



UNIVERSITÄT ZU LÜBECK

Information Systems

CS4130-KP06

Prof. Dr. Sylvia Melzer

SoSe2026





Research Software Engineering

Information Systems

What You Will Learn

- Understand principles of Software Engineering (SE)
- Explain Research Software Engineering (RSE)
- Distinguish SE, RSE, and interdisciplinary RSE
- Describe interdisciplinary RSE workflows
- Understand why sustainable software development matters in research
- Explain why interdisciplinary communication is central in RSE

Why Software Engineering (SE)?

- Writing a **small program** is usually manageable for **one person**
- **Problems** arise when software becomes **larger and more complex**
- **Large systems** involve **many developers, users, and components**
- Different parts of the system must **work together** reliably
- Even **small changes** can create **unexpected errors** elsewhere
- Software must often run for **many years** and continuously evolve

Example:

Development of small system

- One person
- Short-lived
- Local files
- Simple debugging

Development of large system

- Many teams
- Long-lived
- Distributed data
- Coordination overhead

System Development: Small vs. Large

- In small systems, complexity is still visible and understandable to one person
- In large systems, complexity becomes distributed across many people, components, and workflows
- Large software is not only “more code”
- It is mainly:
 - more dependencies
 - more coordination
 - more communication
 - more long-term maintenance

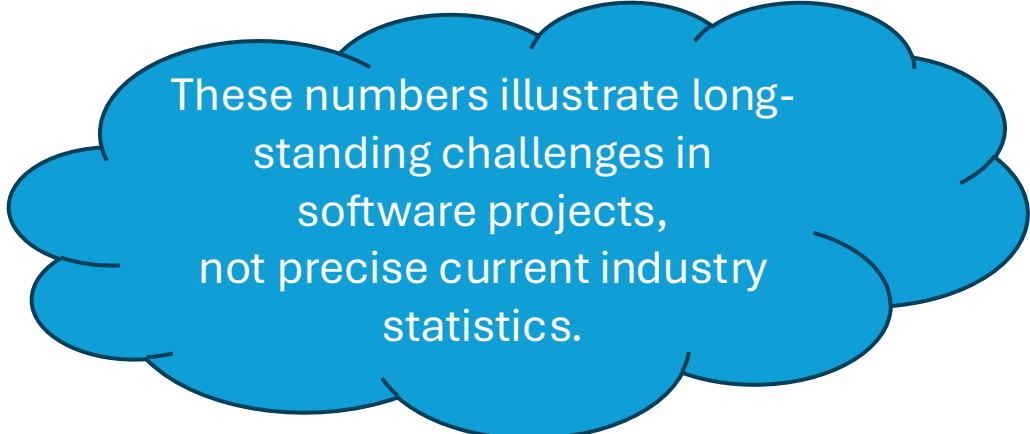
Goals of Software Development

Software engineering tries to balance several important goals:

- Software **should satisfy users requirements**
- Systems should be **reliable** and work correctly over time
- **Maintenance** and future changes should remain **manageable**
- Projects should be completed **within deadlines**
- Development **costs should stay reasonable**
- Software should **perform efficiently** and **respond quickly**
- Components should be **reusable** in future projects

Satisfying User Requirements

- Many programs simply don't do what end users want
- Typical percentages for large-scale commissioned systems:
 - 45% delivered but not used
 - 27% paid for but not delivered
 - 17% abandoned
 - 6% used after changes
 - 5% used as delivered
- Users find it hard to articulate what they want
- Developers find it hard to understand what users say!



These numbers illustrate long-standing challenges in software projects, not precise current industry statistics.

Two Incompatible Ways of Knowing

Humanities Logic

Dimension 1: The Role of Context

Meaning is entirely dependent on context, layered reception, and symbolic weight.

Dimension 2: The Treatment of Ambiguity

Ambiguity is constitutive and intentional (e.g., contested attributions).

Dimension 3: Data Representation

Continuous prose, marginalia, and overlapping structural boundaries.



Computational Logic

Dimension 1: The Role of Context

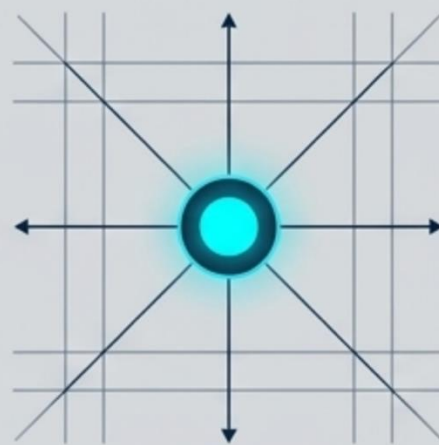
Meaning must be atomized, strictly categorized, and isolated from external ambiguity.

Dimension 2: The Treatment of Ambiguity

Ambiguity is an error. Data must be forced into First Normal Form (1NF).

Dimension 3: Data Representation

Discrete fields, binary relationships, and rigid hierarchical schemas.



High Reliability

Errors in software are commonly called bugs

Bugs can never be completely ruled out

Testing can reveal existing problems

However, proving that no bugs exist is extremely difficult

Reliable software is essential in critical systems

Failures can have serious real-world consequences

Software quality directly affects trust and safety

In research contexts, reliability also affects reproducibility, transparency, and long-term usability of scholarly data

Consequences of Software Failures

- Human Safety
 - Aircraft systems
 - Medical systems
 - Nuclear power systems
- Financial Damage
 - Failed software projects
 - System outages
 - Expensive launch failures
- Reputation Damage
 - Loss of customer trust
 - Negative public perception
 - Long-term business consequences

Consequences of Software Failures in the Humanities

- Research data may become inaccessible or lost
- Digital editions and annotations can become corrupted
- Incorrect analyses may lead to false interpretations
- Poor metadata can make research data difficult to find or reuse
- Broken infrastructures interrupt research workflows
- Researchers can lose trust in digital methods and infrastructures

Maintenance Costs



Software continues evolving after release



Most project costs occur after deployment



Maintenance is often more expensive than initial development



Small changes can unexpectedly affect other components



Understanding existing systems is often difficult



Poor documentation increases maintenance effort



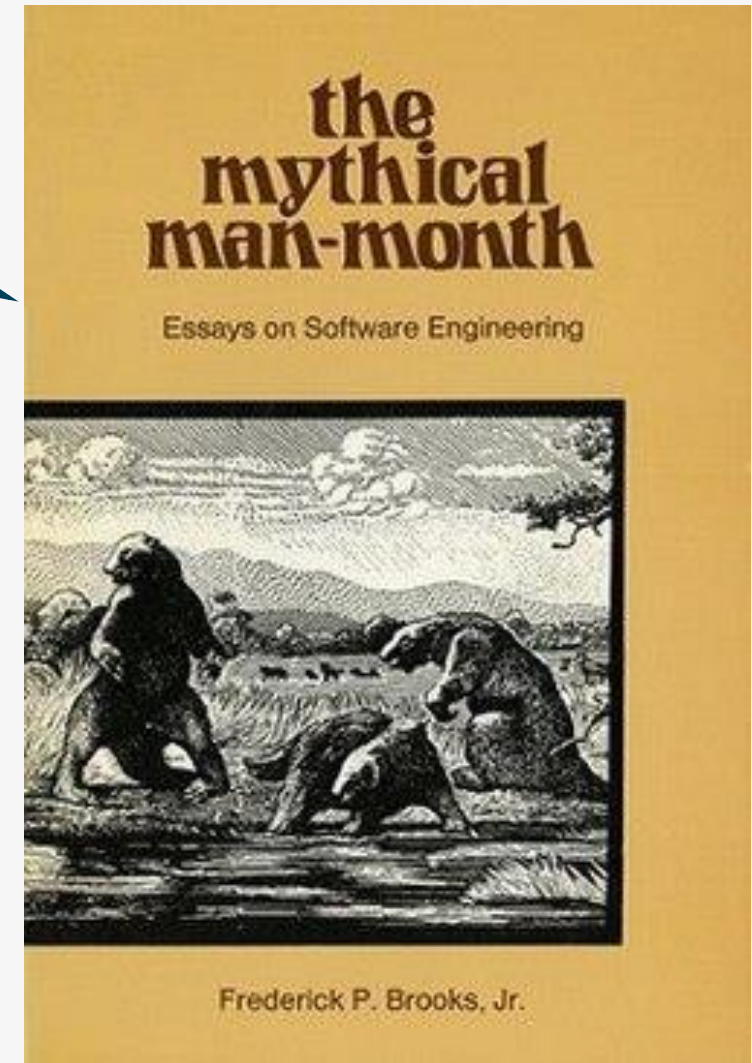
Sustainable software requires maintainable design

- Corrective: Fixing bugs
- Adaptive: Adjusting to new environments
- Perfective: Adding new features and requirements
- Preventative: Improving long-term maintainability
- **Most software engineering work is not writing new code, but understanding and adapting existing systems**

Delivery on Time

Brooks' Law:
"Adding manpower to a
late software project
makes it later."

- Software projects are notorious for overrunning
- It is extraordinarily hard to reliably predict how much effort a software project will require, and when it will be completed
- The relationship between person months devoted and development time is almost never linear:
 - adding person months of effort to a project frequently has no effect;
 - adding person months of effort often makes the project slower



Production Costs

Software production is an enormous industry:

- **The global custom software development** market size is anticipated to grow from \$53.02 billion in 2025 to \$334.49 billion by 2034, at a CAGR of 22.71%. Enterprise software accounted for 61% of the market, with large enterprises holding the largest revenue share of over 61%
- The **global software market** size was \$823.92 billion in 2025 and is expected to reach around \$2,248.33 billion by 2034, expanding at a CAGR of 11.8% from 2025 to 2034
- The software testing segment within the **software engineering** market is expected to hold a share of over 38% by 2032
- **Software market growth** around 10–12% annually
- Software engineering is not only a technical activity, but also a major economic and organizational factor

Performance

- Some systems must operate extremely efficiently
- Optimization improves speed and resource usage
- High-performance systems require specialized solutions
- Optimization often increases system complexity
- Optimized code can become harder to understand
- Highly specialized systems are less flexible
- Performance improvements often create tradeoffs
- Better Performance, BUT
 - harder maintenance
 - reduced readability
 - reduced flexibility
- Example
 - Scientific simulations
 - Real-time systems
 - Large-scale databases

Maximum performance often reduces simplicity and maintainability

Ease of Reuse



Reuse reduces development effort



Reusable components improve reliability



Existing solutions can be adapted to new systems



Reuse accelerates software development



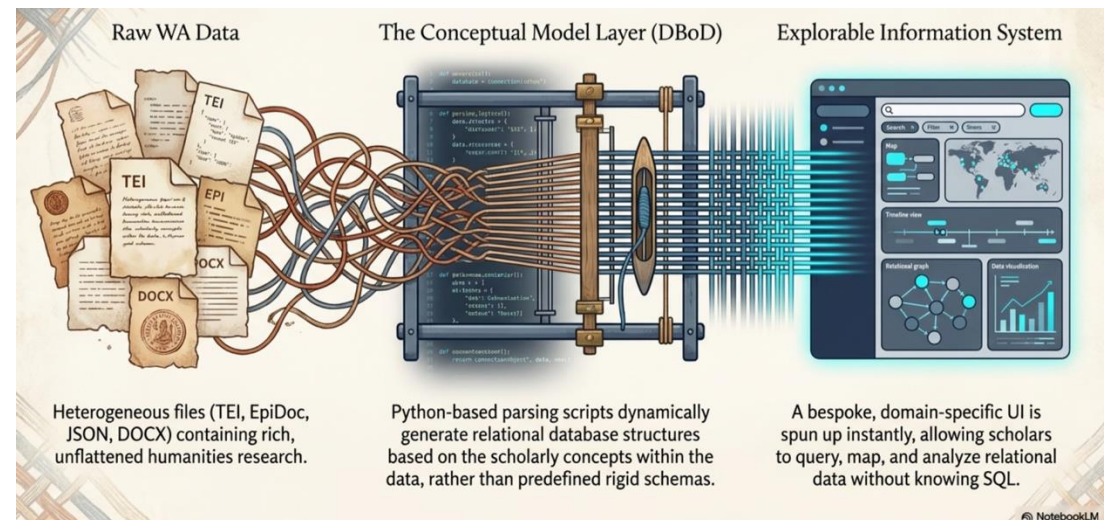
Modular architectures support reuse



Separation of concerns is essential

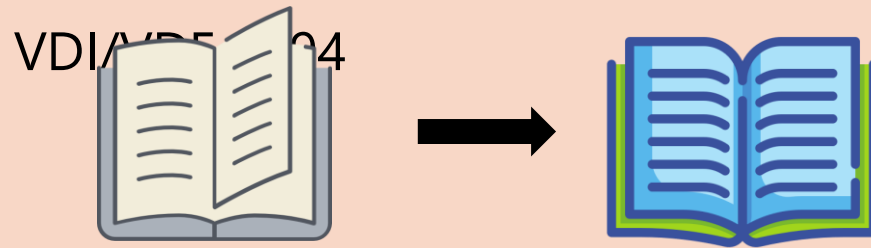


Systems should be designed with reuse in mind



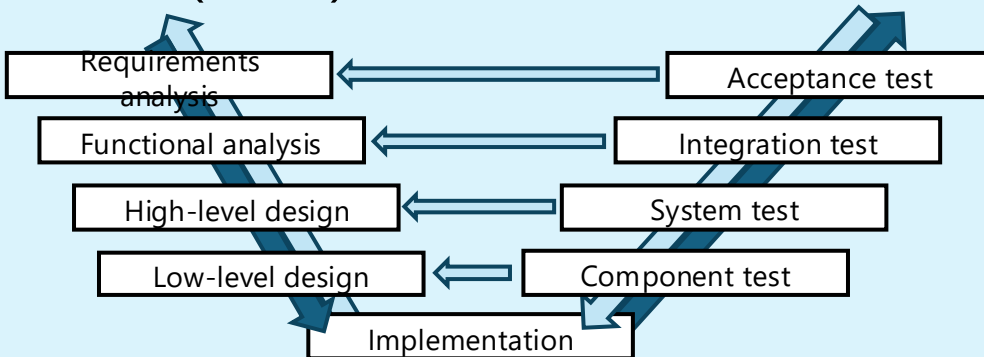
Process models

PRODUCT REQUIREMENTS
DOCUMENT → SCOPE MANAGEMENT
PLAN



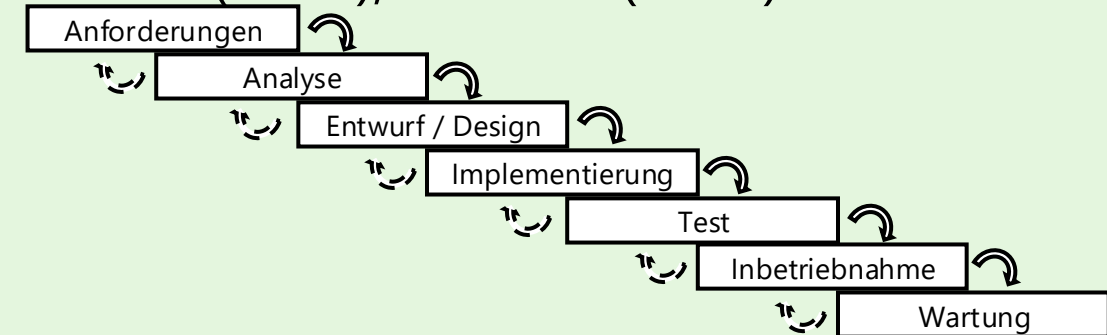
WHAT (FOR)? → HOW?/WITH
WHOM?

BUND (1992)



WATERFALL MODEL

ROYCE (1969), BOEHM (1981)

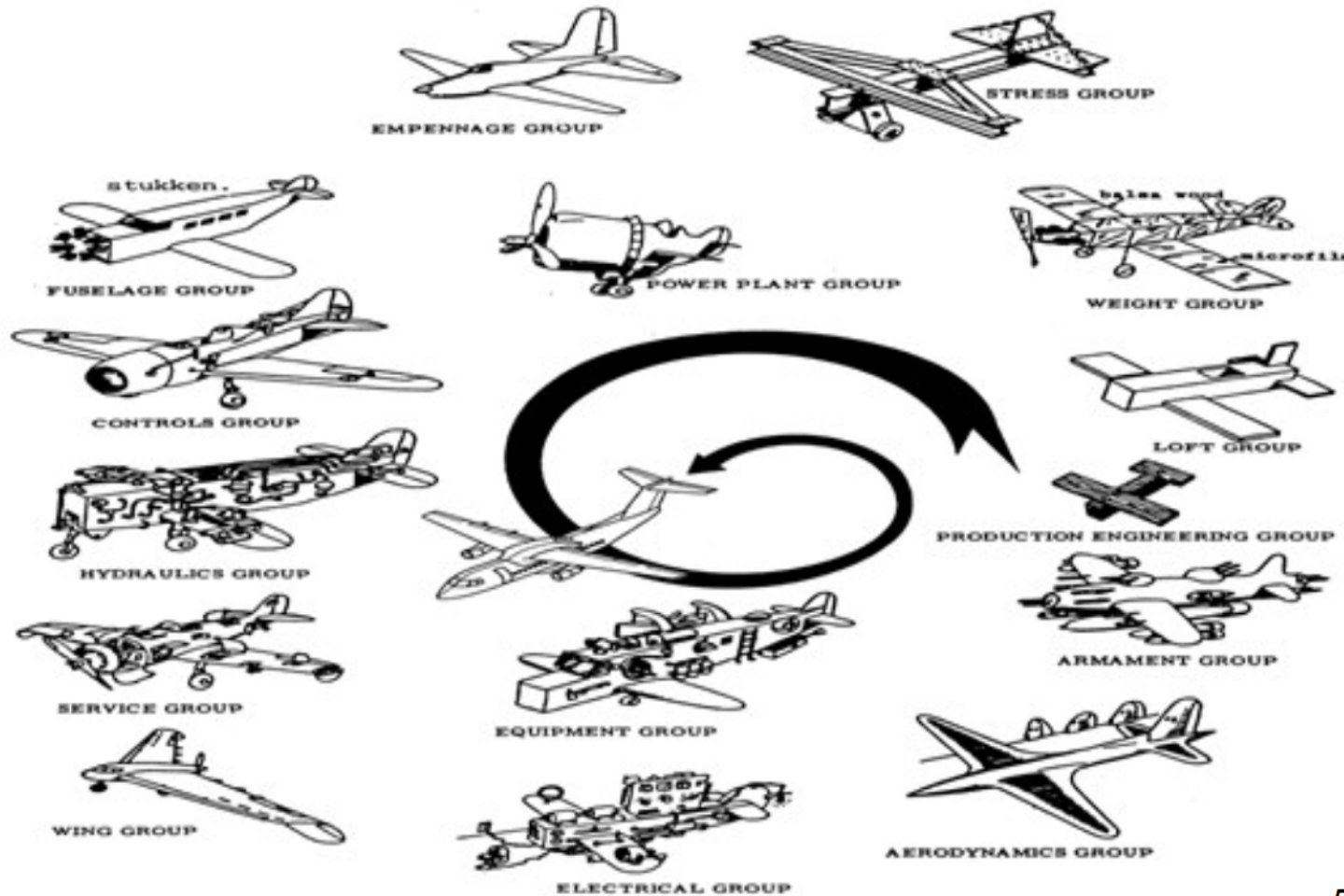


SPIRAL MODEL

TAYLOR (1956), BOEHM (1980)



Process models



„A completed airplane in many ways is a compromise of the knowledge, experience and desires of many engineers that make up the various design and production groups of an airplane company.“
– C.W. Miller

Challenges in Research Contexts



Research requirements constantly evolve



Data are uncertain and heterogeneous



Research workflows are exploratory



Interpretations change over time



Domain expertise becomes central



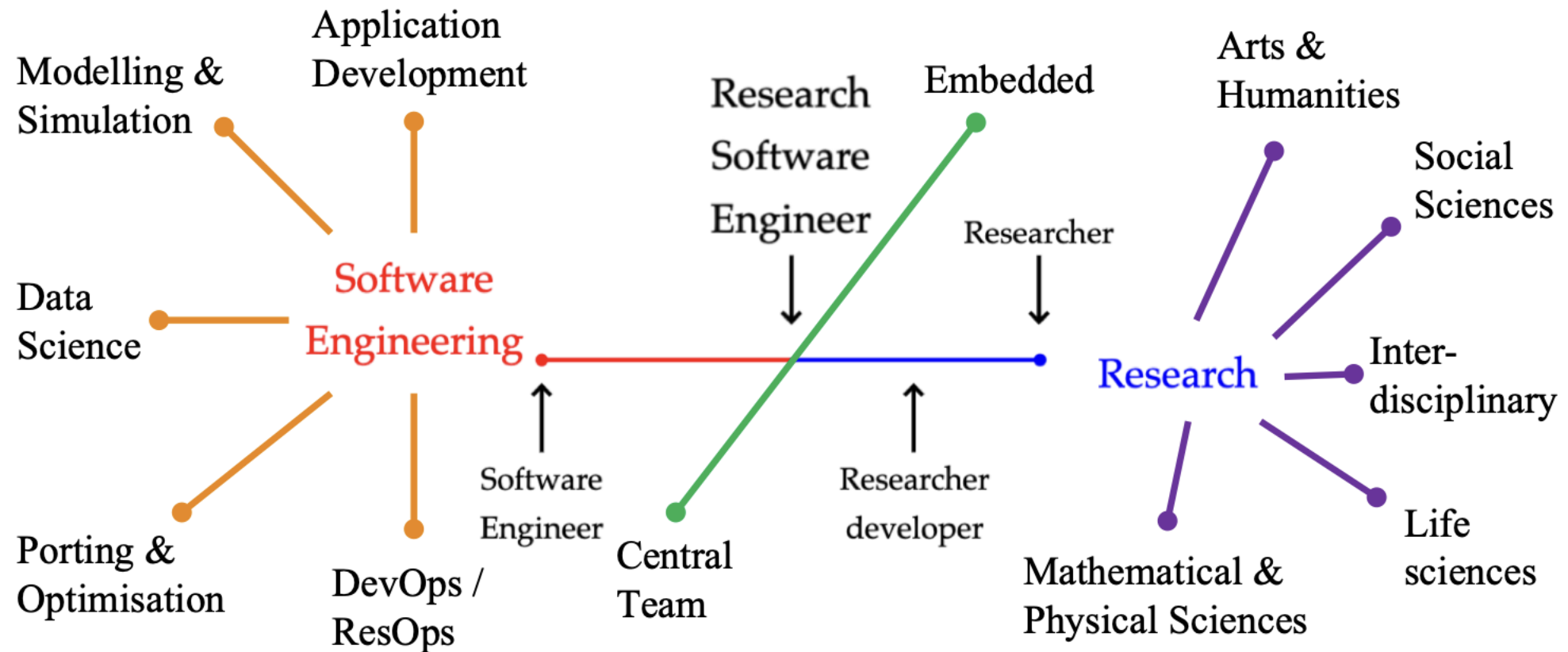
Reproducibility is essential



Sustainability is often difficult

Problem: Classical SE alone is insufficient for research contexts

Software Engineering and Research



Research Software

- **Research Software** includes source code files, algorithms, scripts, computational workflows and executables that were created **during the research process or for a research purpose**.
- **Software components** (e.g., operating systems, libraries, dependencies, packages, scripts, etc.) **that are used for research**, but were not created during or with a clear research intent should be considered software in research and not Research Software.

Research Software Engineer

- A **Research Developer** wants to be judged on their scientific output, is a researcher at heart, however develops a lot of code due to the nature of their research. They would like recognition comparable to a research paper for a software implementation taken up by others.
- A **Research Software Engineer** comes from a research background but is also a skilled software developer, and relishes challenge of not just developing code to solve a problem but doing it well. They want to be recognized for producing tools which others rely on for research.

Analyst Developer Analyst Programmer - SITS (x 3) Analyst/Programmer Applications Developer Applied Scientist Architectural Robotics Developer Assistant Data Programmer Assistant Project Manager Atmospheric Correction and Radiative Transfer Model Scientist Audio Software Developer - KTP Associate Bioinformatician Bioinformatician In Potato Genomics and Genetics Bioinformatician/Computational Bioscientist in Microbiology Bioinformaticians Bioinformatics Analyst Bioinformatics Postdoctoral Researcher Bioinformatics scientist Biometric Software Systems Developer Biorespository Software Developer C++ / 3D Graphics Software Engineer Casebooks Project Editor (Research Assistant/Associate) Climate Researcher (Research Associate) Clinical Study Programmer CoMPLEX Research Associate Computational Biologist / Bioinformatician Computational Scientist Computational Scientist in Computational Fluid Dynamics & Industrial Applications Computational Scientist in Structural Mechanics and Industrial Applications Computer Scientist Computer Vision Researcher Content Developer/Programmer Control Engineer-IMG - 3 posts CREATE Data Specialist Data Analyst Data Integration Coordinator Data Manager x3 Database and Software Engineer Database Manager/Researcher Database Programmer Digital Media Technician E-Learning Portal Manager (KTP Associate) e-Learning Systems Development Analyst e-Learning Systems Development Analyst (Moodle, SQL) E-Learning Web Developer E-Portfolio Learning Technologist Embedded Systems Engineer Engineering Technician Environmental Scientist EPSRC Studentship on Algorithmic Construction of Finsler-Lyapunov Functions Experimental Officer in Bioinformatics Experimental Psychologist Finance Assistant Gaia Alerts Software Developer Gaia Software Developer (Gaia UK Team) GIS Applications Specialist Graduate Programmer / Software Developer Graphics Programmer Health Data Manager / Scientist High Throughput Bioinformatician High Throughput Sequencing Bioinformatician (Two posts) HIVE Manager/ HIVE Coordinator HIVE Senior Researcher and Technical Lead Hydro-informatics Scientific Software Developer Image Analysis Manager for Cancer Imaging Information Systems Developer Instrumentation Engineer Investigator Statistician IT Developer IT Services Manager IT Services Specialist (e-Learning Systems) IT Support Technician (Unix / Windows Systems) Knowledge Transfer Partnership (KTP) Associate: Innovent Technologies LTD Knowledge Transfer Partnerships (KTP) Associate - Software Developer KTP Associate - Robot Vision Scientist (Research Fellow) KTP Associate (Fixed Term Contract for 24 months) KTP Associate (Precision Agriculture Data Analyst) KTP Associate (e-Learning Systems Development) KTP Associate: Electronics / Robotics Engineer Learning Technologist Leicester Respiratory BBU IT Developer Lincolnton Psychologist Marie Curie Early Stage Researcher Marie Curie Early Stage Researcher in Radar Rainfall for Integrated Water Quality Modelling Marine Earth Observation Scientists Medical Statistician Medical Statistician/Senior Medical Statistician Metrology Engineer Mobile Application Developer NASC IT Support - Programmer and Systems Administrator (Fixed-term) NIHR Research Methods Fellow PDRA on EU Project on Automated Multisensor Surveillance Planning Officer Policy Modeller 2014 Post - Doctoral Research Assistant INSTRON Post Doctoral Research Worker Post Doctoral Researcher in the application of Digital Technology Post-Doctoral Research Assistant in Simulation and Visualization Post-Doctoral Research Associate Post-Doctoral Research Associate (Pathogen Genomics) Post-Doctoral Research Fellow Postdoctoral Fellow - population genetics / evolutionary genetics Postdoctoral Fellow in Bioinformatics Postdoctoral Fellow in Cancer Therapeutics Postdoctoral Research Assistant Postdoctoral Research Associate Postdoctoral Research Fellow Postdoctoral Research Scientist Postdoctoral Researcher in Declarative (Logic and Functional) Programming Postdoctoral Researcher Postdoctoral Scientist Postdoctoral statistician Postdoctoral Training Fellow - Statistical and Computational Genetics of Autism Principal / Senior Bioinformatician Principal Bioinformatician Product Development Engineer (Rail) Publishing Portal Web Developer Radio Frequency Engineer Reader in Computer Science Reporting Analyst Research (Software) Engineer Research Assistant Research Associate Research Fellow Research Image Data Manager, Biomedical Engineering Research Officer Research Officer â€” Social Protection Research postgraduate Research Programmer Research Scientist Research Scientist / Senior Research Scientist Research Scientist in Machine Learning and Computer Vision Research Software Developer Research Software Developer for the Herchel Smith Professor of Organic Chemistry Research Software Engineer Research Studentship Research Worker Researcher SAP Trainee Technical Analyst Scientific Officer with Michela Garofalo Scientist SEAHA Studentship: Extracting epidemiological data from collections SEEG Data Archive Manager Senior / Research Associate in Clinical Integration and Image Analysis for Fetal Surgery Senior Analyst Programmer (Business Analysis) Senior Analyst/Programmer Senior Bioinformatician Senior Bioinformatician / Bioinformatician Senior Computational Statistician - Spatial Models Senior Data Acquisition Scientist / Data Acquisition Scientist Senior Data Manager Senior Database Administrator Senior IT Developer Analyst Senior Mathematical Modeller Senior Media Developer Senior Postdoctoral Researcher - Evolutionary and Computational Analysis of Infectious Disease (Phylodynamics) Senior Research Assistant Senior Research Associate Senior Research Associate â€” Molecular Modelling & Simulation Senior Research Associate in Quantitative Clinical MRI Senior Research Fellow Senior Research Fellow/Research Fellow in Vibration Diagnostics and Promotive/Digital Clinical Research Senior Research Technician Senior Research Technician Senior Research Technician Senior Software Developer in Bioinformatics Senior

194 different job titles

A not-so-brief history of Research Software Engineers

Posted by s.aragon on 17 August 2016 - 2:26pm



Image by Marie Brizard and Simon Hettrick

By Simon Hettrick, Deputy Director

On a beautifully sunny day in March 2012, a small group met at Queen's College Oxford and challenged a long-standing problem: why is there no career for software developers in academia? They didn't know it at the time, but this meeting led to a nationwide campaign that created a vibrant and rapidly growing community, and established a new role in research: the Research Software Engineer.

By Simon Hettrick, Deputy Director

History

- 2012 RSE term usage and 1st RSE group established
- 2014 UK RSE Association
- 2016 1st UK RSE conference & de-RSE founded
- 2016 FORGE workshop “sustainable research applications and software” at U Hamburg
- Smith et al. (2016) Software Citation Principles
<https://doi.org/10.7717/peerj-cs.86>
- 2017/18 Dutch + Nordic + US RSE Associations launched
- 2018 Software Management Recommendations (Alliance DE research org)
- 2022 FAIR Principles for Research Software (FAIR4RS)
- ...

FAIR4RS – FAIR for Research Software

- less known/implemented than FAIR data principles
- referenced by ReSA, funders, research org, RSEs, training initiatives, ..., (<https://doi.org/10.5281/zenodo.10816031>)
 - mostly alignment but not required
- several FAIR assessment tools suggested / implemented
 - diverse interpretations lead to diverging FAIRness scores in various implementations, example: <https://github.com/fair-software/howfairis>
- more: <https://zenodo.org/communities/fair4rs/>

FAIR FOR RESEARCH SOFTWARE



Martinez, P. (2023, February 21). FAIR4RS - FAIR for Research Software. Zenodo. <https://doi.org/10.5281/zenodo.7660245>

TOP 10 FAIR DATA & SOFTWARE THINGS. 10.5281/zenodo.3409968

FAIR4RS WG. RDA, ReSA, FORCE11

2016

2018

2019

2020

2022

SOFTWARE CITATION PRINCIPLES
DOI: 10.7717/peerj-cs.86

TOWARDS FAIR PRINCIPLES FOR RESEARCH SOFTWARE
10.3233/DS-190026

FAIR FOR RESEARCH SOFTWARE PRINCIPLES (1.0)
10.15497/RDA00068

DOI: [10.5281/zenodo.7660245](https://doi.org/10.5281/zenodo.7660245)



Research Software Engineering (RSE)

RSE integrates software development into research practice

Research requirements evolve continuously

Research data are heterogeneous and uncertain

Software becomes part of the knowledge process

Domain experts and developers collaborate closely

Reproducibility and sustainability are core goals

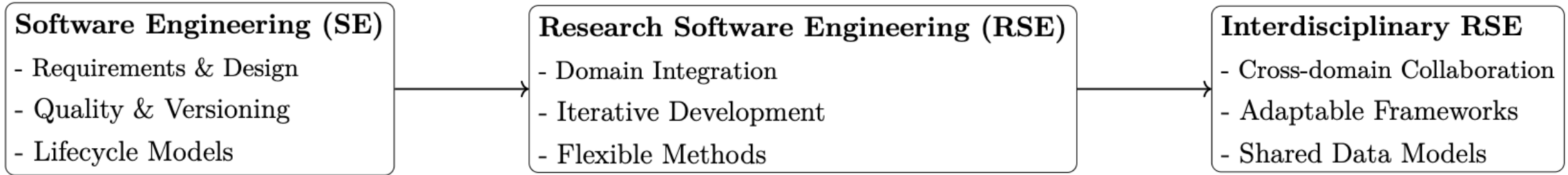
RSE combines technical and epistemic perspectives

The Future of RSE

- Code is **not a condensed method** or self-contained representation of a research method
- Research methods express **epistemic goals and interpretive frameworks**, whereas code implements technical and operational decisions
- Software embodies **assumptions, abstractions, and constraints** that extend beyond methodological description
- Research software is typically developed within **limited project durations**, often around **three years**
- Sustainability, documentation, and maintainability are essential to **enable future reuse**

RSE

- Research Software Engineering (RSE) emerged as a recognized field in the 2010s
- RSE combines software engineering expertise with domain-specific research
- Existing RSE initiatives focus strongly on:
 - Reproducibility
 - Sustainability
 - research infrastructure
 - FAIR principles
- Important organizations and initiatives include:
 - The UK Research Software Engineer Association
 - The Software Sustainability Institute
 - Netherlands eScience Center
 - Research Data Alliance (RDA)
- Existing approaches mainly focus on technical sustainability
- Interdisciplinary humanities research additionally requires:
 - interpretive flexibility
 - heterogeneous data integration
 - epistemic transparency



From SE to Interdisciplinary RSE

From SE to Interdisciplinary RSE

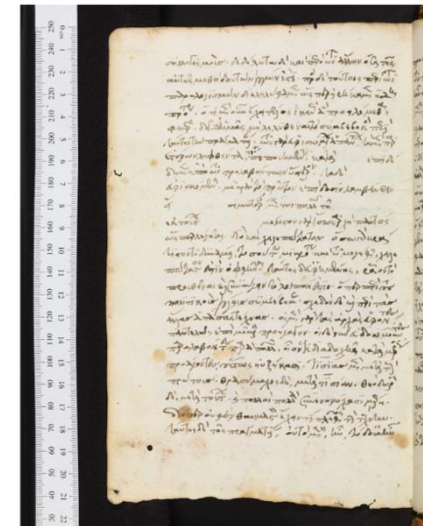
- Interdisciplinary projects involve **multiple epistemologies**
- Different disciplines use **different terminologies**
- **Shared computational environments** are required
- **Data models mediate** between disciplines
- **Communication** becomes a central infrastructure
- **Flexible frameworks** are necessary
- Interdisciplinary RSE enables **integration** across domains

UWA II: Project Groups Examples

PG Writing the Family: Genealogical Written Artefacts in Africa, Asia, and Europe

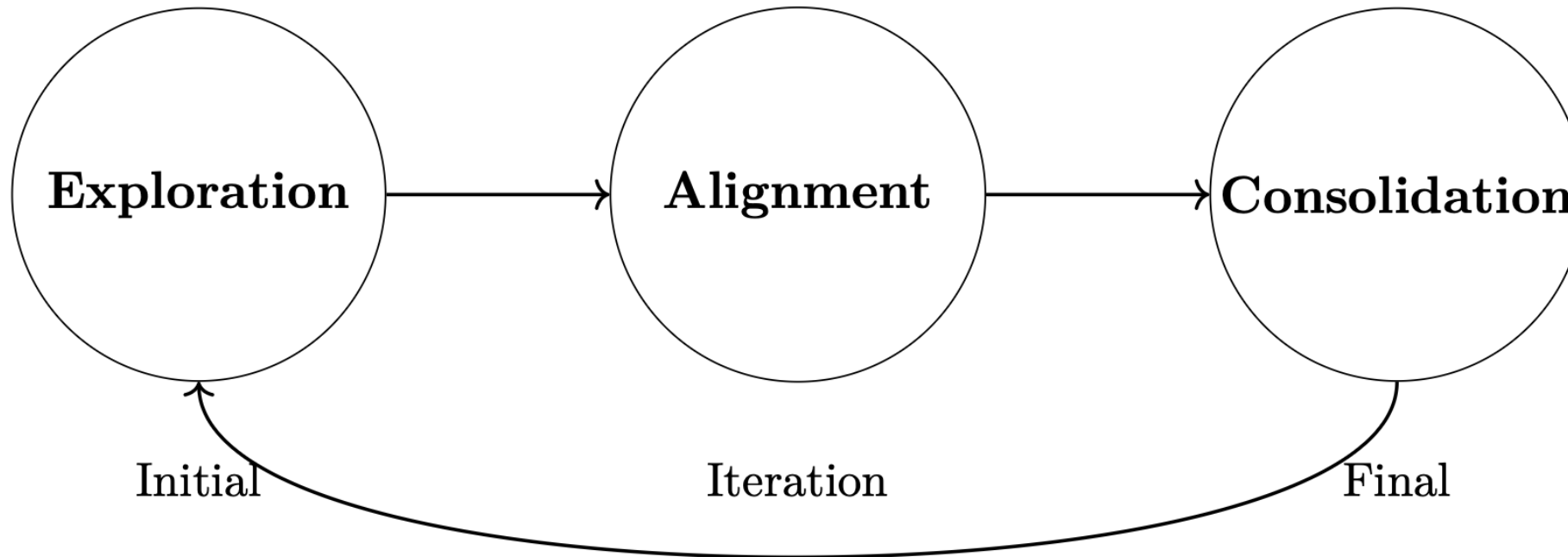


PG Provenancing Plant-Based Writing Supports



Interdisciplinary RSE in Practice

EASE Model



The EASE Approach



Explorable

Researchers must be able to actively navigate through data, discover latent patterns, and understand how elements build interpretive frameworks.



Accessible (in Quality)

The reliability, provenance, and structure of the data must be immediately assessable to the researcher.



Seeable

Data must be perceivable and renderable across entirely new analytical contexts without losing its original integrity.



Easily Checkable

Interpretations drawn from the data must be transparently traceable back to the raw artifact for reuse and verification.

Interdisciplinary RSE: Exploration

This phase focuses on establishing a shared understanding of the data and the research context.

- Researchers and RSEs collaborate early as equal partners
- Align research questions, expectations, and available data
- Identify heterogeneous data sources and domain-specific requirements
- Clarify formats, workflows, infrastructure, and expected data growth
- Balance technical feasibility with scholarly practices

Interdisciplinary RSE: Exploration

- Develop a shared conceptual data model for heterogeneous research data
- Define workflows for data organization, publication, and FAIR reuse
- Establish conventions for data collection, naming, and annotation
- Define guidelines to ensure consistency across contributors and disciplines
- Implement backup and data management strategies from the outset

Interdisciplinary RSE: Alignment

This phase begins as soon as new findings, data or requirements make adjustments necessary, reflecting the inherently exploratory nature of research.

- Reassess requirements and intermediate results collaboratively
- Maintain continuous exchange between researchers and RSEs
- Refine data models and software components iteratively
- Improve mechanisms for linking, filtering, and integrating data
- Align development with technical constraints and research needs

Interdisciplinary RSE: Alignment

- Ensure interoperability with external platforms and standards
- Synchronize data across repositories and research systems
- Manage raw and processed data transparently
- Reduce inconsistencies through guidelines and documentation
- Apply version control and EASE-oriented SE practices

Interdisciplinary RSE: Consolidation

This phase focuses on stabilizing and curating results and preparing them for publication.

- Validate results, data structures, and dissemination strategies
- Curate data while preserving technical and scholarly integrity
- Clean and consolidate datasets for traceability and reproducibility
- Transform data into reusable and accessible research outputs
- Ensure consistent metadata, identifiers, and long-term availability

Interdisciplinary RSE: Consolidation

- Finalize publication and dissemination workflows
- Ensure FAIR compliance, interoperability, and long-term accessibility
- Align datasets and annotations with established standards
- Preserve domain-specific interpretative richness and context

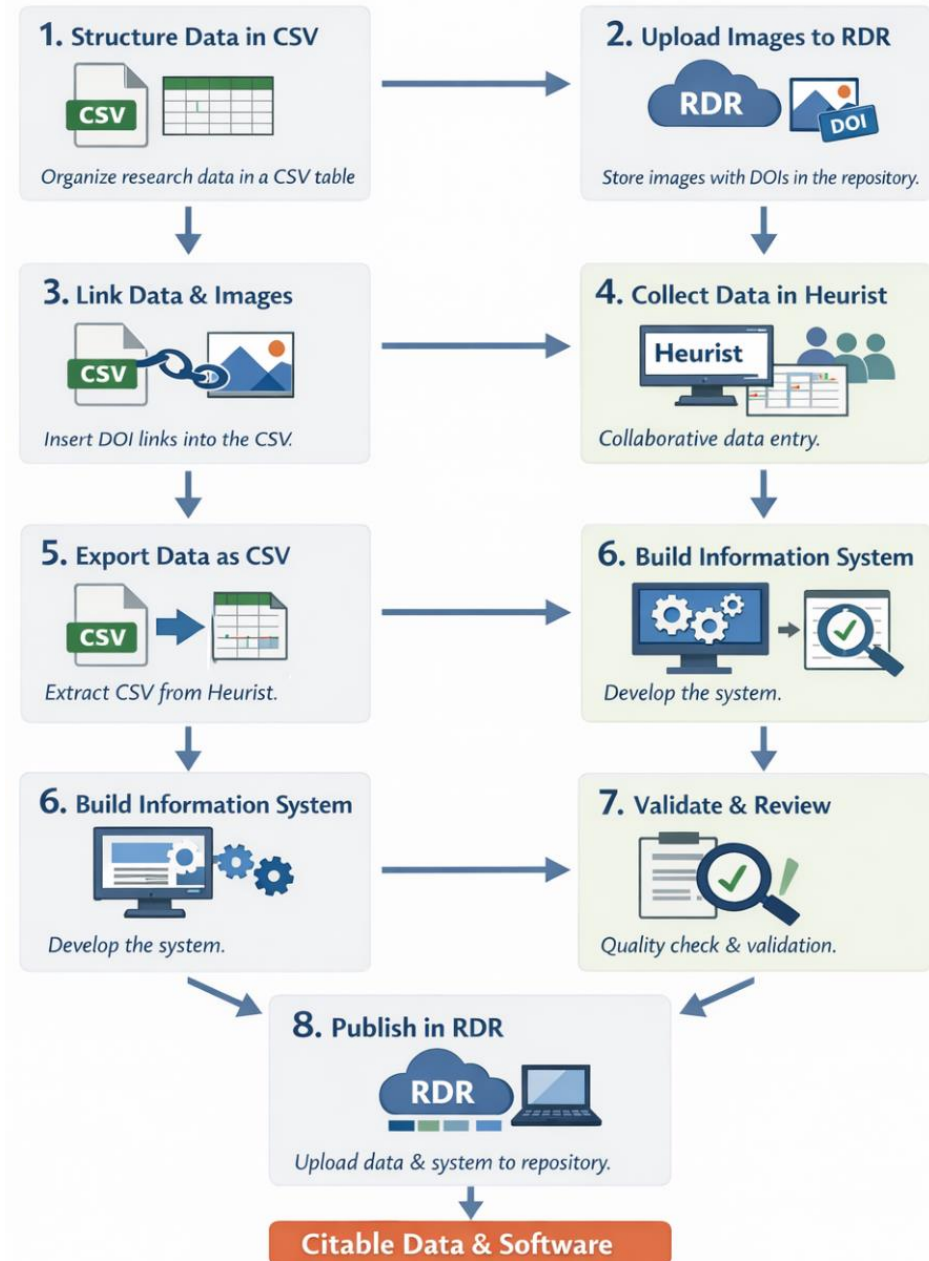
Interdisciplinary RSE: Iteration

The iteration perspective **spans all project phases** and emphasizes that interdisciplinary RSE is not a linear process but a **continuous cycle of refinement, feedback, and adaptation**. Rather than being confined to a single phase, **iteration connects exploration, alignment, and consolidation** through recurring coordination, reassessment, and integration activities.

- Involve research software developers continuously as integral partners in the research process, rather than merely as technical support
- Treat research software and data as evolving, interdependent research objects shaped by both technical and epistemic considerations
- Base decisions on shared understanding of data, domain knowledge, and iterative communication rather than fixed project timelines
- Ensure at least three structured coordination meetings to support alignment across the entire project life cycle, iterative refinement, and task-specific consolidation
- Recognize RSE as a domain-oriented SE practice that bridges disciplinary boundaries and enables sustainable, reusable, and transparent research

Example: Workflow of an Interdisciplinary RSE at UWA I+II

Step-by-Step Implementation of an Interdisciplinary RSE Workflow



Workflow of an Interdisciplinary RSE



Step 1: Structured Data Definition

A structured data model is defined during the initial meeting

Research data are organized in a tabular format (e.g., CSV), where each row represents a research object and columns define attributes, classifications, or references to external resources



Step 2: Image Publication in the RDR

Associated image data are uploaded to an RDR

Each image is assigned a DOI, ensuring long-term accessibility and citability



Step 3: Linking Data and Images

The generated DOIs are inserted into the CSV as URLs in dedicated fields

This creates a stable and explicit link between structured data entries and their corresponding digital images



Step 4: Collaborative Data Collection with Heurist

For data acquisition and curation, Heurist (e.g., hosted at UHH) is used as a collaborative RDM tool

This enables simultaneous data entry by multiple users and ensures structured and controlled data collection as well as automatic data backup via a centralized infrastructure

Workflow of an Interdisciplinary RSE



Step 5: Data Export and Standardization

Once data collection is completed, the dataset is exported from Heurist as a CSV file

This export serves as a standardized interface between data management and software development

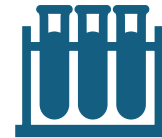
At this stage, the second meeting takes place at the latest so that any necessary adjustments can be made



Step 6: Implementation of the Information System

Based on the structured dataset, an information system is implemented using a generic code base aligned with the EASE model

This allows modular development, iterative refinement, and direct integration of structured research data.



Step 7: Validation and Quality Assurance

The dataset and system are reviewed, cleaned, and validated to ensure consistency, completeness, and reproducibility

The third meeting should take place no later than this stage so that the final adjustments can be made



Step 8: Publication of Data and Software in the RDR

Both the dataset and the developed information system are deposited in the RDR

This ensures that the entire system becomes a citable research output, individual data entries (e.g., CSV rows) remain citable via linked DOIs, and the solution is reusable for future research contexts

Conclusion

- Sustainable software enables sustainable research
- Interdisciplinary collaboration is central to RSE
- Reproducibility and reuse require long-term thinking