Software Architecture: gestern, heute, morgen

Wilhelm (Willi) Hasselbring

Software Engineering, Kiel University, Germany
https://se.informatik.uni-kiel.de

School of Electronics & Computer Science, University of Southampton, UK
https://www.southampton.ac.uk/people/professor-wilhelm-hasselbring

iSAQB, 14. November 2024

C | A | U
Kiel University
Christian-Albrechts-Universität zu Kiel

University of Southampton

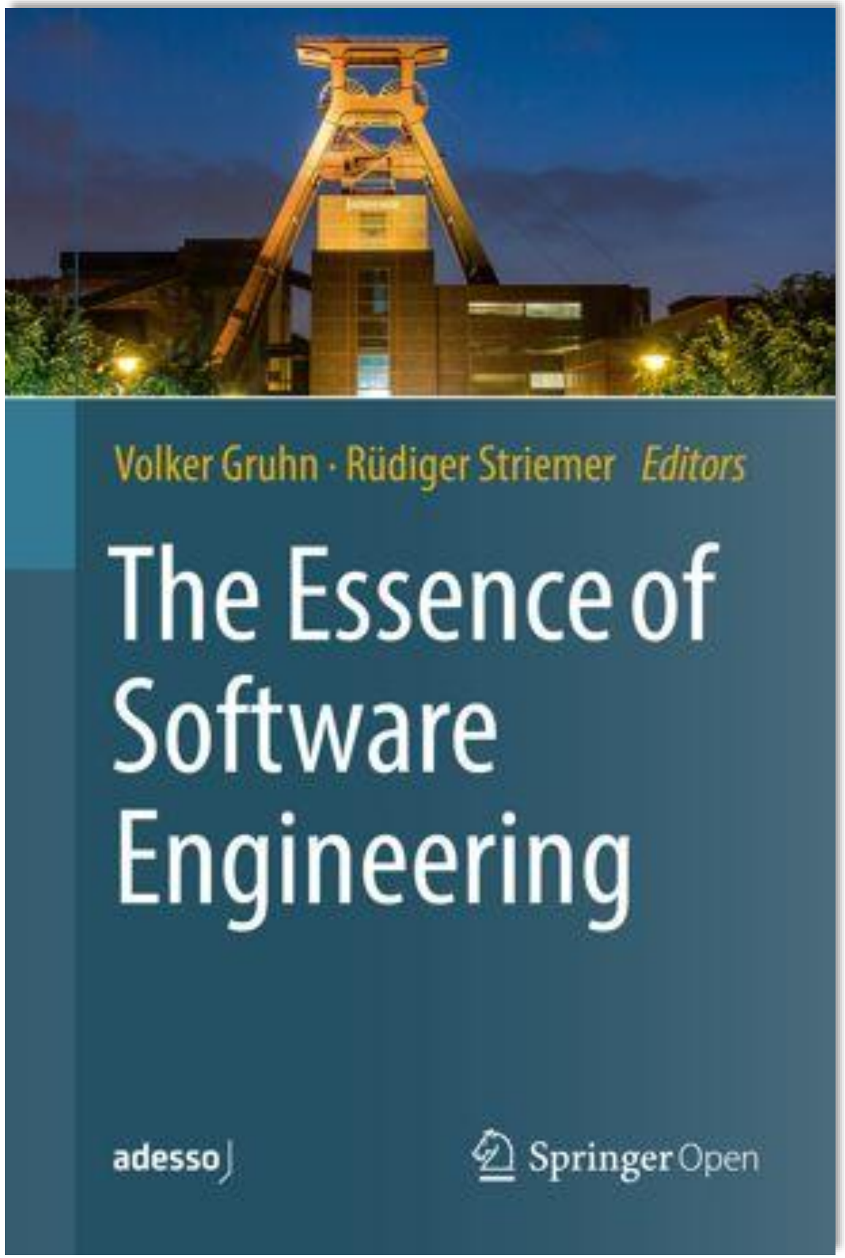
Software Architecture: Past, Present, Future

Wilhelm Hasselbring

1 Introduction

For large, complex software systems, the design of the overall system structure (the software architecture) is an essential challenge. The *architecture* of a software system defines that system in terms of components and connections among those components [55, 58]. It is not the *design* of that system which is more detailed. The architecture shows the correspondence between the requirements and the constructed system, thereby providing some rationale for the design decisions. This level of design has been addressed in a number of ways including informal diagrams and descriptive terms, module interconnection languages, and frameworks for systems that serve the needs of specific application domains. An architecture embodies decisions about quality properties. It represents the earliest opportunity for evaluating those decisions. Furthermore, reusability of components and services depends on how strongly coupled they are with other components in the system architecture. Performance, for instance, depends largely upon the complexity of the required coordination, in particular when the components are distributed via some network. The architecture is usually the first artifact to be examined when a programmer (particularly a maintenance programmer) unfamiliar with the system begins to work on it. Software architecture is often the first design artifact that represents decisions on how requirements of all types are to be achieved. As the manifestation of early design decisions, it represents design decisions that are hardest to change and hence most deserving of careful consideration.

[Hasselbring 2018]

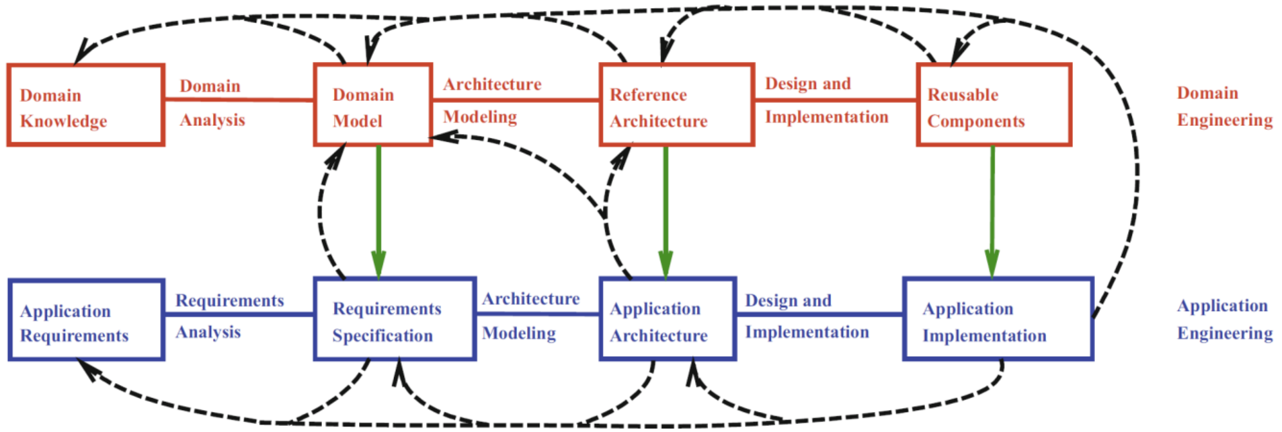


Content

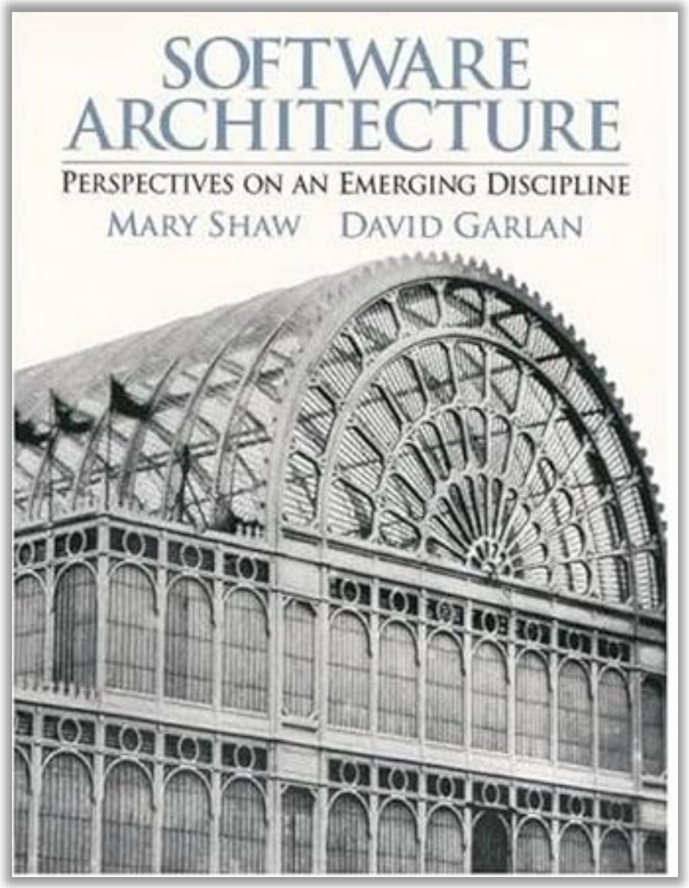
- Software Architecture
  - Past
  - Present
  - Future
- Research Software Architecture
  - Past
  - Present
  - Future
- Research Software Engineering Research

Past:
Focus on Architecture Description and Reuse

- Architecture description languages
- Formalization of architectural models
- Architectural styles and design patterns
- Software product lines for reusing software components



[Hasselbring 2002]



[Shaw and Garlan 1996]

Will Tracz at ICSE 1995, Seattle

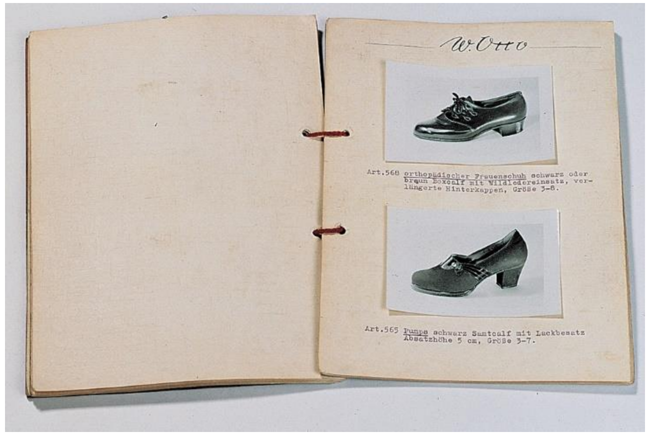
"The state of the art of Software Architecture is like teenage sex:

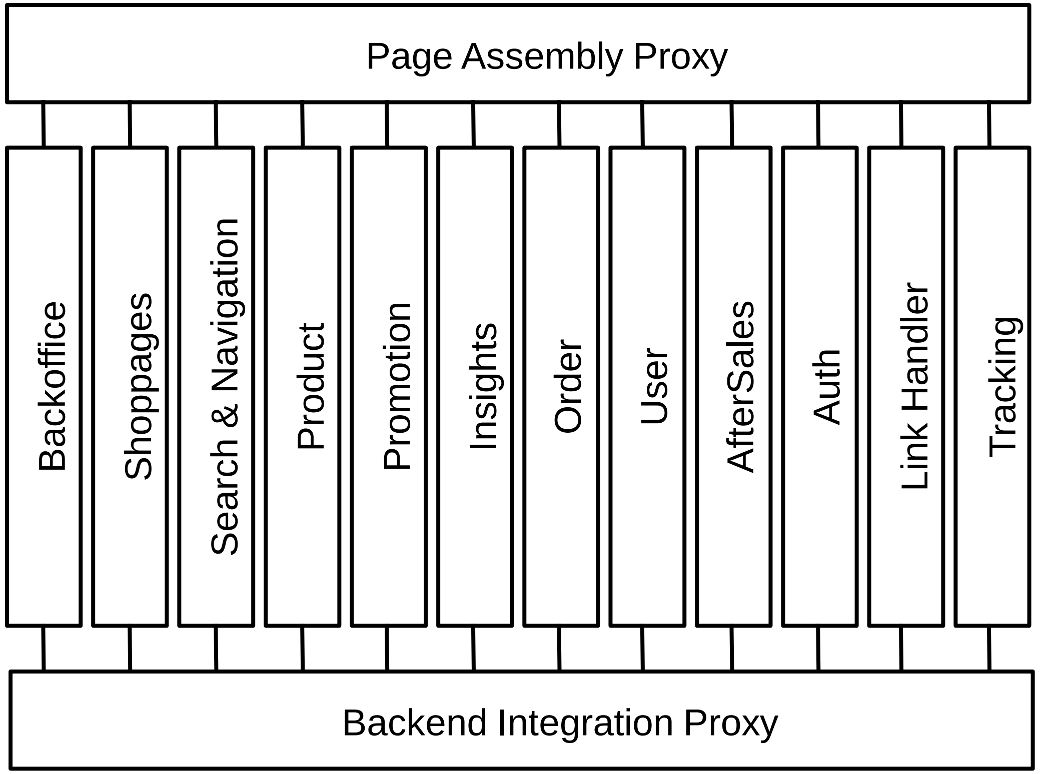
- It's on everybody's mind all the time
- Everyone talks about it all the time
  (but they don't really know what they are talking about)
- Everyone thinks everyone else is doing it
- The few that are doing it:
  - are doing it poorly,
  - think it will be better next time, and
  - are not practicing it safely."

Present: Establishment of Domain-Specific Architectures and Focus on Quality Attributes

- Various domain-specific architectures emerged, particularly from industrial practice. Examples:
  - Data warehouse / lake / data streaming architectures
  - Microservice architectures
- Focus on Quality Requirements
  - Performance
  - Scalability
  - Fault tolerance
  - …

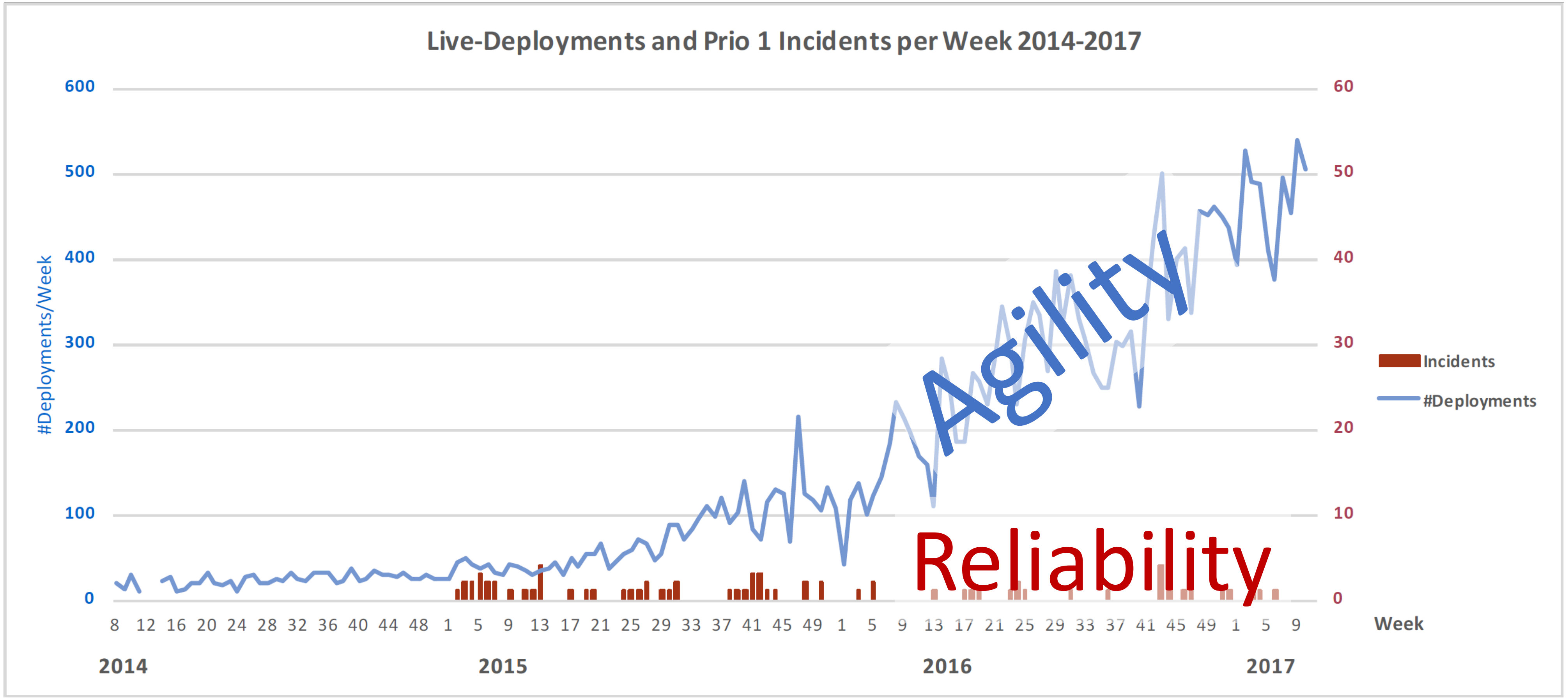
Example: otto.de





Microservices: [Hasselbring 2016, Hasselbring & Steinacker 2017]

Quality: Scalability, Agility and Reliability



[Hasselbring & Steinacker 2017]

Migrating toward Microservices

FOCUS: **MICROSERVICES**

IEEE SOFTWARE

Using Microservices for Legacy Software Modernization

Holger Knoche and Wilhelm Hasselbring, Kiel University

// *Microservices promise high maintainability, making them an interesting option for software modernization. This article presents a migration process to decompose an application into microservices, and presents experiences from applying this process in a legacy modernization project.* //

reduce coordination effort and improve team productivity.

It is therefore not surprising that companies are considering microservice adoption as a viable option for modernizing their existing software assets. Although some companies have succeeded in a complete rewrite of their applications,[2] incremental approaches are commonly preferred that gradually decompose the existing application into microservices.[3] Other approaches to modernization—e.g., restructuring and refactoring of existing legacy applications—are also valid options.[4] However, decomposing a large, complex application is far from trivial. Even seemingly easy questions like "Where should I start?" or "What services do I need?" can actually be very hard to answer.

In this article, we present a process to modernize a large existing software application using microservice principles, and report on experiences from implementing it in an ongoing industrial modernization project. We particularly focus on the process of actually decomposing the

[Knoche and Hasselbring 2018]

Enterprise Modelling and Information Systems Architectures
Vol. 14, No. 1 (2019). DOI:10.18417/emisa.14.1

Drivers and Barriers for Microservice Adoption – A Survey among Professionals in Germany

Holger Knoche[*,a], Wilhelm Hasselbring[a]

[a] Software Engineering Group, University of Kiel, 24118 Kiel, Germany

Abstract. *Microservices are an architectural style for software which currently receives a lot of attention in both industry and academia. Several companies employ microservice architectures with great success, and there is a wealth of blog posts praising their advantages. Especially so-called Internet-scale systems use them to satisfy their enormous scalability requirements and to rapidly deliver new features to their users. But microservices are not only popular with large, Internet-scale systems. Many traditional companies are also considering whether microservices are a viable option for their applications. However, these companies may have other motivations to employ microservices, and see other barriers which could prevent them from adopting microservices. Furthermore, these drivers and barriers possibly differ among industry sectors. In this article, we present the results of a survey on drivers and barriers for microservice adoption among professionals in Germany. In addition to overall drivers and barriers, we particularly focus on the use of microservices to modernize existing software, with special emphasis on implications for runtime performance and transactionality. We observe interesting differences between early adopters who emphasize scalability of their Internet-scale systems, compared to traditional companies which emphasize maintainability of their IT systems.*

Keywords. Microservice architecture • Survey • Software modernization • Microservice adoption

[Knoche and Hasselbring 2019]

Migration von Legacy-Anwendungen

summit – community learning experiences
2.407 Follower:innen
2 Wochen • Bearbeitet •

📢 Last Call – Community „Softwarearchitektur & Softwareentwicklung" am 28. und 29. November 2024 in Leipzig

Schwerpunkt wird die Migration von Legacy-Anwendungen sein. Wir werden über erfolgreiche Migrationsmuster, bewährte Best Practices sowie wichtige Architekturentscheidungen für die Modernisierung und den Austausch von Legacy-Systemen sprechen. Darüber hinaus werden wir modellgetriebene und automatisierte Ansätze für eine effiziente und reibungslose Migration vorstellen.

Dazu teilen diese Expert:innen ihre Erfahrungen:

- 🔻 Dieter Masak (plenum AG - Management Consulting): Legacyzombies in der Cloud
- 🔻 Jannik Zappe (BROCKHAUS AG): Legacy-Software im Wandel: Erfolgreiche Migration zu modernen Architekturen
- 🔻 Heidi Schmidt (PKS Software GmbH): Legacy-Modernisierung in der Praxis einer Förderbank
- 🔻 Andre Lünsmann (Barmenia Krankenversicherung AG): Wenn sich eine Mainframe-Migration wie eine Mondlandung anfühlt: Herausforderungen und Lösungsansätze
- 🔻 Stephan Herold (bitside GmbH): Erfolgreich komplexe Legacy-Systeme modernisieren

https://summit-community.de/veranstaltung/softwarearchitektur-softwareentwicklung/

Future: Proper Integration of Architecture Work into Agile Software Development

- The tension between the agile and architecture communities still is fairly high.
- Ford is often cited for his statements
  - “Architecture is the stuff that’s hard to change later” and
  - “By deferring important architectural and design decisions until the **last responsible** moment, you can prevent unnecessary complexity from undermining your software projects”
- However, making Architectural Decisions is a key IT architect’s responsibility
- Christiane Floyd auf der Tagung Software Engineering im März 2007:

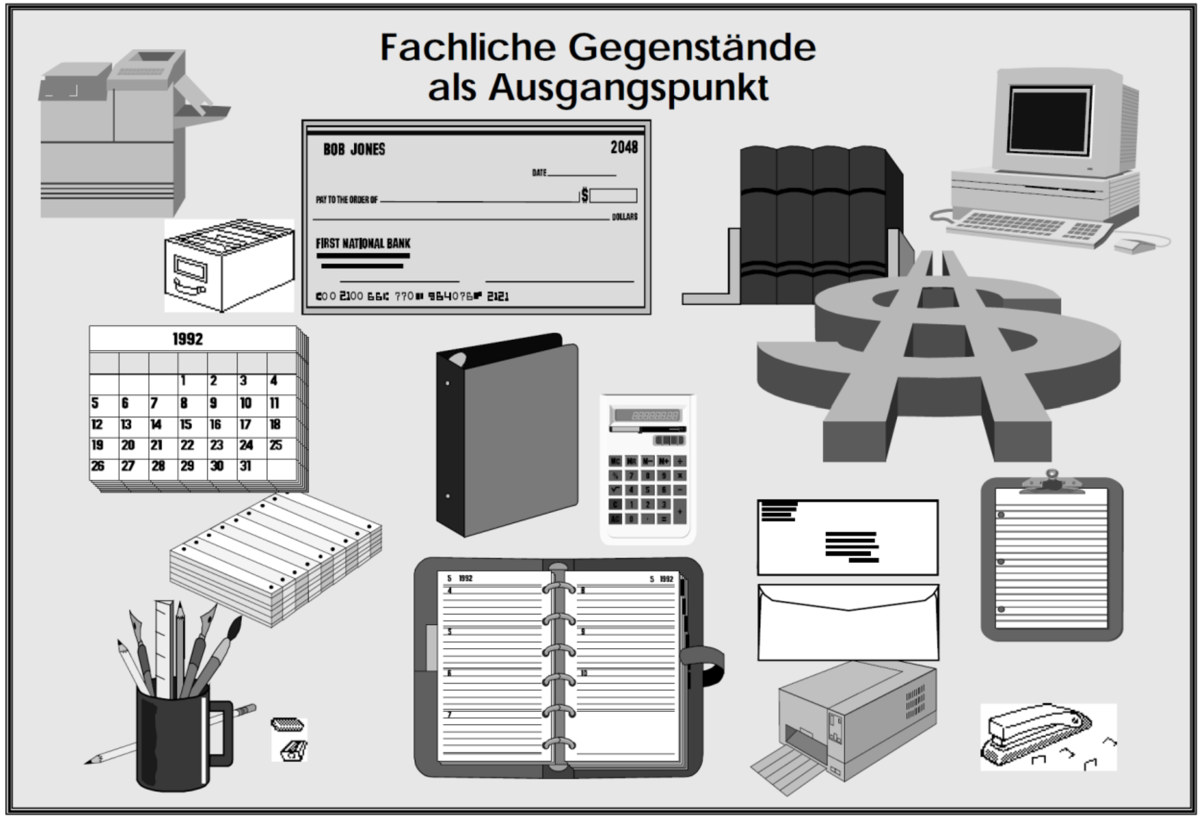
  “Vorgehensmodelle kommen und gehen,
  aber Architekturprinzipien bleiben bestehen.”

Agile Architecture Work

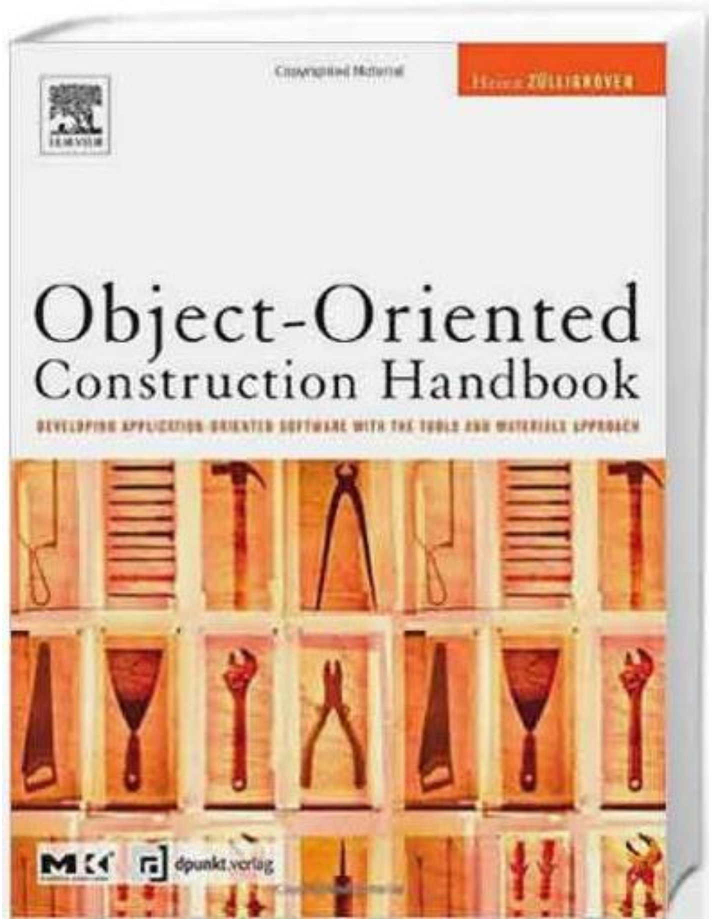
- Finding the **right balance** for architecture work is the art of agile architecture ownership.
- Integrating **Architecture Owners** into Agile Teams
- Architecture owners should decide at the **most responsible** moment, not the **last possible / responsible** moment.
- We can expect a coalescence of architecture work and agile software development practices.

Aus der Forschung in die Praxis

Werkzeug- und Materialansatz (**WAM-Ansatz**)
Werkzeug Automat Material



[Gryczan & Züllighoven 1992]



[Züllighoven 2004]

Flexible Architekturen auch in der Weiterbildung / Zertifizierung

Das CPSA®-Advanced-Level-Modul FLEX – iSAQB®-Training in Flexiblen Architekturmodellen

FLEX

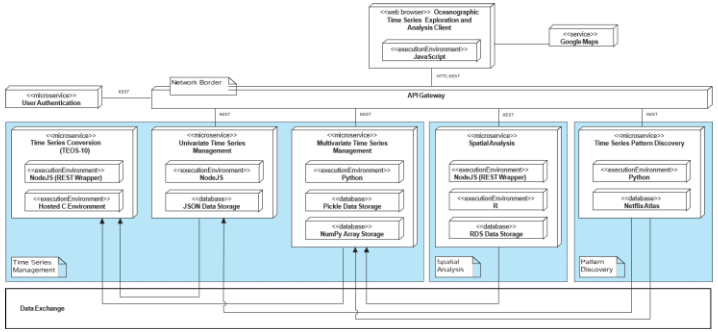
MODUL FLEX

Flexible Architekturmodelle

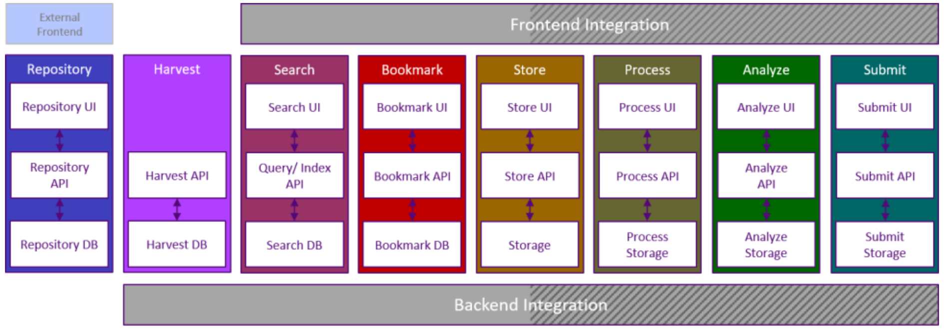
Wie entwirft man besonders flexible Architekturen? Der Lehrplan umfasst moderne Architekturansätze wie Micro-services, Continuous Delivery und Self-contained Systems sowie aktuelle Grundsätze für den Betrieb solcher Lösungen.

iSAQB®
International Software Architecture Qualification Board

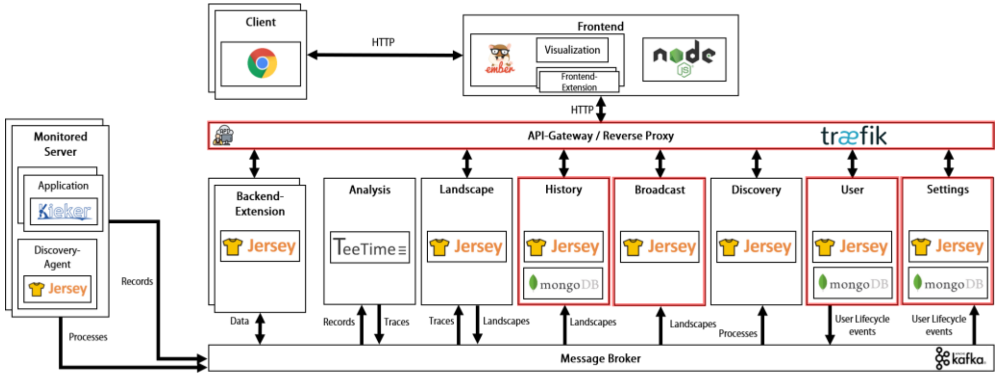
Aus der Praxis in die Forschung



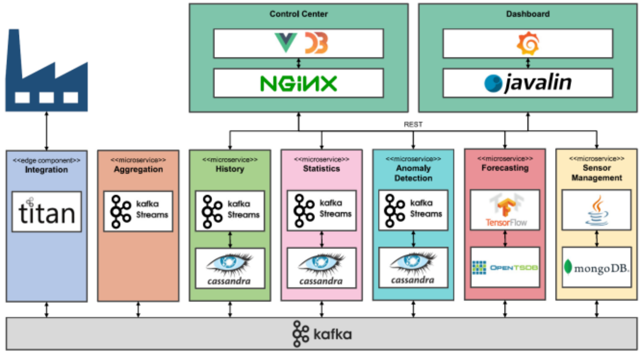
OceanTEA [Johanson et al. 2016]



GeRDI [Tavares de Sousa et al. 2018]



ExporViz [Fittkau et al. 2017]
[Zirkelbach et al. 2019] [Hasselbring et al. 2020]



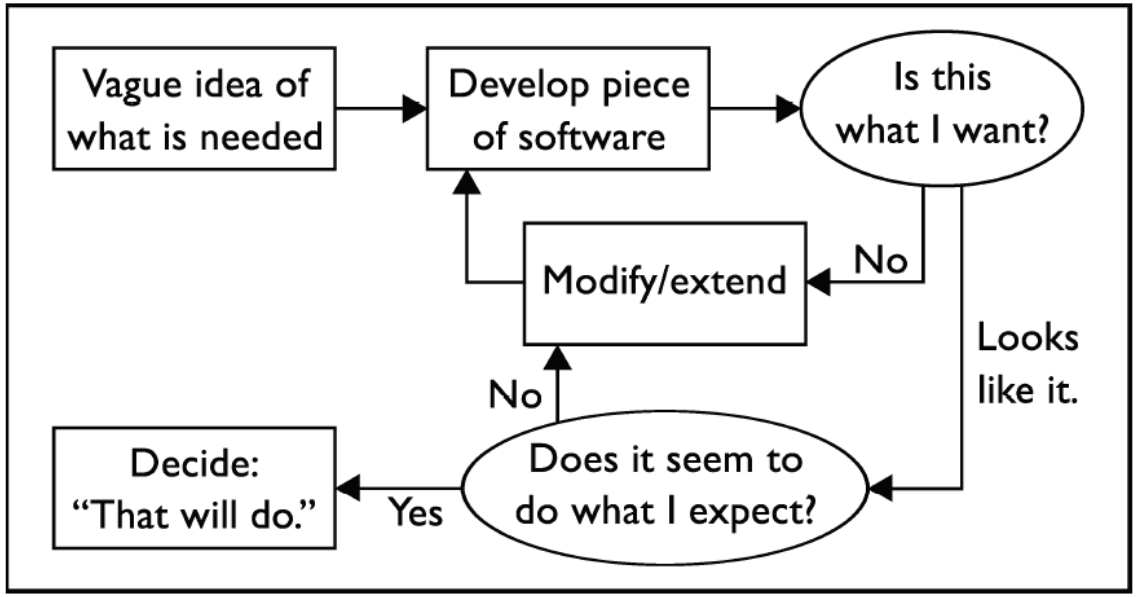
Titan [Henning & Hasselbring 2021]

Some experience with research software

Content

- Software Architecture
  - Past
  - Present
  - Future
- **Research Software Architecture**
  - Past
  - Present
  - Future
- Research Software Engineering Research

Past, Present, Future



[Johanson & Hasselbring 2018]

Software Engineering for Computational Science:

Past, Present, Future

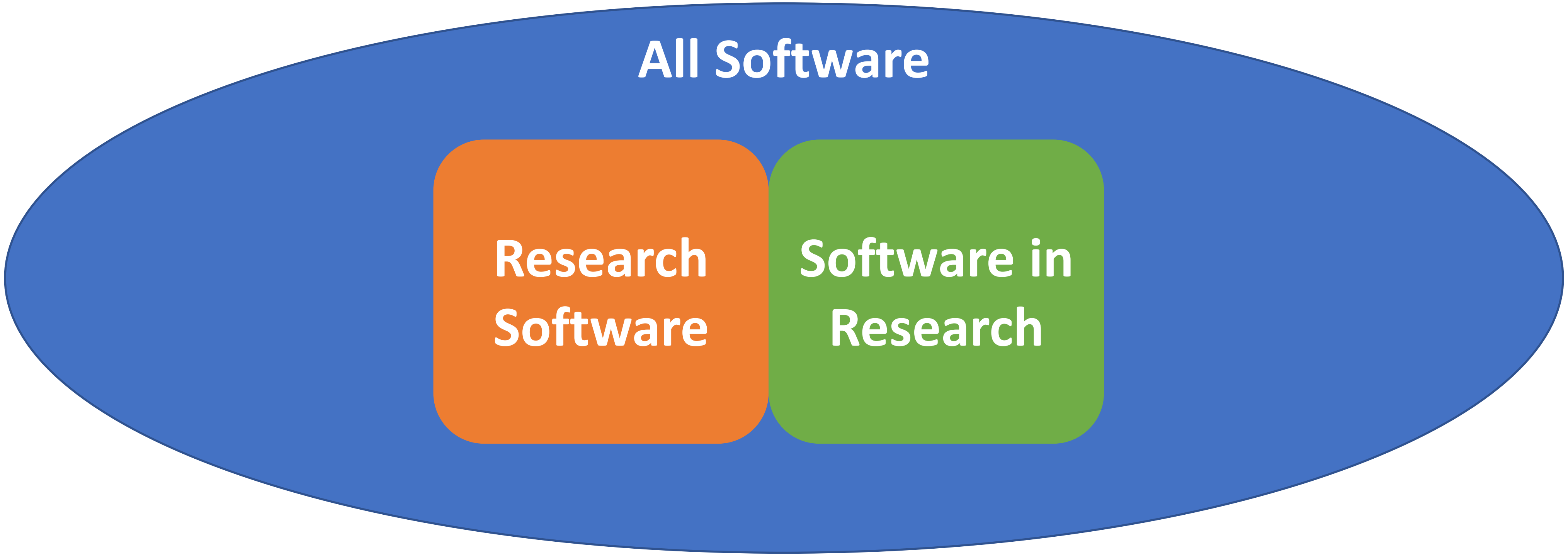
Arne N. Johanson
XING Marketing Solutions GmbH

Wilhelm Hasselbring
Kiel University

Editors:
Jeffrey Carver, carver@cs.ua.edu; Damian Rouson, damian@sourceryinstitute.org

Despite the increasing importance of in silico experiments to the scientific discovery process, state-of-the-art software engineering practices are rarely adopted in computational science. To understand the underlying causes for this situation and to identify ways to improve it, we conducted a literature survey on software engineering practices in computational science. We identified 13 recurring key characteristics of scientific software development that are the result of the nature of scientific challenges, the limitations of computers, and the cultural environment of scientific software development. Our findings allow us to point out shortcomings of existing approaches for bridging the gap between software engineering and computational science and to provide an outlook on promising research directions that could contribute to improving the current situation.

Software Segmentation



[Chue Hong et al. 2022]

Research Software
created during the research process or for a research purpose
Software in Research
used for research but not created during or with research intent

deRSE | GESELLSCHAFT FÜR FORSCHUNGSSOFTWARE

Research Software & Research Software Engineering

- Research software is software
  - that is employed in the scientific discovery process or
  - a research object itself.
- Computational science (also scientific computing) involves the development of research software
  - for model simulations and
  - data analytics

  to understand natural systems answering questions that neither theory nor experiment alone are equipped to answer.

RSE
SOCIETY OF RESEARCH SOFTWARE ENGINEERING

SOFTWARE SUSTAINABILITY INSTITUTE

https://www.software.ac.uk/

RSG Research Software Group

University of Southampton

Driving rese

WELCOME | WHO WE ARE | HOW WE WORK | PROJECTS | TRAINING | HIGH PERFORMANCE COMPUTI | CONTACT US | JOBS

WELCOME

Southampton Research Software Group

Our talented team of **Research Software Engineers** is dedicated to ensuring that software developed for research and enterprise at the University of Southampton is the best it can be.

https://rsgsoton.net/

MANCHESTER 1824

The University of Manchester

Search

Research IT

Research Software Engineering

The Research Software Engineering (RSE) team enhances the University's capacity to produce high quality research software by collaborating with researchers to create correct, efficient, readable, reliable and sustainable code.

https://research-it.manchester.ac.uk/

epcc

Research software engineering

EPCC is one of the founding groups involved in research software en
lead site for the Software Sustainability Institute, it was instrumental
development of research software engineering as a profession and,
an area of research. EPCC collaborates with researchers from across
topics related to research software policy and practice.

https://www.epcc.ed.ac.uk/research/research-themes

Research Software should be open and FAIR

Open Source Research Software

Wilhelm Hasselbring, Kiel University
Leslie Carr, University of Southampton
Simon Hettrick, Software Sustainability Institute and University of Southampton
Heather Packer and Thanassis Tiropanis, University of Southampton

For good scientific practice, research software should be open source. It should be both archived for reproducibility and actively maintained for reusability.

are reused. To study the state of the art in this field, we analyzed research software publishing practices in computer and computational science and observed significant differences: computational science emphasizes reproducibility, while computer science emphasizes reuse.

SOFTWARE ENGINEERING FOR SUSTAINABLE RESEARCH SOFTWARE

[Hasselbring et al. 2020a, 2020b]

Research Software Engineering

Forschungssoftware effizient erstellen und dauerhaft erhalten

| LARS GRUNSKE | ANNA-LENA LAMPRECHT | WILHELM HASSELBRING | BERNHARD RUMPE | **Viele Forschungsprojekte an Universitäten sind ohne entsprechende Software nicht mehr denkbar. Software entwickelt sich zur relevanten Infrastruktur, die gepflegt, weiterentwickelt und gewartet werden muss. Mit Research Software Engineering (RSE) sollen geeignete Rahmenbedingungen geschaffen werden. Handlungsempfehlungen im Überblick.**

Der Begriff „Forschungssoftware" (engl. „research software") bezeichnet Software, die während des Forschungsprozesses oder für einen Forschungszweck erstellt wird. Forschungssoftware ist heute für viele wissenschaftliche Aktivitäten zwingend erforderlich. Sie kann zum Sammeln, Verarbeiten, Analysieren und Visualisieren von Daten, zur Erkennung von Zusammenhängen und zur Modellierung komplexer Phänomene und zur Durchführung anspruchsvoller Simulationen vom Material- über das Zell- und Organverhalten, soziale und ökonomische Beobachtungen, über das Wetter, das Klima der Erde bis hin zu Galaxienhaufen verwendet werden. Forschungssoftware spielt daher heute in fast allen Fächern eine entscheidende Rolle für die Forschung.

AUTOREN


Foto: privat

Lars Grunske ist Professor für Software Engineering an der Humboldt-Universität zu Berlin.

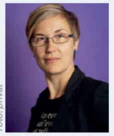


Foto: privat

Anna-Lena Lamprecht ist Professorin für Software Engineering an der Universität Potsdam.


Foto: privat

Wilhelm Hasselbring ist Professor für Software Engineering an der Universität zu Kiel.


Foto: privat

Bernhard Rumpe ist Professor für Software Engineering an der RWTH Aachen.

50 Jahre Software Engineering

Software Engineering (SE) hat sich in fast allen Universitäten und Fachhochschulen als eigenständiges Forschungsgebiet etabliert. Dabei haben die Professorinnen und Professoren durch ihre Forschung ein umfassendes Verständnis über die systematische und ingenieurtechnische Softwareentwicklung erarbeitet und dies nachhaltig in der Industrie etabliert. Dieses Wissen ist in Teilbereichen des Software Engineering wie etwa Anforderungsmanagement, Architektur, Design, Modellierung, Testen, Entwicklungsprozesse und angewandte formale Methoden organisiert, die sich weit über die Programmierung hinaus erstrecken.

Das Gebiet des Software Engineering entwickelt sich dennoch kontinuierlich weiter, weil neue Technologien neue Arten von Software und damit neue Herausforderungen für das Software Engineering mit sich bringen: Software ist sehr heterogen und reicht von eingebetteter Software und autonomen Steuerungen bis hin zu Desktop- und KI-Systemen, Geschäftssoftware und auch Forschungssoftware. Dabei sind die Probleme immer die gleichen:

- Wie lässt sich sicherstellen, dass die Software richtig und korrekt funktioniert?
- Wie kann die Qualität der Software sichergestellt werden?
- Wie lässt sich Software effizient entwickeln?
- Wie kann Software weiterentwickelt und langfristig nutzbar erhalten werden?
- Wie lassen sich Zeitvorgaben und Budgetbeschränkungen einhalten?
- Wie kann Software verallgemeinert werden, um mehr Nutzerinnen und Nutzer zu finden?

Die Lösungen und die sich daraus ergebenden Entwicklungstechniken sind in den verschiedenen Teilaktivitäten der Softwareentwicklung jedoch zumeist sehr unterschiedlich, denn unter anderem die Ausgangssituation, die Art der Software, die Komplexitätstreiber, die benötigten Qualitätsmerkmale, der Kontext, in dem die Software eingesetzt werden soll, sowie die regulatorischen Vorgaben unterscheiden sich stark. Die Software Engineering Community hat durch ihre Forschung schon viele Innovationen angeschoben, die oft auch breitere Nutzung finden. Dazu gehören zum Beispiel Wikis (die Grundlage der Wikipedia), agile Entwicklungsprozesse, Open Source (als Vorlage für Open Science) und eine Vielzahl an Werkzeugen zur Automatisierung in der Produktentwicklung, Informationsgewinnung mit Entwicklungs-Dashboards, für kollaborative Arbeitstechniken, für Versions- und Variantenmanagement und noch einiges mehr. Variantenmanagement mit Produktlinien, explizites Anforderungsmanagement und modellbasierte Entwick-

RSE Praxis
Bewährte Praktiken für die Entwicklung von Software im Forschungsalltag

RSE Training
Entwicklung von (R)SE-Fähigkeiten bei Forschenden und von R(SE)-Fähigkeiten bei Softwareentwickler/-innen

RSE Infrastruktur
Unterstützung bei Entwicklung, Betrieb und Wartung von Forschungssoftware

RSE Community

RSE Karrierepfade
Entwicklung von RSE als eigenes Berufsprofil und Karrierewegen für RSEs

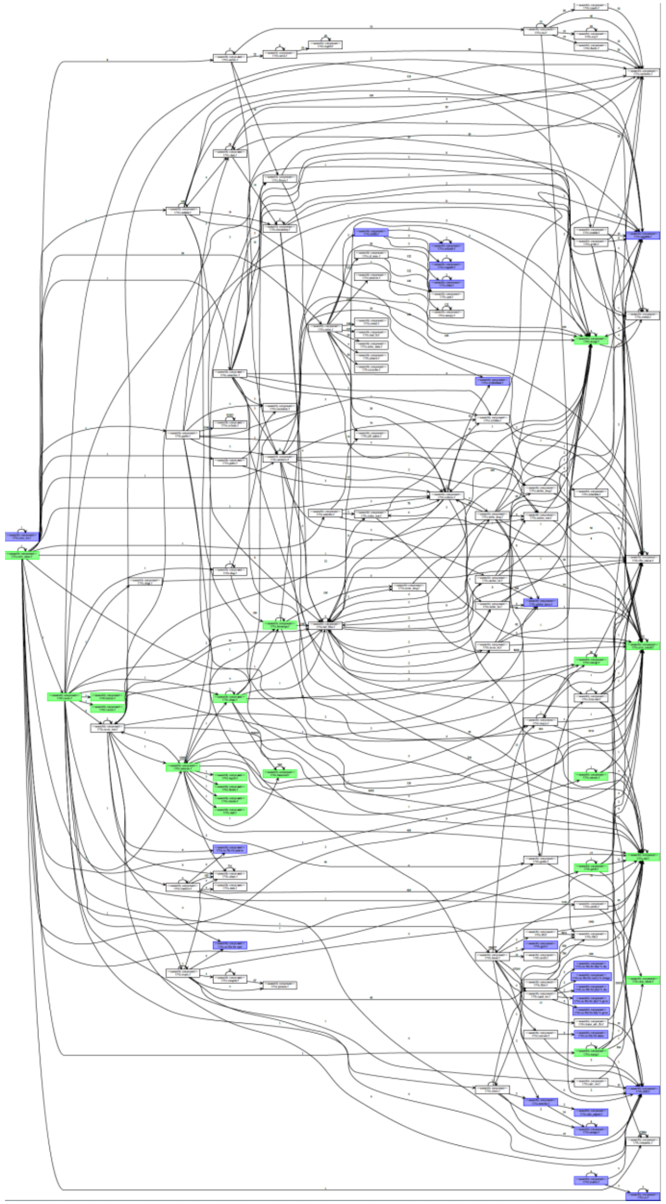
RSE Interessen-vertretung
für institutionelle Unterstützung, Finanzierung und Anerkennung von RSE und RSEs

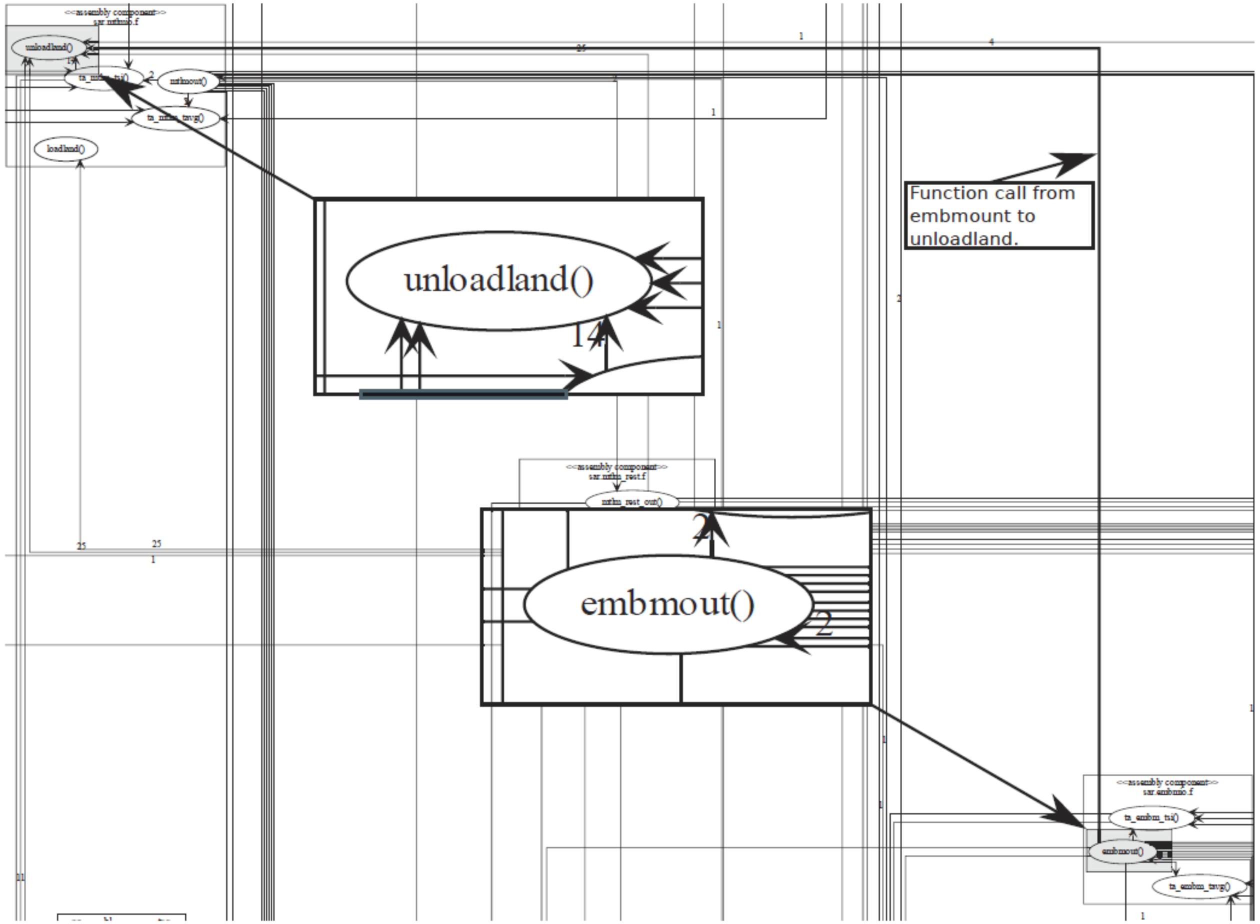
RSE Forschung
Analyse und Verbesserung (des Entwicklungsprozesses) von Forschungssoftware

Content

- Software Architecture
  - Past
  - Present
  - Future
- Research Software Architecture
  - **Past**
  - Present
  - Future
- Research Software Engineering Research

Analyzing the structure of UVic for modularization





[Jung et al. 2021,
Claus et al. 2022]

OceanDSL

Funded by DFG Deutsche Forschungsgemeinschaft German Research Foundation

GEOMAR

Content

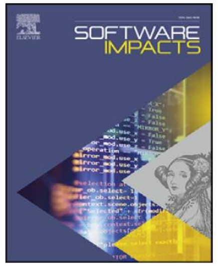
- Software Architecture
  - Past
  - Present
  - Future
- Research Software Architecture
  - Past
  - **Present**
  - Future
- Research Software Engineering Research

Modular Scientific Code

Contents lists available at ScienceDirect

Software Impacts

journal homepage: www.journals.elsevier.com/software-impacts



Eulerian-Lagrangian fluid dynamics platform: The `ch4-project`

Enrico Calzavarini

Highlights

- Ch4-project is a fluid dynamics code used in academia for the study of fundamental problems in fluid mechanics.
- It has contributed to the understanding of global scaling laws in non-ideal turbulent thermal convection.
- It has been used for the characterisation of statistical properties of bubbles and particles in developed turbulence.
- It is currently employed for a variety for research projects on inertial particle dynamics and convective melting.
- **Its modular code structure allows for a low learning threshold and to easily implement new features.**

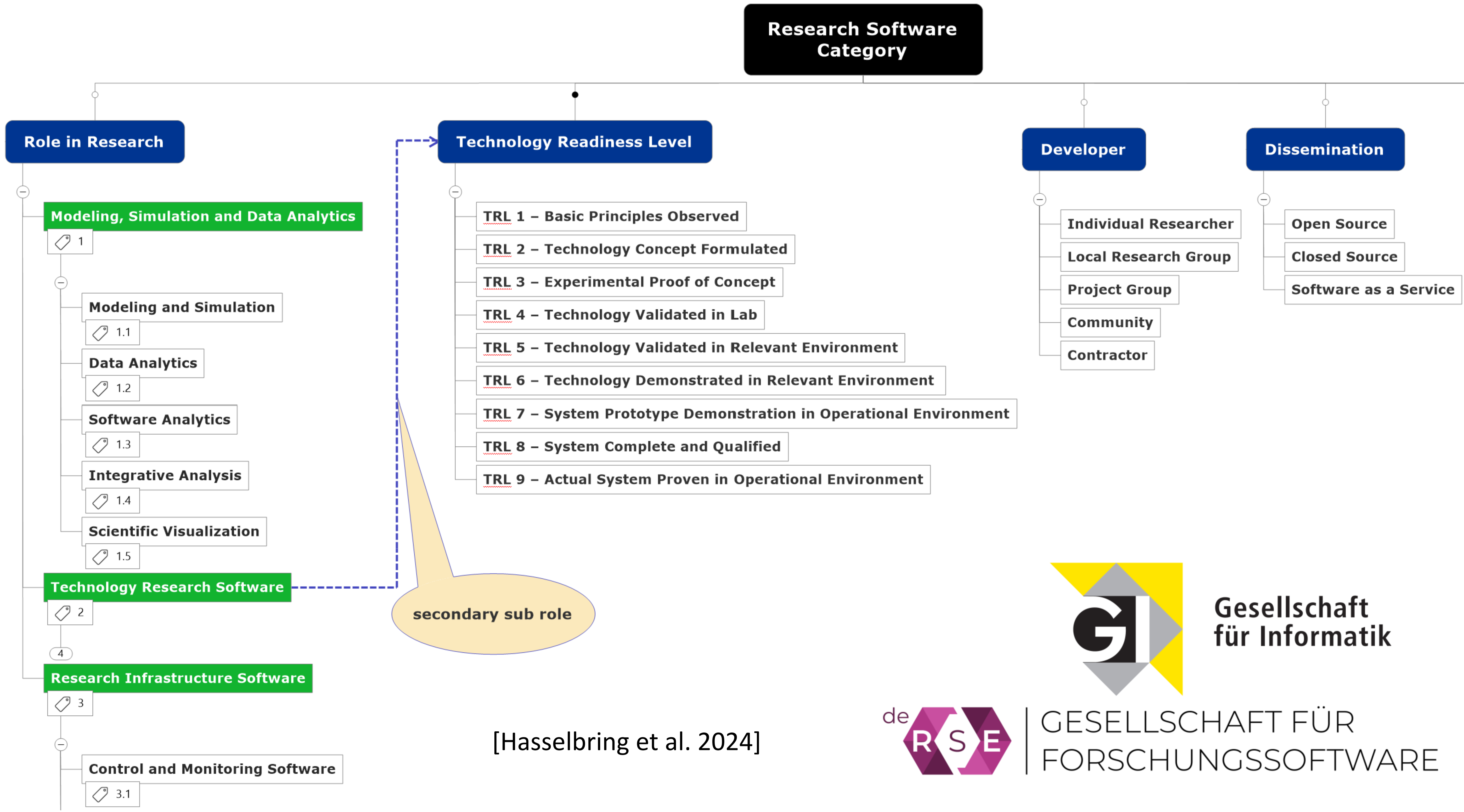
Modular Scientific Code

From [Calzavarini 2019]:

- “A dream for principal investigators in this field is to not have to deal with different (and soon mutually incompatible) code versions for each project and junior researcher in his/her own group.
- In this respect an **object-oriented modular** code structure would be the ideal one,
  - but this makes the code less prone to modifications by the less experienced users.
- The choice made here is to rely on a systematic use of **C language preprocessing** directives and on a **hierarchical naming convention** in order to configure the desired simulation setting in a module-like fashion at compiling time.”

Content

- Software Architecture
  - Past
  - Present
  - Future
- Research Software Architecture
  - Past
  - Present
  - **Future**
- Research Software Engineering Research

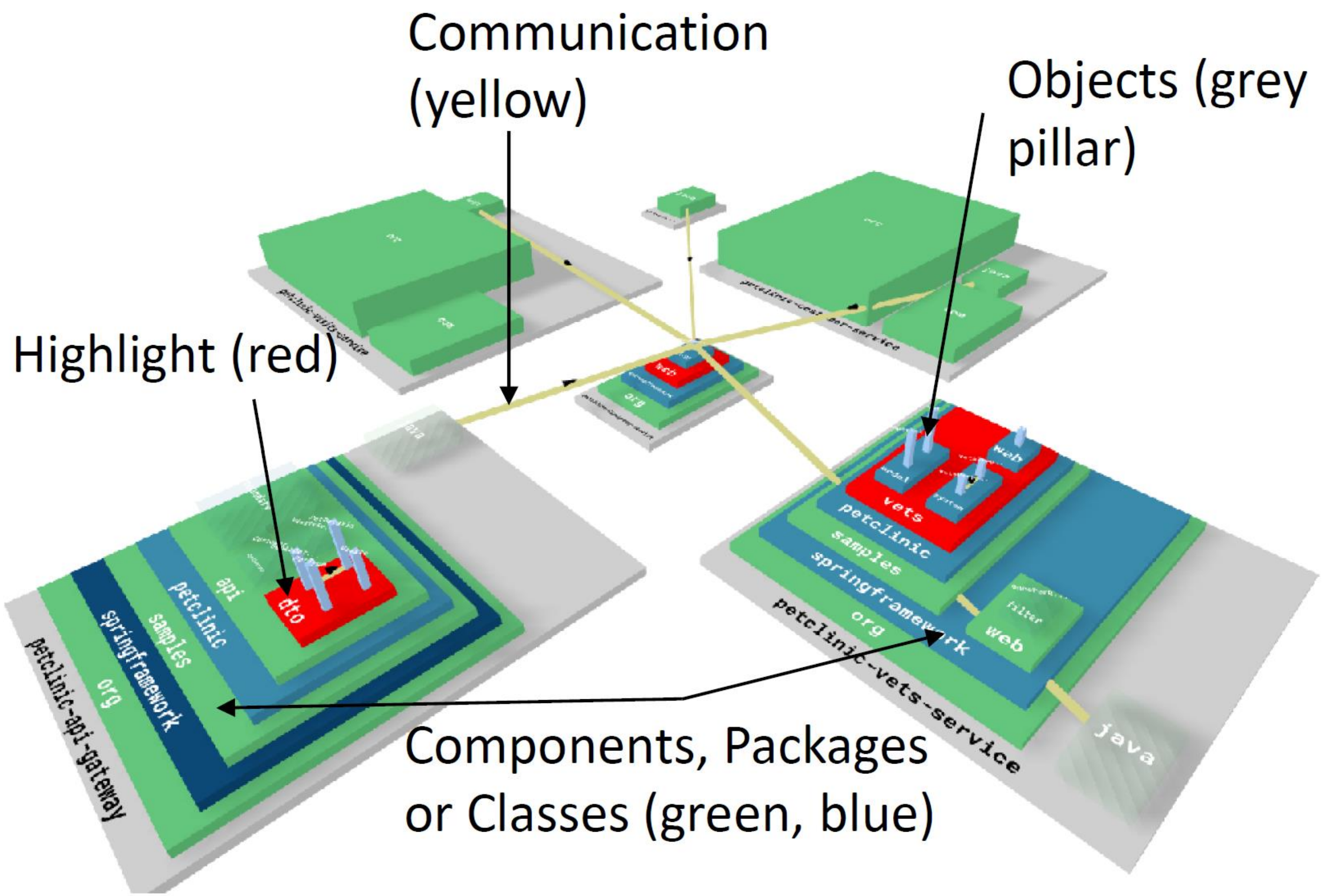


[Hasselbring et al. 2024]

Gesellschaft für Informatik

deRSE GESELLSCHAFT FÜR FORSCHUNGSSOFTWARE

Dynamic Software Analysis with the Technology Research Software ExplorViz



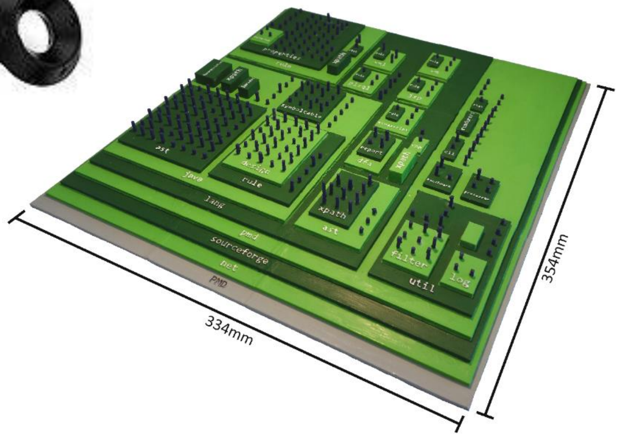
[Fittkau et al. 2013, 2014, 2015e, 2017, Hasselbring et al. 2020]

Experimentation with various Hardware Devices

Virtual Reality

[Fittkau et al. 2015c]

3D Print

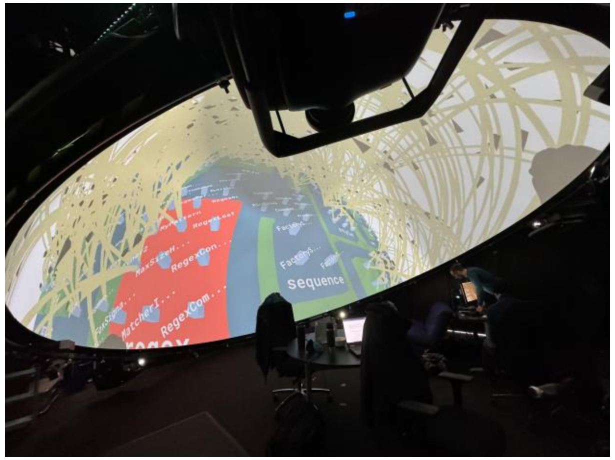


[Fittkau et al. 2015d]

Augmented Reality

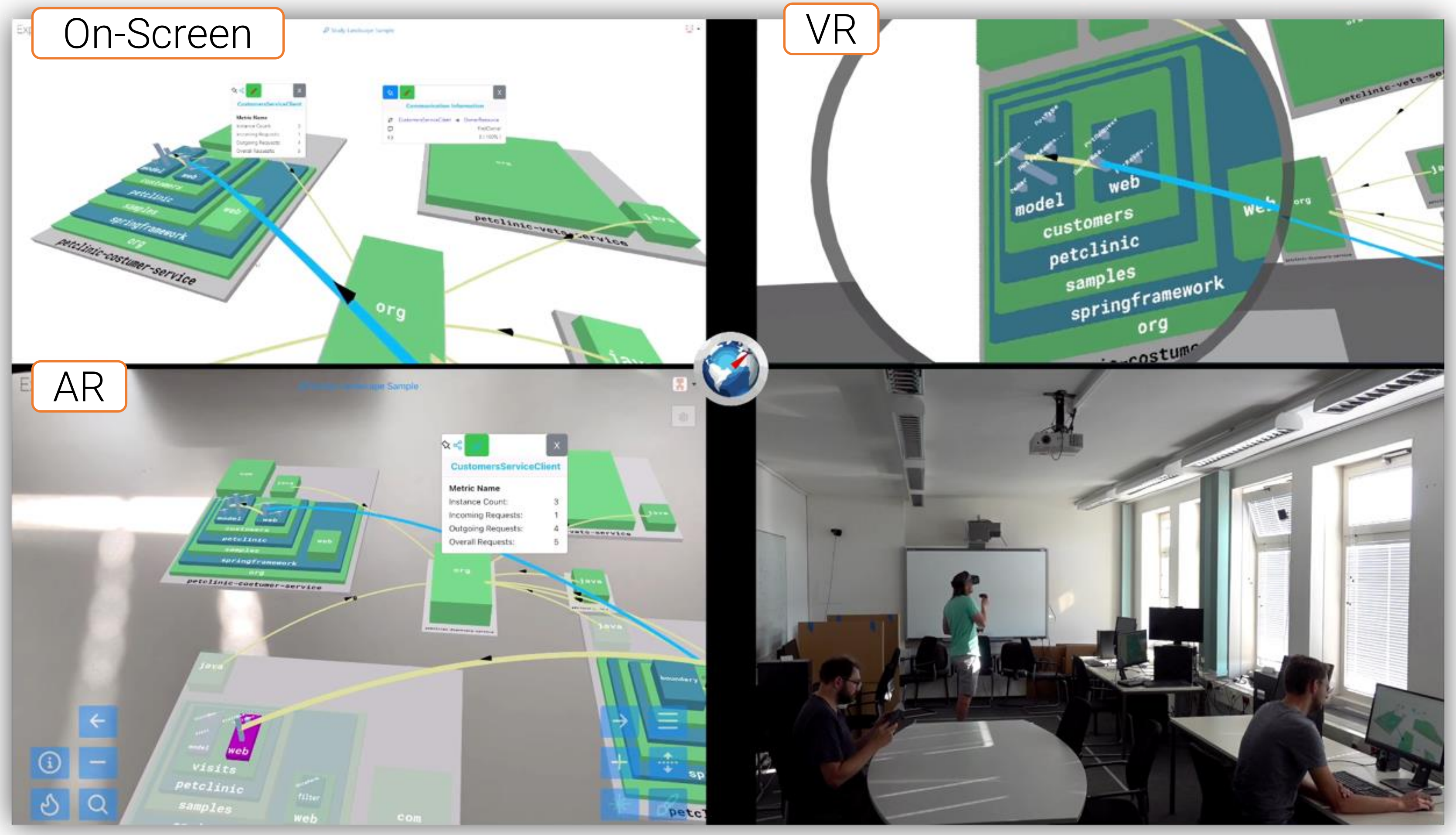
[Krause et al. 2021]

Projection Dome



[Hansen et al. 2024]

Multi-User Collaboration

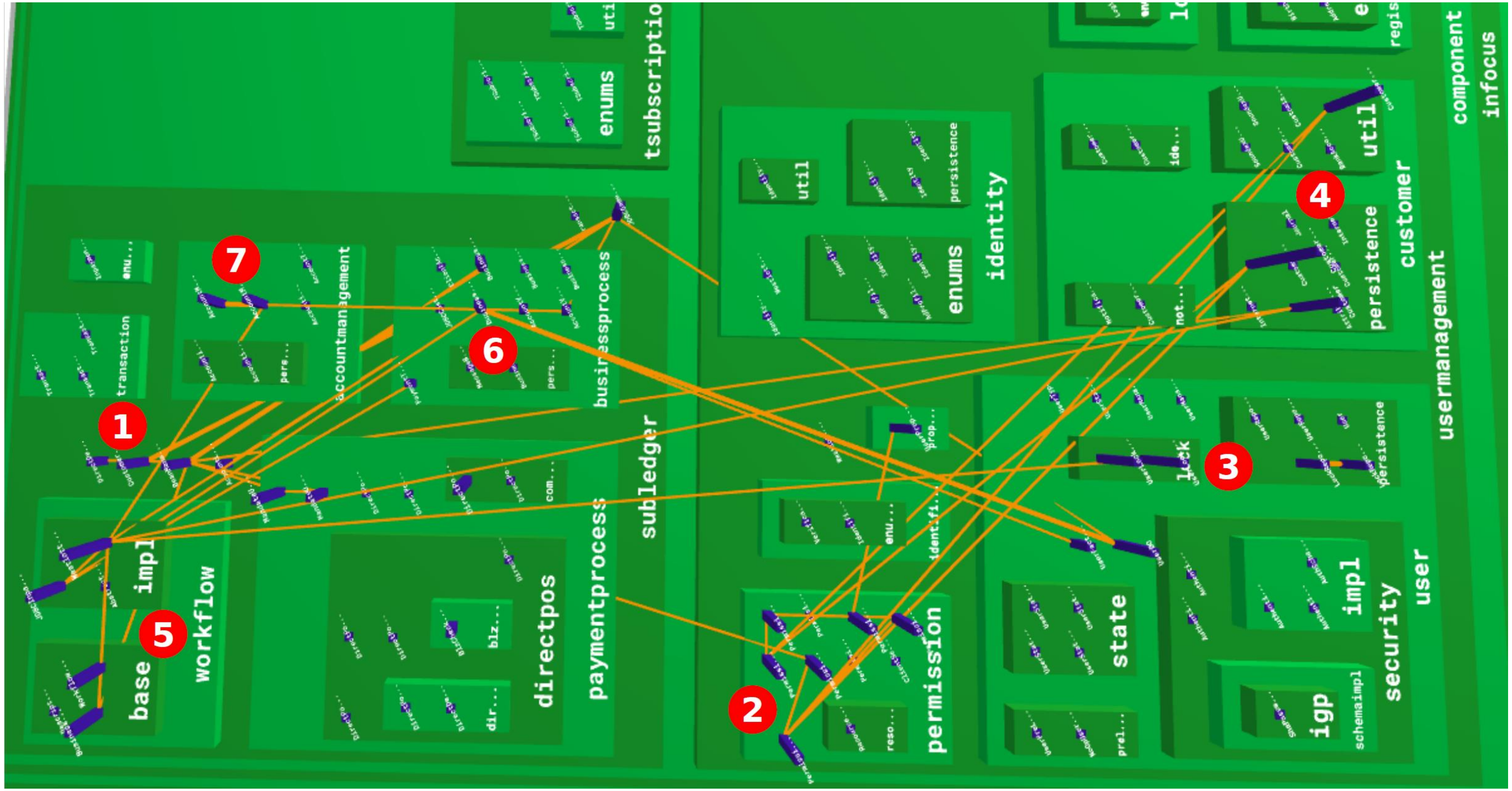


[Krause et al. 2022, Krause-Glau et al. 2022, 2024a, 2024b
Krause-Glau and Hasselbring 2022, 2023]

in|FOCUS

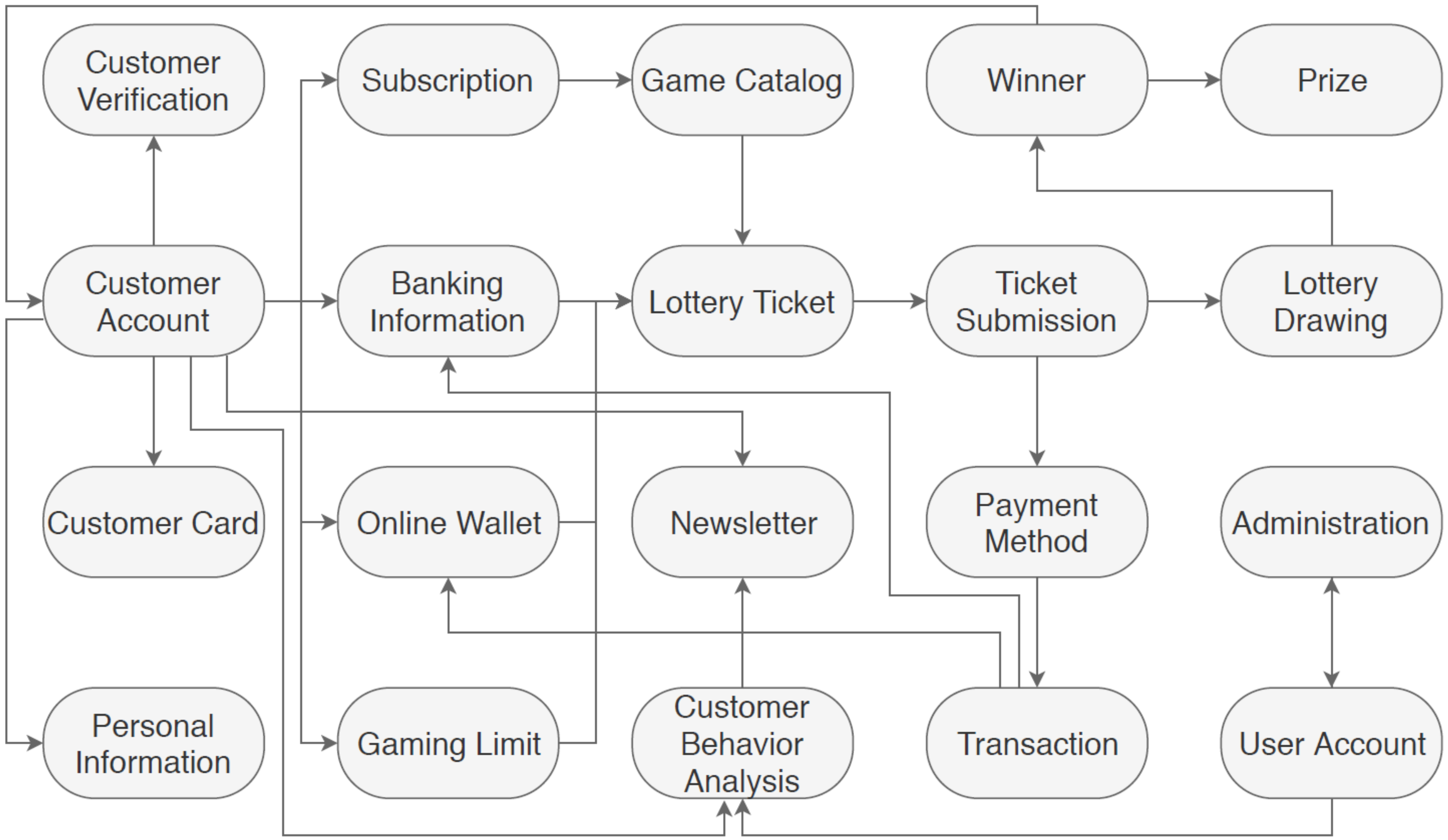
Dynamic Analysis

adesso

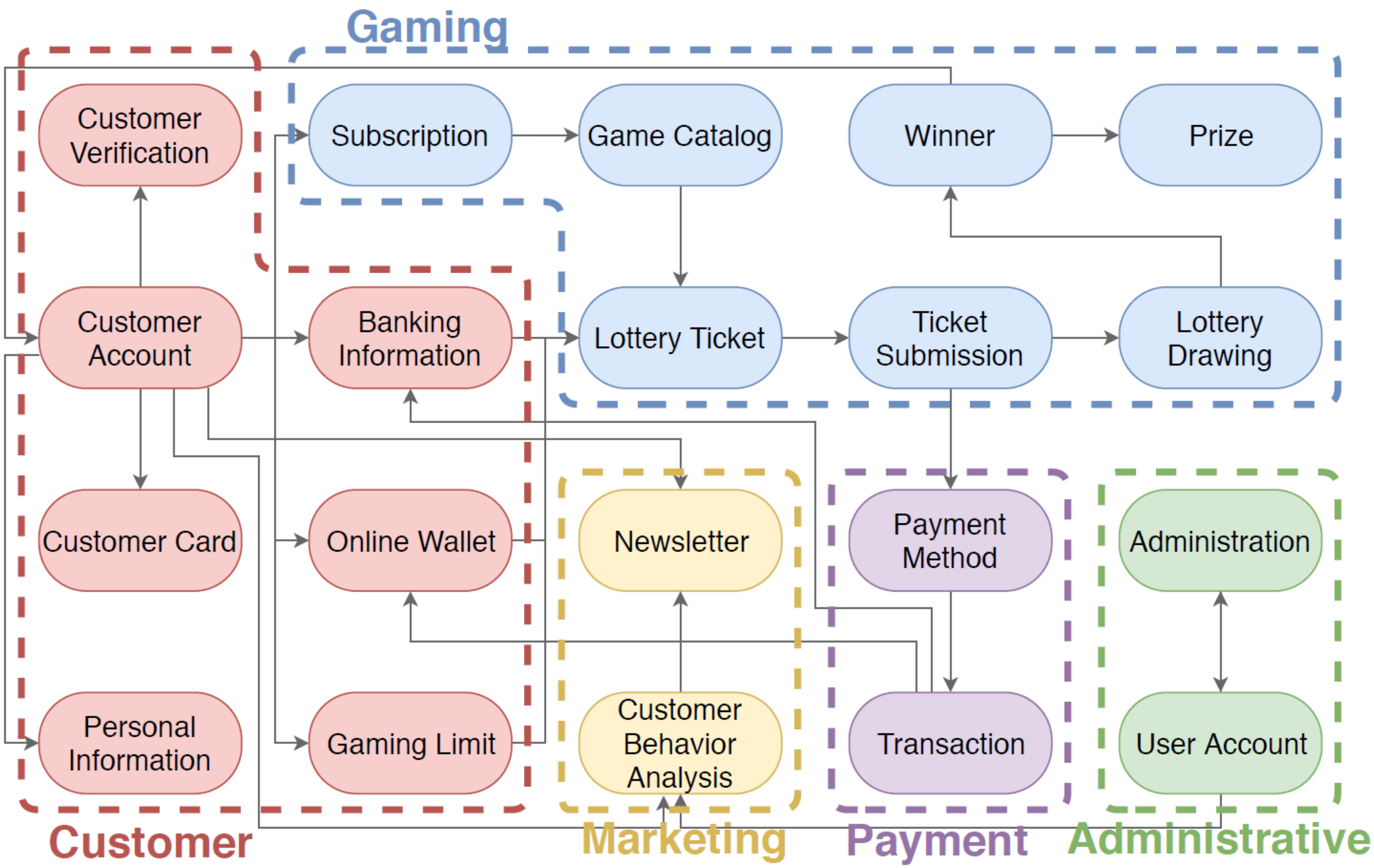


[Krause et al. 2020]

Identified Business Objects (Selection)

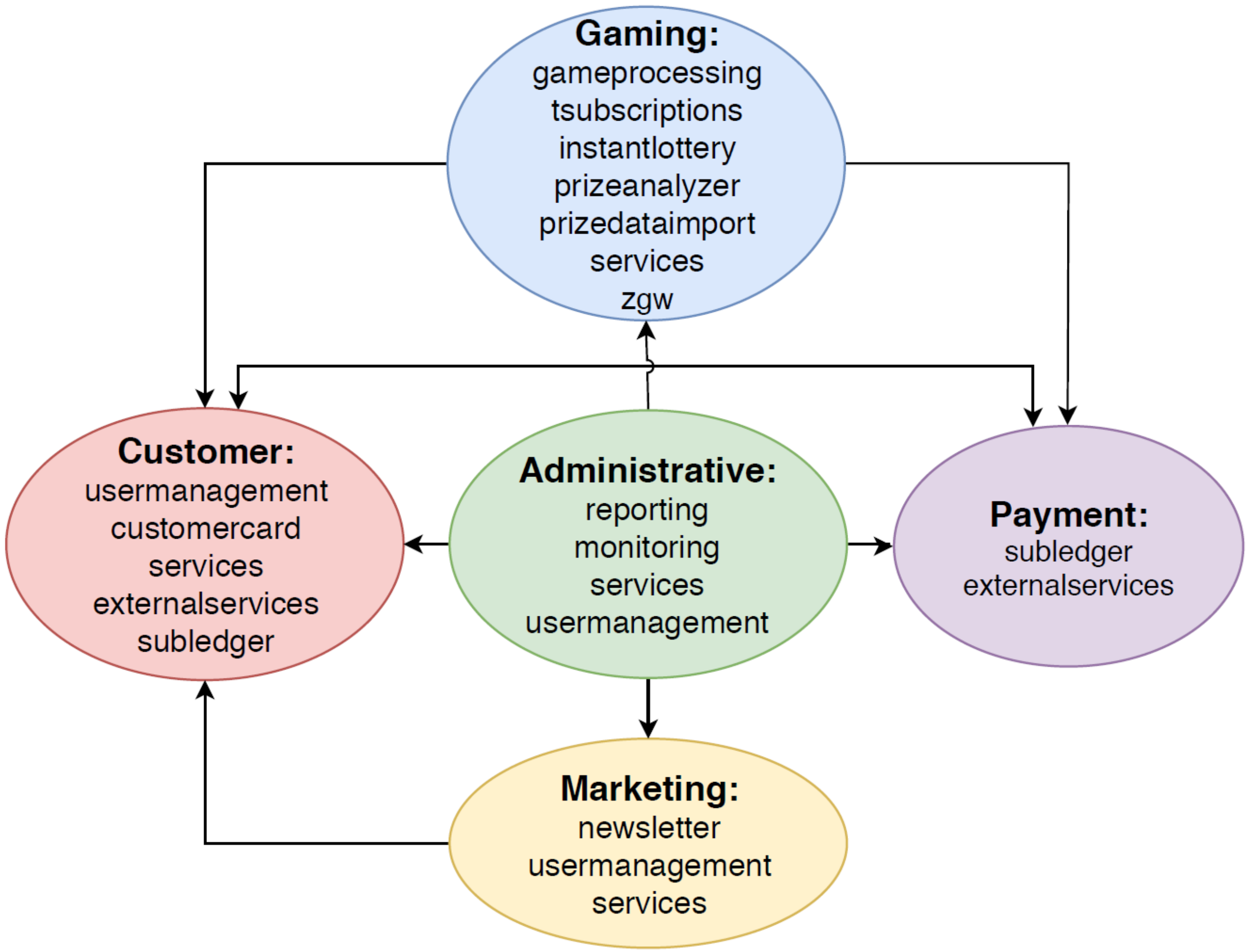


Partitioning into bounded contexts

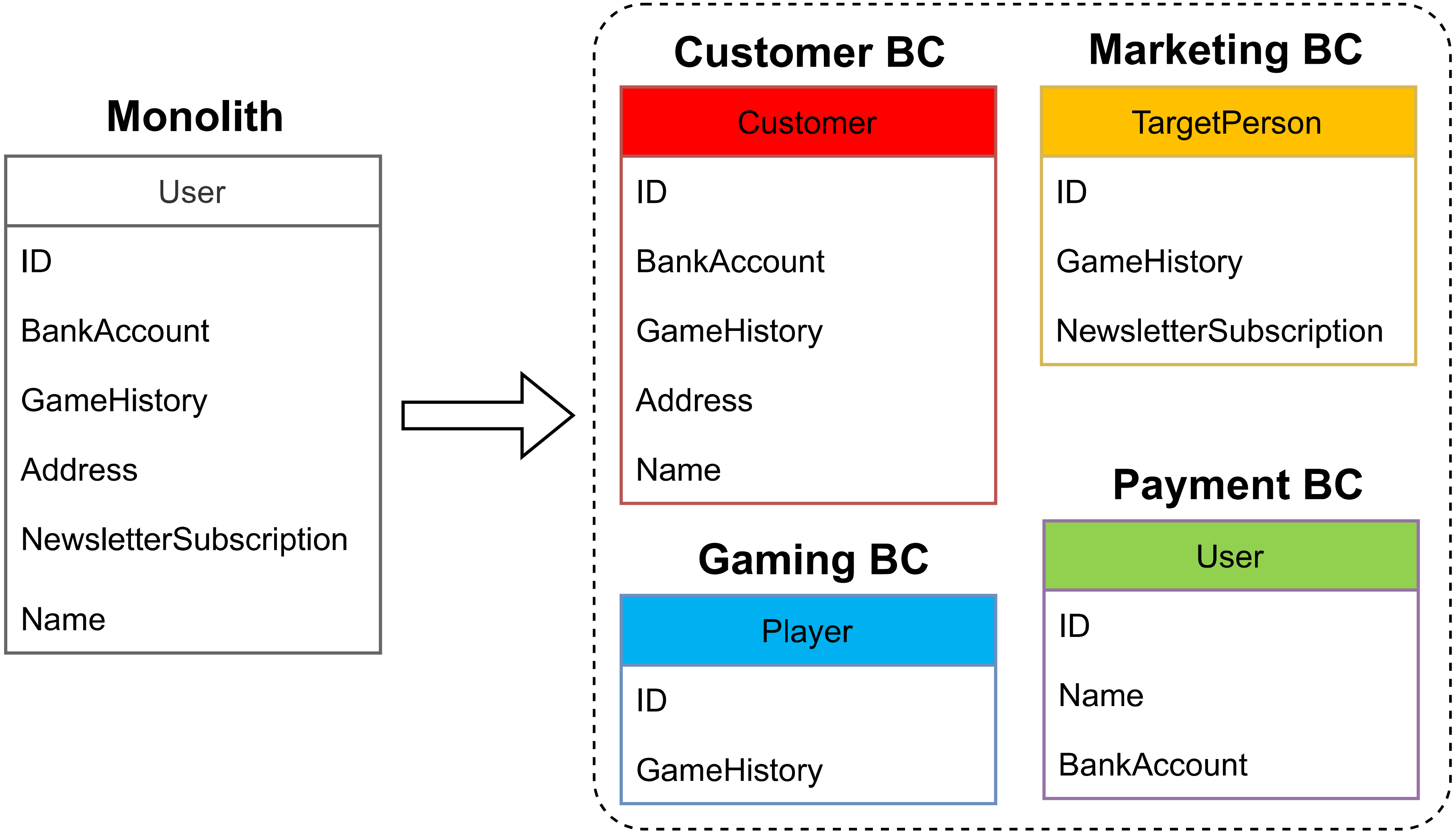


in|FOCUS

Resulting Bounded Contexts



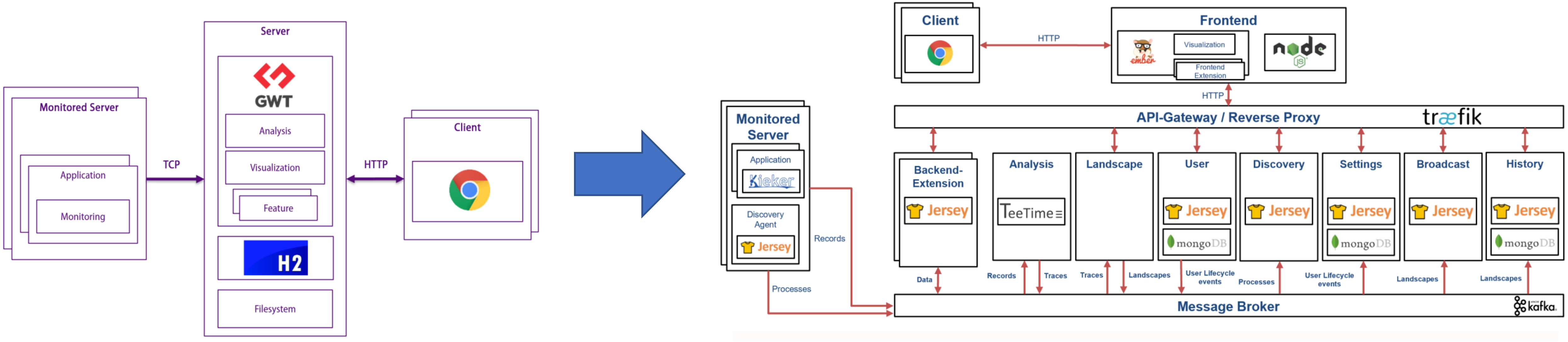
Domain-Driven Database Design



[Krause et al. 2020]

Re-Engineering ExplorViz toward a Microservice Architecture

“eat your own dog food”



[Fittkau et al. 2013b, Fittkau and Hasselbring 2015, Zirkelbach et al. 2018, 2019, 2020, Krause et al. 2018, Krause-Glau and Hasselbring 2022]

Research Software Engineering Research

Research Software Engineering | Software Engineering Research

Research Software Engineering Research aims at understanding and improving how software is developed for research.

RSE Research, in short [Felderer et al. 2023, 2025].

Sample RSE Research Question:
“**Which categories of research software require which software architecture structures?** ”

Slides

https://oceanrep.geomar.de/id/eprint/60906/

References

[Calzavarini 2019] E. Calzavarini: “Eulerian–Lagrangian fluid dynamics platform: The ch4-project”. In: Software Impacts 1, 2019. DOI https://doi.org/10.1016/j.simpa.2019.100002

[Chue Hong et al. 2022] N. P., Chue Hong, et al. (2022). FAIR Principles for Research Software version 1.0. (FAIR4RS Principles v1.0). Research Data Alliance. DOI https://doi.org/10.15497/RDA00068

[Claus et al. 2022] Claus, M., Gundlach, S., Hasselbring, W., Jung, R., Rath, W. und Schnoor, H.: “Modularizing Earth system models for interactive simulation.” Informatik Spektrum, 45. pp. 300-303. 2022, DOI https://doi.org/10.1007/s00287-022-01490-z

[Felderer et al. 2023] Felderer, M., Goedicke, M., Grunske, L., Hasselbring, W., Lamprecht, A. L. und Rumpe, B.: “Toward Research Software Engineering Research”. 2023. DOI https://doi.org/10.5281/ZENODO.8020525.

[Felderer et al. 2025] Felderer, M., Goedicke, M., Grunske, L., Hasselbring, W., Lamprecht, A. L. und Rumpe, B.: “Investigating research software engineering: Toward RSE Research”. Communications of the ACM, 2025. DOI https://doi.org/10.1145/3685265

[Fittkau et al. 2013a] F. Fittkau, J. Waller, C. Wulf, W. Hasselbring: “Live Trace Visualization for Comprehending Large Software Landscapes: The ExplorViz Approach“, In: 1st IEEE International Working Conference on Software Visualization (VISSOFT 2013). DOI https://doi.org/10.1109/VISSOFT.2013.6650536

[Fittkau et al. 2013b] F. Fittkau, J. Waller, P. Brauer, W. Hasselbring: “Scalable and Live Trace Processing with Kieker Utilizing Cloud Computing”. In: Symposium on Software Performance. 2013. URL https://ceur-ws.org/Vol-1083/paper10.pdf

[Fittkau et al. 2014] F. Fittkau, P. Stelzer, W. Hasselbring: “Live Visualization of Large Software Landscapes for Ensuring Architecture Conformance”. In: European Conference on Software Architecture Workshops. 2014. DOI https://doi.org/10.1145/2642803.2642831

[Fittkau et al. 2015a] F. Fittkau, S. Roth, W. Hasselbring: “ExplorViz: Visual Runtime Behavior Analysis of Enterprise Application Landscapes“, In: 23rd European Conference on Information Systems (ECIS 2015). DOI https://doi.org/10.18151/7217313

[Fittkau et al. 2015b] F. Fittkau, A. Krause, W. Hasselbring: “Hierarchical Software Landscape Visualization for System Comprehension: A Controlled Experiment”. In: 3rd IEEE Working Conference on Software Visualization, 2015. DOI https://doi.org/10.1109/VISSOFT.2015.7332413

[Fittkau et al. 2015c] F. Fittkau, A. Krause, W. Hasselbring: “Exploring Software Cities in Virtual Reality”, In: 3rd IEEE Working Conference on Software Visualization, 2015. DOI https://doi.org/10.1109/VISSOFT.2015.7332423

[Fittkau et al. 2015d] F. Fittkau, E. Koppenhagen, W. Hasselbring: “Research Perspective on Supporting Software Engineering via Physical 3D Models”. In: 3rd IEEE Working Conference on Software Visualization. 2015. DOI https://doi.org/10.1109/VISSOFT.2015.7332422

References

[Fittkau et al. 2015e] F. Fittkau, S. Finke, W. Hasselbring, J. Waller: “Comparing Trace Visualizations for Program Comprehension through Controlled Experiments”, In: 23rd IEEE International Conference on Program Comprehension (ICPC 2015), May 2015, Florence. DOI https://doi.org/10.1109/ICPC.2015.37

[Fittkau and Hasselbring 2015] F. Fittkau, W. Hasselbring: “Elastic Application-Level Monitoring for Large Software Landscapes in the Cloud”. In: European Conference on Service-Oriented and Cloud Computing. 2015. DOI https://doi.org/10.1007/978-3-319-24072-5_6

[Fittkau et al. 2017] F. Fittkau, A. Krause, W. Hasselbring: “Software landscape and application visualization for system comprehension with ExplorViz”, In: Information and Software Technology. DOI https://doi.org/10.1016/j.infsof.2016.07.004

[Gryczan and Züllighoven 1992] G. Gryczan, H. Züllighoven: “Objektorientierte Systementwicklung, Leitbild und Entwicklungssystems”. Informatik Spektrum, 15(5): 264-272. Springer, 1992.

[Hansen et al. 2024] M. Hansen, H. Bielfeldt, A. Bernstetter, T. Kwasnitschka, W. Hasselbring: “A Software Visualization Approach for Multiple Visual Output Devices”. In: VISSOFT 2024. DOI https://doi.org/10.48550/arXiv.2409.02620

[Hasselbring 2002] W. Hasselbring: “Component-Based Software Engineering”. In: Handbook of Software Engineering and Knowledge Engineering, World Scientific, pp. 289-305, 2002. DOI https://doi.org/10.1142/9789812389701_0013

[Hasselbring et al. 2020a] W. Hasselbring, L. Carr, S. Hettrick, H. Packer, T. Tiropanis: “Open Source Research Software”. In: Computer, 53 (8), pp. 84-88. 2020. DOI https://doi.org/10.1109/MC.2020.2998235

[Hasselbring et al. 2020b] W. Hasselbring, L. Carr, S. Hettrick, H. Packer, T. Tiropanis: “From FAIR Research Data toward FAIR and Open Research Software”, it - Information Technology, 2020. DOI https://doi.org/10.1515/itit-2019-0040

[Hasselbring et al. 2020c] Hasselbring, W., Krause, A., Zirkelbach, C.: “ExplorViz: Research on software visualization, comprehension and collaboration.” In: Software Impacts, 6, 2020. DOI https://doi.org/10.1016/j.simpa.2020.100034.

[Hasselbring et al. 2024] Hasselbring, W., Druskat, S., Bernoth, J., Betker, P., Felderer, M., Ferenz, S., Hermann, B., Lamprecht, A. L., Linxweiler, J., Prat, A., Rumpe, B., Schoening-Stierand, K., Yang, S.: “Multi-dimensional categorization of research software with examples,” Zenodo, 2024. DOI: https://doi.org/10.5281/zenodo.14082554

[Henning and Hasselbring 2021] Henning, S., Hasselbring, W.: “The Titan Control Center for Industrial DevOps analytics research,” In: Software Impacts, 7, 2021. DOI https://doi.org/10.1016/j.simpa.2020.100050

References

[Henning and Hasselbring 2021] Henning, S., Hasselbring, W.: "Theodolite: Scalability Benchmarking of Distributed Stream Processing Engines in Microservice Architectures." In: Big Data Research, 25 (100209), 2021. pp. 1-17. DOI https://doi.org/10.1016/j.bdr.2021.100209

[Henning and Hasselbring 2022] Henning, S. und Hasselbring, W.: "A configurable method for benchmarking scalability of cloud-native applications." In: Empirical Software Engineering, 27 (6), 2022. p. 143. DOI https://doi.org/10.1007/s10664-022-10162-1

[Johanson et al. 2016] A. Johanson, S. Flögel, C. Dullo, W. Hasselbring: "OceanTEA: Exploring Ocean-Derived Climate Data Using Microservices". In: Sixth International Workshop on Climate Informatics (CI 2016), 2016, DOI http://dx.doi.org/10.5065/D6K072N6

[Jung et al. 2021] R. Jung, S. Gundlach, S. Simonov, W. Hasselbring: "Developing Domain-Specific Languages for Ocean Modeling". In: Software Engineering 2021 Satellite Events, http://ceur-ws.org/Vol-2814/

[Knoche and Hasselbring 2018] H. Knoche and W. Hasselbring: "Using Microservices for Legacy Software Modernization". IEEE Software, 35(3). pp. 44-49, 2018. DOI https://doi.org/10.1109/MS.2018.2141035.

[Knoche and Hasselbring 2019] H. Knoche and W. Hasselbring: "Drivers and Barriers for Microservice Adoption - A Survey among Professionals in Germany". Enterprise Modelling and Information Systems Architectures (EMISAJ) - International Journal of Conceptual Modeling, 14(1). pp. 1-35, 2019. DOI https://doi.org/10.18417/emisa.14.1.

[Krause et al. 2018] A. Krause, C. Zirkelbach, W. Hasselbring: "Simplifying Software System Monitoring through Application Discovery with ExplorViz". In: Symposium on Software Performance. 2018. URL https://dl.gi.de/handle/20.500.12116/40464

[Krause et al. 2020] A. Krause, C. Zirkelbach, W. Hasselbring, S. Lenga, D. Kröger: "Microservice Decomposition via Static and Dynamic Analysis of the Monolith". In: IEEE International Conference on Software Architecture (ICSA 2020). pp. 9-16. DOI https://doi.org/10.1109/ICSA-C50368.2020.00011

[Krause et al. 2021] A. Krause, M. Hansen, W. Hasselbring: "Live Visualization of Dynamic Software Cities with Heat Map Overlays". In: 2021 Working Conference on Software Visualization (VISSOFT). September 27-28, 2021, pp. 125-129 . DOI https://doi.org/10.1109/VISSOFT52517.2021.00024

[Krause et al. 2022] A. Krause-Glau, M. Bader, W. Hasselbring: "Collaborative Software Visualization for Program Comprehension". In: 2022 Working Conference on Software Visualization (VISSOFT). 2022. DOI: https://doi.org/10.1109/VISSOFT55257.2022.00016

[Krause-Glau et al. 2022] A. Krause-Glau, M. Hansen, W. Hasselbring: "Collaborative program comprehension via software visualization in extended reality". In: Information and Software Technology. 2022. DOI: https://doi.org/10.1016/j.infsof.2022.107007

[Krause-Glau and Hasselbring 2022] A. Krause, W. Hasselbring: "Scalable Collaborative Software Visualization as a Service". In: IEEE International Conference on Cloud Engineering. 2022. DOI https://doi.org/10.1109/IC2E55432.2022.00026

References

[Krause-Glau et al. 2024a] A. Krause-Glau, M. Hansen, W. Hasselbring: “Collaborative Design and Planning of Software Architecture Changes via Software City Visualization”. In: VISSOFT 2024. DOI https://doi.org/10.48550/arXiv.2408.16777

[Krause-Glau et al. 2024b] A. Krause-Glau, L. Damerau, M. Hansen, W. Hasselbring: “Visual Integration of Static and Dynamic Software Analysis in Code Reviews via Software City Visualization”. In: VISSOFT 2024. DOI https://doi.org/10.48550/arXiv.2408.08141

[Krause-Glau and Hasselbring 2023] A. Krause, W. Hasselbring: “Collaborative, Code-Proximal Dynamic Software Visualization within Code Editors”. In: 2023 Working Conference on Software Visualization (VISSOFT). October 2023. DOI https://doi.org/10.48550/arXiv.2308.15785

[Shaw and Garlan 1996] M. Shaw, and D. Garlan: “Software Architecture: Perspectives on an Emerging Discipline”. Prentice Hall, 1996

[Tavares de Sousa et al. 2018] N. Tavares de Sousa, W. Hasselbring, T. Weber, D. Kranzlmüller: “Designing a Generic Research Data Infrastructure Architecture with Continuous Software Engineering”, In: 3rd Workshop on Continuous Software Engineering (CSE 2018), March 2018, Ulm, Germany.

[Zirkelbach et al. 2018] C. Zirkelbach, A. Krause, W. Hasselbring: “On the Modernization of ExplorViz towards a Microservice Architecture”. In: Collaborative Workshop on Evolution and Maintenance of Long-Living Software Systems. 2018. URL https://ceur-ws.org/Vol-2066/emls2018paper01.pdf

[Zirkelbach et al. 2019] C. Zirkelbach, A. Krause, W. Hasselbring: “Modularization of Research Software for Collaborative Open Source Development”, In: The Ninth International Conference on Advanced Collaborative Networks, Systems and Applications (COLLA 2019), June 30 - July 04, 2019, Rome, Italy.

[Zirkelbach et al. 2020] C. Zirkelbach, A. Krause, W. Hasselbring: “The Collaborative Modularization and Reengineering Approach CORAL for Open Source Research Software”. In: International Journal On Advances in Software. 2020.

[Züllighoven 2004] H. Züllighoven: “Object-Oriented Construction Handbook: Developing Application-Oriented Software with the Tools & Materials Approach”, Morgan Kaufmann, 2004