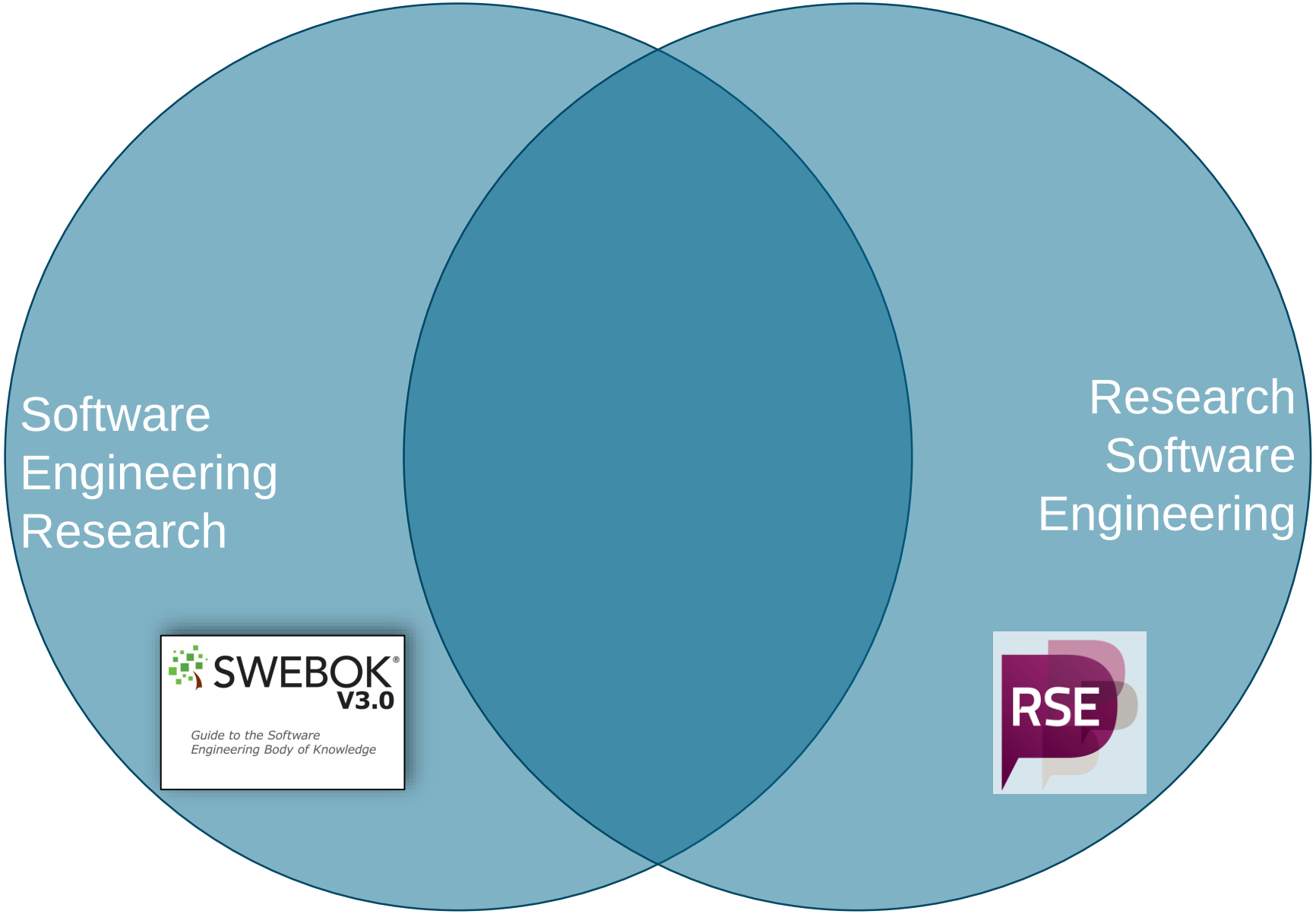
Patterns:

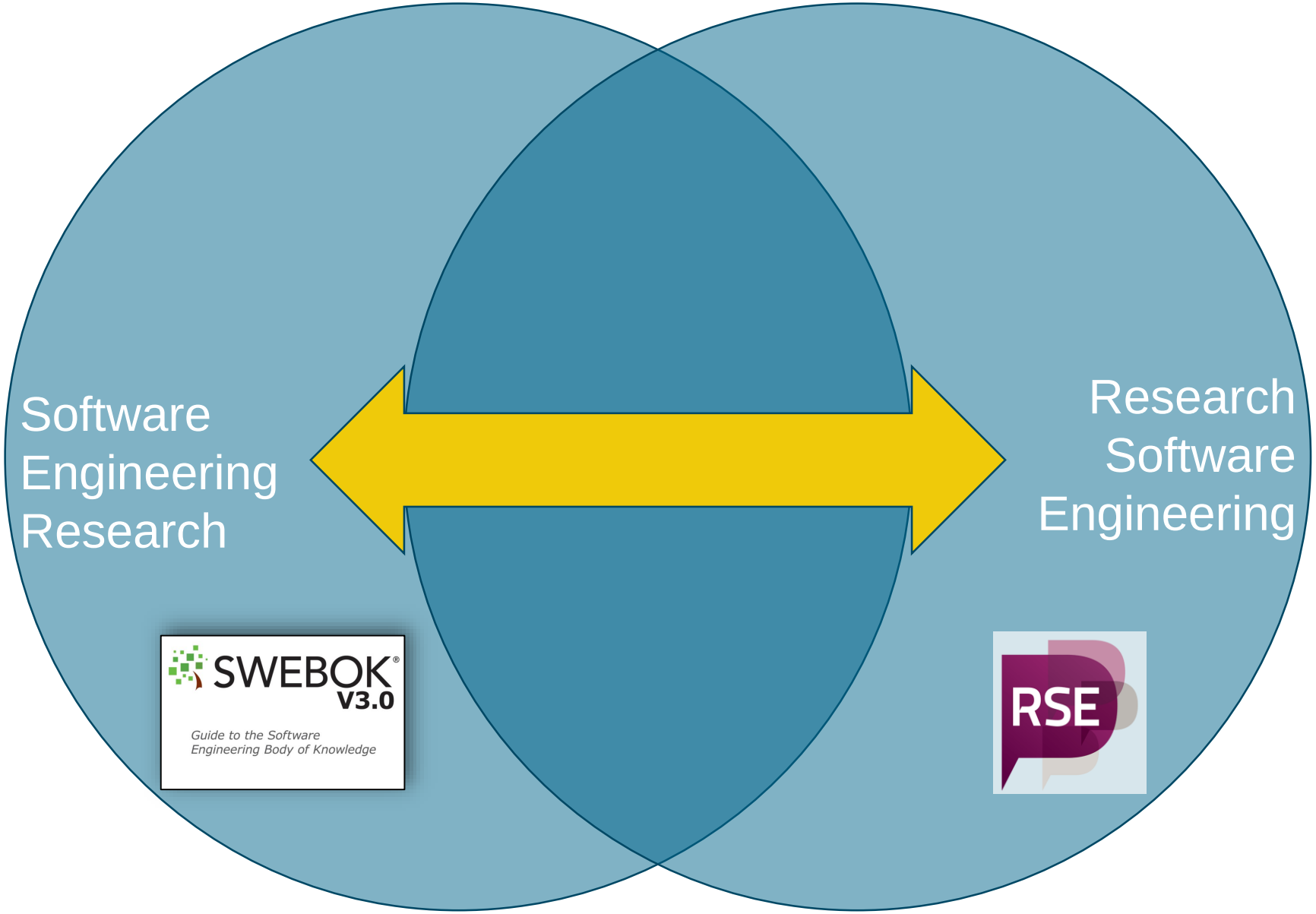
- Proper and active knowledge management (PAKM)
- Ensuring engagement and managing commitment (ENMC)
- Considering and understanding industry's needs, and giving explicit industry benefits (CUIN)
- Having mutual respect, understanding and appreciation (HMRU)
- Being Agile (BA)
- Working in (as) a team and involving the "right" practitioners (WTI)
- Considering and manage risks and limitations (CMRL)
- Researcher's on-site presence and access (ROSP)
- Following a proper research/data collection method (FPRM)
- Managing funding/recruiting/partnerships and contracting privacy (MFRP)
- Understanding the context, constraints and language (UCCL)
- Efficient research project management (ERPM)
- Conducting measurement/ assessment (CMA)
- Testing pilot solutions before using them in industry (TPS)
- Providing tool support for solutions (PTS)

Anti-patterns:

- (Anti-pattern): Following self-centric approach (FSCA)
- Unstructured decision structures (UDS)
- Poor change management (PCM)
- Ignoring project, organizational, or product characteristics (IPOP)

N = 101


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

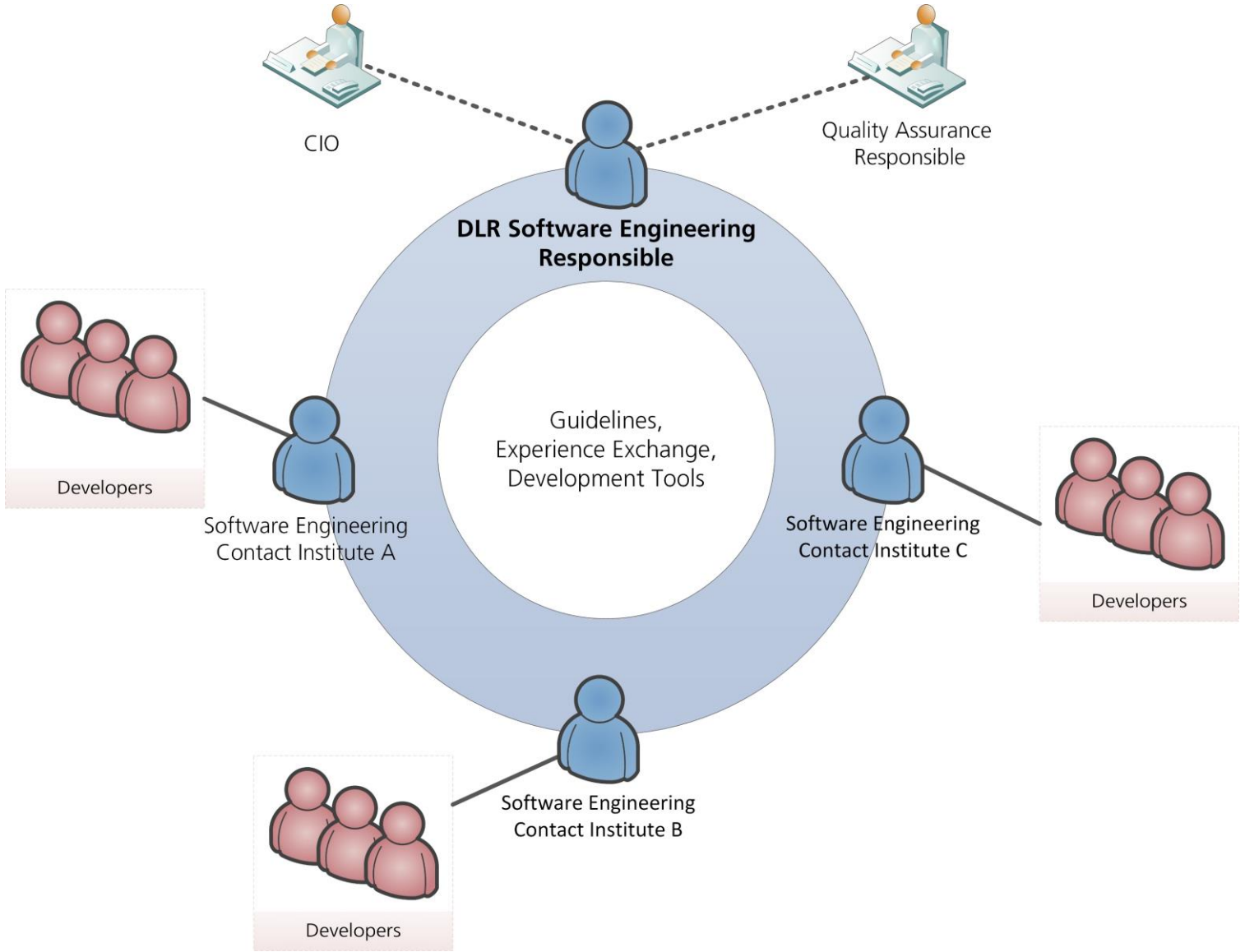
The Most Important Thing: Talking To Each Other

SE 25 SOFTWARE ENGINEERING de RSE CONFERENCE KARLSRUHE 2025

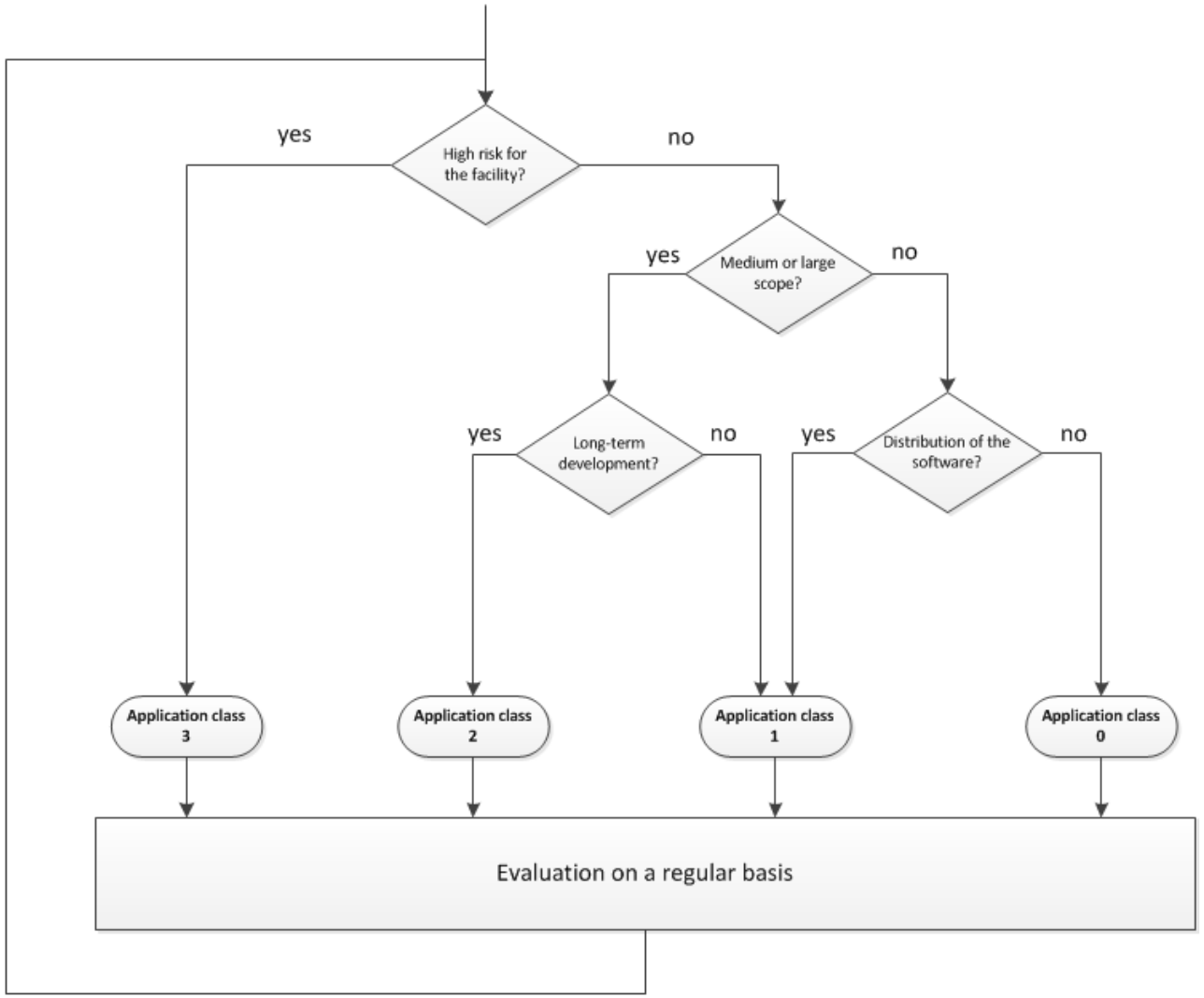
deRSE25 and SE25 Timetables

11:00	**A short introduction to Nextflow: Bring your data science pipelines to the next level** *Marie Latar… et al.*	**Large Language Models(LLMs) in RSE** *Felipe Donoso Aguirre* *Audimax-1, Building 30.95* 11:00 - 12:30	**Workflows for data pipelines** *Joerg Schaarschmidt* *SR A+B, Building 30.95* 11:00 - 12:30	**Research Software working on Medical Data** *Frank Loeffler* *Talk Room TBD, Building 30.70* 11:00 - 12:30	**BoFs: SE and Research** *Upper Seminarroom, Building 30.96* 11:00 - 12:30	**SE: Architectures** *Sven Peldszus* *Audimax-2, Building 30.95* 11:00 - 12:30	**SE: Reproducibil…** *Matthias Tichy* *Vortragsraum 3. OG, Building 30.51 (Bibliothek)* 11:00 - 12:30
12:00							
14:00	**Aspects of Usability in RSE** *Jan Bernoth et al.* *Lower Seminarroom, Building 30.96* 14:00 - 15:30	**ML-assisted and more general data workflows** *Joerg Schaarschmidt* *Audimax-1, Building 30.95* 14:00 - 15:30	**Research Software Engineering in HPC** *Michele Martone* *SR A+B, Building 30.95* 14:00 - 15:30	**Communities around Research Software** *Inga Ulusoy* *Talk Room TBD, Building 30.70* 14:00 - 15:30	**BoF: Bringing together software engineering researchers and research software engineers (SE4Science @ deRSE25)**	**SE: Testing** *Timo Kehrer* *Audimax-2, Building 30.95* 14:00 - 15:30	**SE: Quality Assurance** *Vortragsraum Bibliothek, Building 30.51* 14:00 - 15:30
15:00							
16:00	**How to co…** *Joa…* *Lower Sem…* 16:00 - 16…	**Community in NFDI** *Stephan Janosch* *Audimax-1, Building 30.95* 16:00 - 17:30	**Open Source Community Building** *Inga Ulusoy* *SR A+B, Building 30.95* 16:00 - 17:30	**Reproduci… and Discovery of Research Software** *Bernadette Fritzsch* *Talk Room …* 16:00 - 17…	**BoFs: Challenges for RSEs** *Upper Seminarroo… Building 30.96* 16:00 - 17:30	**SE: Software Evolution** *Wilhelm Hasselbring* *Audimax-2, Building 30.95* 16:00 - 17:30	**SE: Requireme…** *Andreas Vogelsang* *Vortragsraum Bibliothek, Building 30.51* 16:00 - 17:30	**SE: Student Research Competition: Presentati…** *Dr Hamideh Hajiabadi* *Seminar ro…* 16:00 - 17…
17:00	**How to ac…** *Nitai…* *Lower Sem…* 16:45 - 17…							

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

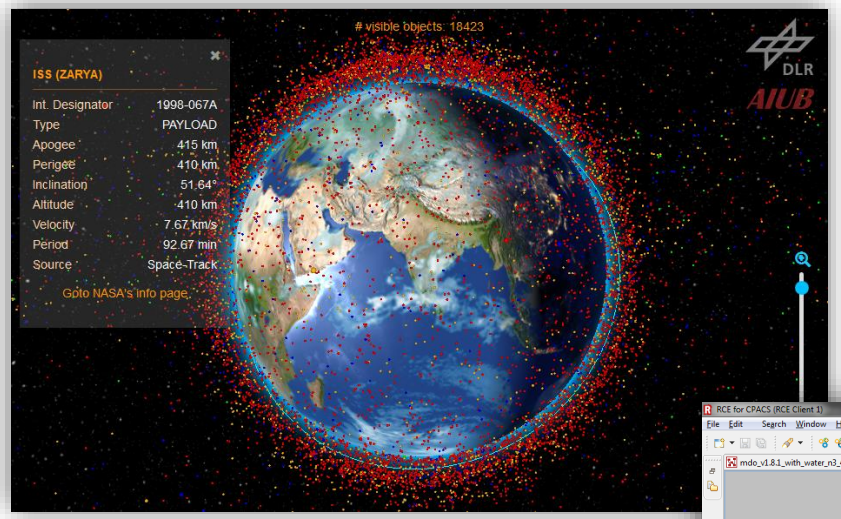
Guidelines „Efficient Development of Research Software“

1. Executive Summary (for Decision Makers)
2. Introduction
3. Guidelines for Software Development
   - Categories of research software
   - Minimum requirements for core competencies
   - Development processes
   - Projekt planning
   - Methodical Foundations
4. Licensing and usage (Legal safeguarding)
5. Support Services
   - Technical (Git, Overleaf, FDM-Tools)
   - Personel through trained RSEs

https://dl.gi.de/handle/20.500.12116/45663 or https://fg-rse.gi.de/

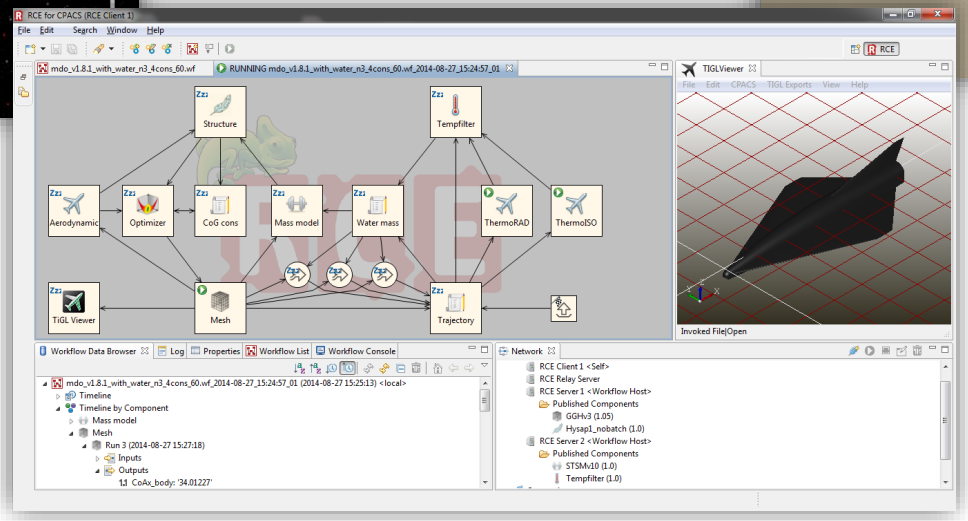
Research software is created during the research process or for a research purpose

Modeling, Simulation, Data Analytics



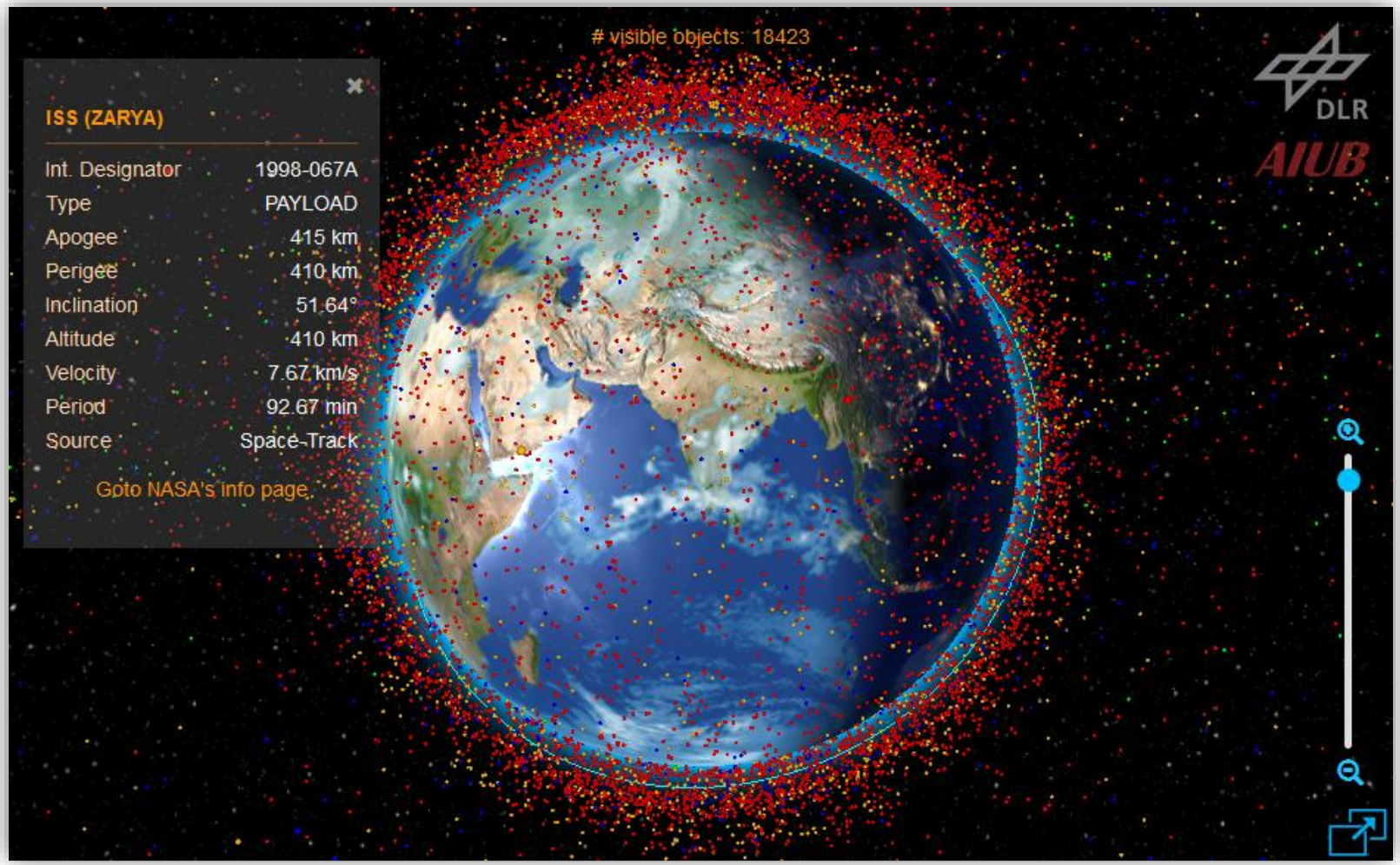
Technology Research Software

Research Infrastructure Software



Hasselbring, W. et al. (2024) Toward Research Software Categories. CoRR abs/2404.14364.

Modeling, Simulation and Data Analytics

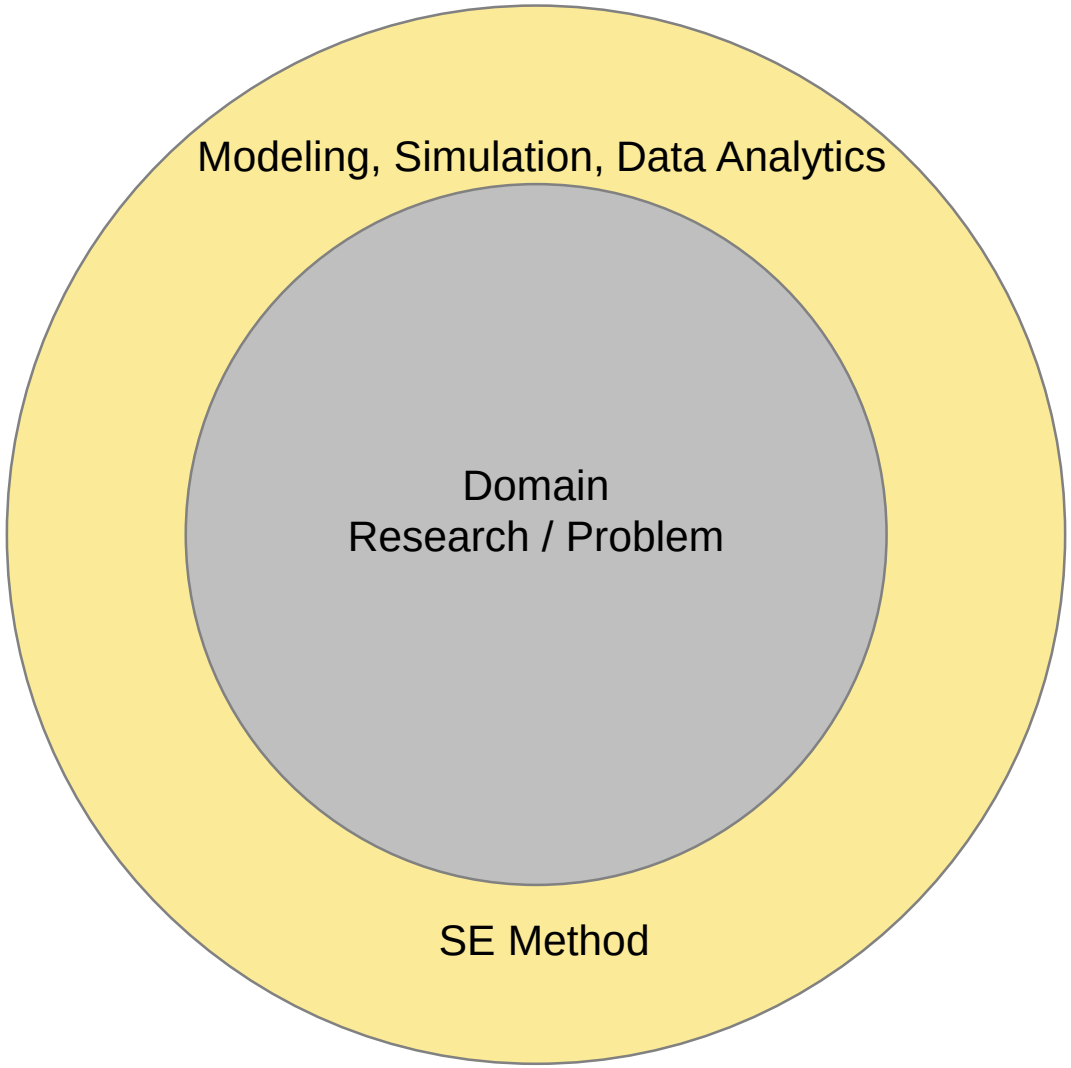


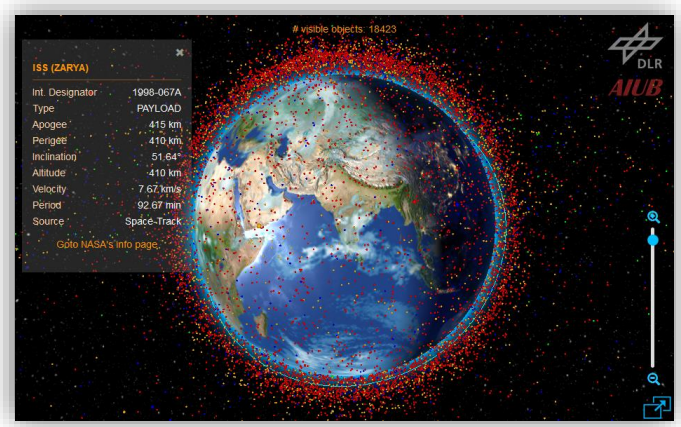
Modeling, Simulation and Data Analytics 1

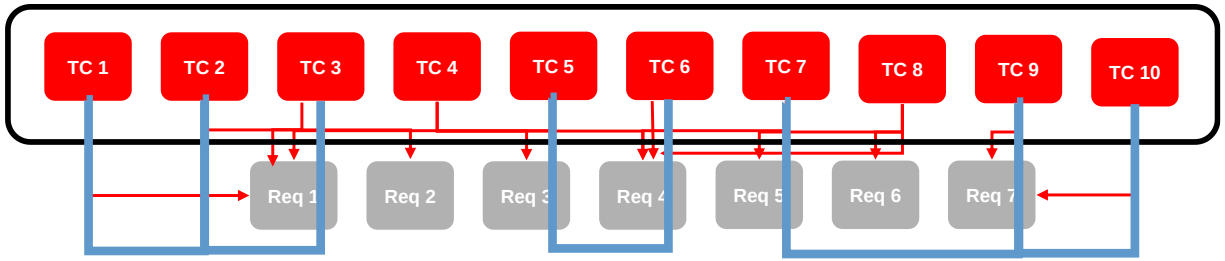
- **Modeling and Simulation** 1.1
- **Data Analytics** 1.2
- **Software Analytics** 1.3
- **Integrative Analysis** 1.4
- **Scientific Visualization** 1.5

BACARDI (Backbone Catalogue of Relational Debris Information) https://bacardi.dlr.de/

Software in Research and SE







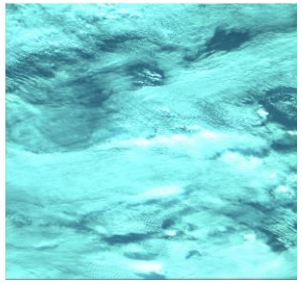
Technology Research Software (1/2)

- **Technology Research Software** (2)
  - **Hardware Related** (2.1)
  - **Software Related** (2.2)
  - **Human Related** (2.3)
  - **Process Related** (2.4)

https://www.dlr.de/en/sc/research-transfer/projects/islandviz

Technology Research Software (2/2)

(a) Good image



(b) Good image

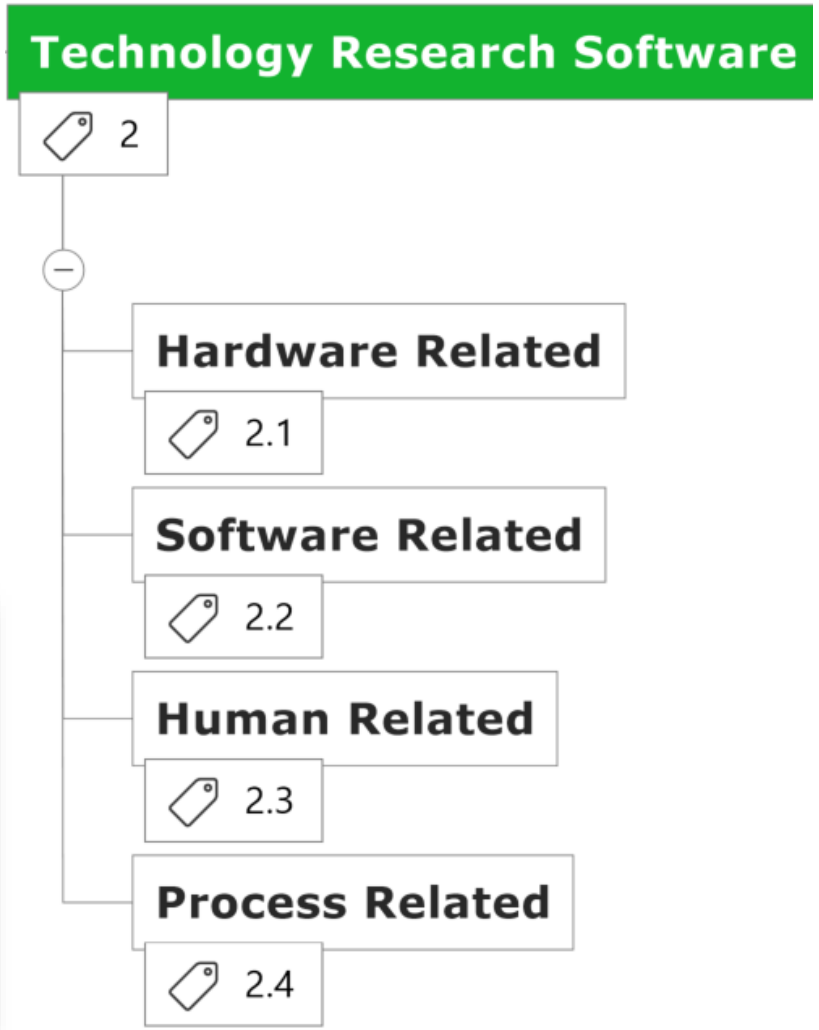
(c) Bad image (high exposure)

(d) Bad image (too dark)

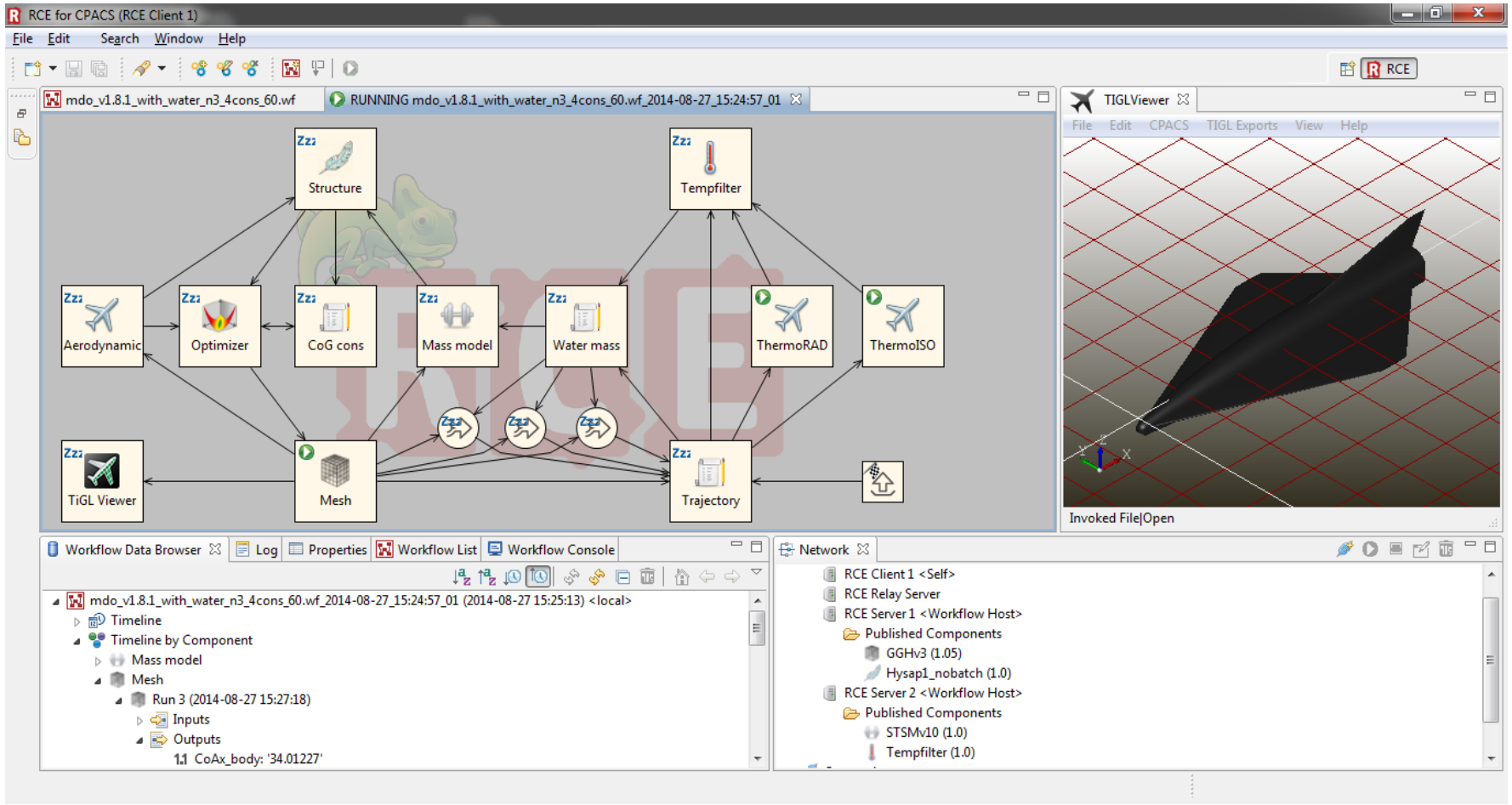
```
LaserCurrentDri   laserStack.stac   laserStackWithC   testSequence.se
device LaserCurrentDriver1_0
  parameter currentA float 0.0 to 0.999984741 default = 0.0
  parameter currentB float 0.0 to 0.999984741 default = 0.0
  parameter globalLaserEnable boolean default = false
  parameter laserADisable 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
```

ScOSA (Scalable On-board Computing for Space Avionics)

https://www.dlr.de/en/sc/research-transfer/projects/scosa-flight-experiment



Research Infrastructure Software (1/2)



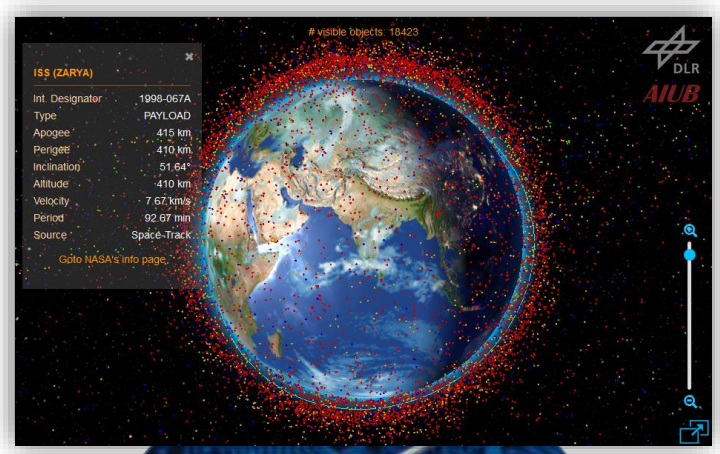
Research Infrastructure Software 3

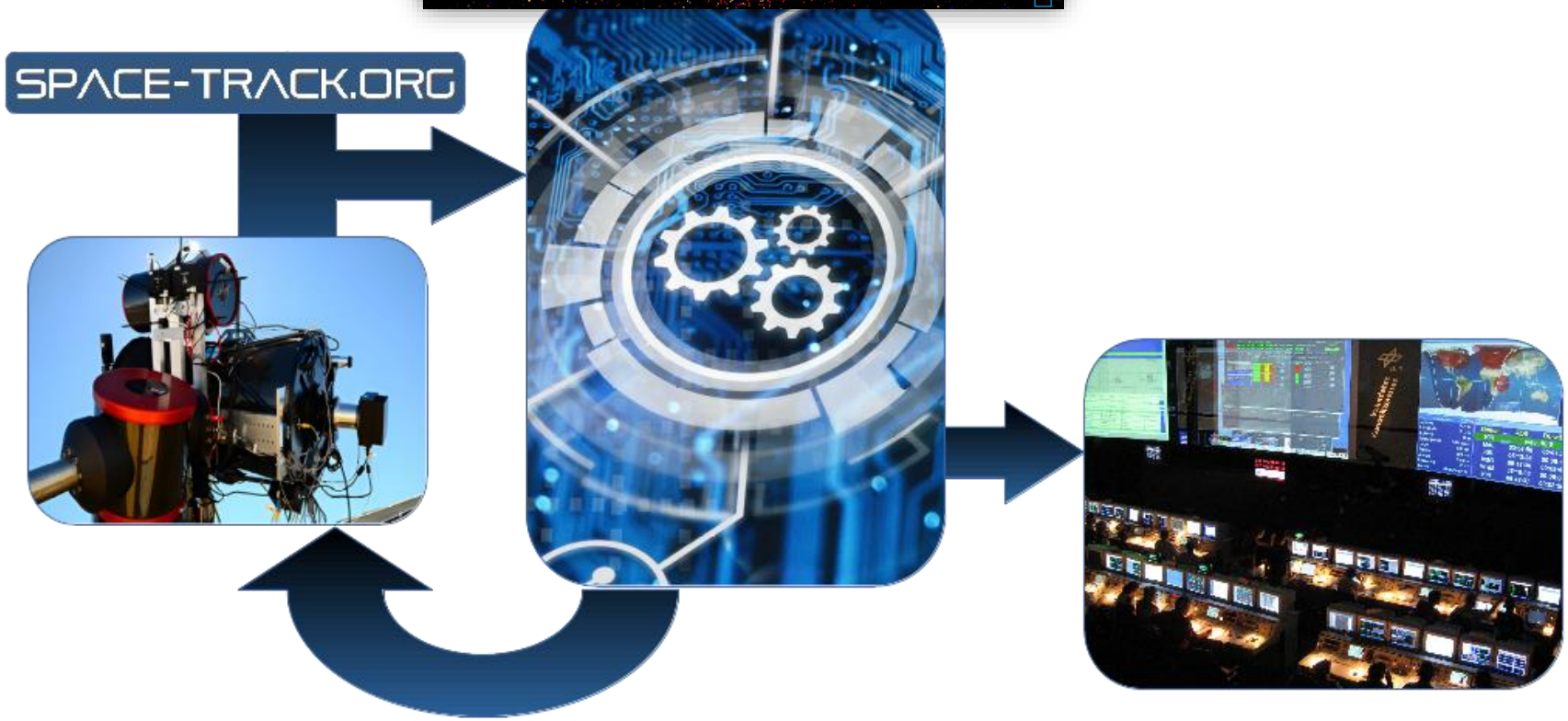
- **Control and Monitoring Software** 3.1
- **Data Collection and Generation** 3.2
- **Pipelines and Tools** 3.3
- **Libraries** 3.4
- **Laboratory Notebooks** 3.5
- **Data Management** 3.6
- **Software Management** 3.7
- **Collaboration and Publication** 3.8

RCE (Remote Component Environment)

https://rcenvironment.de/

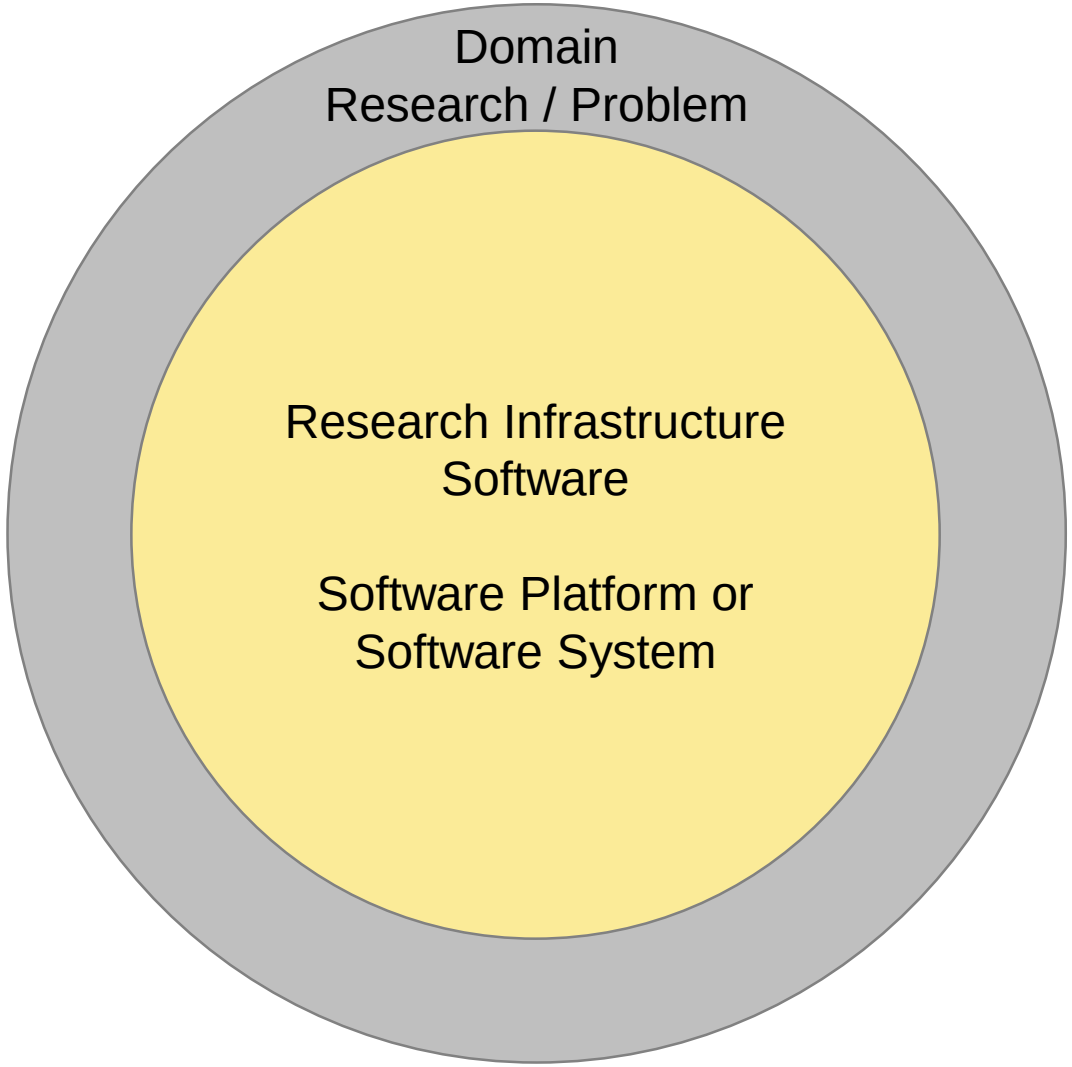
Research Infrastructure Software (2/2)

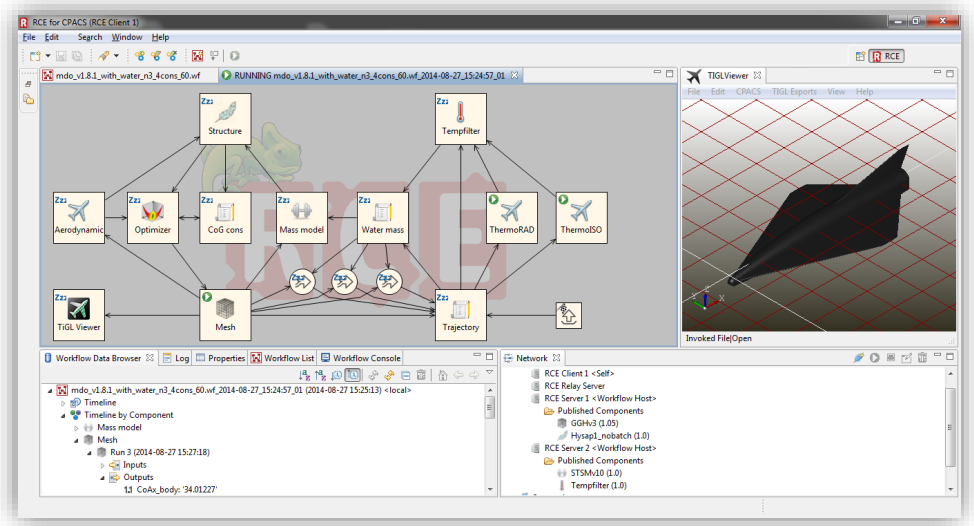




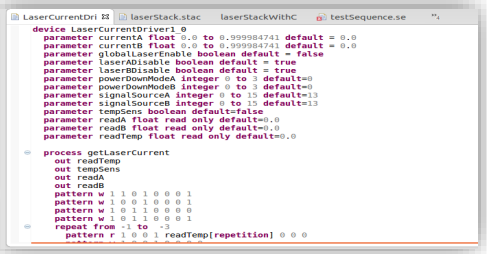
- **Research Infrastructure Software** (3)
  - **Control and Monitoring Software** (3.1)
  - **Data Collection and Generation** (3.2)
  - **Pipelines and Tools** (3.3)
  - **Libraries** (3.4)
  - **Laboratory Notebooks** (3.5)
  - **Data Management** (3.6)
  - **Software Management** (3.7)
  - **Collaboration and Publication** (3.8)

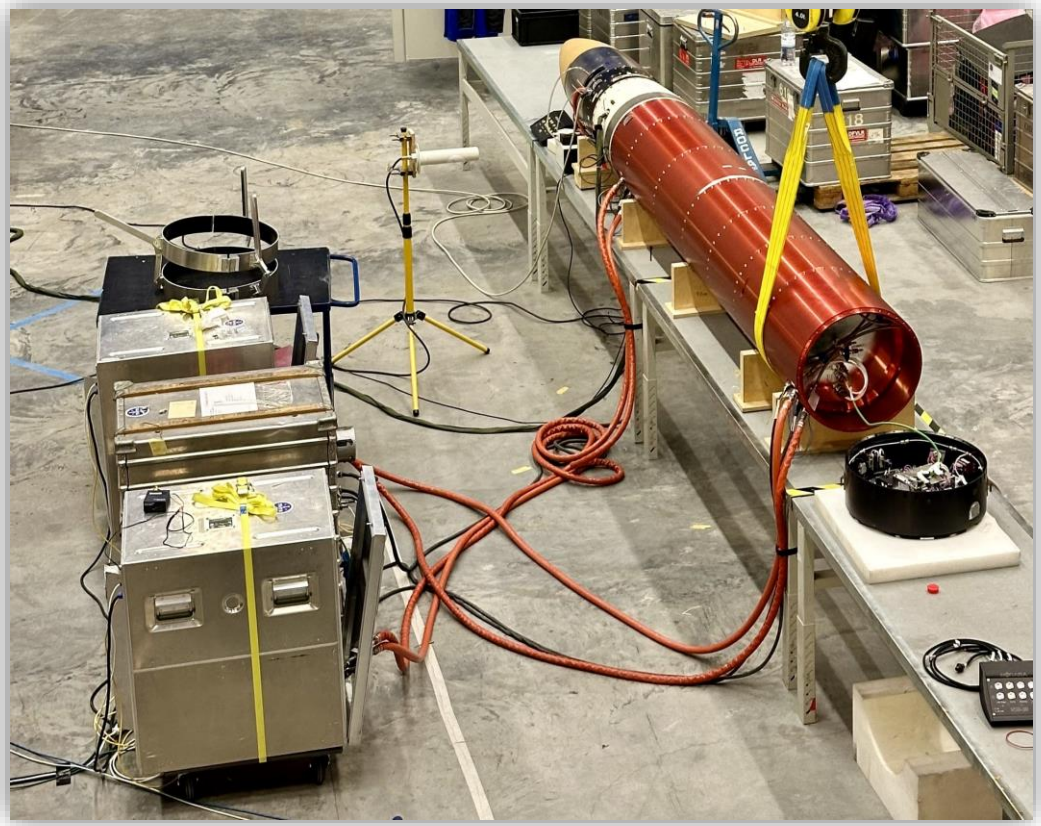
Software in Research and SE





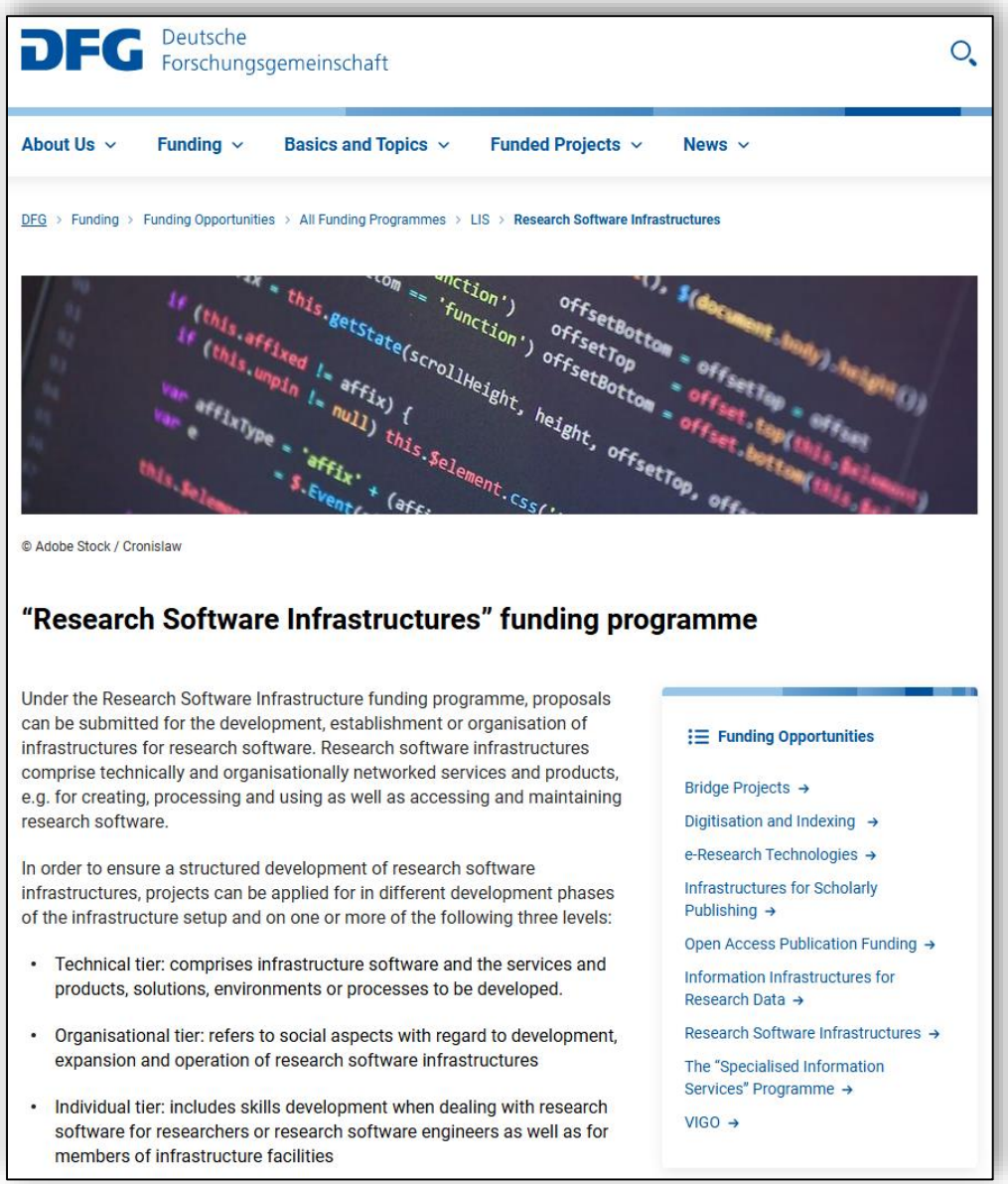
Research Software in Space (TRL 9)





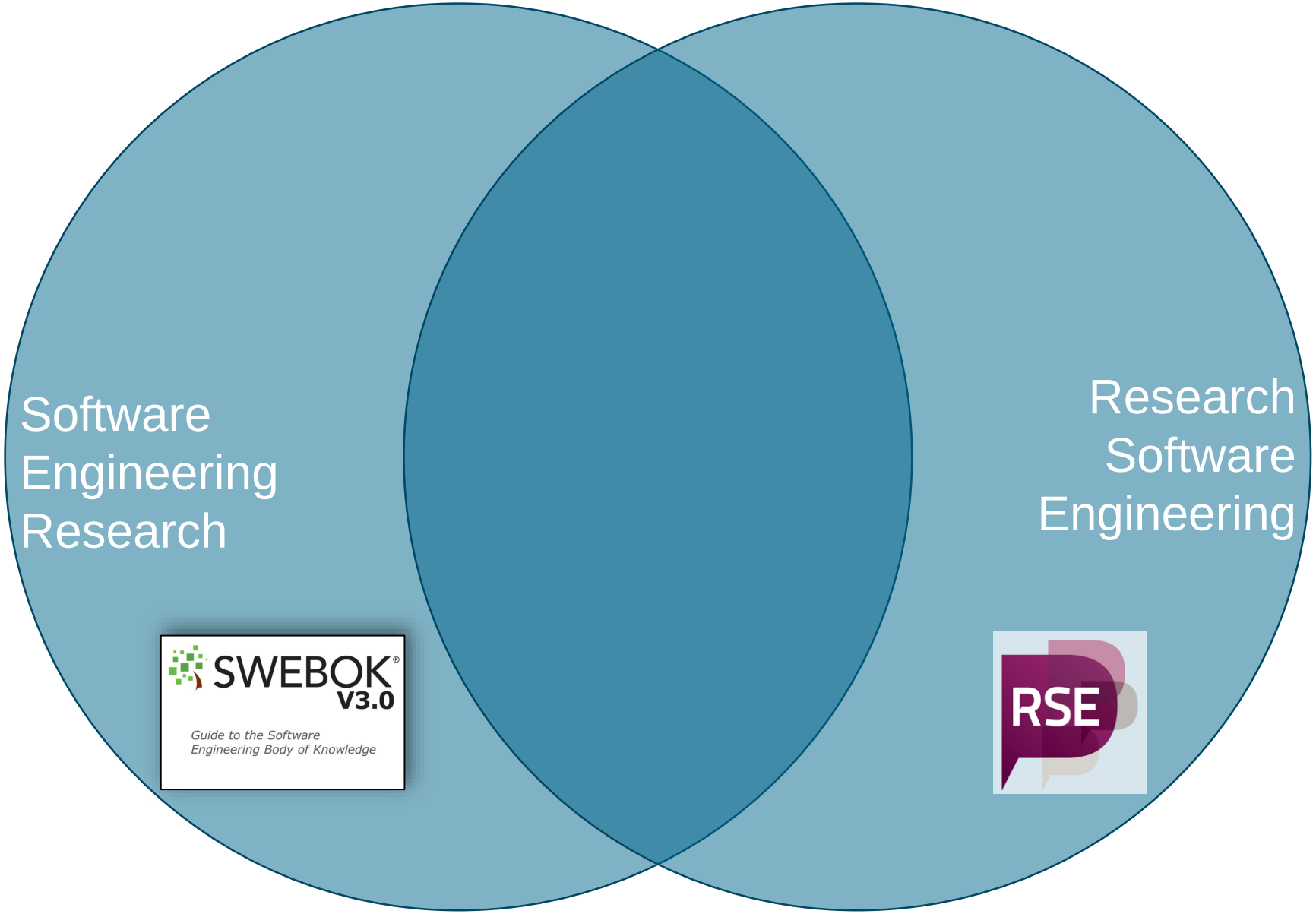
Research Software can also be successful outside research …

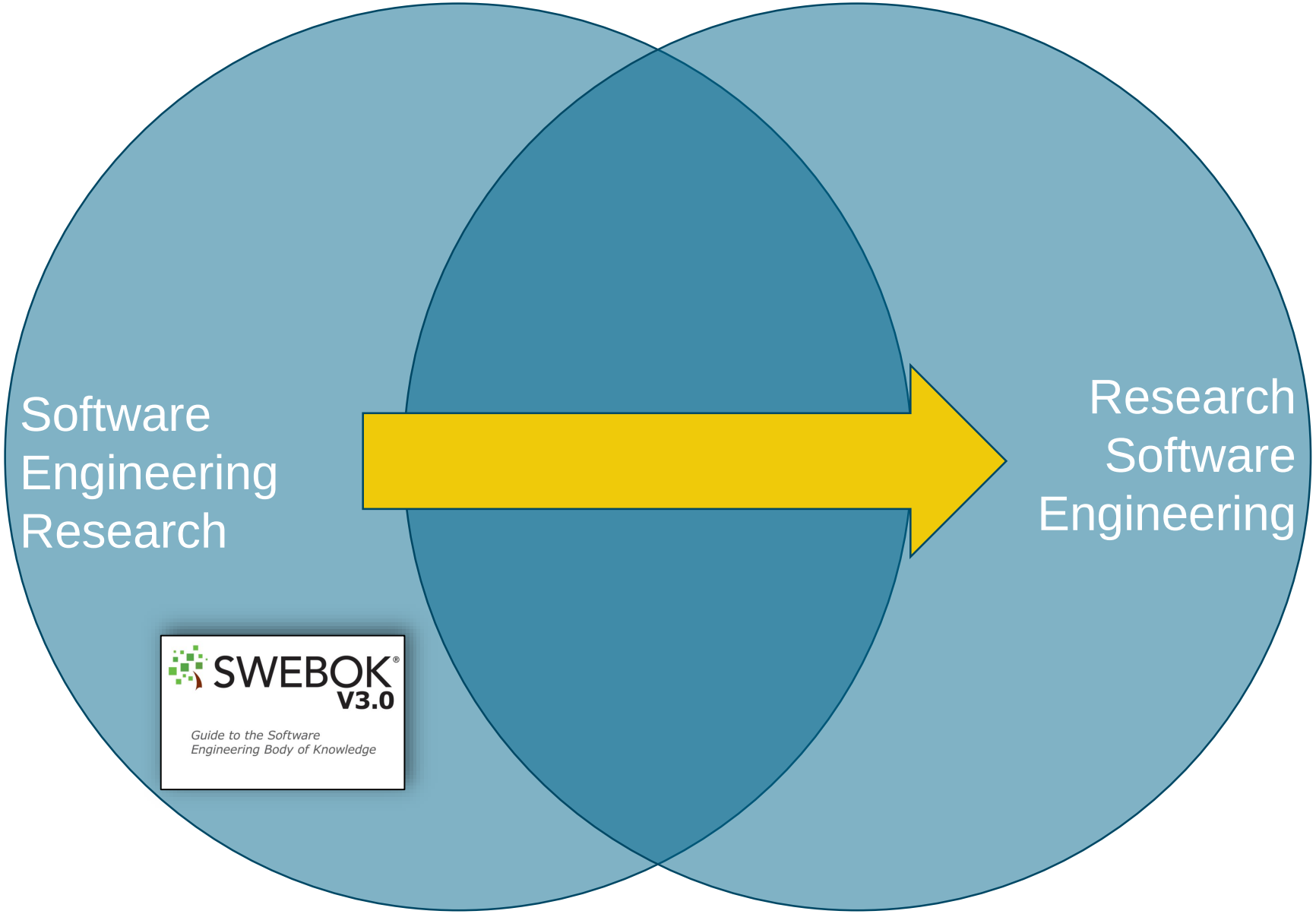
Momentum of the Area Research Software Engineering



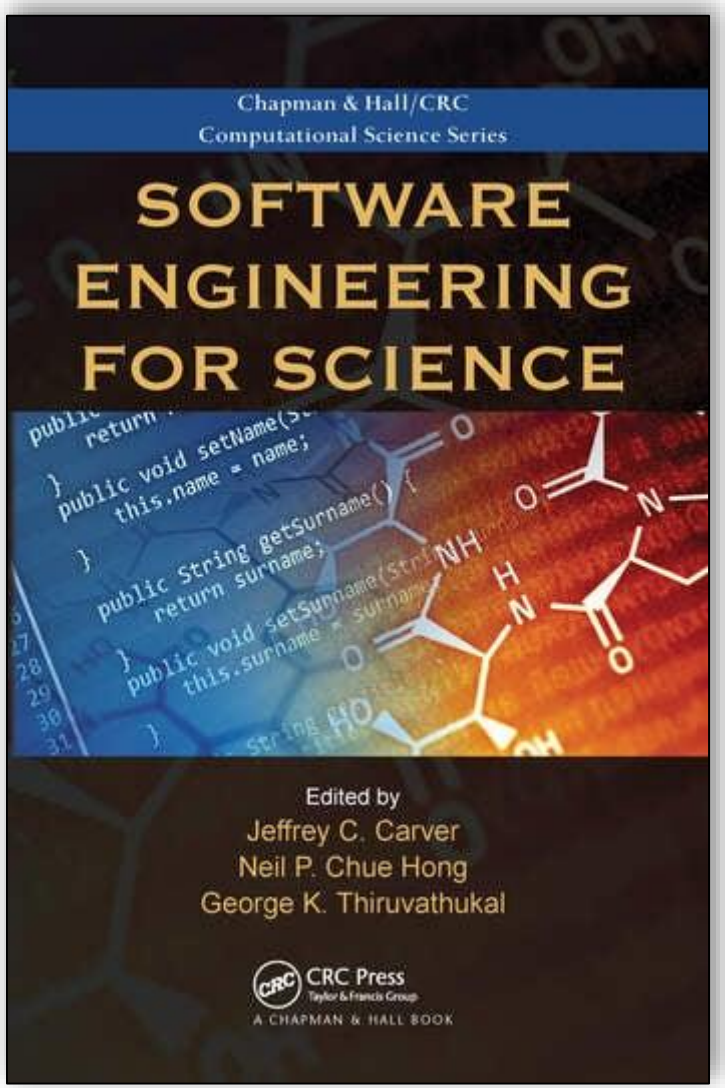
Felderer, M., Goedicke, M., Grunske, L., Hasselbring, W., Lamprecht, A.-L. (2025) Investigating Research Software Engineering: Toward RSE Research. Communications of the ACM.

https://www.dfg.de/en/research-funding/funding-opportunities/programmes/infrastructure/lis/funding-opportunities/research-software-infrastructures


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.)

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

Kanewala et al. (2016) Automated Metamorphic Testing of Scientific Software. Software Engineering for Science. CRC Press

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

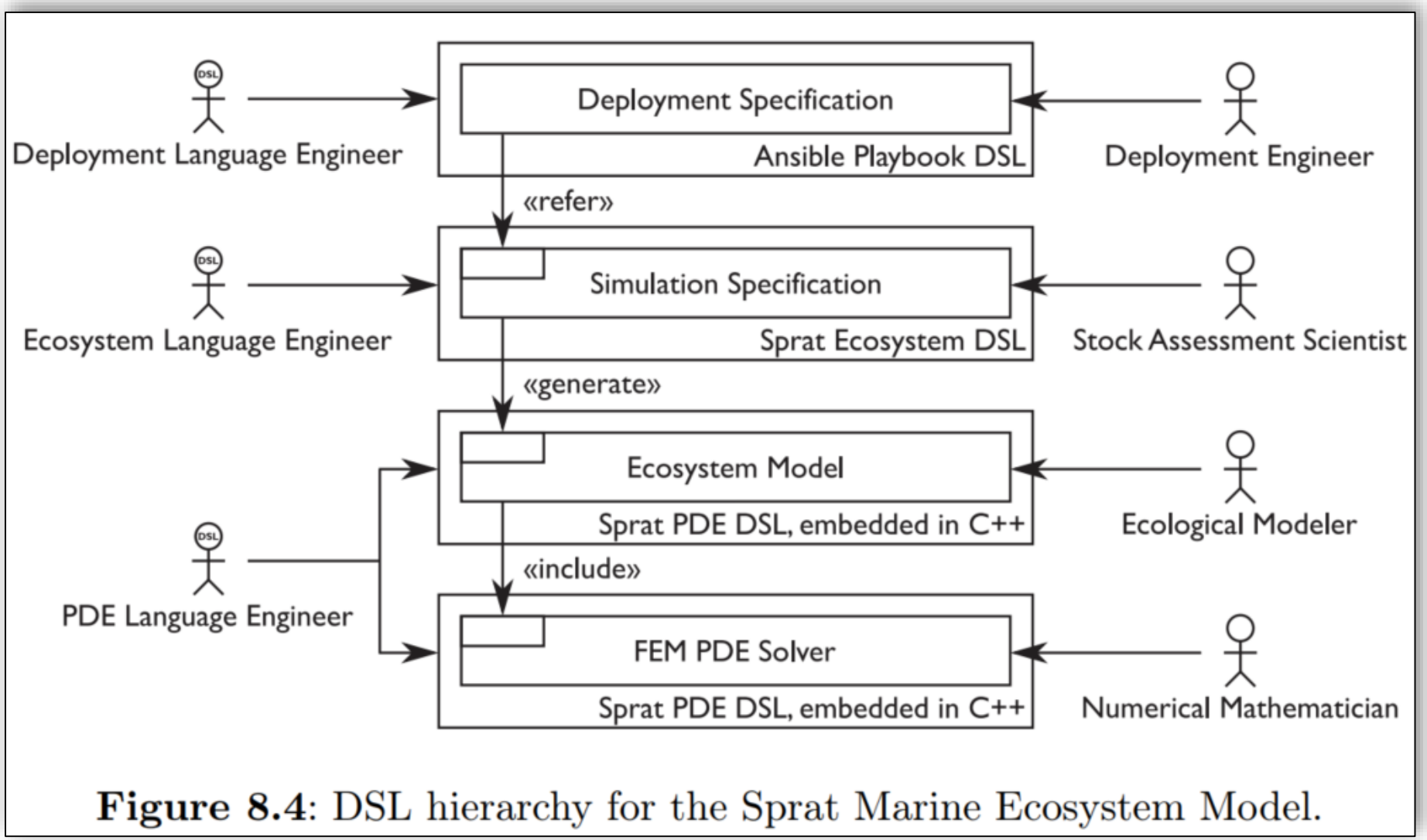


Figure 8.4: DSL hierarchy for the Sprat Marine Ecosystem Model.

Johanson et al. (2016) Evaluating Hierachical Domain-Specific Languages for Computational Science. Software Engineering for Science. CRC Press.

Domain-Specific Languages in Computational Sciences

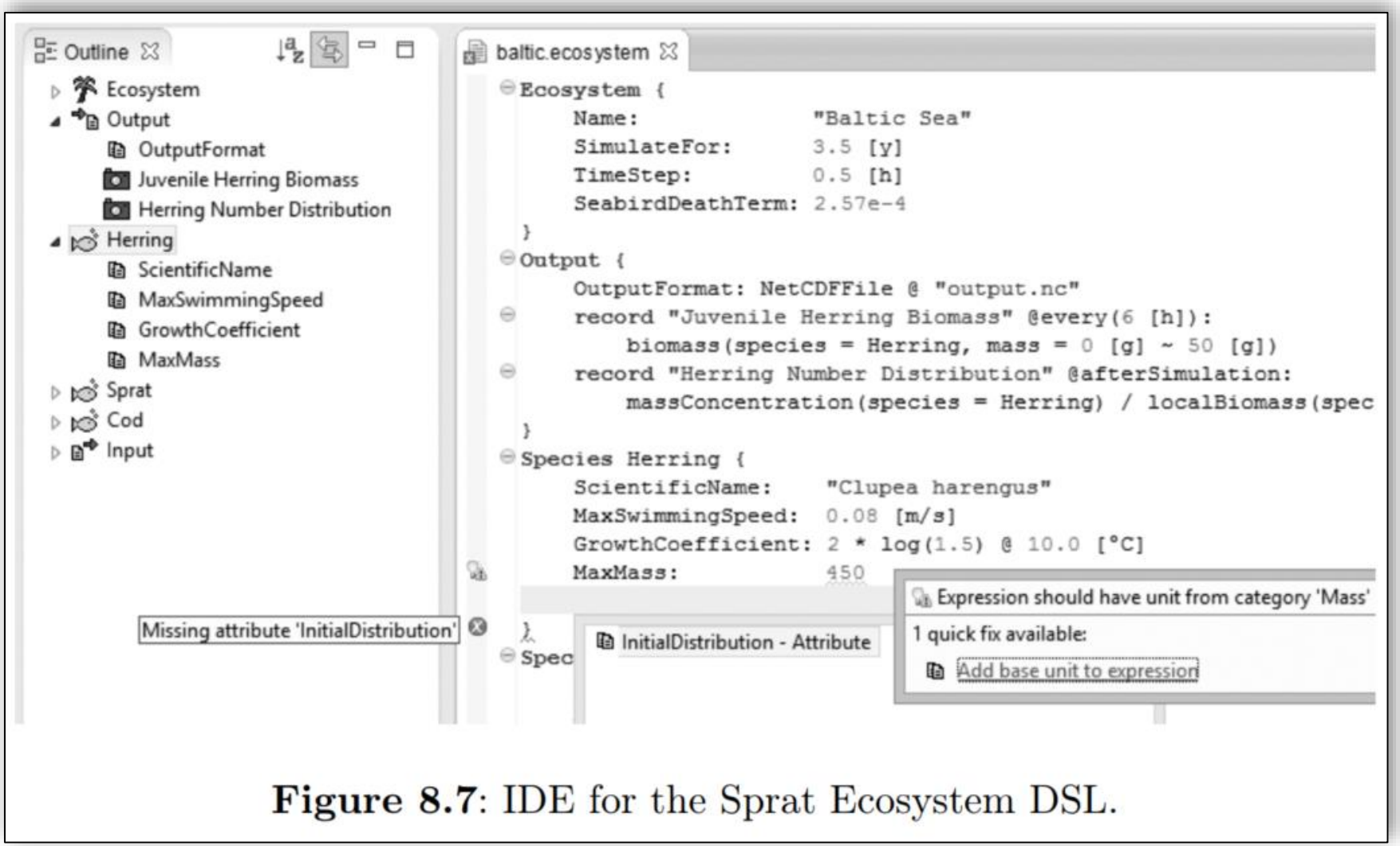


Figure 8.7: IDE for the Sprat Ecosystem DSL.

Many other relevant topics …

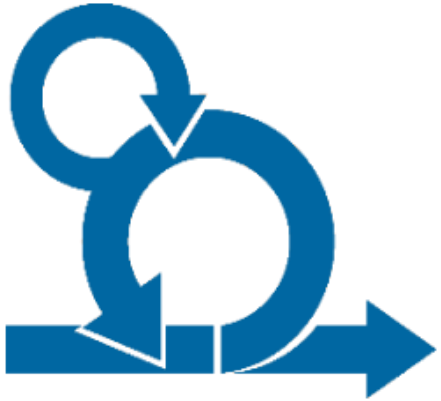
Agile Software Development

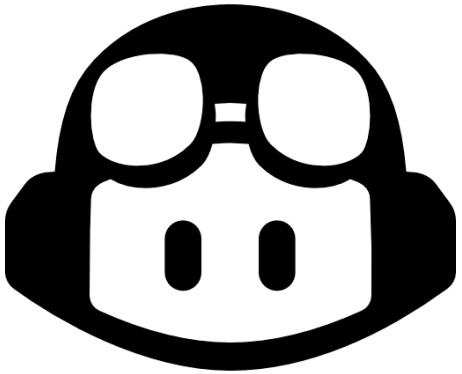
DevOps Practices

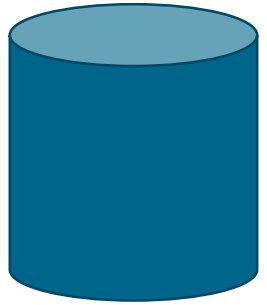
Software Security

GenAI for Software Development

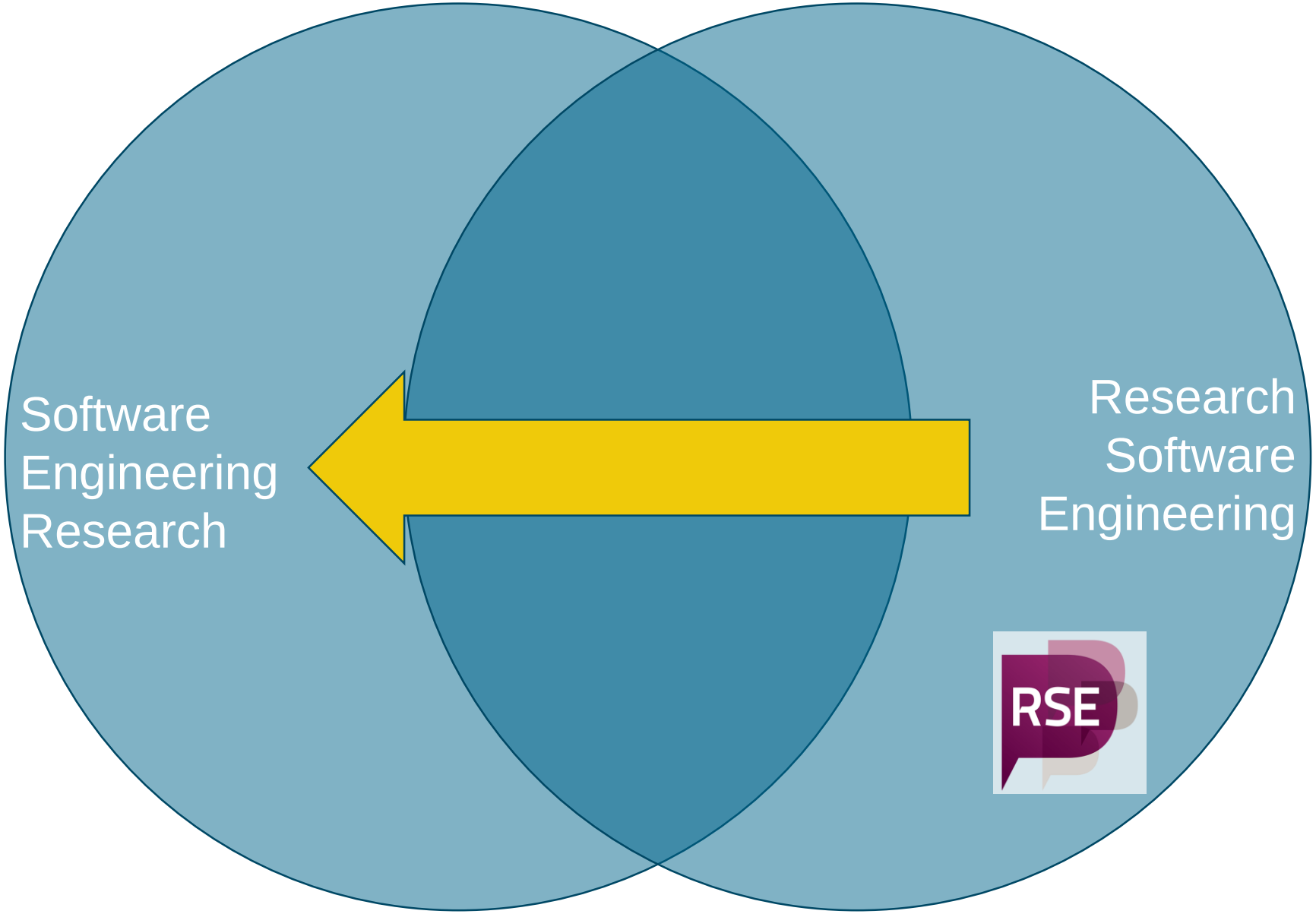
Repository Mining







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


Software
Engineering
Research
Research
Software
Engineering
RSE

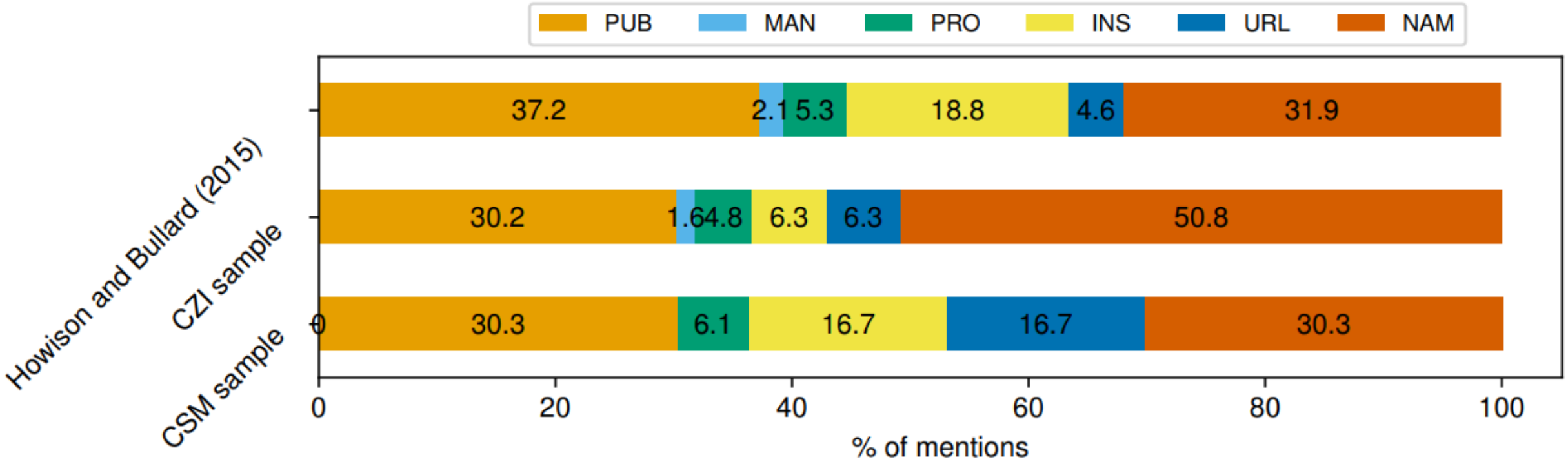
Open Science in Software Engineering

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

Mendez D. et al. (2022) Open Science in Software Engineering. Contemporary Empirical Methods in Software Engineering. Springer.

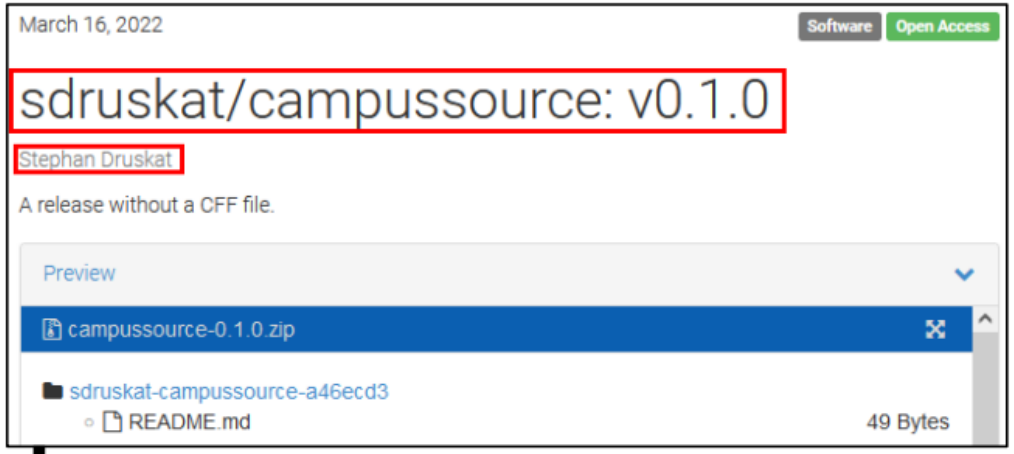
Software is currently mentioned, not cited …



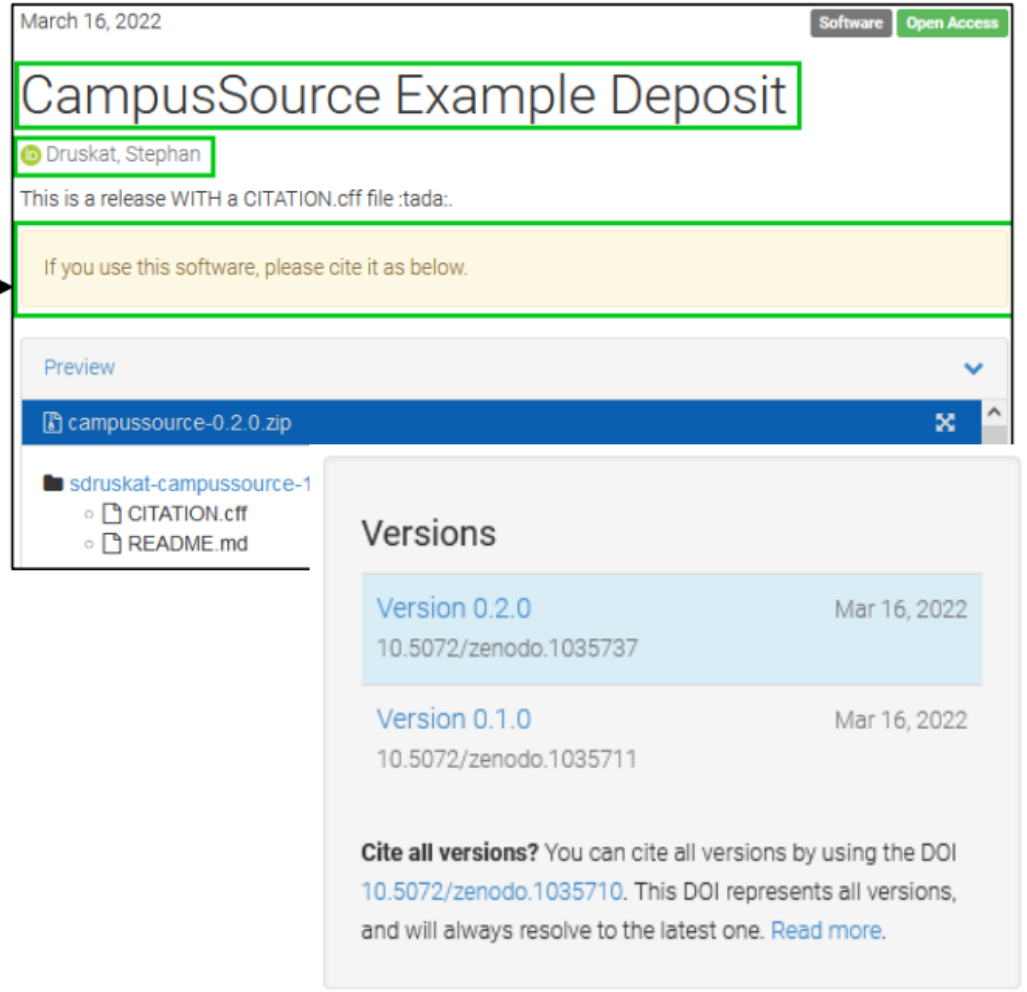
PUB: cites publication
PRO: cites project name/website
INS: instrument-like
URL: URL in text
NAM: in-text name only

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.

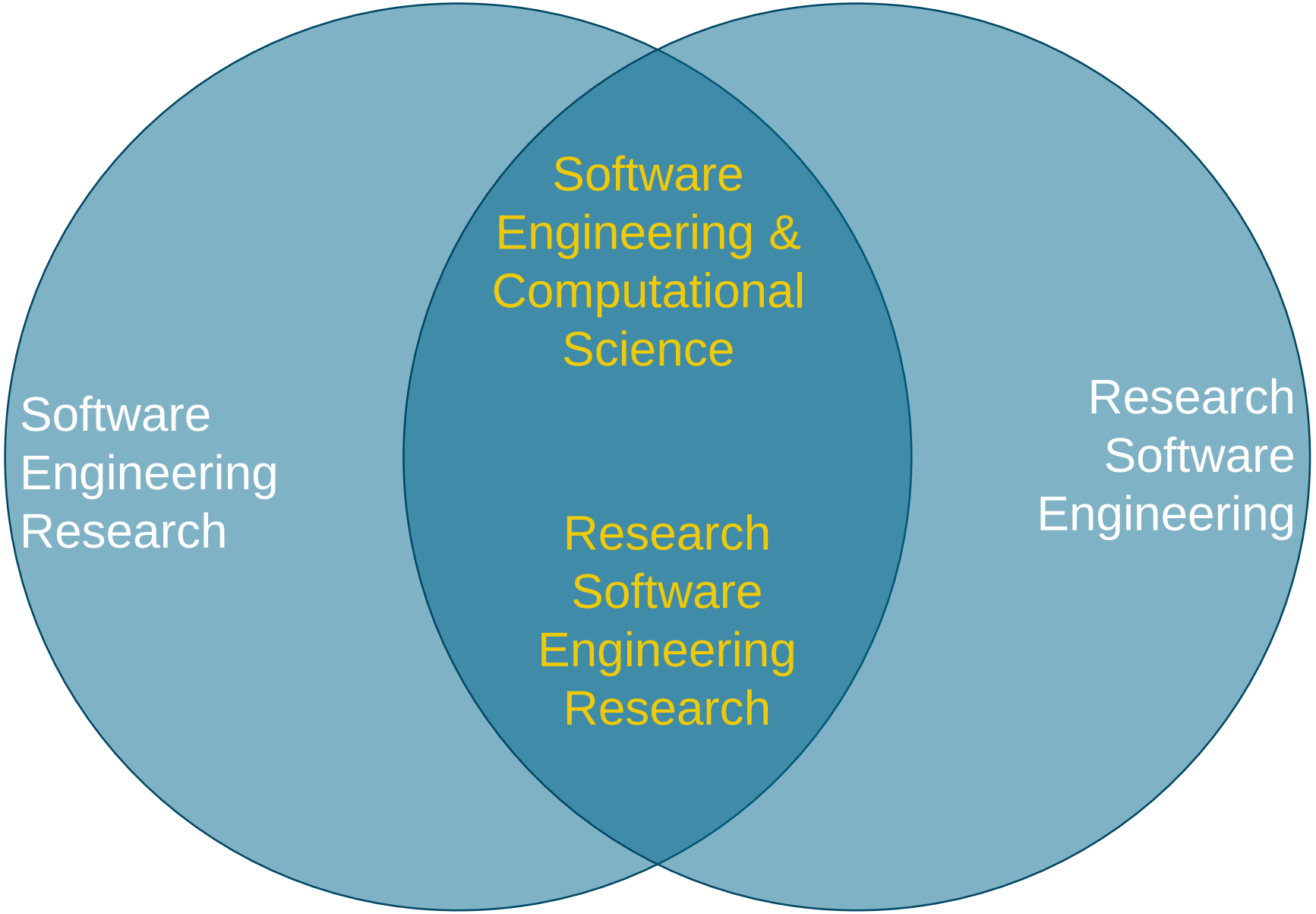
Citation File Format (CFF)

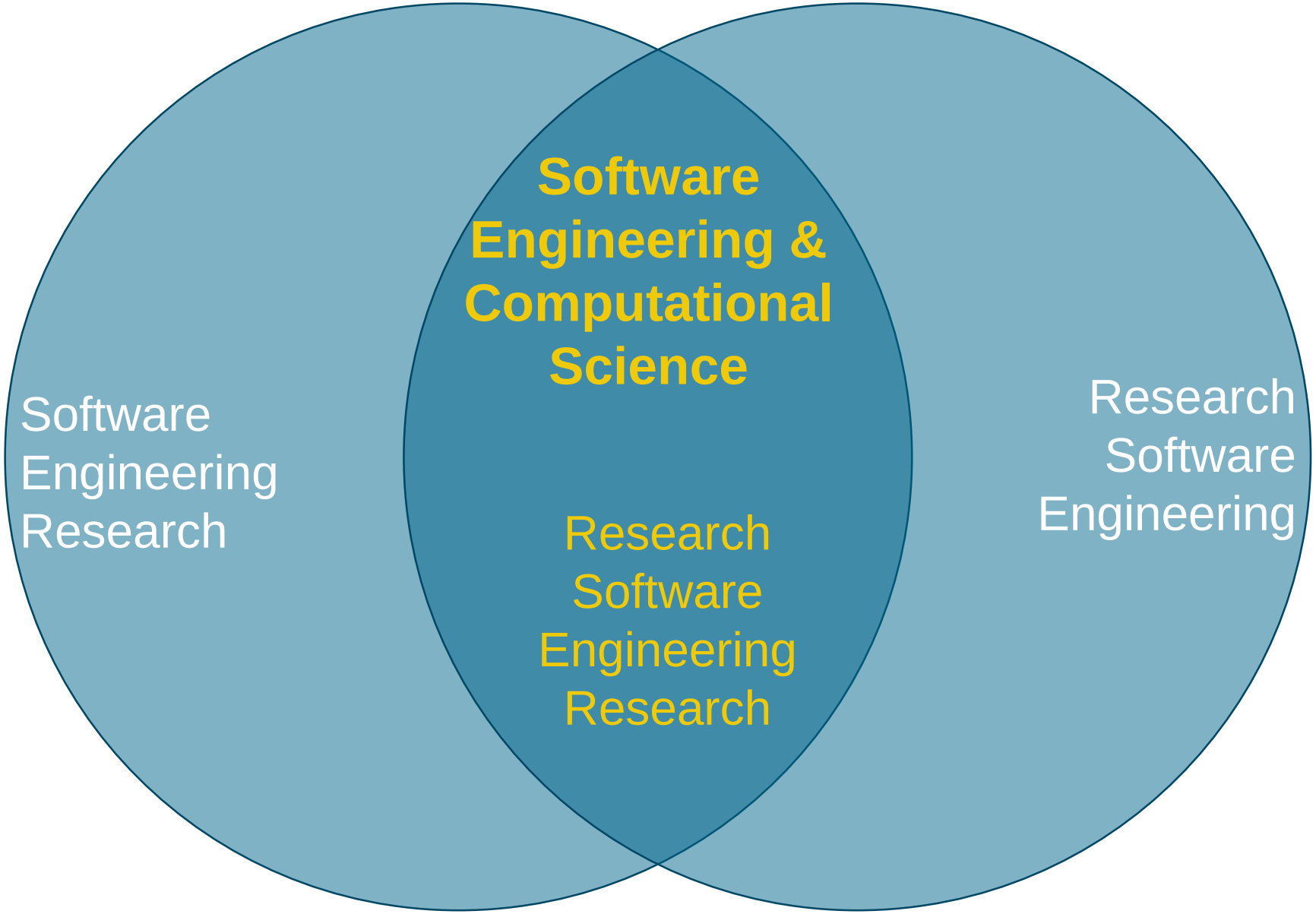



```yaml
cff-version: 1.2.0
message: "If you use this software, please cite it as below."
authors:
- family-names: "Druskat"
  given-names: "Stephan"
  orcid: "https://orcid.org/0000-0003-4925-7248"
title: "CampusSource Example Deposit"
version: 0.2.0
doi: 10.5281/zenodo.1035710
date-released: 2022-03-16
url: "https://www.campussource.de/events/e2203hagen/#Programm"
```



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

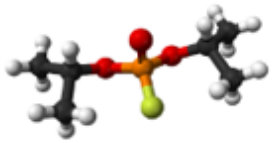

Software Engineering Research
Software Engineering & Computational Science
Research Software Engineering Research
Research Software Engineering


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

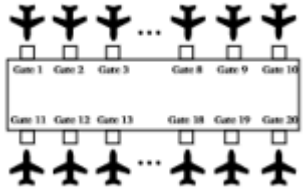
Quantum Computing

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

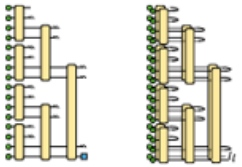
Quantum Simulation



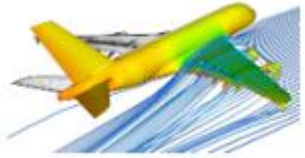
Combinatorial Optimisation



Quantum Enhanced Machine Learning

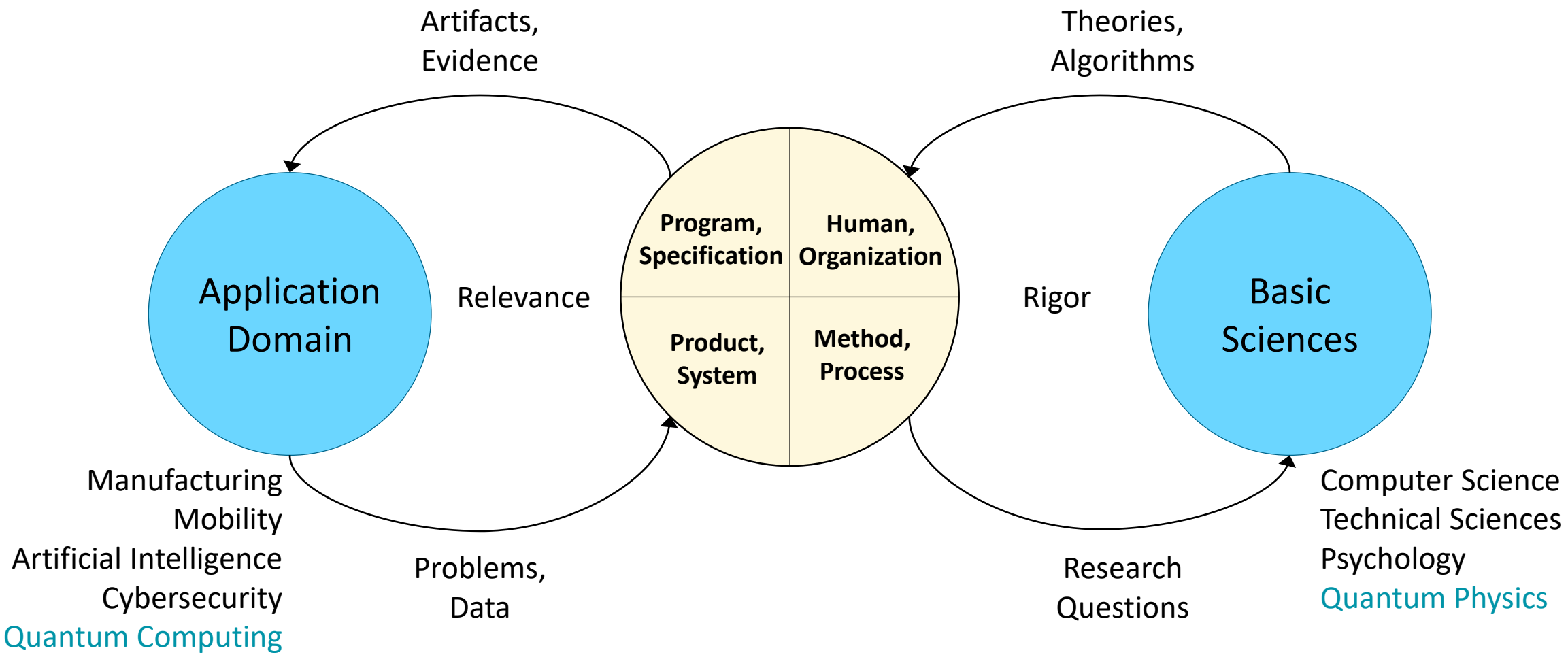


Classical Simulation



Required Error Correction

SE Research Process for Quantum Computing



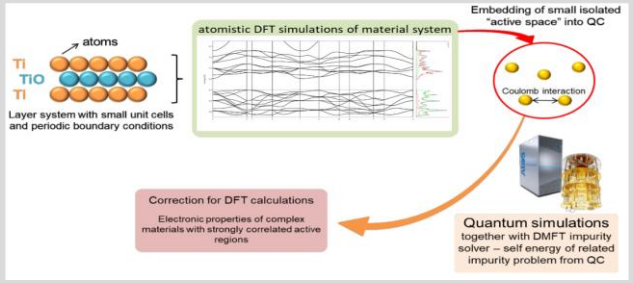
Felderer, M., Reussner, R., Rumpe, B. (2021) Software Engineering und Software Engineering Forschung im Zeitalter der Digitalisierung. Informatik Spektrum.

Project ELEVATE: Use Case Discovery @DLR

ENHANCED PROBLEM SOLVING WITH QUANTUM COMPUTERS

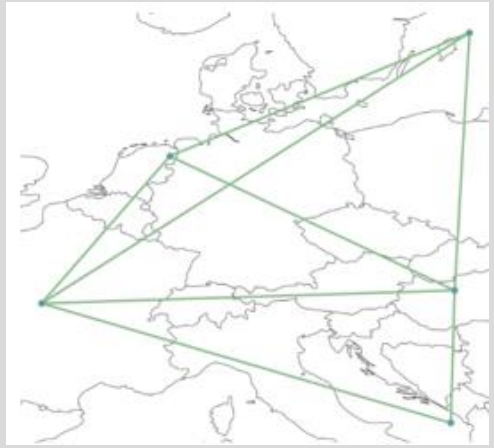
ELEVATE: Investigation of DLR use cases in 42 voucher studies

Atomistic simulation of engineering alloys



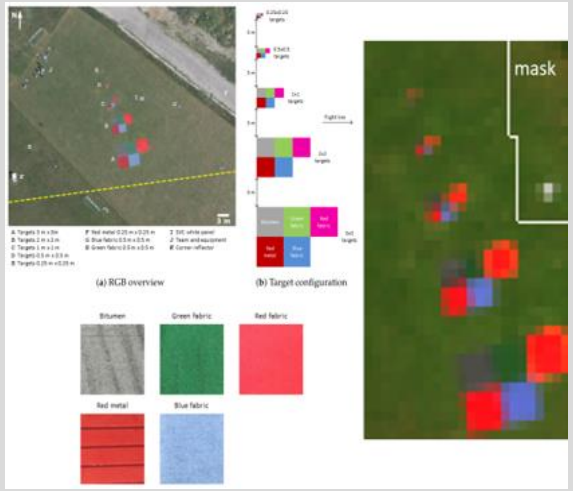
Topic: Improving the prediction of material properties with quantum computing. **(Material Design)**

Transmission Expansion Problem



Topic: Development of a hybrid quantum algorithm for transmission network expansion planning. **(Optimization)**

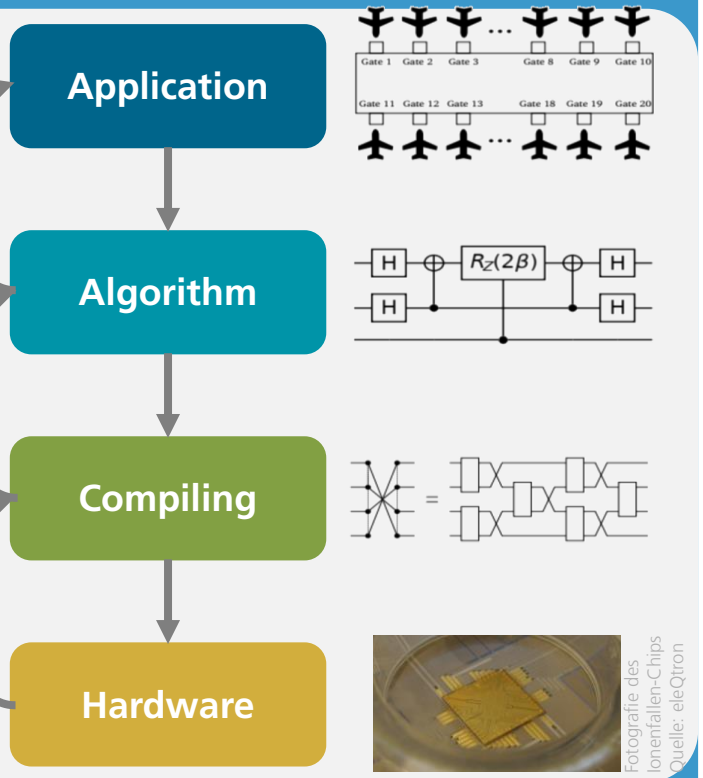
Hyperspectral Data Unmixing



Topic: QML for analysing multimodal spectral data from satellite observations. **(Quantum Machine Learning)**

Key Challenges

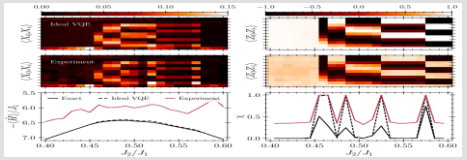
Development



Fotografie des Ionenfallen-Chips
Quelle: eleQtron

How can QC advantages be realised?

Use Case Discovery



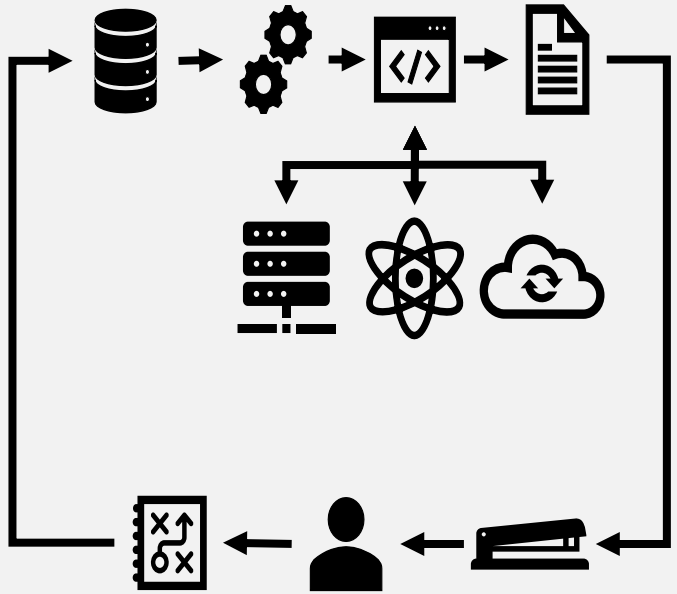
Material Design

Optimization

Machine Learning

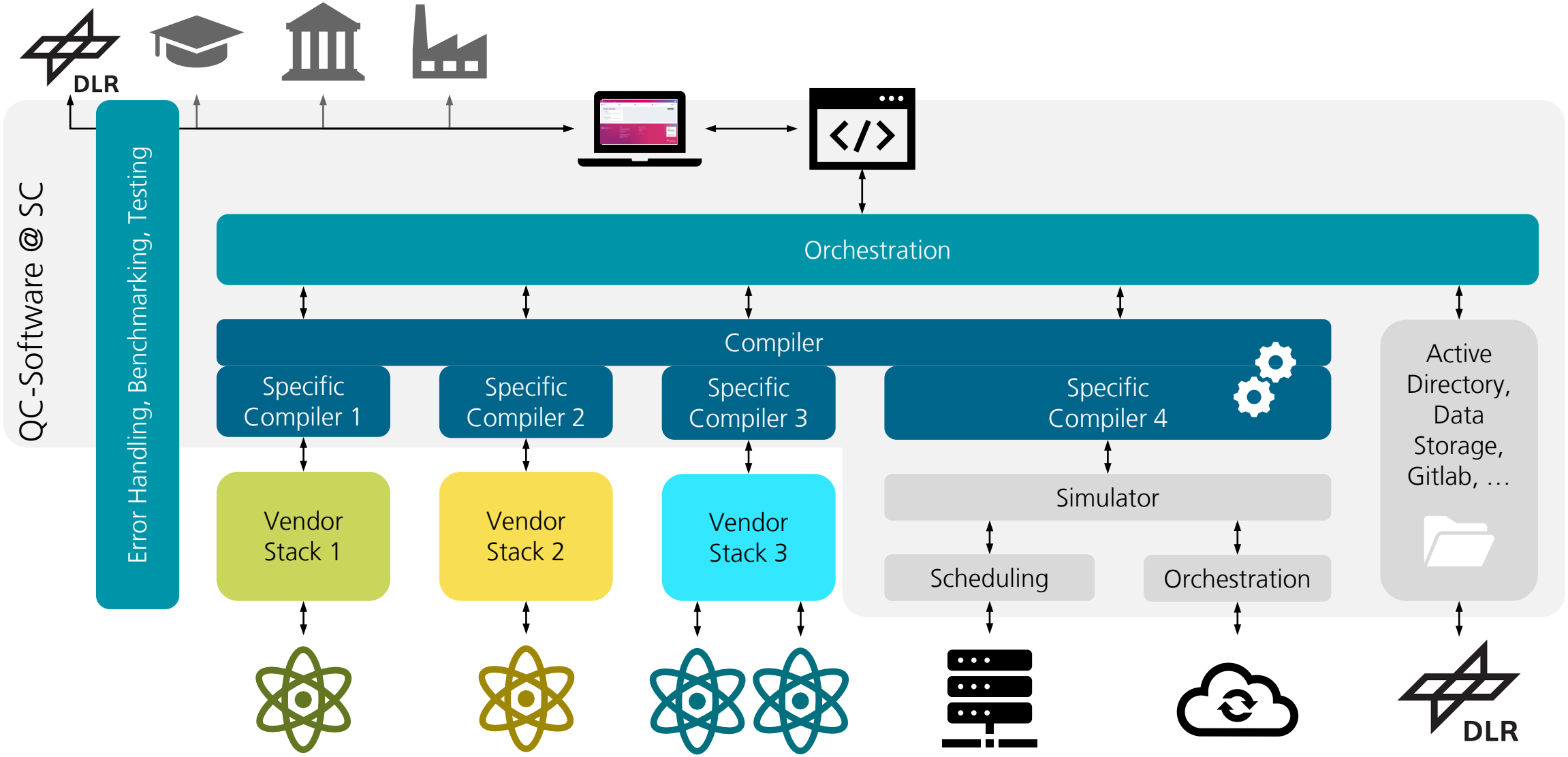
Where does QC offer a realistic benefit?

Integration



How does interaction with quantum computers work?

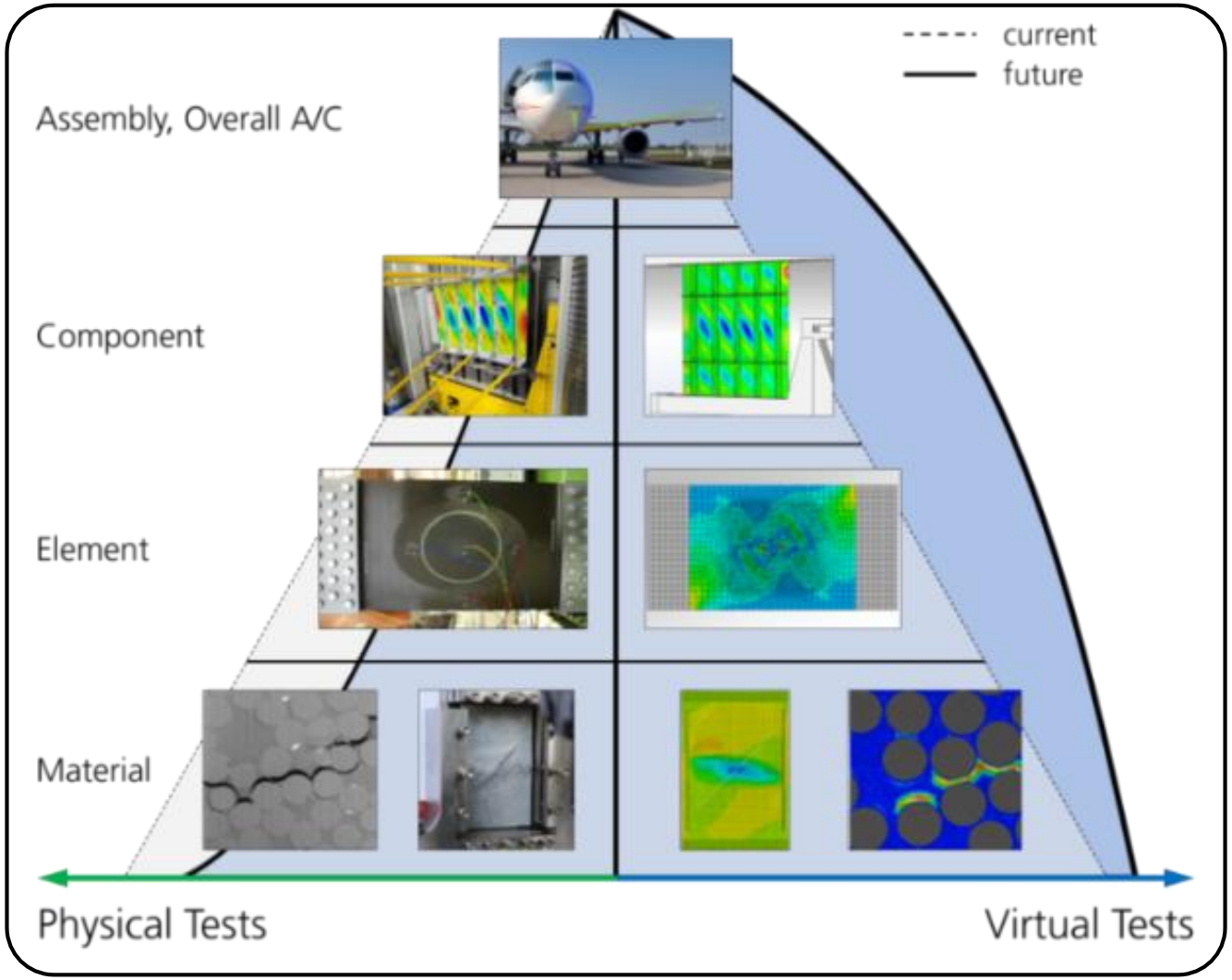
Quantum software as a bridge-builder and research area



Basermann, A., et al. (2024) Quantum Software Ecosystem Design. Quantum Software. Springer.

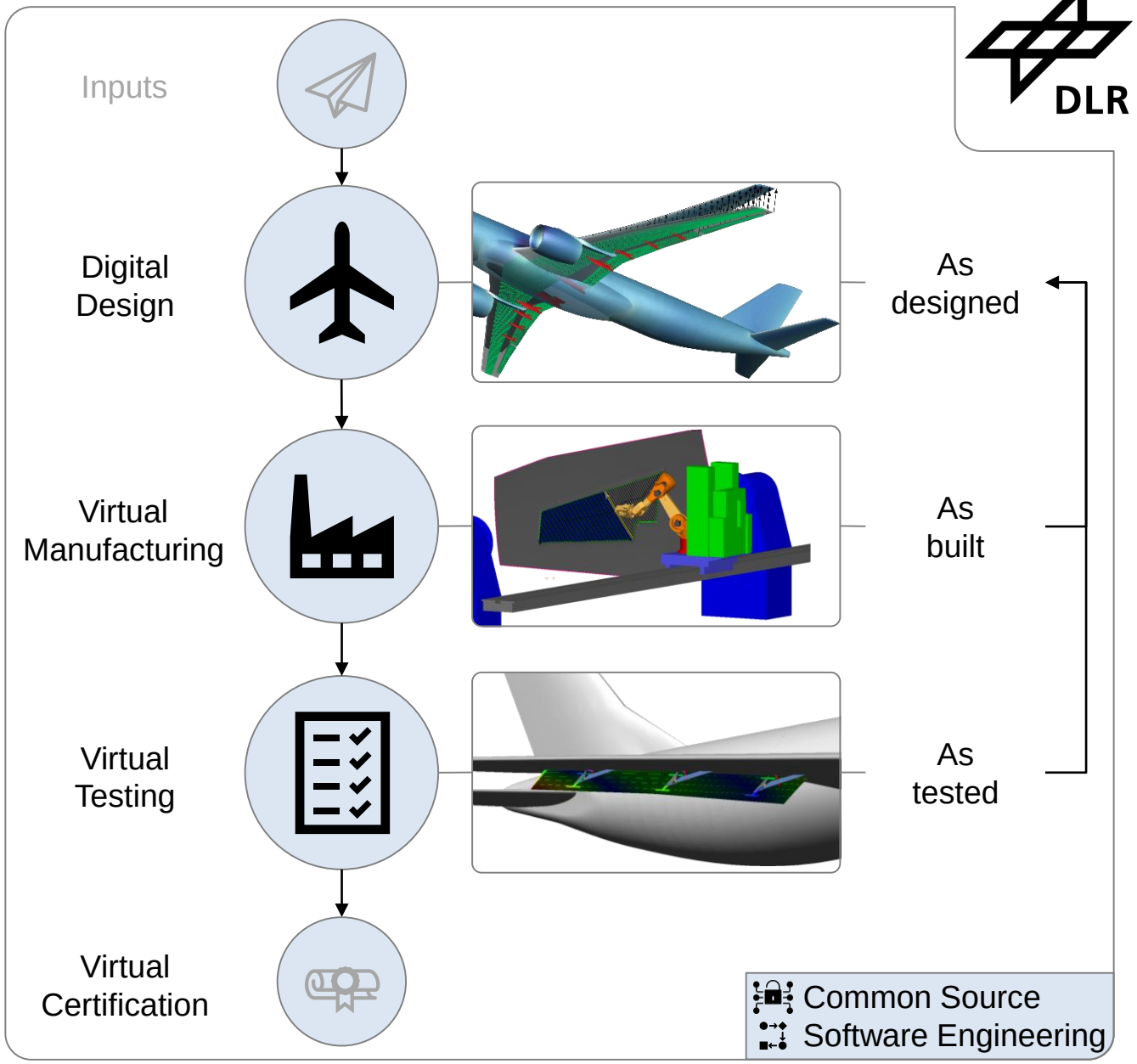
Virtual Product House (VPH): Overview

- Objectives
  - Virtual Aircraft Development & Evaluation
  - Reduce physical tests
  - Improvements in aircraft emissions
  - Virtual Certification
- **Digital Twins** are key



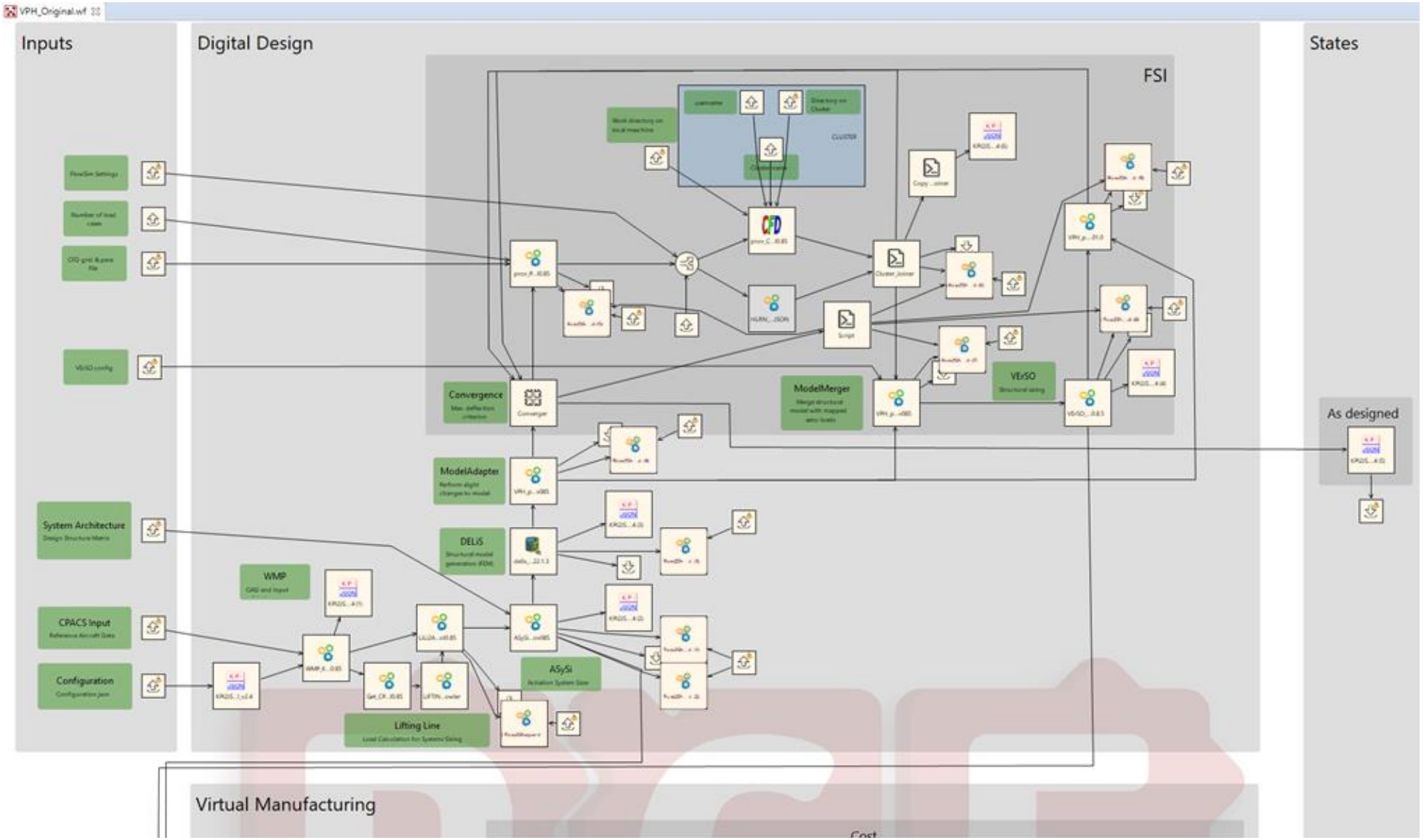
VPH:
Phases Considered

- Key phases are
  - Digital Design
  - Virtual Manufacturing
  - Virtual Testing
- Virtual Certification as vision
- Software platform key enabler and research topic



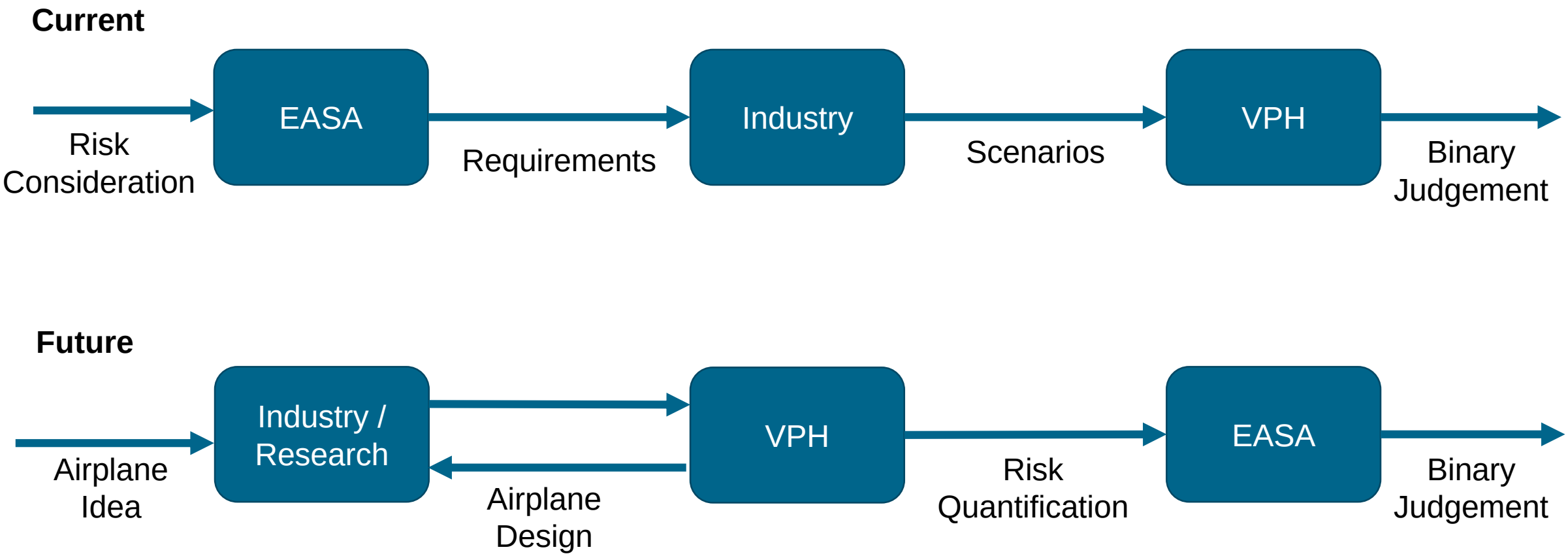
Virtual Product House
Virtual Design

- Digital aircraft (component) design process
- Input: Initial design of aircraft component
- Output: sized component structure (“As designed”)



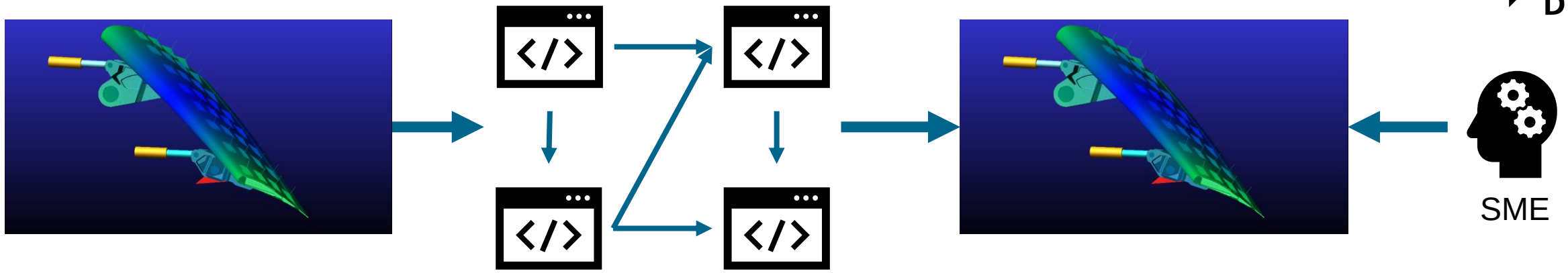
Boden, B. et al. (2021) RCE: An Integration Environment for Engineering and Science. SoftwareX 15:100759.

Current and Future Certification Process



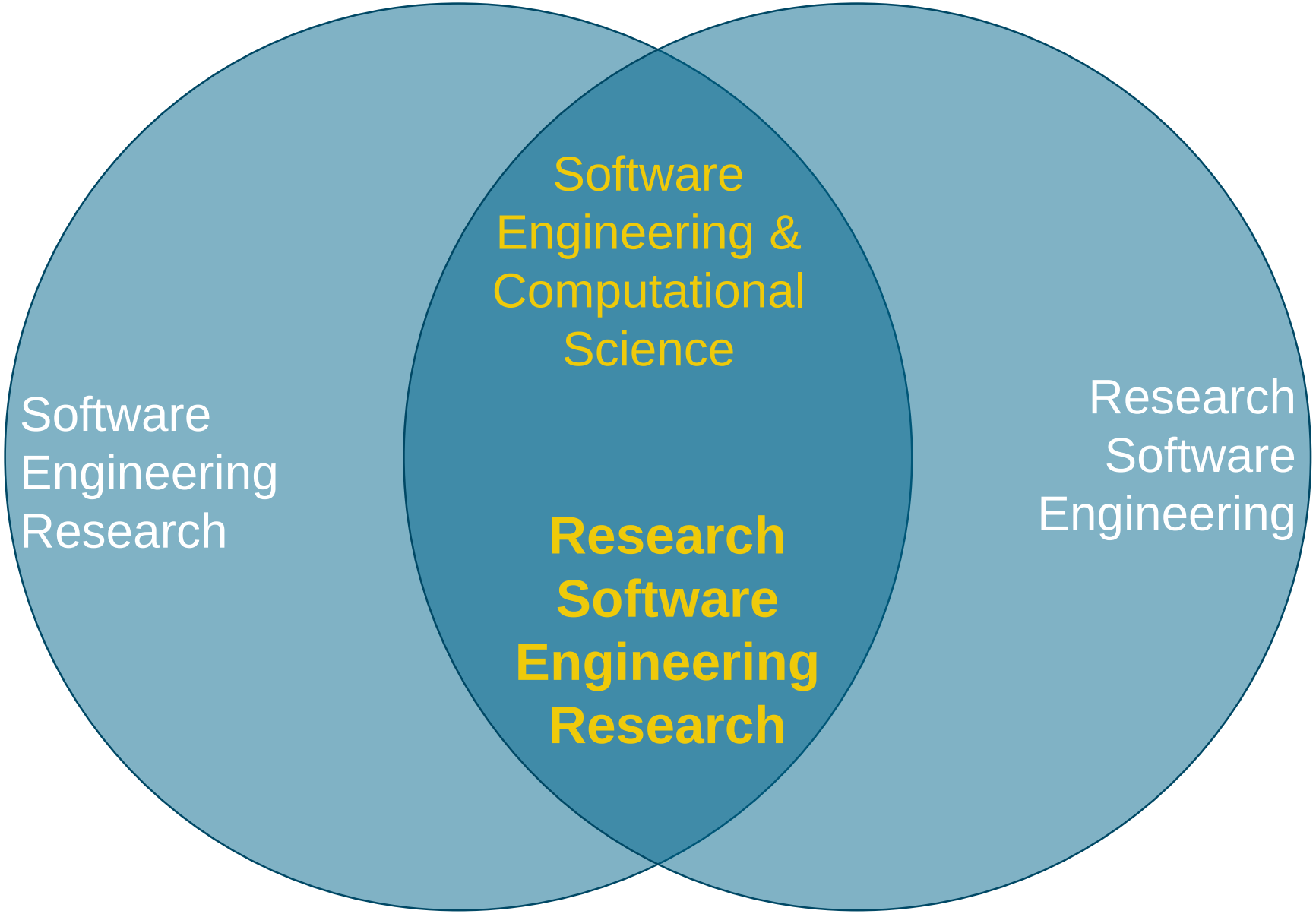
Now: Conservative Airplane Design driven by Top-Down Waterfall Process
Future: Risk-Driven Agile Airplane Design

Advanced Automated Testing for VPH



- Judgement currently based solely on experience of Subject Matter Expert (SME)

- Automated testing support required that takes
  - Provides fast feedback
  - Takes parameter space (length, width, shape, no. of flaps, …) into account
  - Takes massive amounts of data into account
  - Provides suitable oracles
  - Considers human-in-the-loop


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

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

- Dealing with vast configuration spaces
- Dealing with highly dynamic requirements and complex domains
- Dealing with simulations, digital twins and big data
- Dealing with long-lived software artifacts and reuse
- Dealing with development under resource constraints
- 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 is specific about RSE compared to other SE specializations?

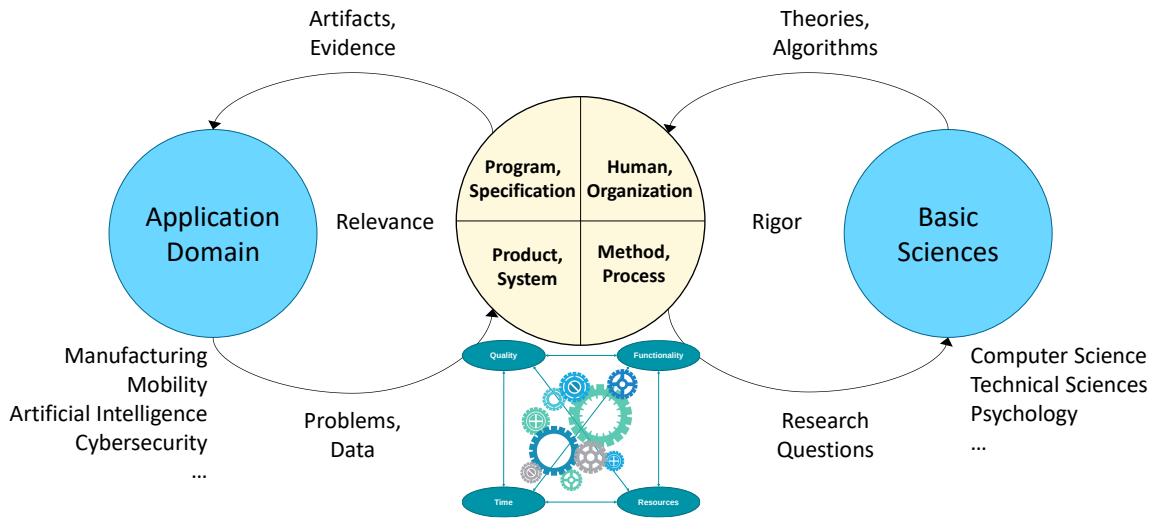
What are suitable lifecycle models for research software?

How to integrate SE techniques into research software development?

Is research software of poorer quality than industry software?

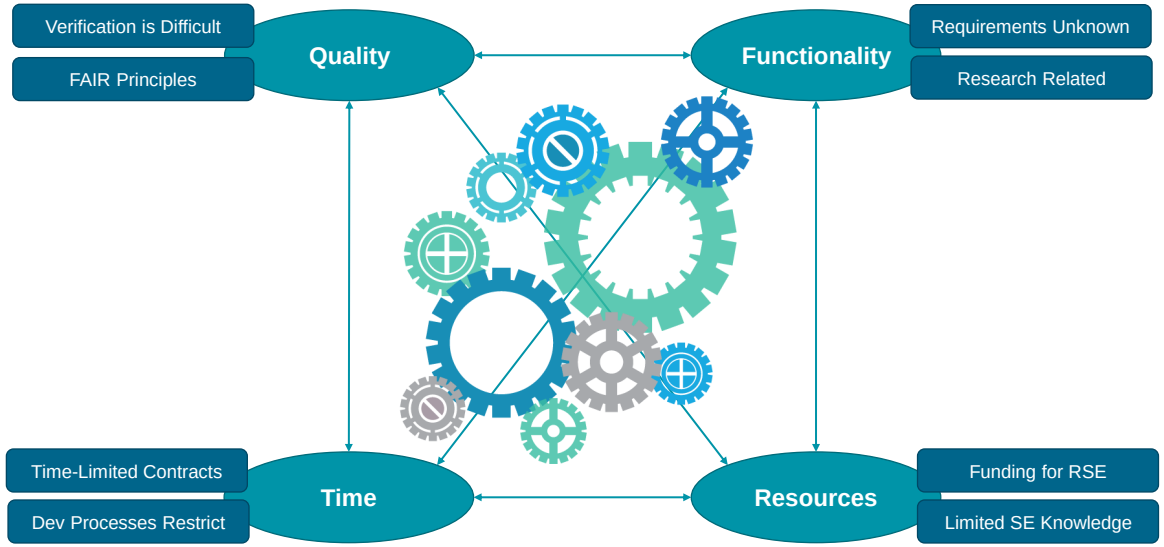
Which skills and educational formats are required for RSE practitioners?

Software Engineering Research

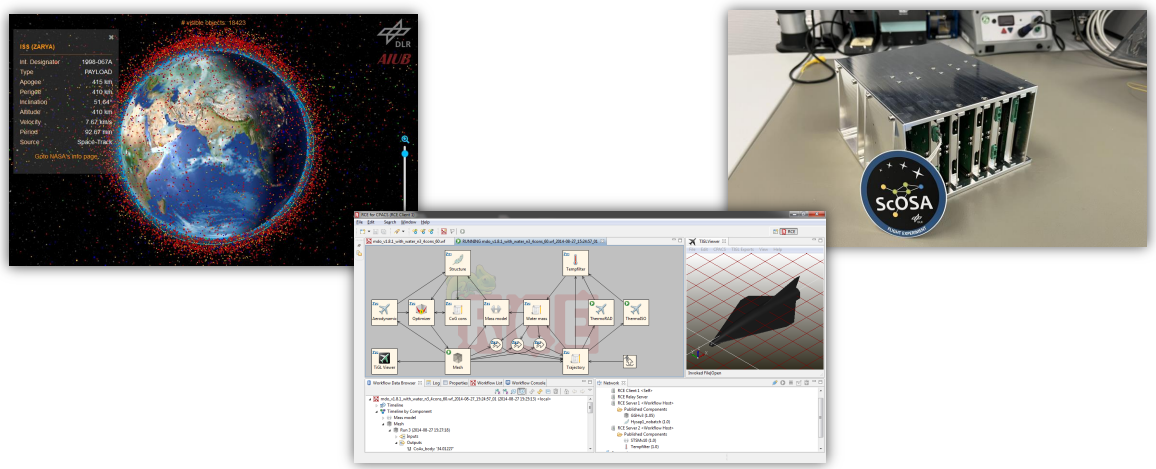


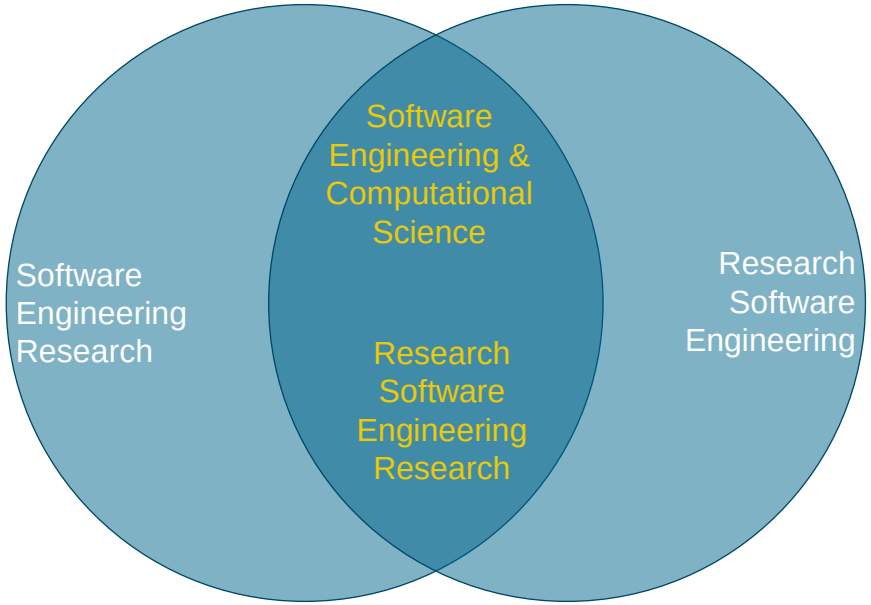
Felderer, M., Reussner, R., Rumpe, B. (2021) Software Engineering und Software Engineering Forschung im Zeitalter der Digitalisierung. Informatik Spektrum.

Trade Off in Software Engineering: Specifics of RSE



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





References

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

[2] Boden, B. et al. (2021) RCE: An Integration Environment for Engineering and Science. SoftwareX 15:100759.

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

[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] Felderer, M., Reussner, R., Rumpe, B. (2021) Software Engineering und Software Engineering Forschung im Zeitalter der Digitalisierung. Informatik Spektrum.

[6] Felderer, M., Goedicke, M., Grunske, L., Hasselbring, W., Lamprecht, A.-L. (2025) Investigating Research Software Engineering: Toward RSE Research. Communications of the ACM.

[7] Hasselbring, W. et al. (2024) Toward Research Software Categories. CoRR abs/2404.14364.

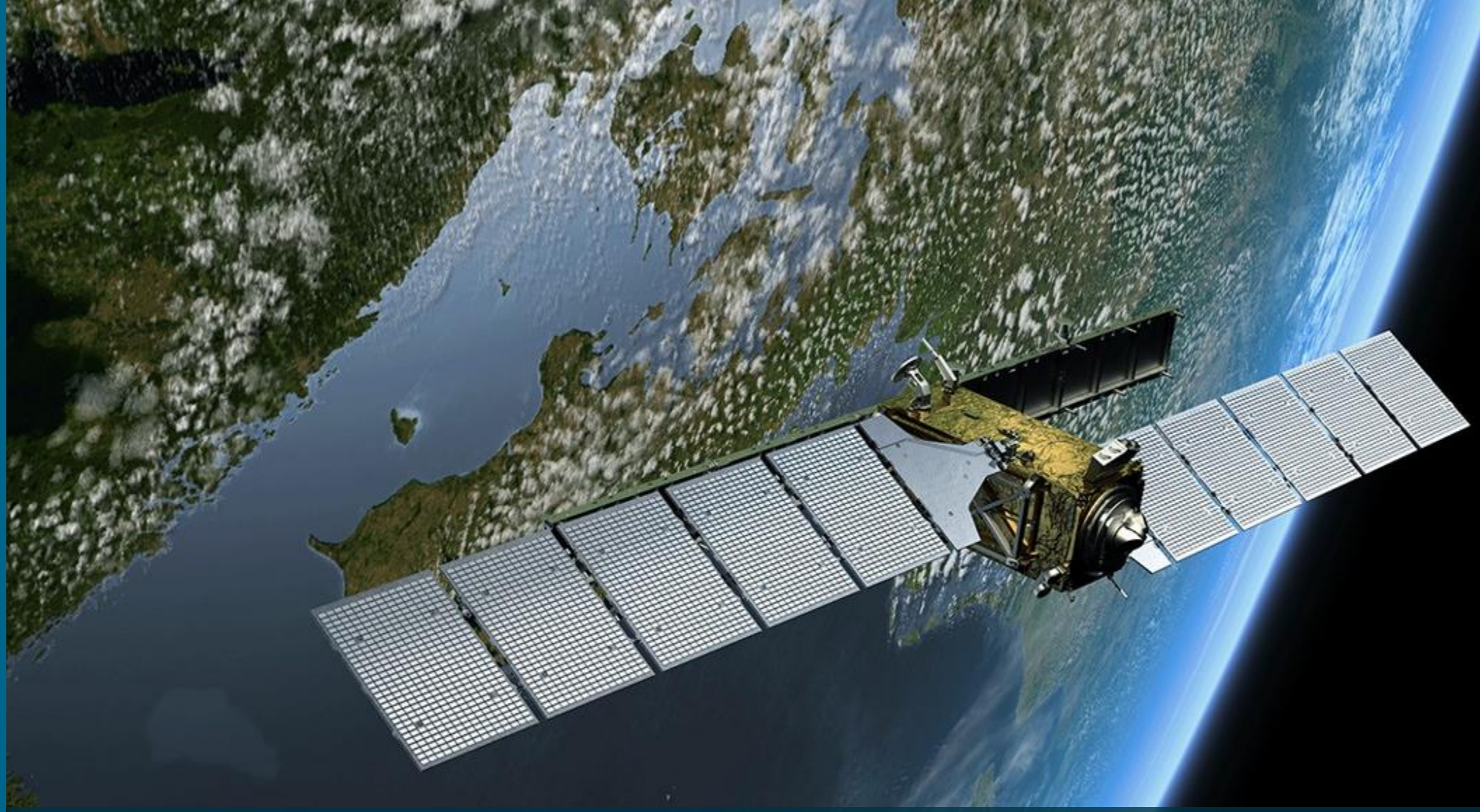
[8] Johanson et al. (2016) Evaluating Hierachical Domain-Specific Languages for Computational Science. Software Engineering for Science. CRC Press.

[9] Kanewala et al. (2016) Automated Metamorphic Testing of Scientific Software. Software Engineering for Science. CRC Press.

[10] Mendez D. et al. (2022) Open Science in Software Engineering. Contemporary Empirical Methods in Software Engineering. Springer.

[11] Tahvili, S., Hatvani, L., Felderer, M., Afzal, W., Bohlin, M. (2019) Automated Functional Dependency Detection Between Test Cases Using Doc2Vec and Clustering. AITest 2019.

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



Prof. Dr. Michael Felderer
Institute of Software Technology

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