

# RESEARCH SOFTWARE NETWORK FOR THE DIGITAL HUMANITIES

Tibor Kálmán

DARIAH-EU & GWDG

deRSE19 – Conference for Research Software Engineers

Potsdam, Germany, 04-06 June 2019



PID of this presentation:  
[hdl.handle.net/21.11101/0000-0007-D423-2](https://hdl.handle.net/21.11101/0000-0007-D423-2)

# WHO AM I?

- Tibor Kálmán
  - Email: *tibor [dot] kalman [at] gwdg [dot] de*
- Based:
  - Göttingen, Germany @ GWDG, [www.gwdg.de](http://www.gwdg.de)
- Background:
  - Computer Science, Phd
- Since mid 90's:
  - hardware, software, software for hardware, 'middleware'
  - industry as well as academia
- Infrastructures
  - Services for eScience @ GWDG
  - CTO of Persistent Identifier Consortium for eResearch (ePIC)
  - co-Head of 'e-Infrastructure' VCC of DARIAH-ERIC (DARIAH VCCI)



today's role

# AGENDA

- Our Context
  - Research Infrastructures
  - ~~Research Software~~ (s. talks on Day #1 and #2)
- Software Engineering
  - ~~Research Software Engineering~~ (s. talks on Day #1 and #2)
  - Infrastructures' View
- Our Network
  - EURISE

# OUR CONTEXT

- RESEARCH INFRASTRUCTURES
- ~~RESEARCH SOFTWARE~~ (S. TALKS ON DAY #1 AND #2)

# RESEARCH INFRASTRUCTURES – THEIR SUSTAINABILITY

- Research projects' limitations
  - RIs ensure **sustainability** per se:
    - clear funding model for their different phases
    - long lifespan
    - binding legal form
      - in case of ESFRI projects: ERIC (specially established for ESFRI projects)
      - NFDI: ???
    - Common understanding that digital RIs do not have a typical project character!
  - Examples:
    - Physics, Climate research, Metrology, since 2006: DH
- European Strategy Forum on Research Infrastructures (ESFRI)
    - established a Roadmap for European RIs
  - **Preparatory Phase:** 2-3 y
    - EU seed funding
  - **(Transition Phase):** 2-3 y
    - integrates national activities
  - **Construction Phase:** 5 y
    - EU member countries share the costs
    - EU funding via research projects
  - **Operational Phase:** 10+/20+ y
    - Members sustain the funding



# EXAMPLE: BRIEF HISTORY OF THE DARIAH-ERIC

- **2006**  
DARIAH @ ESFRI Roadmap
- **2008 – 2011:**  
Preparatory Phase project:  
*Preparing DARIAH*
- **2011 – 2013:**  
Transition Phase  
establishing the DARIAH-ERIC  
integrating national activities
- **2014-:**  
DARIAH-ERIC's construction phase
- **2020-:**  
operational phase for DARIAH-ERIC



## OUR CONTEXT #2: RESEARCH SOFTWARE

Simplified view on research software (for today's talk):

- Facilitates modern research
- Access to and use of data
- Service vs. software vs. application vs. tool vs. etc

“[...] software that is developed within academia and used for the purposes of research: to generate, process and analyse results. This includes a broad range of software, from highly developed packages with significant user bases to short [...] programs written by researchers for their own use.”

➤ Research projects produce demonstrators/prototypes!

Hetrick, Simon (2016): “Research Software Sustainability: Report on a Knowledge Exchange Workshop”

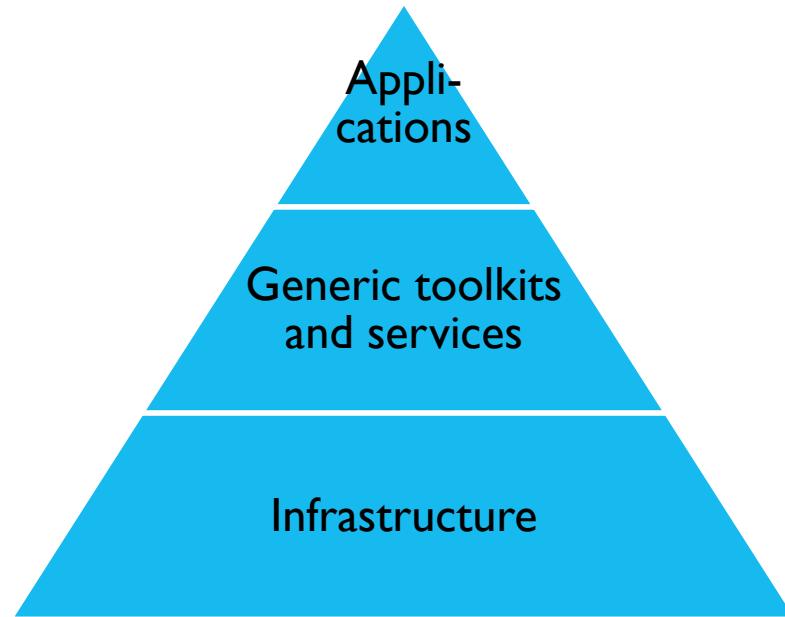
# SOFTWARE ENGINEERING

~~Research Software Engineering~~ (s. talks on Day #1 and #2)

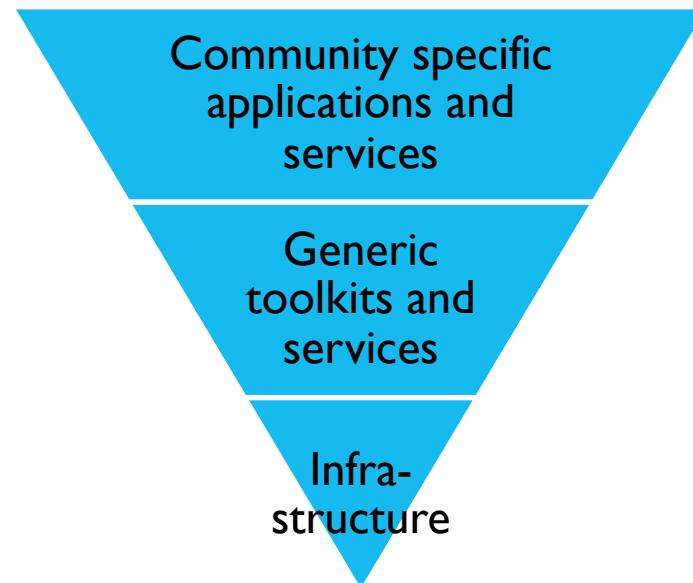
Infrastructures' View

# DIFFERENT INTERESTS IN RESEARCH INFRASTRUCTURES

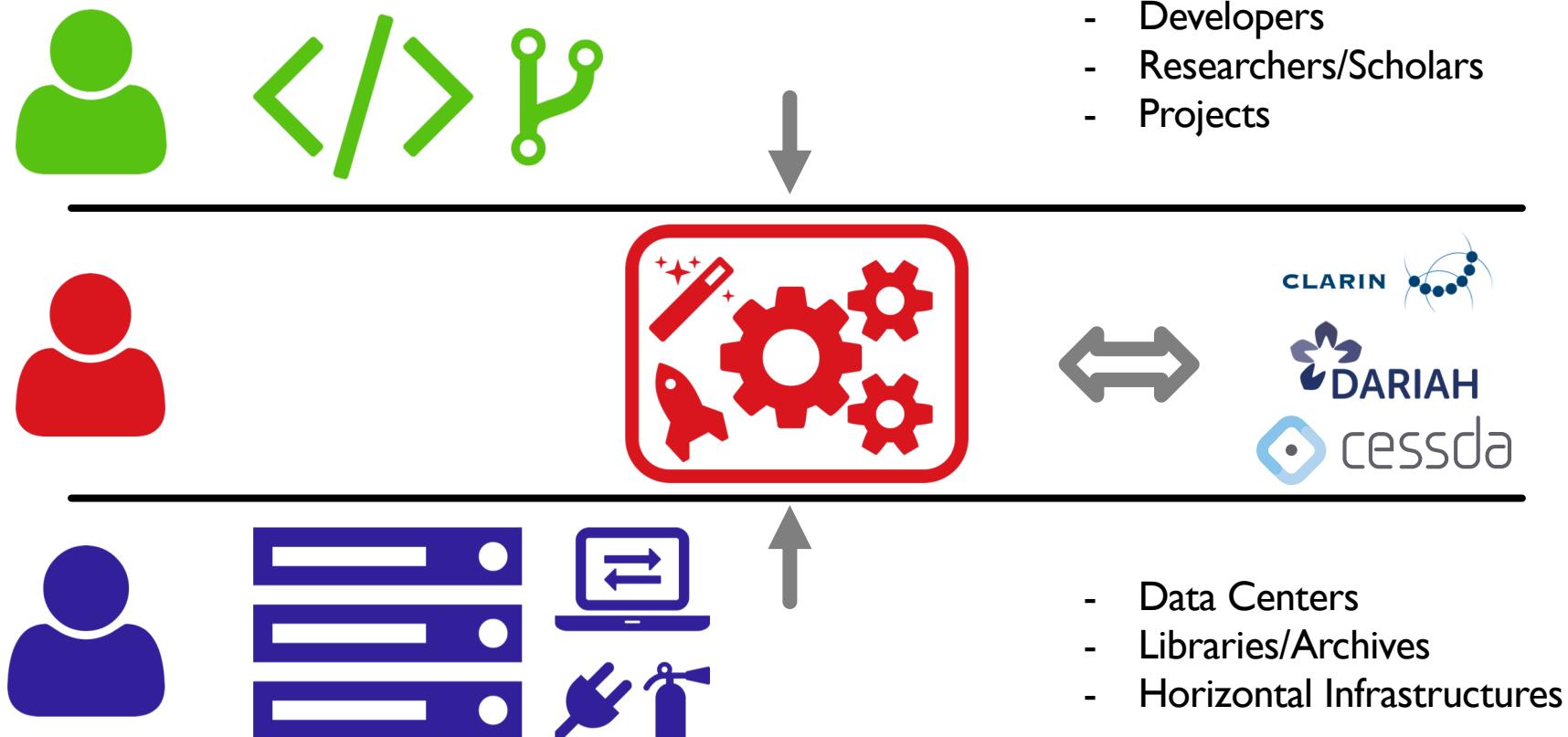
Computing- and Data Centres /  
Providers



Research Institutes / Scholars

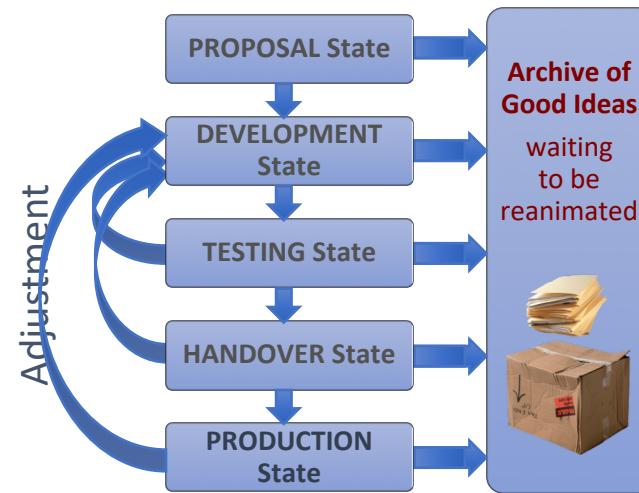


## BRIDGING A GAP



# STANDARDS AND REQUIREMENTS

- Guidelines & Best Practices
- Documentation
- Quality Requirements



## Guidelines for Software Quality

CLARIAH Task 54.100

Maarten van Gompel  
Centre for Language & Speech Technology  
Radboud University, Nijmegen  
Jaaco Noordzij  
Huygens ING

Reinier de Valk & Andrea Scharnhorst  
Data Archiving and Networked Services (DANS)  
Royal Netherlands Academy of Arts and Sciences

September 30, 2016

version 1.1

## Contents

1	Motivation
1.1	Structure of the Document . . . . .
2	Context
3	Introduction - glossary of terms
4	Developer Guidelines: Minimal Requirements
4.1	Configuration 1: Actively Supported End User Software . . . . .
4.2	Configuration 2: Unsupported End User Software . . . . .
4.3	Configuration 3: Actively Supported Experimental Software . . . . .
4.4	Configuration 4: Unsupported Experimental Software . . . . .
5	Quality Assessment Criteria - Usability
5.1	Understandability . . . . .
5.2	Documentation . . . . .
5.3	Learnability . . . . .
5.4	Buildability . . . . .
5.5	Installability . . . . .
5.6	Performance . . . . .
6	Quality Assessment Criteria - Sustainability and Maintainability
6.1	Identity . . . . .
6.2	Copyright & Licensing . . . . .
6.3	Accessibility . . . . .
6.4	Community . . . . .
6.5	Testability . . . . .



## Software Maturity Levels

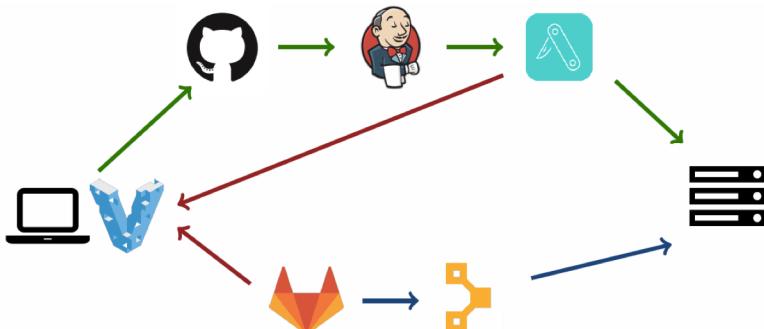
CESSDA Technical Framework Workplan phase 4, D1 (a component of the Technical Architecture)

Status: Public  
Author: John Sheperdson, CESSDA Platform Delivery Director  
Contributors: Members of CESSDA Technical Working Group  
Date: 20 December 2018  
Document: Software Maturity Levels  
Version: Issue 02.00

# SOFTWARE ENGINEERING – INFRASTRUCTURES' VIEW

## Industry Standards

- Automation
- Testing
- Continuous X (Everything)

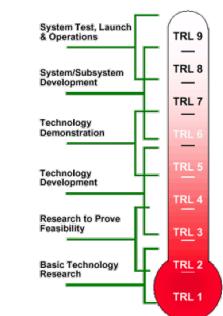


## European Open Science Cloud

- central point of access to digital research in the 21st century
  - technical alignment and interoperability will be key
- EOSC will require Technology Readiness Level 8!
  - We must ensure the software quality of our services
  - We need to be able to provide evidence for our technical maturity
  - We have to operate the services in a reliable manner!

### ➤ Training!

- Software Carpentry &
- Code Refinery



# OPERATIONALISATION

## Operation

Software vs Service

- Running good software the right way

Operation is (also) about management

- Service Lifecycle Management
- Security Management
- Business Models
- ...

Various standards exist:

- ITIL
- FitSM
- REFEDS
- SIRTIFI
- ...



# THE EURISE NETWORK

## European Research Infrastructure Software Engineers' Network

- an umbrella where research infrastructures meet research software engineers
- formed by the three SSH Infrastructures CESSDA, CLARIN & DARIAH

### Events:

- 1st Workshop, Berlin, Autumn 2017 – <https://dhd-blog.org/?p=8685>
- 2nd Workshop, Utrecht, Spring 2019 – <https://dhd-blog.org/?p=11310>
- Activities at RI and RSE Conferences

<https://eurise-network.github.io>

# THE EURISE NETWORK

## Goals

- » Re-Usability of Research Software
- » Infrastructure Sustainability
- » Software Quality
- » Training & Education



The screenshot shows the EURISE Network website. At the top, there is a navigation bar with links for About, Goals, Team, News, and Infrastructures. Below the navigation is a large blue header with the text "EURISE Network" in white. Underneath the header, it says "European Research Infrastructure Software Engineers' Network". The main content area features a section titled "What is the EURISE Network?" with a brief description: "The EURISE Network has been formed by the three Social Sciences and Humanities ERICs CESSDA, CLARIN & DARIAH to create an umbrella where research infrastructures meet research software engineers." There are four main pillars described: "Standards" (represented by a document icon), "Best practices" (represented by a checkmark icon), "Education" (represented by a graduation cap icon), and "Sustainability" (represented by a person icon). Each pillar has a short description below it.

# TECHNICAL REFERENCE

- First common baseline of the three infrastructures
- Collection of best practices & community standards
- Agnostic of platform/vendor
  - It does not define any technology stack, licences, or hosting services
  - Compatible with existing requirements & not-invented-here-syndrome
- For developers, maintainers and operators
  - To gauge the quality of ongoing developments
  - Better yet, use it when starting a new project

The screenshot shows a web page titled "EURISE Network Technical Reference". The header includes a "Docs" link, a "View page source" link, and a "EURISE Network Technical Reference" title. The main content area has a dark background with white text. It features a sidebar with a "CONTENTS" section listing "Developer Guidelines", "Operational Guidelines", "Policy Recommendations", "Software Quality Checklist", "Glossary", and "Bibliography". The main text area discusses the purpose of the document, its funding, and usage guidelines. It also mentions the availability of the document in PDF and epub formats, and its licensing under Creative Commons Attribution 4.0 International. A copyright notice for 2018, CC-BY 4.0, is at the bottom.

# TECHNICAL REFERENCE (CONT'D)

The screenshot shows the 'Basics' section of the EURISE Network documentation. The left sidebar includes links for 'Developer Guidelines', 'Basics', 'The README', 'Documentation', 'Tooling', 'Interoperability', 'Changelog', 'Operational Guidelines', 'Policy Recommendations', 'Software Quality', 'Glossary', 'Bibliography', and a 'DigitalOcean' logo. The main content area starts with a note about open source licences, followed by a numbered list of best practices:

1. Use version control right from the beginning of a new project.
  - If in doubt, use [Git](#).
  - Implement a [Code Hosting Policy](#).
  - Use meaningful commit messages, cf. [\[ProGit\] Sec. 5.2](#):
    - Capitalised summary with a maximum of 50 characters followed by a blank line.
    - Detailed but concise explanations in paragraphs or bullet points at 72 characters line length.
    - Explain what you do and why, but not how.
2. Use an appropriate [OSI](#) approved license.
  - Decide on an appropriate license before you first commit.
  - Ensure the license is compatible with all dependencies.
  - If in doubt, choose [APACHE-2.0](#) or [EUP-1.2](#).
  - Add the text in a [LICENSE.txt](#).
  - Add license statements to code files, consider using [SPDX](#) <<https://spdx.org/specifications>> identifiers.
3. Maintain a [README](#).
4. Document your software properly.
5. Use existing [tooling](#) to support development workflows.
6. Ensure maximal [interoperability](#).
7. Ensure your software is usable and accessible.
8. Implement a [release policy](#) and keep a [changelog](#).
9. Add a code of conduct in a [CODE\\_OF\\_CONDUCT.md](#), like [we do](#). Don't ignore [non-code contributions](#). A legitimate policy can be that external contributions are not accepted and merged.

The screenshot shows the 'Code Hosting Policies' section. The left sidebar includes links for 'Developer Guidelines', 'Documentation', and 'Edit on GitHub'. The main content area has a heading 'Code Hosting Policies' and a sub-section 'General Rules'. It states: 'Make sure to publish your code in a version control repository.' Below this is a bulleted list:

- There are a number of well-known commercial solutions, such as
  - [GitHub](#)
  - [GitLab](#)
  - [Bitbucket](#)

They all offer some free options and using them has a number of advantages, e.g.

- Good and established usability
- High visibility of your code
- Low barrier for findability and re-use
- Good integration with other services and solutions

When using commercial and in particular external services, you must have a backup and data extraction strategy in place, which ensures that you can always move to another solution.

- There are a number of possibilities to host your own solution
  - The commercial solutions above.
  - [GitLab Community Edition](#)
  - [Gogs](#)
  - [Gitea](#)
  - [gitlite](#)

The screenshot shows the 'Documentation' section. The left sidebar includes links for 'Developer Guidelines', 'Documentation', and 'Edit on GitHub'. The main content area starts with a note about the importance of documentation for software usability and usefulness. It states: 'Documentation is fundamental to ensure usability and usefulness of the software. It must be stored along the code, ideally in the repository's [docs](#) folder. Basic documentation should also be included in the [README](#).'

Documentation is relevant in many forms, each of which should be addressed for different audiences with varying degree of experience and knowledge.

1. **User documentation:** Include a documentation for end users, including e.g.
  - Examples
  - Tutorials
  - How-Tos
  - FAQs
  - Screen-casts
  - API documentation
2. **Developer documentation:** Provide instructions for developers.
  - How to set up the environment.
  - Dependencies, including
    - Supported operating systems
    - Required libraries
    - External dependencies
  - Requirements, e.g. hardware, architecture, CPU, RAM, disk space and network bandwidth.
  - How to build the code.
  - How to package the code.Additionally, inline code documentation should be used as appropriate.
3. **Administration documentation:** Provide instructions for installation, configuration and

# QUESTIONS?

Tibor Kálmán – DARIAH-EU & GWDG  
tibor [dot] kalman [at] gwdg [dot] de

deRSE19 – Conference for Research Software Engineers  
Potsdam, Germany, 04-06 June 2019



PID of this presentation:  
[hdl.handle.net/21.11101/0000-0007-D423-2](https://hdl.handle.net/21.11101/0000-0007-D423-2)