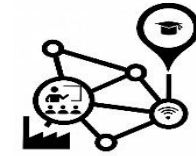




Co-funded by the
Erasmus+ Programme
of the European Union

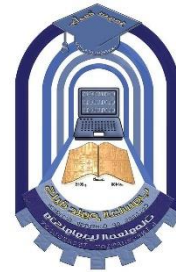


IoTrain
Master of Engineering in Internet of Things

Course Seminar

University of Sumer

College of Computer Science and Information Technology





Co-funded by the
Erasmus+ Programme
of the European Union



Course Seminar

Course description and goals

Seminars encourage students to practice a scientific presentation based on the peer review of scientific papers on a given topic to an expert team, e.g., professors, graduate students, to receive their feedback and finalize their report which is a state-of-the-art of a technical topic. Technical topics are based on the subjects taught in the previous semesters. The topics can deepen already existing professional interests and emphases.

A few lectures will be provided to students to gain some knowledge on the following topics:

- 1) literature search; 2) literature analysis; 3) Literature survey/review; 4) write the literature review report; 6) report presentation; 7) Seminar presentation (i.e., presentation, Questions/answers)

At the end, the students will be able to provide the followings:

- Introduce their advisor and committee members.
- Give an introduction and background information on their topic. What relevant research has been performed previously?
- State the problem(s) that remain unanswered.
- Clearly state their objectives and give the specific hypotheses they wish to test.
- Describe the methodology they will use to test their hypotheses. Be sure they fully understand your chosen methods. Give reasons why they chose these methods over other approaches.
- Present any data they have collected thus far.
- Describe what remains to be done, and what they expect to find.
- Explain the significance of their findings (or potential future findings).



Co-funded by the
Erasmus+ Programme
of the European Union



Course Seminar

Learning outcomes and competencies that the course develops

Factual knowledge

- The technical contents are secondary to the desired methodological competencies and key qualifications and may supplement a focus chosen in the elective area.

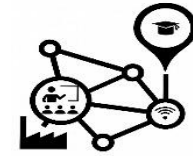
After passing this course, the students acquire the key qualifications below:

- To access material on a given topic from literature databases and other open science sources
- To read, understand and prepare original literature
- To design a lecture on a more complex scientific topic in front of a specialist audience (i.e. to design it didactically correctly) and to give it using standard media
- To contribute to discussions in a scientific lecture
- To write texts of approx. 10 - 20 pages, usually to explain technical / scientific matters
- To conform research ethics

To understand and implement research verification methods



Co-funded by the
Erasmus+ Programme
of the European Union



IoTrain
Master of Engineering in Internet of Things

Course Seminar

Topics to be covered

Scientific topic can depend on the master thesis topic.

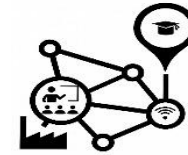
Technical topics are as mentioned below:

- literature search;
- literature analysis;
- Literature survey/review;
- write the literature review report;
- report presentation;

Seminar presentation (i.e., presentation, Questions/answers)



Co-funded by the
Erasmus+ Programme
of the European Union



IoTrain
Master of Engineering in Internet of Things

Course Seminar

Text books & Content resources

- Michael Alley, “The Craft of Scientific Presentations Critical Steps to Succeed and Critical Errors to Avoid,” <https://nerds.itu.dk/wp-content/uploads/sites/74/2022/01/Scientific-Presentation.pdf>
 - 2nd edition: <https://link.springer.com/content/pdf/10.1007/978-1-4419-8279-7.pdf>
- Challenging the status quo of scientific presentations, Debnath Chatterjee¹ | Myron Yaster¹ | Justin L. Lockman² | Nancy L. Glass³ | Mark S. Schreiner² | Jina L. Sinskey⁴ | John E. Fiadjoe² <https://onlinelibrary.wiley.com/doi/pdf/10.1111/pan.14064>

Section 1. Writing Scientific Texts

Outline

- Introduction
- Generic Structure of Scientific Texts
- Academic Writing Guidelines
- The Small Things

Outline

- **Introduction**
- Generic Structure of Scientific Texts
- Academic Writing Guidelines
- The Small Things

What is a scientific text?

- A scientific text is a report about new research results.
- A scientific text describes clearly identifiable, distinguished contributions which increase the knowledge in some field of research.
- A scientific text uses textual descriptions combined with different other elements like figures, tables, diagrams etc. to present the results in a graspable way.

What makes Scientific Writing challenging?

- Scientific texts should describe non-trivial content in a clear and comprehensible language.
- Scientific texts should neither over-complicate or obfuscate, nor over-simplify research results.
- Scientific texts should have a clear focus, but should also put the results into a larger context.
- Scientific texts are written for a particular research community, yet should be as self-explanatory as possible.
- Scientific texts describe novel aspects or phenomena which often require newly introduced terminology.
- Scientific texts usually have a strict page limit which is not sufficient to describe everything in full depth.

Outline

- Introduction
- **Generic Structure of Scientific Texts**
- Academic Writing Guidelines
- The Small Things

Example: Structure of a Research Paper



Front Matter
Prologue
Main Part
App.

Title
List of Authors

Abstract, Keywords etc.

Introduction Section

Related Work Section

Background Section

Methodology Section

Evaluation Section

Conclusion Section

Acknowledgements

References, Appendices

ArchJava: Connection Software Architecture to Implementation

Jonathan Aronoff Craig Chambers David Notkin
Department of Computer Science and Engineering
University of Washington
Box 352020
Seattle, WA 98195-2020 USA
+1 206 543-5460
jonathan.aronoff@cs.washington.edu

Abstract
ArchJava describes the structure of a system, enabling software architects to describe the structure of a system, making design, development, and testing easier. ArchJava is a domain-specific language for describing software architecture. It is designed to be used by software architects to describe the structure of a system, making design, development, and testing easier. ArchJava is a domain-specific language for describing software architecture. It is designed to be used by software architects to describe the structure of a system, making design, development, and testing easier.

Introduction
ArchJava is the architecture of a system as a collection of components, connections, and constraints. ArchJava is a domain-specific language for describing software architecture. It is designed to be used by software architects to describe the structure of a system, making design, development, and testing easier.

2. Previous Work
ArchJava is a domain-specific language for describing software architecture. It is designed to be used by software architects to describe the structure of a system, making design, development, and testing easier.

Many of these languages support automated analysis and reasoning. For example, Wright [18] allows architects to specify temporal constraints, and check against such constraints via model checking. SAE [19] allows architects to specify constraints in a declarative language, and check against such constraints via model checking. Wright [18] allows architects to specify temporal constraints, and check against such constraints via model checking. SAE [19] allows architects to specify constraints in a declarative language, and check against such constraints via model checking.

3.1. Components and Parts
A component is a special kind of object that communicates with other components. A component is a special kind of object that communicates with other components. A component is a special kind of object that communicates with other components.

3.2. Component Composition
A component is a special kind of object that communicates with other components. A component is a special kind of object that communicates with other components. A component is a special kind of object that communicates with other components.

Figure 1: A generic component in ArchJava. The generic component class has two parts to implement. The generic component class has two parts to implement. The generic component class has two parts to implement.

```
public interface Component {
    public void activate();
    public void deactivate();
    public void activate();
    public void deactivate();
    public void activate();
    public void deactivate();
    public void activate();
    public void deactivate();

```

Figure 2: A graphical compiler interface and its ArchJava representation. The graphical compiler interface and its ArchJava representation. The graphical compiler interface and its ArchJava representation.

Figure 3: Precision and recall for classifiers trained with differing amounts of data from the Eclipse project. Precision and recall for classifiers trained with differing amounts of data from the Eclipse project. Precision and recall for classifiers trained with differing amounts of data from the Eclipse project.

Classifier	Precision	Recall
Naive Bayes	0.95	0.95
Decision Tree	0.92	0.92
Support Vector	0.98	0.98
Neural Network	0.96	0.96
Random Forest	0.97	0.97

Figure 4: Results of using a component-based classifier. Results of using a component-based classifier. Results of using a component-based classifier.

Classifier	Precision	Recall
Naive Bayes	0.95	0.95
Decision Tree	0.92	0.92
Support Vector	0.98	0.98
Neural Network	0.96	0.96
Random Forest	0.97	0.97

3.6. Limitation of ArchJava
ArchJava is a domain-specific language for describing software architecture. It is designed to be used by software architects to describe the structure of a system, making design, development, and testing easier.

3.7. Evaluation
ArchJava is a domain-specific language for describing software architecture. It is designed to be used by software architects to describe the structure of a system, making design, development, and testing easier.

4.1. Methodology
ArchJava is a domain-specific language for describing software architecture. It is designed to be used by software architects to describe the structure of a system, making design, development, and testing easier.

Figure 4: The developer's drawing of ArchJava's architecture. The developer's drawing of ArchJava's architecture. The developer's drawing of ArchJava's architecture.

Figure 5: The developer's drawing of ArchJava's architecture. The developer's drawing of ArchJava's architecture. The developer's drawing of ArchJava's architecture.

Figure 4: Precision and recall for classifiers trained with differing amounts of data from the Eclipse project. Precision and recall for classifiers trained with differing amounts of data from the Eclipse project. Precision and recall for classifiers trained with differing amounts of data from the Eclipse project.

Classifier	Precision	Recall
Naive Bayes	0.95	0.95
Decision Tree	0.92	0.92
Support Vector	0.98	0.98
Neural Network	0.96	0.96
Random Forest	0.97	0.97

Figure 4: Results of using a component-based classifier. Results of using a component-based classifier. Results of using a component-based classifier.

Classifier	Precision	Recall
Naive Bayes	0.95	0.95
Decision Tree	0.92	0.92
Support Vector	0.98	0.98
Neural Network	0.96	0.96
Random Forest	0.97	0.97

4.4. Case Study Summary
ArchJava is a domain-specific language for describing software architecture. It is designed to be used by software architects to describe the structure of a system, making design, development, and testing easier.

5. Conclusion and Future Work
ArchJava is a domain-specific language for describing software architecture. It is designed to be used by software architects to describe the structure of a system, making design, development, and testing easier.

6. Acknowledgments
ArchJava is a domain-specific language for describing software architecture. It is designed to be used by software architects to describe the structure of a system, making design, development, and testing easier.

7. References
ArchJava is a domain-specific language for describing software architecture. It is designed to be used by software architects to describe the structure of a system, making design, development, and testing easier.

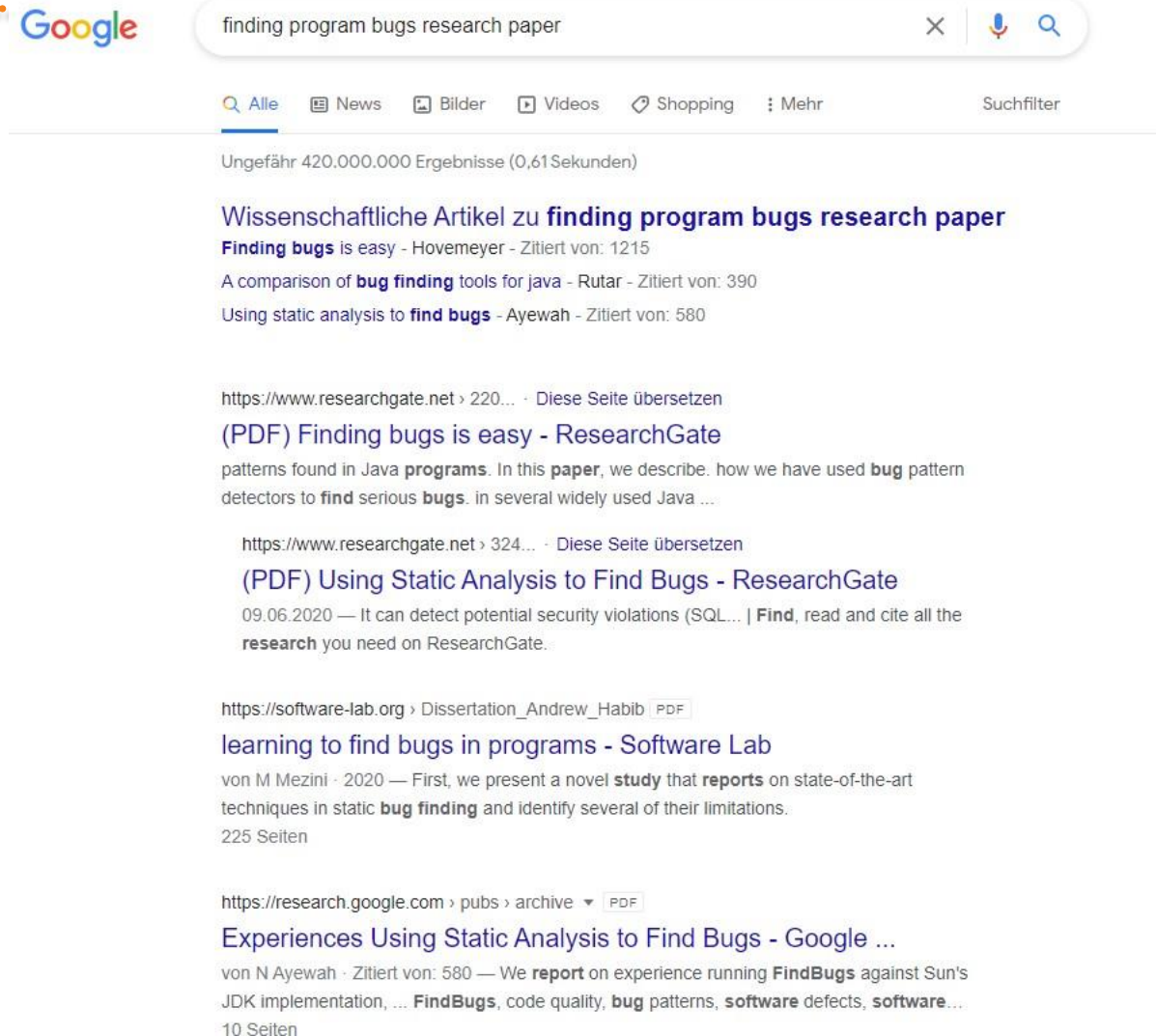
Title



- The title should summarize in a meaningful way the approach and key contributions described by the scientific text.
- The title should pique interest, yet being honest about the actual scientific achievements.

Title

- The title should cover relevant key words to have a good chance to be highly ranked by search engines for respective queries.



Google

finding program bugs research paper

Alle News Bilder Videos Shopping Mehr Suchfilter

Ungefähr 420.000.000 Ergebnisse (0,61 Sekunden)

Wissenschaftliche Artikel zu **finding program bugs research paper**

Finding bugs is easy - Hovemeyer - Zitiert von: 1215
A comparison of **bug finding** tools for java - Rutar - Zitiert von: 390
Using static analysis to **find bugs** - Ayewah - Zitiert von: 580

<https://www.researchgate.net> › 220... · Diese Seite übersetzen
(PDF) Finding bugs is easy - ResearchGate
patterns found in Java **programs**. In this **paper**, we describe. how we have used **bug** pattern detectors to **find** serious **bugs**. in several widely used Java ...

<https://www.researchgate.net> › 324... · Diese Seite übersetzen
(PDF) Using Static Analysis to Find Bugs - ResearchGate
09.06.2020 — It can detect potential security violations (SQL... | **Find**, read and cite all the **research** you need on ResearchGate.

<https://software-lab.org> › Dissertation_Andrew_Habib PDF
learning to find bugs in programs - Software Lab
von M Mezini · 2020 — First, we present a novel **study** that **reports** on state-of-the-art techniques in static **bug finding** and identify several of their limitations.
225 Seiten

<https://research.google.com> › pubs › archive PDF
Experiences Using Static Analysis to Find Bugs - Google ...
von N Ayewah · Zitiert von: 580 — We **report** on experience running **FindBugs** against Sun's JDK implementation, ... **FindBugs**, code quality, **bug** patterns, **software** defects, **software**...
10 Seiten

Title

- Participants of scientific conferences, first of all, skim through the list of paper titles to decide which talks they are going to attend.

08:00	15m	✎	Developing a Multi-Cultural Conversational Agent With a Globally Distributed Team ICGSE: Experience Reports Clayton Rouse University College Dublin, Anthony Wedderburn Lee and University College Dublin, Yusef Smith Microsoft	
08:15	15m	✎	Occurrence Frequency and All Historical Failure Information Based Method for TQP in CI ICSSP Ying Sheng Beijing University of Chemical Technology, Qianguo Li Beijing University of Chemical Technology, Teng Yang Beijing University of Chemical Technology, Zhenqiang Li University of Constance	
08:30	15m	✎	How to Treat the Use of Gray Literature in Software Engineering ICSSP Xin Zhou State Key Laboratory of Novel Software Technology, Software Institute, Nanjing University	
09:00 - 18:00			Session 3 at ICSE/ICSE ICGSE: Pablo Carroño University of Seville, Paul Clarke, Marco Canosa Northern Arizona University ICGSE Theme: Onboarding and community evolution ICSSP Theme: Machine Learning, AI and Microservices Architectures	ICGSE Research Papers / ICGSE Experience Reports / ICSE/ICSE [Joint Event] / ICSE/ICSE / ICGSE Journal First
09:00	5m	✎	Opening ICGSE: Leif Eriksson - ICSSP/ICGSE Pablo Iñigo IT University of Copenhagen, David Haffa Portland State University	Reviews
10:00	15m	✎	TacRes: A Framework for Task Recommendation in Crowdsourcing ICGSE: Research Papers Korner Adhikari Amrita Vishwa Vastu Sangrahalaya, Gurpreet Kaur Ghoshal Indian Institute of Information Technology Delhi, Algeria Oudry Amara University, Sakshi Jain Indian Institute of Information Technology Lucknow	
10:30	15m	✎	From Art to Science: Evolution of Community Development Talk Shane Mueller Cornell, Daniel Sappido Cornell	
10:35	15m	✎	Process Implications of Executable Domain Models for Microservices Development ICSSP Bai Wang University of Southern California, Sherry Swaters University of Southern California	
10:50	15m	✎	Do Instance-Level Review Diagrams Support Validation Processes of Cyber-Physical System Specifications? ICSSP Marten Damm University of Duisburg-Essen, Jennifer Brings University of Duisburg-Essen, Hans-Joerg Pohlmann University of London	
10:55	15m	✎	Onboarding Bot for Newcomers to Software Engineering ICSSP James Damento Carleton University, Charles Hilder Carleton University, Fuguo Rodriguez Carleton University	
10:55	15m	✎	How do newcomers learn work process in Global Software Development (GSD)? A survey study from the perspective of newly project leaders ICGSE: Experience Reports Rajeev Chandra SDIA Institute of Science and Technology, Fernando Souza SDIA Institute of Science and Technology, Francisco Lima SDIA Institute of Science and Technology, Bruno Damasceno Universidade Federal do Rio de Janeiro - UFRJ	
10:55	15m	✎	Designing Engineering Onboarding for 80+ Nationalities ICGSE: Experience Reports Julien Perly Connected Cloud	
10:55	15m	✎	From Ad-Hoc Data Analytics to DataOps ICSSP Aparajita Managayya Carleton University of Technology, David Isaac Malina Carleton University of Technology, Jan Seifert, Helena Holmérholm Chalmers Applied University, Anas Dabbak Chalmers	
17:00	15m	✎	Emerging and Changing Tasks in the Development Process for Machine Learning Systems ICSSP Mehmet Ali Chalkas University of Göttingen, Samuel Okoro Chalmers University of Göttingen, Johan Falkberg Chalmers University of Göttingen, Hagen Puhj University of Göttingen	
17:30	15m	✎	Developing ML/DL Models: A Design Framework ICSSP Mauro Mary John North University, Helena Holmérholm Chalmers Applied University, Jan Seifert	
Sun 28 Jun				
Displayed time zone: (UTC) Coordinated Universal Time change				
08:00 - 18:00			Session 4 at ICSE/ICSE ICGSE: Ricardo Brito Chalmers University of Technology, Klaus-Jens Sidel University College Cork and Limerick, Israel Yaron Shalom University ICGSE Theme: Process and requirements ICSSP Theme: Empirical studies and experience reports on agile and hybrid processes	ICGSE Experience Reports / ICSE/ICSE Industry Talks / ICGSE [Joint Event] / ICSE/ICSE
08:00	5m	✎	Opening ICGSE: Leif Eriksson - ICSSP/ICGSE Pablo Iñigo IT University of Copenhagen, David Haffa Portland State University	Reviews
08:00	15m	✎	More-STP: A Novel Approach for Requirement Definition for GSD Projects in a Mobile Ecosystem ICGSE: Experience Reports Rikman M. Ghorasdes SDIA R&D Institute, Yasmin G. Var SDIA R&D Institute, Eberth F. Chor SDIA R&D Institute, Rafael E. Silva SDIA R&D Institute, Linaker Souza SDIA R&D Institute, Fábio M. Azevedo SDIA R&D Institute, Eduardo U. Sardinha SDIA R&D Institute, Paulo F. Ferreira SDIA R&D Institute, Cleonir Azevedo De Lara P&R SDIA Research Institute	
08:20	15m	✎	Using a Tool-based Approach to Comply with Smartphone User Manual Regulations in Latin America Countries ICGSE: Experience Reports Rikman M. Ghorasdes SDIA R&D Institute, Yasmin G. Var SDIA R&D Institute, Eberth F. Chor SDIA R&D Institute, Rafael E. Silva SDIA R&D Institute, Linaker Souza SDIA R&D Institute, Fábio M. Azevedo SDIA R&D Institute, Eduardo U. Sardinha SDIA R&D Institute, Paulo F. Ferreira SDIA R&D Institute, Cleonir Azevedo De Lara P&R SDIA Research Institute	

2014 23rd Australian Software Engineering Conference

Correctness by Construction with Logic-Labeled Finite-State Machines – Comparison with *Event-B*

Vladimir Estivill-Castro
School of Information and
Communication Technology

René Hexel
School of Information and
Communication Technology

- Avoid excessively long and over-specific titles with unclear focus.
- Avoid cascades of "by", "with", "using", "based on" etc. phrases.

Quantum measurement occurrence is undecidable

J. Eisert,¹ M. P. Müller,² and C. Gogolin¹

¹*Qmio Group, Dahlem Center for Complex Quantum Systems, Freie Universität Berlin, 14195 Berlin, Germany*

²*Perimeter Institute for Theoretical Physics, 31 Caroline Street North, Waterloo, Ontario N2L 2Y5, Canada*

In this work, we show that very natural, apparently simple problems in quantum measurement theory can be undecidable even if their classical analogues are decidable. Undecidability hence appears as a genuine quantum property here. Formally, an undecidable problem is a decision problem for which one cannot construct a single algorithm that will always provide a correct answer in finite time. The problem we consider is to determine whether sequentially used identical Stern-Gerlach-type measurement devices, giving rise to a tree of possible

- Very clear and groundbreaking research results may be directly announced in the title in a straight-forward manner.

PROGRAM SLICING*

Mark Weiser

Computer Science Department
University of Maryland
College Park, MD 20742

Abstract

Program slicing is a method used by experienced computer programmers for abstracting from programs. Starting from a subset of a program's behavior, slicing reduces that program to a minimal form which still produces that behavior. The

behavior is of interest. For instance, during debugging a subset of behavior is being corrected, and in program modification or maintenance a subset of behavior is being improved or replaced. In these cases, a programmer starts from the program behavior and proceeds to find and modify the corresponding portions of program code. (code not

- Entirely novel concepts can be established by a preferably short and catchy phrase.

Title

A practical guide for using statistical tests to assess randomized algorithms in software engineering

Conference Paper in Proceedings - International Conference on Software Engineering - June 2011

DOI: 10.1145/1985793.1985795 - Source: IEEE Xplore

CITATIONS

651

READS

2,023

2 authors:



Andrea Arcuri
Westerdals Oslo School of Arts, Communication and Technology



L. Briand
University of Luxembourg

A Novel Algorithm for Optimised Real Time Anomaly Detection in Timeseries

Krishnam Kapoor¹

¹ Indian Institute of Technology, Kharagpur, India
krishnamkapoor@iitkgp.ac.in

Quantum measurement occurrence is undecidable

J. Eisert,¹ M. P. Müller,² and C. Gogolin¹

¹Qnio Group, Dahlem Center for Complex Quantum Systems, Freie Universität Berlin, 14195 Berlin, Germany
²Perimeter Institute for Theoretical Physics, 31 Caroline Street North, Waterloo, Ontario N2L 2Y5, Canada

In this work, we show that very natural, apparently simple problems in quantum measurement theory can be undecidable even if their classical analogues are decidable. Undecidability hence appears as a genuine quantum property here. Formally, an undecidable problem is a decision problem for which one cannot construct a single algorithm that will always provide a correct answer in finite time. The problem we consider is to determine whether sequentially used identical Stern-Gerlach-type measurement devices, giving rise to a tree of possible

User Study: Programming Understanding from Similar Code

By

Anush Ramsurat Ganapathi Agraharam Sivasankar

ag3630@columbia.edu

Abstract

The aim of the user study conducted is primarily threefold:

- The title may reflect the type of research methodology or contribution: guidelines, formal proofs, new algorithms, literature surveys, user studies...

Towards a Method of Programming with Assertions

David S. Rosenblum

AT&T Bell Laboratories
600 Mountain Avenue
Murray Hill, NJ 07974

Abstract

Embedded assertions have long been recognized as a potentially powerful tool for automatic runtime detection of software faults during debugging, testing and mainte-

1 Introduction

Assertions are formal constraints on software system behavior that are commonly written as annotations of a source text. The primary goal in writing assertions is to specify *what* a system is supposed to do rather than *how*

- Preliminary results of ongoing research projects or early research ideas are often marked by corresponding title prefixes like "Towards..."

Title

#ifdef Considered Harmful, or Portability Experience With C News

Henry Spencer – Zoology Computer Systems, University of Toronto
Geoff Collyer – Software Tool & Die

ABSTRACT

We believe that a C programmer's impulse to use `#ifdef` in an attempt at portability is usually a mistake. Portability is generally the result of advance planning rather than trench warfare involving `#ifdef`. In the course of developing C News on different systems, we

- Attract readers by controversial statements which are clarified in the text.

Who Should Fix This Bug?

John Anvik, Lyndon Hiew and Gail C. Murphy
Department of Computer Science
University of British Columbia
(janvik, lyndonh, murphy)@cs.ubc.ca

ABSTRACT

Open source development projects typically support an open bug repository to which both developers and users can re-

However, this potential advantage also comes with a significant cost. Each bug that is reported must be *triaged* to determine if it describes a meaningful new problem or

- Attract readers by asking interesting questions which are answered in the text.

JDeodorant: Identification and Removal of Type-Checking Bad Smells

Nikolaos Tsantalis^{*}, Theodoros Chaikalis, Alexander Chatzigeorgiou
Department of Applied Informatics, University of Macedonia
54006 Thessaloniki, Greece
{nikos, chaikalis}@java.uom.gr , achat@uom.gr

Abstract

In this demonstration, we present an Eclipse plug-in that automatically identifies Type-Checking bad smells

2. Methodology

Our methodology consists of two parts. The first deals with the identification of type-checking bad

- Funny wordplays may sometimes also help to attract attention - as long as they do not offend readers...

Title



Finding Bugs is Easy *

David Hovemeyer and William Pugh
Dept. of Computer Science, University of Maryland
College Park, Maryland 20742 USA
{daveho,pugh}@cs.umd.edu

ABSTRACT

feature or library API. Automatic detectors for many bugs

Distinguished Paper

Debugging Reinvented: Asking and Answering Why and Why Not Questions about Program Behavior

Andrew J. Ko and Brad A. Myers
Human-Computer Interaction Institute

School of Computer Science, Carnegie Mellon University

- Do not frustrate readers by promising too much or being too general.

Assignment 1

Your task is to write a scientific text proposing "your" new sorting algorithm *QuickSort*.

1. Discuss good paper titles.
2. Discuss bad paper titles.

List of Authors

- The list of authors may contain every person
 - who has considerably contributed to the research results described in the scientific text and
 - who has agreed to be listed as an author.
- The ordering of authors often has some meaning.

Autorinnen und Autoren wissenschaftlicher Veröffentlichungen tragen die Verantwortung für deren Inhalt stets gemeinsam. Autorin oder Autor ist nur, wer einen wesentlichen Beitrag zu einer wissenschaftlichen Veröffentlichung geleistet hat. Eine sogenannte „Ehrenautorschaft“ ist ausgeschlossen.

From the Denkschrift "Sicherung guter wissenschaftlicher Praxis", German Research Foundation (DFG) - 2013.

Rules of Authorship

Anyone listed as Author on an ACM manuscript submission must meet all the following criteria:

- they have made substantial intellectual contributions to some components of the original work described in the manuscript; and*
- they have participated in drafting and/or revision of the manuscript and*
- they are aware the manuscript has been submitted for publication; and*
- they agree to be held accountable for any issues relating to correctness or integrity of the work.*

From the criteria for authorship of the Association for Computing Machinery (ACM) - last updated August, 2021.

Ordering of Authors

Automatically Finding Patches Using Genetic Programming *

First author

Westley Weimer
University of Virginia
weimer@virginia.edu

ThanhVu Nguyen
University of New Mexico
nguyen@cs.unm.edu

Claire Le Goues
University of Virginia
legoues@virginia.edu

Stephanie Forrest
University of New Mexico
forrest@cs.unm.edu

Supervisor

Abstract **Co-authors**

Automatic program repair has been a longstanding goal in software engineering, yet debugging remains a largely manual process. We introduce a fully automated method for locating and repairing bugs in software. The approach works on off-the-shelf legacy applications and does not re-

To alleviate this burden, we propose an automatic technique for repairing program defects. Our approach does not require difficult formal specifications, program annotations or special coding practices. Instead, it works on off-the-shelf legacy applications and readily-available test-cases. We use genetic programming to evolve program variants until one is found that both retains required function-

- Ordering reflects amount of contribution (e.g., first author did most of the work).
- Ordering reflects seniority (e.g., students first, then assistants, and lastly supervisors - or vice versa).
- Or: alphabetical order.

Handling Authorships

- Discuss the list and ordering of authors of a scientific text as early as possible
 - to clarify who is responsible for the content of the text and
 - to avoid conflicts during later phases of preparation (or even while finishing or submitting the final text).
- In some cases, co-authors from outside of academia like collaborators from industrial companies require a scientific text to be approved by some authority before submission (e.g., to avoid publishing confidential details, to protected intellectual properties etc.).

Abstract

- The abstract provides a brief summary of the scientific text.
- The abstract should be short (150 - 200 words) and reading it should require no more than 5 minutes.
- This sounds easy but writing good abstracts is very difficult and usually requires several years of experience.

What the abstract should do:

- Provide enough background such that any reader is able to at least understand the meaning of the title.
- Motivate the research problem and summarize the novelty of the key contributions.
- Give interested readers from the same field a rough idea if the research results are relevant for him/her or not.
- Make potential readers from outside the field curious about the research problem.

Abstract

What the abstract should NOT do:

- Explain concepts and theories in detail.
- Extensively discuss related work.
- Describe the outline of the paper.
- ...

Abstract

- Problem-oriented abstracts focus on the motivation of open problems:

"Problem XY is widely considered relevant..."

"Problem XY is important..."

- Solution-oriented abstracts focus on the descriptions of concepts and contributions:

"We present a novel approach..."

"In this paper, we propose a novel technique..."

Is Mutation an Appropriate Tool for Testing Experiments?

J.H. Andrews
Computer Science Department
University of Western Ontario
London, Canada
andrews@csd.uwo.ca

L.C. Briand Y. Labiche
Software Quality Engineering Laboratory
Systems and Computer Engineering Department
Carleton University
Ottawa, Canada
{briand, labiche}@sce.carleton.ca

ABSTRACT

The empirical assessment of test techniques plays an important role in software testing research. One common practice is to instrument faults, either manually or by using mutation operators. The latter allows the systematic, repeatable seeding of large numbers of faults; however, we do not know whether empirical results obtained this way lead to valid, representative conclusions. This paper investigates this important question based on a number of programs with comprehensive pools of test cases and known

One problem in the design of testing experiments is that real programs of appropriate size with real faults are hard to find, and hard to prepare appropriately (for instance, by preparing correct and faulty versions). Even when actual programs with actual faults are available, often these faults are not numerous enough to allow the experimental results to achieve statistical significance. Many researchers therefore have taken the approach of introducing faults into correct programs to produce faulty versions.

Feedback-directed Random Test Generation

Carlos Pacheco¹, Shuvendu K. Lahiri², Michael D. Ernst¹, and Thomas Ball²
¹MIT CSAIL, ²Microsoft Research
{cpacheco, mernst}@csail.mit.edu, {shuvendu, tball}@microsoft.com

Abstract

We present a technique that improves random test generation by incorporating feedback obtained from executing test inputs as they are created. Our technique builds inputs incrementally by randomly selecting a method call to apply and finding arguments from among previously-constructed inputs. As soon as an input is built, it is executed and checked against a set of contracts and filters. The result of the execution determines whether the input is redundant, illegal, contract-violating, or useful for generating more inputs. The technique outputs a test suite consisting of unit tests for the classes under test. Passing tests can be used

tion [18], model checking, and symbolic execution [28].

It is difficult to generalize the results of these studies with regard to the relative advantages of random and systematic testing. The evaluations were performed on very small programs. Because the small programs apparently contained no errors, the comparison was in terms of coverage or rate of mutant killing [21], not in terms of true error detection, which is the best measure to evaluate test input generation techniques. While the systematic techniques used sophisticated heuristics to make them more effective, the type of random testing used for comparison is unguided random testing, with no heuristics to guide its search.

Our work addresses random generation of unit tests for

Four-questions-template for writing abstracts:

Answer the following questions, each by 1-2 sentences:

- ① What is the open research problem under consideration?
- ② Why is this an interesting / important / relevant / non-trivial research problem?
- ③ What is the proposed solution?
- ④ Why is this a reasonable / promising / successful solution?

Keywords

Categories and Subject Descriptors

D.2.5 [Testing and Debugging]

General Terms

Experimentation, Verification.

Keywords

Real faults, Hand-seeded faults, Mutants

- ACM computing classification system: <https://dl.acm.org/ccs>
- The choice of keywords may influence rankings by search-engines, selection of reviewers, ...

- The introduction section may serve as an extended abstract.
- However, besides providing more details, it may contain additional information not covered by the abstract.

The introduction section may be structured as follows:

1. Background: Overview about the problem domain.
2. Motivation: Relevance of the problem domain.
3. Research question(s): Research problem under consideration.
4. Research goal(s): Which aspects of the research problem are considered?
5. Contribution(s): Proposed solution to achieve the research goals.
6. Outline: Summary of the structure of the text.
7. References to external resources (URL, GitHub, ...).

In practice, we often find two complementary styles to write introductions:

1. A *concise* introduction usually not exceeding 1.5 pages is essentially being structured like the abstract, yet provides some more details.
2. An *exhaustive* introduction often spanning over 2-4 pages is containing additional material like discussion of related work, illustrating examples, concept figures etc.

Introduction Section

- Example: Concise introduction section

< 1 page

Mathematical Programming for Anomaly Analysis of Cluster Models

Markus Weickens, Malte Lochau, Tilo Deuschke, Michael Ritz, Malte Lochau, Tilo Deuschke, Andy Schuster, Tilo Deuschke

ABSTRACT
Cluster analysis (CA) has become an indispensable tool for data analysis. The underlying mathematical models are often NP-hard. This paper introduces a novel approach to the CA problem, which is based on mathematical programming. The proposed approach is able to solve the CA problem for arbitrary data sets and to provide a certificate of optimality for the solution. The approach is able to handle large data sets and to provide a certificate of optimality for the solution.

INTRODUCTION
Cluster analysis (CA) is a fundamental tool for data analysis. It is used to identify groups of objects that are similar to each other. The most common CA method is hierarchical clustering. However, this method is NP-hard. This paper introduces a novel approach to the CA problem, which is based on mathematical programming. The proposed approach is able to solve the CA problem for arbitrary data sets and to provide a certificate of optimality for the solution.

CCS CONCEPTS
Software and its engineering, Software problem domains, Mathematics of computing, Design programming.

KEYWORDS
Cluster analysis, Mathematical programming, Anomaly analysis.

123

Problem and Solution Space

Before the problem and solution space are defined, it is necessary to define the problem and the solution space. The problem is defined as a set of constraints and the solution space as a set of possible solutions. The problem and solution space are defined in terms of the variables and parameters of the problem.

2.1 BACKGROUND AND MOTIVATION
The motivation for this work is the need for a systematic approach to the problem and solution space. The proposed approach is able to solve the problem for arbitrary data sets and to provide a certificate of optimality for the solution.

2.2 Modeling Example
The modeling example is a simple problem with a few variables and parameters. The proposed approach is able to solve the problem for arbitrary data sets and to provide a certificate of optimality for the solution.

2.3 Mapping Between Problem Space and Solution Space
The mapping between the problem space and the solution space is a key concept in the proposed approach. It allows for a systematic approach to the problem and solution space.

- Example: Exhaustive introduction section

4 pages

ACM Reference format

Deuschke, Tilo, Ritz, Michael, Lochau, Malte, and Weickens, Markus. 2019. Automated N-way Program Merging for Building Family-based Analyses of Variant-rich Software. *ACM Trans. Softw. Eng. Technol.* 30, 4, Article 13 (July 2019), 59 pages. <https://doi.org/10.1145/3301170>

1 INTRODUCTION
In many modern application domains, software systems are produced in numerous different, yet similar variants. It is necessary to develop an efficient and systematic approach to the problem and solution space. The proposed approach is able to solve the problem for arbitrary data sets and to provide a certificate of optimality for the solution.

Need techniques for filling established quality assurance techniques. The main testing [35] and model-checking [41]. In variant-rich software, the need for a systematic approach to the problem and solution space is a key concept in the proposed approach.

N-way Program Merging for Family-based Analysis 133

In these parts (during family-based analysis) to exactly those program variants the parts originate from.

In the case of close-on-own development practices, a representation satisfying all these requirements is usually not available a priori and, even in the cases of small- and medium-size families of program variants, it is impossible to be obtained manually on-demand. In addition, even if a product-line implementation exists that already superimposes all variants of a program family, these representations are usually defined in the spirit of *ad hoc* directives of the C programming language and are unsatisfactory. In contrast to this, a feasible representation for facilitating most recent family-based analyses requires one-time variability based on compile-time encoding (i.e., preprocessor conditions) that can be encoded as conventional conditional program statements to simulate program variants by means of conventional conditional flow branches [19, 20].

Existing attempts to automatically derive product-line representations from a set of program variants have several essential drawbacks making them unsuitable for family-based analysis. First, most approaches either apply N-way merging techniques by superimposing N (usually visual) design models rather than program variants [26, 34, 57, 58], or they perform purely local or syntactic matching (therefore, computationally cheap) preparation of match-candidates via inherently iterative results that may lead to non-associative and even unassociative merges as well as merge conflicts, which cannot be resolved automatically [14, 15, 31, 33]. In addition, most existing N-way model merging approaches require all variants to be available at once for merging, whereas in practice, program families continuously evolve, by starting from an initial core program and consecutively deriving further variants on-the-fly. Hence, a feasible N-way merging technique must further allow for incrementally merging arbitrary subsets of N variants/variants in a step-wise and compositional manner such that performing, for instance, N-1 consecutive N-way merging steps would essentially yield a result being similar to performing one N-way merge in one step.

Second, the problem of finding sufficiently precise and generally applicable match-candidates with respect to the program path-coverage problem when comparing the global context of program elements, in addition with the combinatorial explosion problem arising from the number of all possible matches among these elements [35].

Finally, most existing approaches to N-way merging are either not variant-preserving at all, or they augment superimposed representations with additional variability annotated along with related directives. Hence, those approaches may compromise validity that often is even partial variability if the superimposed representation is used for family-based analysis.

To tackle these issues, we present a novel methodology and accompanying tool support, called SMPROSE, for deriving a superimposed representation of a given set of N program variants that satisfies all aforementioned requirements and also provides a systematic approach to the problem and solution space. The proposed approach is able to solve the problem for arbitrary data sets and to provide a certificate of optimality for the solution.

Figure 1: Overview of the approach and our core.

and merge [5]. However, to cope with the inherent complexity of N-way model merging in general and with path-based semantic program models like CIA, in particular, our approach includes several technical novelties, as compared to existing techniques.

First, we utilize principles of *imitation-programming* [19, 44] to internalize similarity-based, yet locally restricted and, therefore, computationally cheap preparation of match-candidates via incremental path-based matching of CIA elements, to reduce the number of potential N-way matches to be considered. In this way, we obtain a reasonable and explicitly controllable trade-off between precision and computational efficiency of the applied similarity measures and matching procedure.

Second, to ensure syntactic well-formedness, semantic soundness and variant-preserving, we encode variability information by dedicated, yet regular conditional CIA fragments, so-called variant points, into the merged CIA [34, 52]. Due to this construction—and in contrast to most other existing approaches—our approach incorporates a fully automated resolution of merge conflicts. In addition, variant points enable compositional merging in the sense that the overall problem of N-way program merging may be decomposed by incrementally deriving the final N-way superimposition from intermediate partial superimpositions of subsets of program variants. This is of particular relevance if not all variants are initially available at once from scratch, but rather being developed (and updated) consecutively (e.g., according to the Groves-and-Prater Model [23]).

Contributions. To summarize, we make the following contributions:

- **Efficient N-way CIA Matching.** We present an efficient, yet sufficiently precise heuristic approach for approximating similarity matches among N CIA variants.
- **Variant-Preserving N-way CIA Merging.** We present an N-way program-merging operator relying on the concept of variant points and our proof-practice well-formalizable, semantic soundness and variant-preserving nature of the resulting superimposed program variants, thus enabling sound applications of family-based program-analysis techniques. The operator further allows for automatically resolving merge conflicts and for incrementally merging N variants/variants being decomposed into arbitrary subsets, into one superimposed representation in a consecutive and compositional manner.
- **Tool Implementation.** We present a tool implementation of SMPROSE, which integrates our algorithm with an existing tool for automated family-based program analysis, based on the software model-checker CBMC [24]. In our evaluation papers, our tool further includes also alternative algorithms a line-based tool enabling merging approach based on

Source code example

Figure 2: Variants of C program created using a clone-and-merge approach.

GNI/Derrn [61] [33] as well as an adoption of the generic NeM Algorithm (NmA) for N-way model matching/merging [51].

- **Experimental Evaluation.** We provide results of our experimental evaluation gained from applying our tool to a rich collection of realistic variant systems. Implemented in C, we investigate applicability and efficiency/effectiveness trade-offs of our approach, by comparing our results with those obtained by a variant-by-variant approach. As analyzed in Section 2, we consider automated (white-box) test generation as well as (software) model-checking. We further investigate the precision of our approach by additionally comparing our results with optimally superimposed representations available for selected community benchmarks. Frequently used for evaluating family-based merging [41]. In addition, we also compare merge precision and computational effort of SMPROSE when merging all N variants at once, as compared to incremental step-wise merging, especially focusing on how accumulating merging. Finally, we conduct a comparison of SMPROSE with the two major existing matching/merging techniques diff and NvM. Here, we observe that SMPROSE outperforms diff in terms of matching/merging precision and NvM in terms of scalability that constitutes, in the average case, the best trade-off between efficiency and effectiveness.

Outline. The remainder of this article is structured as follows. In Section 2, we introduce a running example, a small collection of program variants that is used to motivate our approach and to illustrate the proposed methodology. Section 3 first provides the necessary preliminary definitions concerning CIA and N-way model merging, before the concepts of SMPROSE are described in full detail in Section 4. In addition, Section 5 provides a proof of correctness for our proposed technique as well as an illustrative description of possible practical scenarios of the framework. A tool implementation of SMPROSE and the results of our experimental evaluation, gained from applying SMPROSE to realistic programs, are discussed in Section 6. In Section 7, we give an overview on related work, and Section 8 concludes and provides a look on potential future work.

Verifiability. To make our results reproducible, we provide the tool implementation and all experimental results as well as raw data on a supplementary web page [1].

2 BACKGROUND AND MOTIVATION
In this section, we provide the necessary background and motivation for the concepts described in the remainder of this article.

Program Variants and Program Families. Consider the C code (A) in Figure 2. This function takes as input three integers x , y , and z , assigns the smaller value of x and y to local variable $clone$ (line 2-4), doubles the value of $clone$ (line 5) before subtracting it from z . While the resulting value of z is a regular integer (i.e., is a repetitively added to a float), the value is a floating-point number, assuming that software developers observe the given program to meet user requirements.

- The related work section categorizes and summarizes existing (i.e., already published) scientific works which address similar research problems as the approach described in the scientific text.
- For each such work, the related work section should contain a description what is equal and what is different to the approach described in the scientific text.
- Instead of following the introduction section, the related work section might be alternatively placed at the end (before the conclusion) or inlined into the introduction or background section.

Assignment 2

Discuss the advantages and disadvantages of having the related work section at the beginning or at the end of a scientific text.

Related Work Section

Mnemonic: References like "[42]" are no proper grammatical objects, but should be treated like (invisible) footnotes. “

Counterexample:

"[42] describes the approach XY which is also used in this paper to [...]"

Instead, use one of the following two patterns:

1. „In this paper, we also use approach XY to ... [42].“
2. „Approach XY as described by **Smith et al. in [42]** is also used in this paper to [...].“

Related Work Section

Mnemonic: In natural sciences, engineering and similar disciplines, there are usually only few word-by-word quotations of particular sentences or entire paragraphs. Instead, citations usually refer to facts, concepts, definitions, experimental results etc. In addition, it is usually sufficient to cite a reference only once in a section or paragraph.

Counterexample:

Smith et al. conclude that „Problem XY is undecidable.“ [42, p. 42, Thm. 42].

In addition, they show that „Problem XY is NP-complete“ [42, p. 43, Thm. 43].

Instead:

Undecidability as well as NP-completeness of problem XY has been recently shown by Smith et al. [42].

Mnemonic: A better name for the "related work" section would be "differences to competitive work" section:

1. Not only list *which* works are related to the approach described in the paper, but also describe *how* they are related (*similarity* with the own approach) and how they essentially differ (*novelty* of the own approach).
2. Works building the *theoretical / conceptual foundation* of the described approach should not be discussed in the related work section, but rather in the background section.

Related Work

Example: Suppose a scientific text to describe a novel solution for the graph-coloring problem using quantum computing.

The related work section may be subdivided into three paragraphs:

1. Summary of (most important) existing graph-coloring solutions.
Difference: none of them uses quantum computation so far.
2. Summary of existing quantum computing approaches for solving (most closely related) graph-problems.
Difference: none of them solves this exact same problem so far.
3. Summary of existing quantum computing approaches for solving the graph-coloring problem.
Precisely work out the differences (efficiency, accuracy, ...)

- The background section describes conceptual foundations and basic terminology required to understand the motivation of the research problem and the contributions in the subsequent main part.
- It is a good practice to introduce an example illustrating the concepts and terminology and to motivate the research problem(s) and to use this example also in the main part to describe how the proposed approach solves the problem(s).
- Mnemonic: The background section only contains state-of-the-art introduced elsewhere and does not contain any novelty and contributions.

Background Section

40 S. Ruland et al.

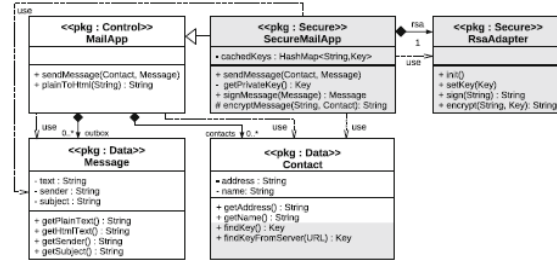


Fig. 1. UML class diagram of MAILAPP

we apply MOMoT [11], a generic framework for search-based model transformations. Our experimental evaluation results gained from applying GOBLIN as well as the recent tools JDEODORANT [12] and CODE-IMP [27] to a collection of real-world JAVA programs provide us with in-depth insights into the subtle interplay between traditional code-quality metrics and attack-surface metrics. Our tool and all experiment results are available on the GitHub site of the project¹.

2 Background and Motivation

We first introduce a running example to provide the necessary background and to motivate the proposed refactoring methodology.

Running Example. We consider a (simplified) e-mail client, called MAILAPP, implemented in JAVA. Figure 1 shows the UML class diagram of MAILAPP, where security-critical extensions (in gray) will be described below. We use stereotype $\langle\langle\text{pkg} : \text{name}\rangle\rangle$ to annotate classes with package declarations. Central class MailApp is responsible for handling objects of classes Message and Contact both encapsulating application data and operations to access those attributes. The text of a message may be formatted as plain String, or it may be converted into HTML using method plainToHtml().

Design Flaws in Object-Oriented Programs. The over-centralized architectural design of MAILAPP, consisting of a predominant controller class (MailApp) intensively accessing inactive data classes (Message and Contact), is frequently referred to as *The Blob* anti-pattern [7]. As a consequence, method plainToHtml() in class MailApp frequently calls method getPlainText() in class Message across

¹ <https://github.com/Echtzeitsysteme/goblin>.

Attack Surface of OO Refactorings 41

class- and even package-boundaries. *The Blob* and other design flaws are widely considered harmful with respect to software quality in general and program maintainability in particular [7]. For instance, assume a developer to extend MailApp by (1) adding further classes SecureMailApp and RsaAdapter for encrypting and signing messages, and by (2) extending class Contact with public RSA key handling: method findKey() searches for public RSA keys of contacts by repeatedly calling method findKeyFromServer() with the URL of available key servers. This program evolution further decays the already flawed design of MAILAPP as class SecureMailApp may be considered as a second instance of *The Blob* anti-pattern: method encryptMessage() of class SecureMailApp intensively calls method findKey() in class Contact. This example illustrates a well-known dilemma of agile program development in an object-oriented world: *Class-Responsibility Assignment* decisions may become unbalanced over time, due to unforeseen changes crosscutting the initial program design [31]. As a result, a majority of object-oriented design flaws like *The Blob* anti-pattern is mainly caused by low cohesion/high coupling ratios within/among classes and their members [5, 6].

Refactoring of Object-Oriented Programs. Object-oriented refactorings constitute an emerging and widely used counter-measure against design flaws [13]. Refactorings impose systematic, semantic-preserving program transformations for continuously improving code-quality measures of evolving source code. For instance, the *MoveMethod* refactoring is frequently used to update CRA decisions after program changes, by moving method implementations between classes [34]. Applied to our example, a developer may (manually) conduct two refactorings, R1 and R2, to counteract the aforementioned design flaws:

- (R1) move method plainToHtml() from class MailApp to class Message, and
- (R2) move method encryptMessage() from class SecureMailApp to class Contact.

However, concerning programs of realistic size and complexity, tool support for (semi-)automated program refactorings becomes more and more inevitable. The major challenges in finding effective sequences of object-oriented refactoring operations consists in detecting flawed program parts to be refactored, as well as in recommending program transformations applied to those parts to obtain an improved, yet behaviorally equivalent program design. The complicated nature of the underlying optimization problem stems from several phenomena.

- **Very large search-space** due to the combinatorial explosion resulting from the many possible sequences of (potentially interdependent) refactoring-operation applications.
- **Multiple objectives** including various (inherently contradicting) refactoring goals (e.g., O1–O3).
- **Many invalid solutions** due to (generally very complicated) constraints to be imposed for ensuring behavior preservation.

Further research especially on the last phenomenon is required to understand to what extent a refactoring actually alters (in a potentially critical way) the

42 S. Ruland et al.

original program. For instance, for refactoring R2 to yield a correct result, it requires to relax declared accessibility constraints: method encryptMessage() has to become public instead of protected after being moved into class Contact to remain accessible for method sendMessage, and, conversely, method getPrivateKey() has to become public instead of private to remain accessible for encryptMessage(). Although these small changes do not affect the functionality of the original program, it may have a negative impact on extra-functional properties like program security. Therefore, the amount of invalid solutions highly depends on the interaction between constraints and repair mechanisms.

Attack Surface of Object-Oriented Programs. The attack surface of a program comprises all conventional ways of entering a software from outside such that a larger surface increases the danger of exploiting vulnerabilities (either unintentionally by some user, or intentionally by an attacker) [20]. Concerning JAVA-like programs in particular, explicit restrictions of accessibility of class members provide an essential mechanism to control the attack surface. Hence, refactoring R2 should be definitely blamed as harmful as the enforced relaxations of accessibility constraints, especially those of the indeed security-critical method getPrivateKey(), unnecessarily widen the attack surface of the original program. In contrast, refactoring R1 should be appreciated as it even narrows the attack surface by setting method plainToHtml() from public to private.

Challenges. As illustrated by our example, the attack surface of a program is a crucial, but yet unexplored, factor when searching for reasonable object-oriented program refactorings. However, if not treated with special care, accessibility constraints may seriously obstruct program maintenance by eagerly suppressing any refactoring opportunity in advance. We therefore pursue a model-based methodology for automating the search for optimal sequences of program refactorings by explicitly taking accessibility constraints into account. We formulate the underlying problem as constrained multi-objective optimization problem (MOOP) incorporating explicit control and minimization of attack-surface metrics. This framework allows us to facilitate search-based model transformation capabilities for approximating optimal solutions.

3 Search-Based Program Refactorings with Attack-Surface Control

We now describe our model-based framework for identifying (presumably) optimal sequences of object-oriented refactoring operations. To explicitly control (and minimize) the impact of recommended refactorings on the attack surface, we extend an existing EMF meta-model for representing JAVA-like programs with accessibility information and respective constraints. Based on this model, refactoring operations are defined as model-transformation rules which allow us to apply search-based model-transformation techniques to effectively explore candidate solutions of the resulting MOOP.

Methodology Section

- The content and style of presentation of the main section(s) highly depend on
 - the type of text,
 - the research domain and community culture,
 - the type of contribution(s),
 - the research methodology applied,
 - ...

- New/extended/improved mathematical theory \Rightarrow definitions, theorems, proofs etc.
- New/extended/improved modeling/programming language concepts \Rightarrow syntax, semantics, pragmatics etc.
- New/extended/improved algorithms \Rightarrow pseudo-code, complexity analysis, correctness proofs etc.
- New/extended/improved tool chain \Rightarrow flow diagrams, data formats etc.
- Empirical studies \Rightarrow design of experiments etc.
- ...

Evaluation Section

- The methodology described in the previous section is experimentally evaluated with respect to the research goals.
- The experimental evaluation may be conducted in many different ways:
 - quantitative measurements/metrics,
 - qualitative comparisons with related approaches,
 - case studies and user studies
 - questionnaires and experiences reports, ...
- The type of experimental evaluation highly influences the way the results are documented and presented in the evaluation section.

Generic Structure of an Evaluation Section

1. Research Questions / Hypothesis
2. Methods and Experimental Design
 - Subject Systems
 - Data Collection
 - Tool Support
 - Measurement Setup
3. Results
4. Discussion
5. Reproducibility
6. Threat to Validity
 - Internal
 - External

Example: Evaluation Section

Example: Evaluation section for a scientific text proposing „your“ new sorting algorithm *QuickSort*.

1. Research Questions

- What is the average run-time required for QuickSort as compared to state-of-the-art sorting algorithms?
- What is the average memory consumption of QuickSort as compared to state-of-tge-art sorting algorithms?

Example: Evaluation Section

2. Methods and Experimental Design

- Subject Systems: Which data collections are used as input by the sorting algorithms (realistic data or synthetically generated data, number of collections, number and type of entries per collection, diversity and distribution of entries per collection etc.).
- Data Collection: How are experimental runs conducted (ordering of subject systems, number of repetitions per subject system etc.).
- Tool Support: Describe relevant details of own prototype implementation of *QuickSort* and those of existing tools for state-of-the-art sorting algorithms used for comparison.
- Measurement Setup: How are measurements performed (CPU time vs. system time, memory footprint etc.)?

Example: Evaluation Section

3. Results

- Show measurement results (e.g., tables, diagrams etc.), grouped by research questions.
- Describe and summarize measurement results (e.g., highlighting min/max values, outliers etc.), grouped by research questions.
- Do NOT answer the research questions and do NOT further discuss or interpret the results.

Example: Evaluation Section

4. Discussion

- Give answers to the research questions based on the measurement results.
- Answers might be:
 - „Yes/No“ (Yes, *QuickSort* requires less CPU time than X-Sort for all subject systems)
 - A factor/percentage value (QuickSort performs, on average, 2.42 times faster than X-Sort)
 - ...
- Summarize the insights gained from the evaluation results (*QuickSort* is improving the state-of-the-art in solving sorting problems).
- Optional: Give a subjective assesement of the evaluation results (*QuickSort* should become the default sorting algorithm of collection API of recent programming languages).

Example: Evaluation Section

5. Reproducibility

- Describe how the reader can reproduce the evaluation results.
- Where to download the measurement raw data?
- Where to download the tool(s) and how to install and run them?
- Where to get the subject systems?
- How to generate own measurement results from experimental runs?
- ...

Example: Evaluation Section

5. Threats to Validity

- Which aspects might harm the feasibility of the evaluation results?
- *Internal* threats are concerned with aspects related to the proposed approach itself and the design of experiments:
 - Are the research questions relevant?
 - Is the new algorithm sound and is the prototype tool bug-free?
 - Is the collection of subject systems representative?
 - ...
- *External* threats are concerned with aspects not directly controllable by the authors:
 - Are the libraries called by the prototype implementation bug-free?
 - Are the tools used for comparison bug-free?
 - ...

Conclusion Section

The conclusion section consists of three parts:

1. Summary of the approach (abstract in past tense)
"In this paper, we proposed a novel approach [...]"
2. Summary of the main results (most important contributions)
"Our evaluation results show that our approach, on average, improves performance of solving XY as compared to recent approaches by [...]"
"Our main theorem proves that XY is indeed undecidable [...]"
3. Outlook on possible future work (what may be done next)
"Based on the insights gained from our experiments, we believe that further improvements may be achievable by [...]"
"Based on our novel proof technique, further open problems like XY may become likewise solveable [...]"
"Based on our new tool, industrial case studies might be conducted [...]"

Acknowledgments

- List of funding bodies, research grants etc. supporting the work of the authors. Example:

Acknowledgments. This work has been supported by the German Research Foundation (DFG) in the Priority Programme SPP 1593: Design For Future – Managed Software Evolution (SCHU 1309/6-1, AP 206/5), by the DFG grants AP 206/4 and AP 206/6, as well as by the Austrian National Research Network S11403 and S11405 (RiSE) of the Austrian Science Fund (FWF), and by the Vienna Science and Technology Fund (WWTF) through grant PROSEED.

- People who have contributed to the research results, but not being involved in the writing process. Example:

Acknowledgement. The authors' thank goes to John Doe for helpful discussions supporting the paper.

References Appendix

- Layouting of lists of references depends on the formatting style:

References

1. Baldan, P., Bruni, A., Corradini, A., König, B., Rodríguez, C., Schwoon, S.: Efficient unfolding of contextual Petri nets. *Theor. Comput. Sci.* **449**, 2–22 (2012). <https://doi.org/10.1016/j.tcs.2012.04.046>
2. Baldan, P., Corradini, A., Montanari, U., Rossi, F., Ehrig, M.: Different semantics of algebraic graph transformation. In: *Handbook of Algebraic Graph Transformation*, vol. 3, pp. 1–44. Springer, 1999. https://doi.org/10.1142/9789812814951_0003
3. Bunke, H.: Programmed graph grammars. In: Claus, V., (eds.) *Graph Grammars 1978*. LNCS, vol. 73, pp. 155–193. Springer, 1979. <https://doi.org/10.1007/BFb0025718>
4. Corradini, A., Montanari, U., Rossi, F.: Graph processing. *Journal of Supercomputing*, **26**(3/4), 241–265 (1996). <https://doi.org/10.3233/FI-1996-2634>
5. Corradini, A., et al.: On the essence of parallel independent and sesqui-pushout approaches. In: Heckel, R., Taentzer, G. (eds.) *Graph Transformation, Specifications, and Nets*. LNCS, vol. 10800, pp. 1–24. Springer, 2018. https://doi.org/10.1007/978-3-319-75396-6_1
6. Dassow, J., Păun, G., Salomaa, A.: Grammars with context. *Journal of Supercomputing*, **26**(3/4), 267–290 (1996). <https://doi.org/10.3233/FI-1996-2634>

Springer

9. REFERENCES

- [1] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Pearson Education, 1994.
- [2] D. L. Parnas, “Software Aging,” in *Proceedings of the International Conference on Software Engineering (ICSE)*. IEEE, 1994, pp. 279–287.
- [3] M. Abbes, F. Khomh, Y.-G. Gueheneuc, and G. Antoniol, “An Empirical Study of the Impact of Two Antipatterns, Blob and Spaghetti Code on Program Comprehension,” in *Proceedings of the European Conference on Software Maintenance and Reengineering (CSMR)*. IEEE, 2011, pp. 1–10.
- [4] D. Sjöberg, A. Yamashita, B. C. D. Anda, and T. Dyba *et al.*, “Quantifying the Effect of Code Smells on Maintenance Effort,” *TSE*, vol. 39, no. 8, pp. 1144–1156, 2013.
- [5] S. Olbrich, D. Cruzes, and D. I. Sjöberg, “The Impact of Code Smells on Program Comprehension,” in *Proceedings of the European Conference on Software Maintenance and Reengineering (CSMR)*. IEEE, 2011, pp. 1–10.

ACM

References

- [AK09] S. Apel and C. Kästner. An Overview of Feature-Oriented Software Development. *Journal of Object Technology*, 8(5):49–84, 2009.
- [BLL⁺14] J. Bürdek, S. Lity, M. Lochau, M. Berens, U. Goltz, and A. Schürr. Staged Configuration of Dynamic Software Product Lines with Complex Binding Time Constraints. In *VaMoS '14*, pages 16:1–16:8. ACM, 2014.
- [CE00] K. Czarnecki and U. W. Eisenecker. *Generative Programming: Methods, Tools, and Applications*. ACM Press/Addison-Wesley Publishing Co., 2000.
- [CHU04] Krzysztof Czarnecki, Simon Helsen, and Eisenecker Ulrich. Staged Configuration Using Feature Models. In *SPLC'04*, pages 266–283, 2004.
- [FDG08] R. Froschauer, D. Dhungana, and P. Grünbacher. Managing the Life-cycle of Industrial Automation Systems with Product Line Variability Models. In *EUROMICRO*, 2008.
- [HHPS08] S. Hallsteinsen, M. Hinchey, Sooyong Park, and K. Schmid. Dynamic Software Product Lines. *Computer*, 41:93–95, 2008.

LNI

- Appendices contain additional material which may help interested readers to gain more information about the contents of the paper.
- The additional material „does not count“ as a contribution and should not be required to understand the scientific text.
- In some cases, the scientific text contains extracts or simplified versions of some content, which is then completely listed in an appendix.
- Examples: Mathematical proofs, tables with measurement results, code listings of algorithms, visual/mathematical models of a theory, survey results, ...

Outline

- Introduction
- Generic Structure of Scientific Texts
- **Academic Writing Guidelines**
- The Small Things

Disclaimer

In the following, we discuss some guidelines for academic writing. These guidelines are not formal rules enforced by some authority, but rather constitute an informal collection of experiences and best practices in terms of unwritten laws widely adhered to in academic writing practices. However, depending on the particular research community or personal preferences of (leading) people involved in an academic writing project, some of the guidelines may substantially differ from what is discussed in the following.

Guideline 1

1. Use simple and precise language.

- Mnemonic: A sentence should require repetitive reading only due to its difficult content, but not due to complicated language.

- Write short and concise sentences each dealing with exactly one clearly identifiable topic and avoid nested sentences. On the other hand, also avoid staccato style.

Counterexample:

„Algorithm A, whose worst-case run time complexity is in $O(n \log n)$, is shown in code listing 42 using C-like syntax.“

Instead:

„Algorithm A is shown in code listing 42 using C-like syntax. The worst-case run-time complexity of Algorithm A is in $O(\log n)$.“

More on Guideline 1

- Write sentences preferably in *S-P-O style* having a clearly identifiable subject with predicate and object located nearby.

Counterexample:

„In order to be better comprehensible for the reader, S-P-O-style should be preferred for writing sentences.“

What is subject, predicate, object in this sentence?

Instead:

„Sentences ^S should be ^P written in ^O S-P-O-style in order to be better comprehensible for the reader.“

- Agree on the usage of a minimal set of established terminology and invent new terminology only if really necessary.

Counterexample:

„We consider **edge-integer-value-labeled** graphs (i.e., graphs whose edges are labeled with integer values).“

Instead:

„We consider **weighted** graphs (i.e., graphs whose edges are labeled with integer values).“

More on Guideline 1

- Use terminology consistently: use the same terms for the same things throughout the whole text.

Counterexample:

„A **test case** consists of a set of input values together with the expected output value [...] Executing a **test** on a given program unit means to [...]“

Is this the same thing as a test case?

Instead:

„[...] Executing a **test case** on a given program unit means to [...]“

More on Guideline 1

- Repeat phrases and terms whenever it prevents from confusion.
- Mnemonic: Necessary repetitions are not bad style.

Counterexample:

“A graph is a discrete structure used in graph theory. **It** consists of a set of nodes and a set edges. Each of **them** connects exactly **two**.”

Graph or graph theory?

Instead:

„A **graph** is a discrete mathematical structure in graph theory. A **graph** consists of set of nodes and set of **edges**. Each **edge** connects exactly two **nodes**...”

Edges or nodes?

Nodes or edges?

- Avoid meaningless fill-words and further redundancy and use reduced vocabulary.

Counterexample:

„In particular, in the context of this paper, we decided to define a so-called graph to consist of exactly two different sets of objects, namely one containing nodes as well as the other one which contains the edges of our graph.“

Instead:

„A graph consists of a set of nodes and a set of edges.“

More on Guideline 1

- Further examples:

furthermore, moreover, in addition, additionally, obviously, often, thus, therefore, hence, as a matter of fact, to this end, as a consequence, consequently, in other words, actually, however, nevertheless, particularly, as already described above, it shall be pointed out, usually, in general, generally, ...

- Excessive usage of those phrases does not only obstruct comprehensiveness and unnecessarily bloats the text, but also leaves the impression that the authors want to distract the readers from unsure or imprecise content.

Guideline 2

2. Use active form and present tense.

- Mnemonic: Write the text as if (1) the authors and readers are simultaneously reading the text together and as if (2) the text describes everlasting facts.

More on Guideline 2

Counterexample:

„Theorem 1 **is proven** as follows [...]“

Instead:

„**We prove** Theorem 1 as follows [...]“

Counterexample:

„The authors of the related paper **have proven** [...]“

Instead:

„The authors of the related paper **prove** [...]“

Counterexample:

„In the following section, we **will discuss** [...]“

Instead:

„In the following section, we **discuss** [...]“

Guideline 3

3. Use linear lines of argumentation.

- Mnemonic: Each sentence of a paragraph should always be a direct consequence of the preceding one.

More on Guideline 3

- Avoid forward or backward pointers in descriptions and explanations.

Counterexample:

„A weighted graph (V,E,w) consists of a set V of vertices and a set E of edges which we have already described in the previous section. We ignore the third component w for now which is described in more detail in the next section.

Instead:

„A graph (V,E) consists of a set V of vertices and a set $E \subseteq V \times V$ of edges being a subset of pairs of vertices. In the next section, we further extend this definition to weighted graphs.“

More on Guideline 3

- Consecutive sentences should be logically connected. If not, begin a new paragraph.

Counterexample:

Edges may denote available connections between vertices in a communication network modeled as a graph. A weighted graph can be used to further specify communication delays between vertices. **In this work, we consider client-server communication as network protocol.**

Is this a consequence of using (weighted) graphs as a network model?

Guideline 4

4. Use inductive descriptions.

- Mnemonic: Whenever you ask yourself if the reader needs an example to better understand what you just wrote than the answer is always: yes. In addition, this example should even precede what you just wrote.

- First introducing a concrete example and then generalizing the illustrated concepts into a theory is usually more convenient for both the reader and the authors than the other way round.

Example:

1. Show and describe a figure of a (weighted) graph describing an example for a communication network.
2. Give a mathematical definition of a weighted graph comprising the previous example as a possible instance.

- To describe what something is, it is helpful for the reader to additionally describe what it is not.

Example:

Edges may denote available connections between vertices in a communication network modeled as a graph. **However, the representation as a plain graph abstracts from further details like the distance between vertices, the connection type and protocol used for communication. Instead, edges solely express the fact that some connection exists between vertices.**

Guideline 5

5. Constantly pick up the reader and do not build up tension.

- Mnemonic: The reader should never feel „lost“ in the text.

More on Guideline 5

- Frequently remind the reader what has been described so far and what follows next and why.
- It should be always clear why something is described at some point.

Example:

„In the following section, we first recapitulate the mathematical theory of weighted graphs building the foundation for the approach proposed in the main part.

[...]

To conclude this section, we have recapitulated the mathematical theory of weighted graphs. In the next section, we will use weighted graphs to model delays in communication networks.“

Guideline 6

6. Use non-textual elements with care.

- Mnemonic: Check for each non-textual element (figures, tables etc.) you plan to add to the text if this element (1) increases the overall information for the reader by complementing the textual descriptions and/or (2) reduces the overall amount of required textual descriptions.

More on Guideline 6

- Non-textual elements should be meaningful and should serve a useful purpose (i.e., not just being a page filler or nice-looking gimmick).

Counterexample:

„[...] The essential concepts of Industry 4.0 are shown in Fig.1. [...]“

Does this figure really help the reader to gain a better understanding of what Industry 4.0 actually is?



Fig.1: The Essence of Industry 4.0

More on Guideline 6

- Spend effort on the visual quality of non-textual elements and use a coherent style.

Counterexample:

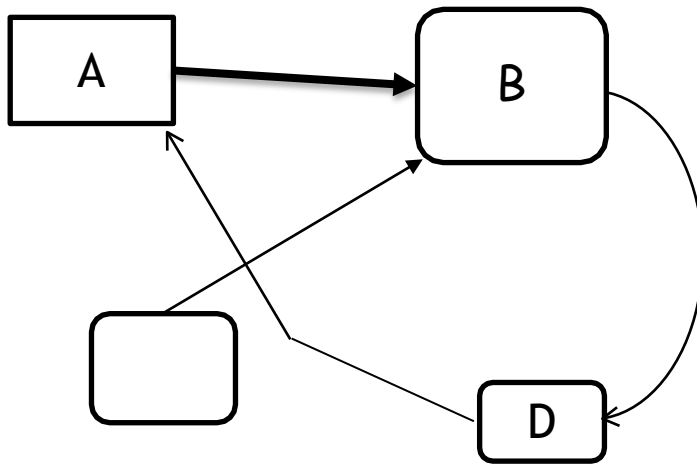


Fig. 1: A Graph denoting a Network Topology

Instead:

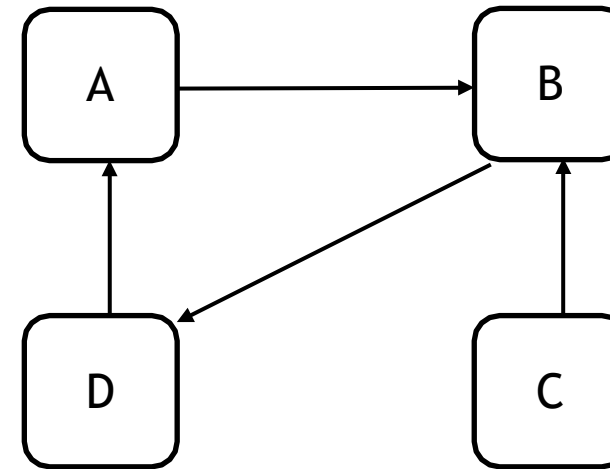


Fig. 1: A Graph denoting a Network Topology

Assignment 3

Your task is to make a list of guidelines improving the visual quality of self-drawn figures by considering the previous example.

More on Guideline 6

- Everything contained in a non-textual element must be described in the text. If something is not relevant, do not show it.

Counterexample:

„Figure 1 shows an example of graph representing a network topology.“

Instead:

„Figure 1 shows an example of graph representating a network topology that consists of four nodes named A, B, C, D. Nodes are visualized as vertices wich are depicted as reactangles with curved angles being labeled by the respective node names [...]“

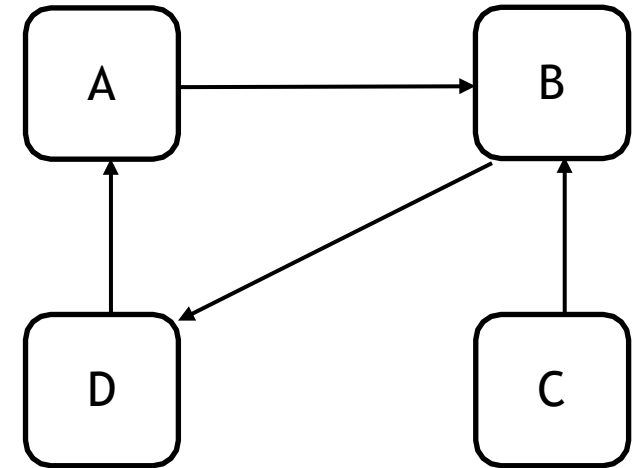


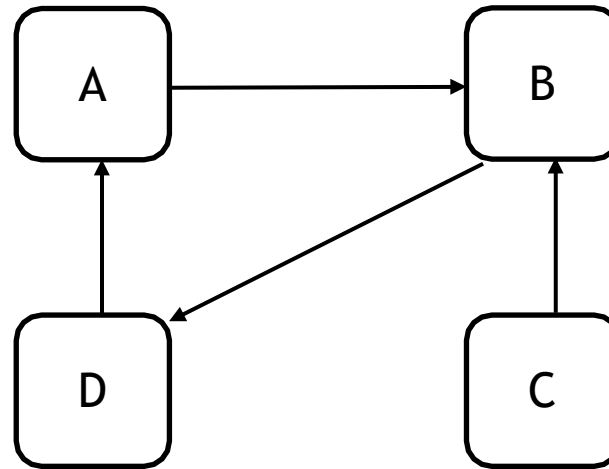
Fig. 1: A Graph denoting a Network Topology

More on Guideline 6

- Insert compound non-textual elements as labeled floating objects (i.e., no manually enforced positioning).
- Refer to those objects in the text only by their labels to enable dynamic repositioning of those objects in case of later changes.

Counterexample:

„An example of a graph representing a network topology is shown **here:**“



Guideline 7

7. Harmonize the text structure and layout.

- Mnemonic: Try to make each page of your text roughly look the same, at least from a bird-eye perspective.

-
- Avoid too many hierarchy levels and too small sections, paragraphs etc.
 - Avoid too long sections, paragraphs, ... with too much/diverse content and walls of text.
 - Avoid singular pages (e.g., pages only containing figures, only containing walls of texts, ...).
 - Balance the length of item-lists (rule of thumb: 4-8 items).
 - ...

Guideline 8

8. Separate opinions from facts.

- Rule of thumb: Check for each statement if it describes a provable or citeable fact or if it expresses your own opinion. In both cases, make sure that this is clearly marked in the text.

More on Guideline 8

Counterexample:

„As shown in Table 1, the measured execution times for validating the input data ranges from 1ms to 10s which is acceptable in an industrial setting.“

Instead:

- Only describe the range of measurement values in the „Results“ section and discuss the interpretation of those results in a separate „Discussion“ section.
- In any case, also opinions must be justified (here: *why* do you think that this is acceptable in an industrial setting?)

Guideline 9

9. If in doubt, provide more explanation.

- Mnemonic: Whenever you ask yourself if the sentence you just wrote is comprehensible for the reader, then the answer is: no. Add more explanation or reminders, but without simply repeating yourself.
- Mnemonic: The average reader usually knows / remembers / understands less than you expect.

More on Guideline 9

Counterexample:

„Figure 1 shows a network topology using the **well-known** graph representation.“

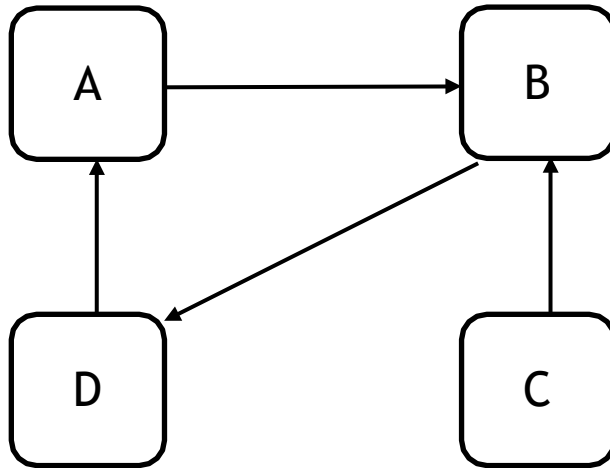


Fig. 1: A Graph denoting a Network Topology

Is this really well-known by any reader or does it require more explanation?

Guideline 10

10. Be honest about weaknesses.

- Mnemonic: You cannot hide any weakness as there is always some reader out there who will eventually recognize it.

- Do not hide or obfuscate unappreciated facts or results by „tuning“ / „over-fitting“ experimental setups or the selection of subject systems to your approach.
- Comparisons with related approaches should be reasonable and fair.
- The subsection on „Threats to Validity“ might be the right place to describe potential weaknesses.
- In the worst, blame the whole idea or contribute it as a negative result.

Outline



- Introduction
- Generic Structure of Scientific Texts
- Academic Writing Guidelines
- **The Small Things**

Finishing a Scientific Text

Summarizing the current status of a scientific text as „almost finished“ is often misleading as there are usually numerous tedious and time-consuming „small things“ left to be done:

- commenting and polishing,
- shortening to reach the page limit,
- final proof reading.

Commenting and Polishing

- Process of repeatedly/concurrently letting co-authors read and comment the current (prefinal/feature-complete) version of the text.
- The leading author decides for every comment, if and how it should be addressed in the next version of the text.
- The leading author requires a good overview on the whole text in order to estimate the impact of changes.
- The leading author must be open for criticism and diverse points of view.
- Co-authors commenting a draft try to take the perspective of reviewers.

Commenting: Example

40 S. Ruland et al.

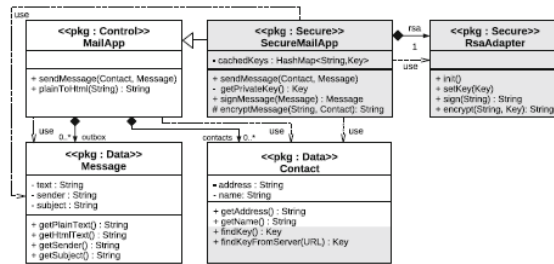


Fig. 1. UML class diagram of MAILAPP

we apply MOMoT [11], a generic framework for search-based model transformations. Our experimental evaluation results gained from applying GOBLIN as well as the recent tools JDEODORANT [12] and CODE-IMP [27] to a collection of real-world JAVA programs provide us with in-depth insights into the subtle interplay between traditional code-quality metrics and attack-surface metrics. Our tool and all experiment results are available on the GitHub site of the project¹.

2 Background and Motivation

We first introduce a running example to provide the necessary background and to motivate the proposed refactoring methodology.

Running Example. We consider a (simplified) e-mail client, called MAILAPP, implemented in JAVA. Figure 1 shows the UML class diagram of MAILAPP, where security-critical extensions (in gray) will be described below. We use stereotype `<<pkg : name>>` to annotate classes with package declarations. Central class MailApp is responsible for handling objects of classes Message and Contact both encapsulating application data and operations to access those attributes. The text of a message may be formatted as plain String, or it may be converted into HTML using method `plainToHtml()`.

Design Flaws in Object-Oriented Programs. The over-centralized architectural design of MAILAPP, consisting of a predominant controller class (MailApp) intensively accessing inactive data classes (Message and Contact), is frequently referred to as *The Blob* anti-pattern [7]. As a consequence, method `plainToHtml()` in class MailApp frequently calls method `getPlainText()` in class Message across

¹ <https://github.com/Echtzeitsysteme/goblin>.

Comment by John Doe: The figure looks nice but requires more explanation in the text. In addition, the example does not contain all interesting cases which I require in my descriptions in the main part.

Comment by John Doe: Comma missing after „and“.

Comment by John Doe: The title of this section is misleading as the motivation follows in the subsequent section. Remove „Motivation“.

Comment by John Doe: The first sentence of this paragraph is too long and hard to understand. Please split up.

Comment by John Doe: We should add a citation to [42] at the end of this sentence.

Comment by John Doe: This paragraph might be shortened or even removed as it is not really relevant for the remainder of the text.

Page Limit

- Most scientific texts have to adhere to a strict page limit in a predefined document template and font formatting style.
- Example:

Each submission will be carefully reviewed by at least three members of the research track program committee. The page limit is 10 pages of content (+ 2 pages for references) for full papers and 5 pages of content (+ 2 pages for references) for short papers. Submissions must adhere to the latest ACM Master Article Template:

<https://www.acm.org/publications/proceedings-template>

- In almost all cases, the authors would like to write more than the page limit permits.
- However, it is usually not helpful to have the page limit in mind already at the beginning of writing.
- Instead: Agree with co-authors on a budget of pages per section, try to describe the content roughly within these boundaries and then shorten the text as a whole.

Example: Page Budget

1. Front Matter, Abstract, Introduction Section	[1.5 pages]
Author: Jane Doe	
2. Background Section	[2.0 pages]
Author: John Doe	
3. Methodology Section	[3.0 pages]
Author: John Doe	
4. Experimental Evaluation Section	[2.5 pages]
Author: John Doe	
5. Related Work Section	[0.5 pages]
Author: Jane Doe	
6. Conclusion Section	[0.5 pages]
Author: Jane Doe	
	<hr/>
	[10 pages]

Shortening

- Page limits are usually strict thus requiring tedious shortening of the final text before submission.
- Mnemonic: Instead of removing actual content like a whole paragraph or figure, it is usually more effective to go through the whole text again and to perform a lot of small shortening measures (e.g. removing fill-words, slightly down-scaling figures etc.)
- Make sure to have enough time left for shortening.

- In the last step before submission thoroughly read the whole text once again and look for typos, layouting and formatting issues and obvious inconsistencies etc.
- The actual content of the text should be freezed at this point in order to avoid a „moving target problem“ leading to infinite iterations (e.g., adding further descriptions may require further shortenings etc.).

Section 2. Reviewing Scientific Texts

Outline



- Introduction
- Evaluation Criteria for Reviews
- Generic Structure of Reviews
- Writing Guidelines for Reviews
- Literature Review (from Scribbr.com)

Outline

- **Introduction**
- Evaluation Criteria for Reviews
- Generic Structure of Reviews
- Writing Guidelines for Reviews

- The goal of a review is to evaluate/ensure the quality of a scientific text.
- A review is usually prepared by a scientific expert (reviewer) coming from the same research community as the authors of the scientific text under review (peer review).
- A reviewer has the responsibility that the scientific text sufficiently meets the criteria required for being accepted as a contribution at a scientific conference and/or for being published in a scientific journal.

Example: Paper Review

Number of review for
this paper

reviewer's name

reviewer's decision

evaluation text
for the authors

expert level of
reviewer

Review 1	
PC member	[REDACTED]
Time	Dec 09, 18:05
	2: (strong accept)
Overall evaluation	<p>This work presents a way to give users feedback about a configuration they have selected and based on a history of previously selected configurations. By feedback, authors mean whether a configuration can be inconsistent with regards to a variability model and from which combination of features the problem comes from. To do so, they rely on an observation matrix and matrix factorization (Singular Value Decomposition).</p> <p>The idea is interesting to the community, well presented and well motivated. Even if the original matrix may not be new, using matrix factorization to highlight important elements is interesting and not completely straightforward. I think it is a nice start and that the idea should be pushed further.</p> <p>In particular, I wonder whether the fact that the initial is sparse is a problem? The cost to fill it completely will be high for sure, but could authors rely on a pairwise selection strategy to help cover the configuration space? Maybe it could also balance the fact that the matrix is sparse.</p> <p>Do authors think that the form of Table 4 can be exploited/interesting? I mean they do not seem to use it in their approach (online phase), but maybe the coefficients can be of some kind of interest? Is it a problem that numbers in this matrix are not positive and between 0 and 1?</p> <p>What is the impact of k over the results? (accuracy specifically; runtime will probably increase)</p>
Reviewer's confidence	4: (high)
Confidential remarks for the program committee	

Number of Reviews



- A scientific text submitted as a paper to a conference or as a manuscript to a journal is usually reviewed by at least three reviewers.
- The overall review decision is made by the PC chair or the journal editor based on the reviews, usually by some kind of majority vote in case of inconclusive reviews.

- None-blind reviews: the reviewers have access to the names of the authors and vice versa.
- Single-blind reviews: the reviewers have access to the names of the authors, but not vice versa.
- Double-blind reviews: the reviewers have no access to the names of the authors and vice versa.
- Triple-blind reviews: in addition to double-blind, the different reviewers have no access to the names of the others reviewers and/or their reviews.

Conferences papers:

- Strong accept (+3)
- Accept (+2)
- Weak accept (+1)
- Borderline (0)
- Weak reject (-1)
- Reject (-2)
- Strong reject (-3)

Journal article manuscripts:

- Accept without changes
- Accept with small changes
- Accept after minor revision
- Accept after major revision
- Reject
- Desk reject

Outline

- Introduction
- **Evaluation Criteria for Reviews**
- Generic Structure of Reviews
- Writing Guidelines for Reviews

Categories of Evaluation Criteria

-
- (A) Format
 - (B) Language
 - (C) Structure
 - (D) Soundness and Completeness
 - (E) Contribution
- } Style and Presentation
- } Content

- Is the text format in compliance with the document template (ACM, Springer, IEEE, ...)?
- Is the paper length within the given limits (usually 8 - 10 pages)?
- Is all front matter information included (title, authors, keywords, ...)?
- Further formatting guidelines are, in most cases, automatically enforced by the document template (text font, page layout, bibliographic style etc.)

- Is the text written in the required language (English in most cases)?
- Is the text linguistically correct (spelling, grammar, punctuation etc.)?
- Is the text comprehensible (correct usage of terms, appropriate sentence structure and length, logical coherence etc.)?
- Is the text written according to scientific writing rules?

- Is the text structured according to established conventions (Introduction, Background, ..., Conclusion)?
- Is the length of sections, subsections, paragraphs etc. appropriate and balanced?
- Is the usage of non-textual elements (figures, tables, listings etc.) sufficient and appropriate?
- Are titles of sections, labelings of non-textual elements etc. appropriate and meaningful?

Soundness and Completeness

- Are textual descriptions, claims and conclusions factually correct and/or sufficiently justified by citations?
- Are examples sufficiently used, meaningful and do they help to illustrate and explain the concepts, descriptions, results etc.?
- Are mathematical definitions and proofs sound and complete?
- Are experimental evaluations methodically conducted according to scientific practices, documented sufficiently and are the drawn conclusions justified by the results?

Soundness and Completeness

- Are all descriptions comprehensive enough to enable readers with different background knowledge to at least understand the key contribution without consulting any further sources (self-containedness)?
- Are the descriptions coherent and consistent?
 - Does the text stay on a clearly identifiable topic or does it digress?
 - Do title and abstract properly reflect the main content and contributions or does it promise too much/less?
 - Does the experimental setup of the evaluation properly address the research questions?
 - Do mathematical theories used constitute sound abstractions for the phenomena under consideration?

Contribution



- Is the contribution precisely described? Does the proposed solution address a clearly identified research problem?
- Is the contribution relevant? Is the addressed research problem open, important and non-trivial?
- Is the contribution novel? Has the proposed solution for the research problem never been considered before and is the approach compared with all relevant related works?
- Is the contribution convincing? Does the proposed solution really tackle the research problem and provide novel insights into the root of the problem?

Contribution



- Is the contribution reproducible? Is a web page, tool etc. available which allows readers to repeat the experiments?
- Is the contribution generalizable? Has the proposed solution potentials to be applicable or adaptable to similar research problems?

Outline

- Introduction
- Evaluation Criteria for Reviews
- **Generic Structure of Reviews**
- Writing Guidelines for Reviews

Generic Structure of Reviews

1. Summary
2. General assessment
3. Detailed evaluation
4. Overall decision
5. Minor remarks

Summary

- The reviewer first summarizes the text under review in his/her own words (4-8 sentences).
- The summary should be as *objective* as possible.
- Mnemonic: Pretend that you have to write your own abstract for the text under review.
- Example:

Summary

The paper describes an integrated approach for applying principles of model-driven engineering (MDE) to software product lines in the contexts of automation engineering systems and robotics. The approach focuses on UML models dealing with the description of system and software architectures at a coarse-grained level such as the composite structures of the software system. One goal of the envisioned approach is to only use modeling constructs being a native part of UML to express variability in an integrated way, instead of extending and/or separating those aspects from the base models. The approach combines recent tools such as FeatureIDE to support developers in using the proposed methodology.

General Assessment

- The actual review should begin with a paragraph summarizing the overall impression of the reviewer.
- This paragraph is subjective, yet it is good practice to begin with some positive points.
- Mnemonic: Answer the following two questions in 2-3 sentences:
 - Which positive point(s) about the text are particularly memorable for you?
 - Which negative point(s) about the text are particularly memorable for you?

General Assessment

- Example: Positive points

First of all, I would like to emphasize that I like the idea to use historical data to tackle the problem described in the paper. The problem considered in the paper is relevant and of high relevance for the VAMOS community. In addition, the paper is, in most parts, very readable and easy to follow. The style format is adhered by the authors and figures/tables are properly scaled and positioned. US English is used uniformly throughout the work which is fine. In the sections 'Working Example' and 'Conflict Detection', the authors describe foundations of feature models, configurations and conflict detection in a comprehensible way. They use suitable figures and tables for their explanations.

- Example: Negative points

Although the motivated problem and the proposed solution are passable, there are several issues with the conceptual description of this problem, the realization of the evaluation, and feasibility of the overall approach in general. Details are described in the following.

Detailed Evaluation

- The main part of the review describes in detail *everything* the reviewer has to say about all evaluation criteria (C)-(E).
- Mnemonic: Sort points in descending order from (E) to (C).

General Assessment

- **Example:**

- The approach described in the paper misses a convincing motivation (which is also due to the very short introduction). No problem statements or research questions are given. What is the problem with existing approaches and why? Why not just using an existing solution such as UML profile as proposed by CVL or one of the many variability-encoding approaches? What are the specific needs for variability modeling and variability-aware MDE in the considered application domains of industrial automation and robotics?
- The paper lacks a clear contribution: As far as I understand the proposed approach, it essentially comprises a 150% modeling approach as usual and employs build-in constructs from UML modeling to encode variability into plain base models. So far so good, but this is nothing new and has been done in similar ways in many previous works before (e.g., CVL and Clafer, just to name two of the most prominent approaches). In addition, it is well-known that 150% modeling approaches (a.k.a. model superimposition or annotative modeling) are problematic in practice as developers require a-priori knowledge about all derivable variants in advance to come up with a proper base model – which has shown to be unrealistic and often also infeasible in practice. In addition, the authors mention support for MDE techniques to motivate their approach. However, the quite basic and coarse-grained structural models alone as considered in this paper are not sufficient to exploit MDE principles which therefore additionally require behavioral models and/or code units etc. Nevertheless, adding those further artifacts to the problem is far from being trivial.
- The related work section is also not satisfactory. As correctly mentioned by the authors, there is a huge body of related approaches out there, so it would be even more important to be very clear about the contribution and novelty of the proposed approach to distinguish it from recent approaches. Here, only a small selection of works is discussed which not sufficiently covers the related work. Although, the closely related works of Haugen et al. are indeed mentioned, no clear distinctions as compared to those works are discussed.
- The presentation of the proposed approach in the main part of the paper is very shallow. On the one hand, the general idea the approach is based upon remains very abstract (also due to the lack of an illustrating example). On the other hand, the descriptions given mostly comprise some technical details which are not very helpful to understand the main ideas and novelties behind the approach. The algorithms shown in the paper are mostly trivial and do not provide any further insights for the reader.
- Finally, no proper evaluation results concerning a systematic study of usability, applicability etc. of the approach – at least with respect to the use case – are given. Has the approach successfully been applied in the application domain and provided some recognizable improvements there?

Detailed Evaluation

- Recommend further relevant related work which have not been referenced by the authors.
- Mnemonic: Do not aggressively enforce the authors to add references to your own work.
- Example:

While the authors provide a comprehensive overview of the related work with regards to the motivation of their research, the authors missed contributions to the solution side. For instance, the reviewers previously used FeatureIDE to configure products, which in fact provides users with explanations why features must or cannot be selected [1]. The reviewers would find a discussion on how this mechanism can be integrated into the authors' technique intriguing but at the very least would like to see a discussion of the upsides of the authors' technique. Furthermore, previous work involving matrix factorization by Pereira et al. from 2017 [2] has not been addressed, while being closely related to the authors' approach.

[1] Explaining Satisfiability Queries for Software Product Lines, Timo Günther, 2017

[2] A Feature-Based Personalized Recommender System for Product-Line Configurations, Pereira et al., 2017

Overall Decision

- Itemize the most important positive and negative points and give the final review decision.
- Example: Summary of positive and negative points

- + Motivation
- + Well written
- o Extensive Background
- Incomplete related work
- Very limited evaluation

- Example: Final decision

To sum up, I vote to reject this paper due to the unclear motivation, the confusing descriptions of the technical details and the very weak evaluation.

Minor Remarks

- Itemize all minor issues related to evaluation categories (A) and (B).
- Mnemonic: If the number of language issues exceed an unacceptable threshold, it is not necessary anymore to list all of them explicitly.
- Example:

-----Minor corrections-----

* Line 83: "Like us"

* Figures 1 and 2 are quite hard to read when printed.

* Use the capital letter to refer to text elements. "figure 1" > "Figure 1", "algorithm 1" > "Algorithm 1"

* The text need a careful proofreading.

Outline

- Introduction
- Evaluation Criteria for Reviews
- Generic Structure of Reviews
- **Writing Guidelines for Reviews**

Use first person perspective or passive. Do not address authors directly, but refer to them as „the authors“.

Examples:

- „I do not understand the statement made by the authors on page 42 [...]“
- „The statement made by the authors on page 42 is hard to understand [...]“

Refer to concrete locations within the text whenever possible.

Examples:

- „For the statement on page 42, line 13-17, a reference should be provided by the authors.“
- „The second paragraph on page 23 should be moved to Section 3.“

Describe points of criticism as polite as possible and show appreciation for the authors' effort.

- Example: „The prove of the main theorem is an impressive theoretical work. However, the authors shall carefully revisit step 3 on page 42 [...]“
- Mnemonic: While writing the review, always remind yourself how much work it is to conduct scientific work and to prepare a scientific text.

Writing Guidelines for Reviews

Sort points for the detailed evaluation in descending order of importance.

- Example: A mistake in the proof of the main theorem of the mathematical theory is more important than a missing comma.

Criticism should be as constructive as possible.

- Example: „The descriptions in this paragraph are hard to understand as they remain very abstract. It would help the reader to see a concrete example showing [...]“.
- Mnemonic: Criticism without any justification and proposals for improvement is frustrating and useless.

Criticism may also contain positive aspects.

- Example: „Although some parts of the mathematical theory are flawed, I see a lot of potential in the proposed approach to tackle the problem [...]“.
- Mnemonic: If you like something, tell it to the authors and encourage them to go on working on a good idea even in case of rejection.

Proposals for improvement should be practicable.

Counterexamples:

- „The authors should first prove that $P \neq NP$ to convince me that the research problem is really that hard to solve.“
- „The authors should perform experimental measurements on at least 100.000 subject systems to show feasibility of the approach.“
- „The authors should add a further section describing all concepts of the C programming language.“

Be open for new ideas, alternative viewpoints and different styles and flavors.

Counterexamples:

- „I do not like the assumption made by the authors that [...], so I skipped all the following descriptions.“
- „Referring to a variable as x is clearly wrong, it should be called v !“

Be honest about your expertise.

- Example: „I am not an expert in the research area the paper is dealing with. As far as I am able to follow the descriptions, the research problem appears to be interesting and the proposed solution seems reasonable. However, other reviewers being more experts in this field should be better able to judge about the feasibility and correctness of the proposed approach.“

Extra notes



- Scientists often complain about the reviewing process.
- Nevertheless, peer reviewing as a voluntary community service is one of the most important pillars of effective research.

Literature reviews

Introduction and step-by-step guide

What is a literature review?

A literature review...

- ↳ Surveys scholarly sources on a specific topic
- ↳ Provides an overview of current knowledge
- ↳ Points out gaps in existing research
- ↳ Appears as part of a dissertation or on its own

Purpose of the literature review

Demonstrate familiarity with the topic and scholarly context

Develop a theoretical framework and methodology

Position your approach in relation to other researchers

Show how your research fits in

Conducting a literature review: 5 steps



How to write a
Literature
Review

A young woman with long dark hair, wearing a brown turtleneck sweater, is shown from the chest up. She is holding a large stack of colorful books in her left arm and has her right hand on her head, looking slightly to the right with a frustrated or overwhelmed expression. The background behind her is a teal circle containing the Scribbr logo and the text 'How to write a Literature Review'.

Step 1

Search for relevant literature

Defining your research problem

- ↳ Effects of social media
 - Social media & body image
 - Social media & body image among Gen Z

What is the impact of social media on body image among Generation Z?

Identifying keywords

- 🔗 **Social media**, Facebook, Instagram, Twitter, Snapchat, TikTok
- 🔗 **Body image**, self-perception, self-esteem, mental health
- 🔗 **Generation Z**, teenagers, adolescents, youth

Where to search

- 🔗 Your university's library catalogue
- 🔗 [Google Scholar](#)
- 🔗 [JSTOR](#)
- 🔗 [EBSCO](#)
- 🔗 [Project Muse](#) (humanities and social sciences)
- 🔗 [Medline](#) (life sciences and biomedicine)
- 🔗 [EconLit](#) (economics)
- 🔗 [Inspec](#) (physics, engineering and computer science)

Searching efficiently

- 🔗 Use boolean operators (**AND, OR, NOT**)
- 🔗 Read abstracts
- 🔗 Check bibliographies for more sources
- 🔗 Note recurring citations

Step 2

Evaluate and select sources

Questions to ask about sources

What question is addressed?

What are the key concepts?

What are the key theories and methods?

What are the results and conclusions?

How does it relate to other studies?

What are the key insights and arguments?

What are the strengths and weaknesses of the research?

Taking notes

- 🔗 Quotes
- 🔗 Summaries of key points
- 🔗 Source information:
 - Author name
 - Title & journal name
 - Year of publication
 - Page numbers

Step 3

Identify themes, debates, and gaps

What to look for

- 🔗 Trends in the literature over time
- 🔗 Key themes
- 🔗 Debates and disagreements
- 🔗 Pivotal publications
- 🔗 Research gaps

Examples of trends and gaps

Most research focused on young women

Increasing interest in the visual aspects of social media

Lack of research on platforms like Instagram and Snapchat

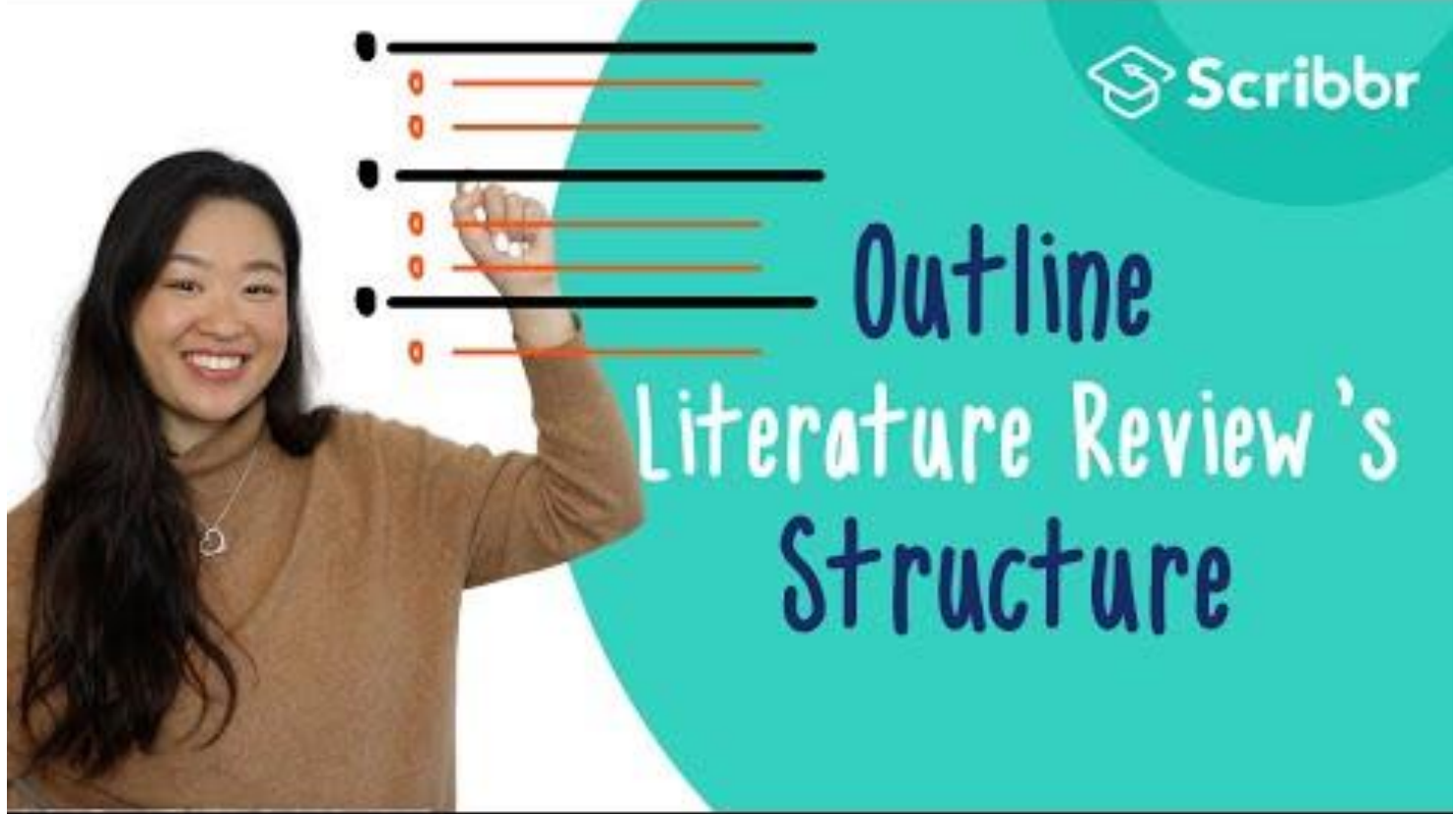
- This is a gap your research could fill


Step 4

Outline your structure

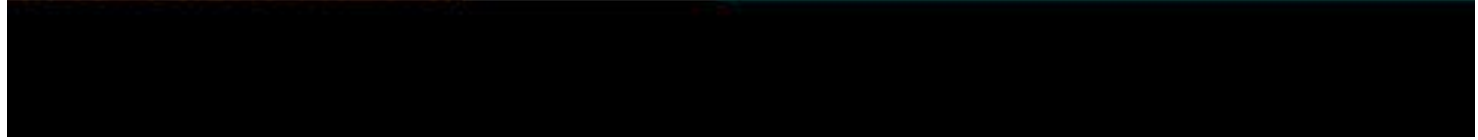
Common structures

- 🔗 **Chronological:** Organize by time
- 🔗 **Thematic:** Organize by theme
- 🔗 **Methodological:** Organize by methodology
- 🔗 **Theoretical:** Organize by theoretical approach



 **Scribbr**

Outline
Literature Review's
Structure



Step 5

Write your literature review

Format of a literature review

1. **Introduction** establishing purpose
2. **Body** analyzing the literature
3. **Conclusion** summarizing key findings

The introduction

Stand-alone literature review:

- 🔗 Provide background on the topic
- 🔗 Describe the objectives of the literature review

Dissertation, thesis, or research paper:

- 🔗 Reiterate the central problem
- 🔗 Briefly summarize the scholarly context

The body

- 🔗 May be divided into sections
- 🔗 Analyze and interpret
- 🔗 Critically evaluate
- 🔗 Synthesize different sources
- 🔗 Use well-structured paragraphs
- 🔗 Cite your sources

Scribbr

Let's Write!

Literature
Review

The conclusion

Stand-alone literature review:

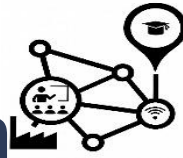
- 🔗 Discuss the overall implications
- 🔗 Make suggestions for future research

Dissertation, thesis, or research paper:

- 🔗 Show how the literature review has informed your approach
- 🔗 State what gaps your research will address

Reference: <https://www.scribbr.com/methodology/literature-review/>

Guidelines for using this presentation



IoTrain
Master of Engineering in Internet of Things

This presentation can be freely used and modified for educational purposes. You may:

- Display this presentation in a classroom environment

- Modify or delete slides

- Distribute this presentation in print or in private student environments (e.g. Moodle, BlackBoard, Google Classroom)

Please do give credit to Scribbr for creating this resource.

Questions or feedback? Email shona@scribbr.com and we'll be in touch!

Section 3. Preparing Scientific Presentations

Outline



- Introduction
- Generic Structure of Scientific Presentations
- Layout & Design Principles
- The Small Things
- Giving a Talk
- Mastering Q & A Sessions

Outline



- Introduction
- Generic Structure of Scientific Presentations
- Layout & Design Principles
- The Small Things
- Giving a Talk
- Mastering Q & A Sessions

What is a Scientific Presentation?

- A scientific presentation is in most cases related to a particular scientific text.
- A scientific presentation informs about the open research problem and the contributions to solve this problems as described in the scientific text.
- A scientific presentation should attract the audience to read the related scientific text.
- A scientific presentation consists of a talk which is usually support by some visuals (slides, white-boards sketches etc.) and a subsequent Q & A session.

What makes a Scientific Presentation challenging?



- Scientific presentations are not lectures: scientific presentations have, in almost all cases, a strict time limit between 20-30 minutes (including Q & A) which makes it impossible to explain everything described in the scientific text in detail.
- The first major challenge is to find an appropriate level of abstraction and to set the right focus on those aspects being most relevant for the presumed target audience.
- The second major challenge is that, in contrast to scientific texts, presentations constitute acts of direct and synchronous communication which only succeed if the presentation, the presenter and the target audience match in some way.

Disclaimer

-
- In contrast to the (mostly generally agreed) guidelines for writing scientific texts, it is difficult to propose similarly general guidelines for preparing scientific presentations.
 - In the following, we will discuss some selected best practices which are, however, not obligatory.

Outline



- Introduction
- **Generic Structure of Scientific Presentations**
- Layout & Design Principles
- The Small Things
- Giving a Talk
- Mastering Q & A Sessions

Generic Structure of Scientific Presentations

- In the following, we discuss some rules for preparing a presentation of a scientific paper at a conference.
- The time slot for this talk is 20 minutes + 10 minutes for Q & A.
- Rule of thumb: A talk of n minutes requires $n/2$ slides.

Generic Structure of a Scientific Presentation

-
- | | |
|------------------------------------|------------------------|
| 1. Title | [1 slide] |
| 2. (Table of Contents) | [0-1 slide] |
| 3. Background / Motivation / Goals | [1-2 slides] |
| 4. Concepts / Contributions | [2-4 slides] |
| 5. Evaluation Results | [1-3 slides] |
| 6. (Related Work) | [0-1 slide] |
| 7. Conclusion / Future Work | [1 slide] |
| 8. Closing Slide | [0-1 slide] |
| | ----- |
| | [\approx 10 slides] |

FastSort - A new Sorting Algorithm based on Divide & Conquer

Prof. John Doe, Prof. Jane Doe (University of Siegen)

2022-02-22

International Conference on
Sorting Algorithms



Title of the talk

List of authors with
affiliation (underline the
one who is giving the
talk)

Date of the talk

Venue

Table of Contents

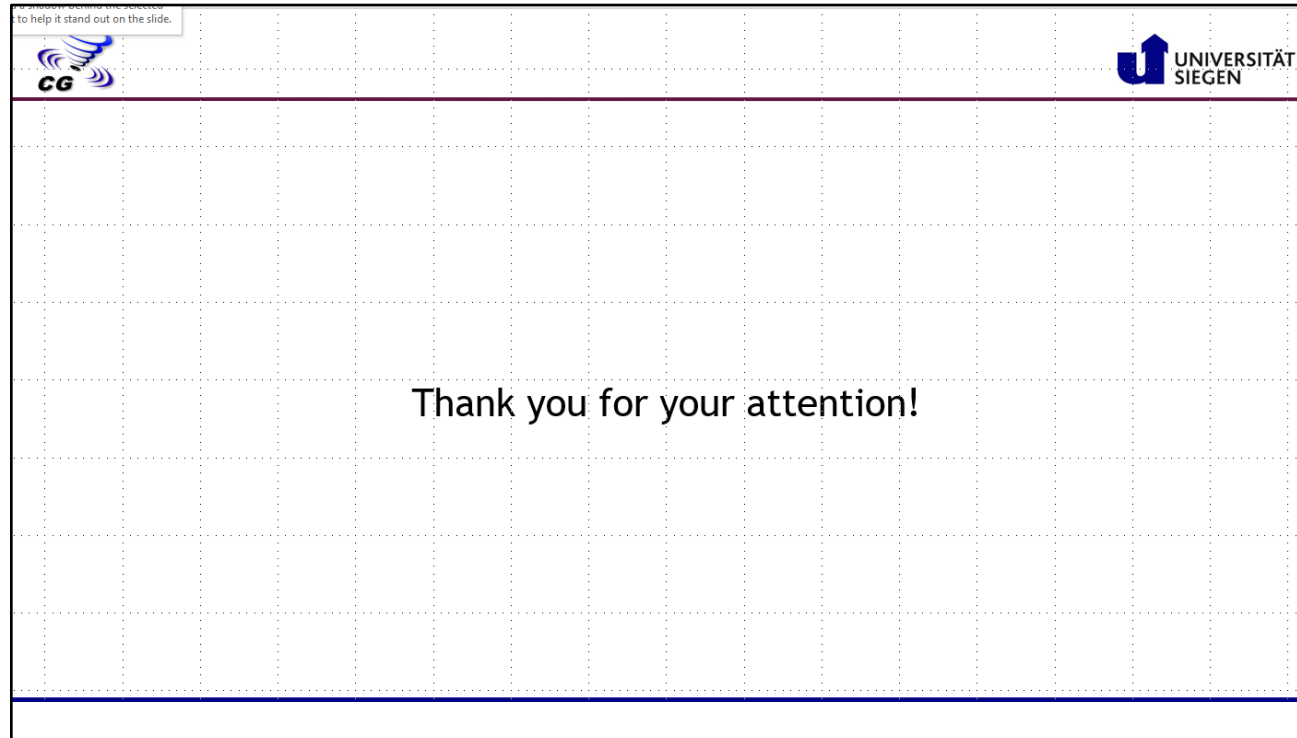
- Mnemonic: A dedicated slide showing the table of contents of the talk is only necessary for talks lasting 45 minutes or longer.

Main Part

-
3. Background / Motivation / Goals [1-2 slides]
 4. Concepts / Contributions [2-4 slides]
 5. Evaluation Results [1-3 slides]

- Overview of and/or extract from the corresponding parts in the scientific text.
- Content, structure and layout depend on the type of contribution, the presenter and the target audience.
- If possible, always use a running example throughout the talk.

Closing Slide



- The only purpose of this slide is to indicate the end of the talk.
- Useful to avoid unpleasant silence after the last word is spoken.

Outline



- Introduction
- Generic Structure of Scientific Presentations
- **Layout & Design Principles**
- The Small Things
- Giving a Talk
- Mastering Q & A Sessions

Guideline 1

1. The slides should serve the presenter, not vice versa.

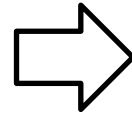
- Mnemonic: If you realize while practicing your talk that you have difficulties to explain your slides, than change the slides, not your explanations.
- Mnemonic: Avoid any impression of „style-over-substance“ as well as „paper-copy-on-slides“.

More on Guideline 1

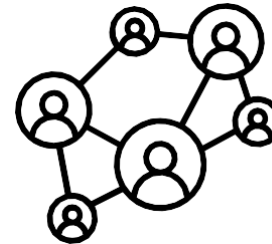
- Example: Explaining graphs

Definition: Graph

A graph \mathcal{G} is a pair (V, E) consisting of a set W of nodes and a set $E \subseteq W \times W$ of edges.



Example: Social Network



- While practicing the talk, you may realize that, while explaining the slide with the definition, you already refer to the upcoming example to explain what nodes and edges may represent.
- In this case, you may flip the ordering of both slides and start with the example.

Guideline 2

2. The audience must be forced to memorize something from your talk.

- Mnemonic: Think about 1-2 key points you definitely want the audience to memorize from your talk and how you can proactively, yet gently, achieve this.

- A common miscomprehension of (enthusiastic) presenters is to assume a „perfect audience“ being interested in everything, constantly listening to the presenter and immediately understanding and memorizing all the details.
- In reality, the amount of content of a presentation actually reaching the audience is (frustratingly) small and diverse.
- To counteract this problem, think about 1-2 highlights you want everyone from the audience to memorize after your talk and focus on these highlights in the preparation of your slides.

More in Guideline 2

- Summarize the essence of your contribution(s) in 1-2 catch phrases or rules of thumbs which are repeated at least two times during the talk.

Examples:

"90% of all programming errors are located in 10% of the source code."

"Programming errors are nearly equally distributed over the source code."

- Open the talk with a mind-blowing claim or provokative question which is resolved/explained during and/or at the end of the talk.

Examples:

"After listening to my talk, you may wish to restart your current programming project from scratch..."

"Are you interested in a simple workaround for the halting problem?"

"Are computer scientists bad programmers?"

Assignment 1

Your task is to prepare a scientific presentation of „your“ new sorting algorithm *QuickSort*.

- Think about 1-2 catch phrases for your talk.
- Think about opening claims or questions for your talk.

Guideline 3

3. The audience fully depends on your slides and what you explain.

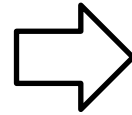
- Mnemonic: Readers of scientific texts are, at least up to a certain degree, self-determined in how they read and perceive the text. In contrast, the way the audience perceives the content of a scientific talk is fully determined by how it is presented.

More on Guideline 3

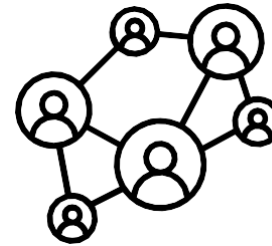
- Example: Explaining graphs

Definition: Graph

A graph \mathcal{G} is a pair (V, E) consisting of a set W of nodes and a set $E \subseteq W \times W$ of edges.



Example: Social Network



- The reader of a paper may first skim (or even skip) the definition and proceeds to the example and then step back and try again to understand the definition...
- In contrast, the audience of a talk depends on the ordering and duration in which the slides are shown and how they are explained.

Guideline 4

4. Different people perceive information in different ways.

- Mnemonic: There is no "golden standard" or "one-size-fits-all" approach for explaining difficult content.

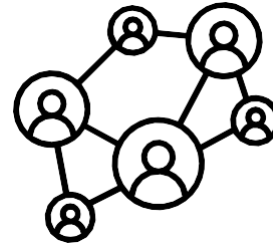
More on Guideline 4

- Example: Explaining graphs

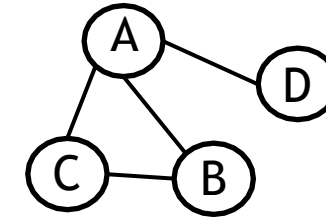
Formal Definition

A graph G is a pair (V, E) consisting of a set V of nodes and a set $E \subseteq V \times V$ of edges.

Concrete Example (Visual)



Abstract Example (Visual)



Textual Definition

A graph is a mathematical structure to specify binary relationships between objects...

Concrete Example (Textual)

A social network consists of a set of members. Two particular members might be connected by friendship relation...

Abstract Example (Formal)

For example, the set of nodes may be given as $V = \{A, B, C, D\}$ and the set of edges...

More on Guideline 4



- Find a balanced mixture of the at least two different ways to explain concepts, preferably combined on the same slide.
- However, everything that is shown on the slides must be explicitly explained during the talk.

Assignment 2

Reflect on the type of audience you are. What kind of presentation do you prefer in which ordering to perceive a new concept?

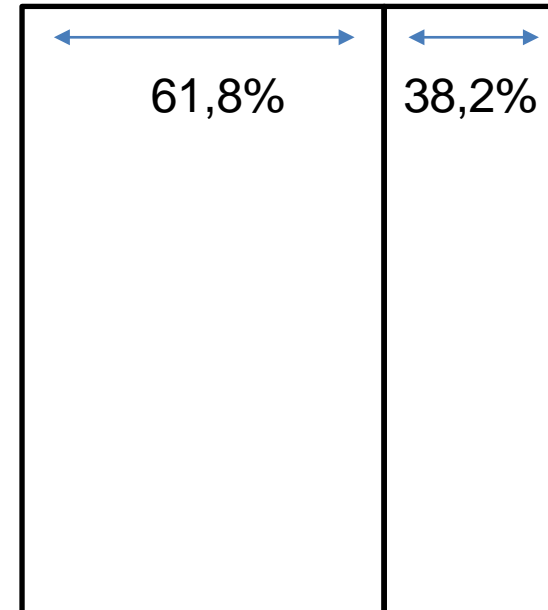
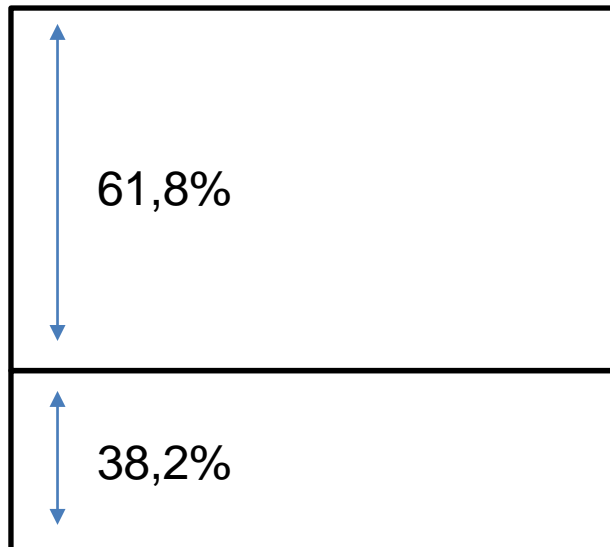
Guideline 5

5. Harmonize your slide layout.

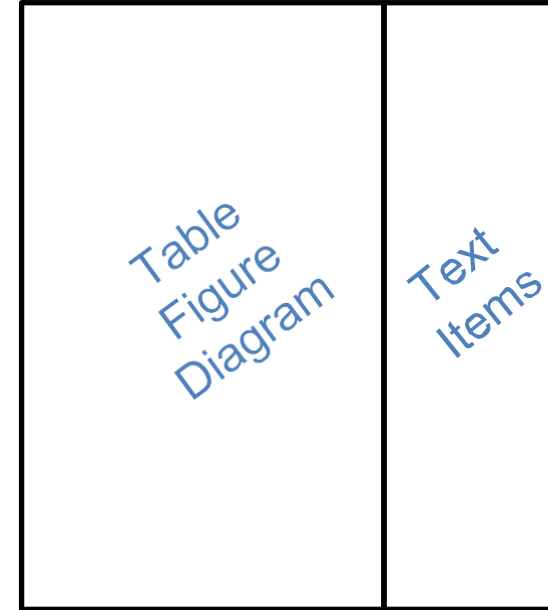
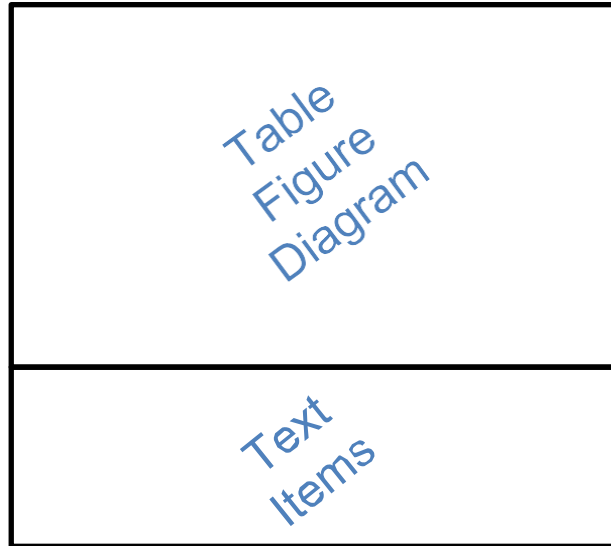
- *Rule of thumb: The layout of the slide should serve the content and presenter, not vice versa.*

More on Guideline 5

- Golden Ratio (Goldener Schnitt)



More on Guideline 5



- Think about a proper composition of the slide content.
- Think about proper transitions between subsequent slides.
- Do not overfill slides: each slide should explain one particular aspect.

More on Guideline 5

Definition: Graph

A graph is a mathematical structure consisting of a set of nodes and a set of edges. Nodes denote objects from the domain of discourse, whereas edges denote some type of relationship holding between certain pairs of nodes ...

wot

vs.

Definition: Graph

A graph consists of:

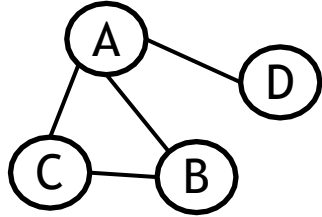
- Set of nodes denoting objects,
- Set of edges denoting relationships between objects
- ...

itemized headline-style

- Avoid „wall-of-text“ (wot) slides.
- Instead: itemize text in „headline-style“.

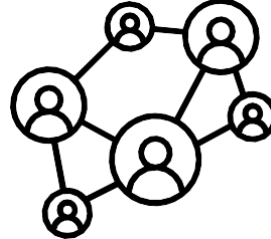
More on Guideline 5

Definition: Graph



For example, the set of nodes may be given as $W = \{A, B, C, D\}$ and the set of edges...

A graph G is a pair (W, E) consisting of a set W of nodes and a set $E \subseteq W \times W$ of edges.





- What is supposed to be the center / anchor of your slide?
- In which ordering do you want the audience to perceive the content of your slide? How to enforce this?

- Take care of a consistent layout of your slides.

Examples:

- Use the same symbol for item-lists on all slides (bullet, dash, ...).
- Emphasize important text in the same way on all slides (bold, underline, ...)
- ...

Outline



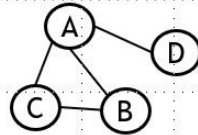
- Introduction
- Generic Structure of Scientific Presentations
- Layout & Design Principles
- **The Small Things**
- Giving a Talk
- Mastering Q & A Sessions

Slide Master

Title of the Slide

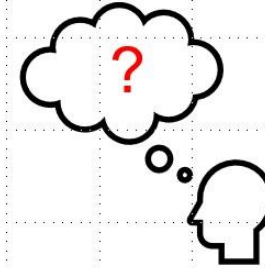
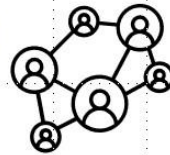
More on Guideline 5

Definition: Graph



A graph G is a pair (V, E) consisting of a set V of nodes and a set $E \subseteq V \times V$ of edges.

For example, the set of nodes may be given as $V = \{A, B, C, D\}$ and the set of edges...



- What is supposed to be the center / anchor of your slide?
- In which ordering do you want the audience to perceive the content of your slide?

35

Number of the slide

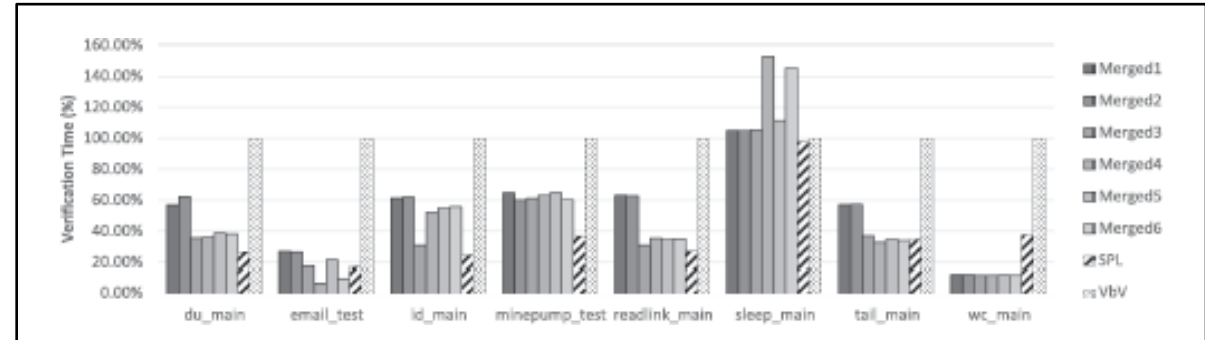
- It is quite common to have a very bare slide master in order to keep the focus of the audience on the actual content and to have enough space for it.
- However, the slide master should at least contain some title and the number of the slide.

- Take care to avoid presumably „minor“ issues like typos, misaligned figures, incoherent fonts etc.
- There are always people in the audience feeling disturbed and distracted by sloppy preparations...

Tables, Diagrams, Formulas...

Table 2. Aggregated Experimental Results for Test-generation of BusyBox

Coverage Criterion	Name	# Test Goals	Avg. # Test Goals per Function	CPU Time (h)	Avg. CPU Time per Function (m)	CPU Time per Goal (s)	Timeouts (%)	Avg. Speedup to VbV (Factor)	# Test Cases	Avg. # Test Cases per Function	# Covered Goals per Test Case	Detected Faults (%)	
C0	VbV	14,753	951.3	7.2	10.2	1.7	2%	1.0	2417	57.5	6.1	98.7%	84.1%
	SPL	3,053	72.7	1.7	2.5	2.1	2%	4.1	551	13.1	5.5	96.6%	84.3%
	Merged1	6,489	154.5	3.0	4.2	1.6	10%	2.4	1240	29.5	5.2	92.0%	76.8%
	Merged2	5,052	120.3	2.8	4.0	2.0	12%	2.5	1060	25.2	4.8	87.1%	74.4%
	Merged3	4,653	110.8	3.0	4.3	2.3	10%	2.4	976	23.2	4.8	87.7%	85.7%
	Merged4	5,014	119.4	3.3	4.7	2.3	14%	2.2	1005	23.9	5.0	84.7%	70.4%
	Merged5	6,875	163.7	3.8	5.4	2.0	10%	1.9	1066	25.4	6.4	80.3%	79.2%
Merged6	7,205	171.5	3.8	5.4	1.9	12%	1.9	1060	25.2	6.8	79.3%	81.7%	
C3	VbV	10,856	258.5	7.5	10.7	2.5	2%	1.0	2664	63.4	4.1	98.8%	84.1%
	SPL	1,832	43.6	1.7	2.5	3.4	2%	4.3	521	12.4	3.5	96.0%	84.3%
	Merged1	4,838	115.2	3.0	4.3	2.2	7%	2.5	1258	30.0	3.8	91.1%	76.8%
	Merged2	3,883	92.5	2.9	4.1	2.7	12%	2.6	1075	25.6	3.6	83.7%	76.8%
	Merged3	3,649	86.9	3.0	4.3	3.0	12%	2.5	970	23.1	3.8	81.6%	83.7%
	Merged4	3,406	81.1	3.3	4.7	3.5	14%	2.3	944	22.5	3.6	78.5%	79.6%
	Merged5	4,462	106.2	3.7	5.3	3.0	14%	2.0	987	23.5	4.5	73.5%	86.8%
Merged6	4,644	110.6	3.8	5.4	2.9	10%	2.0	999	23.8	4.6	72.0%	80.7%	



- If you want to show large tables, diagrams, or complicated formulas, think about how to describe them in a reasonable way during the talk.
- Focus on average values and highlight some distinguished entries, outliers or peaks worth being explained in more detail; refer to the paper for further details.

Citations

Formal Definition [1]

A graph \mathcal{G} is a pair (V, E) consisting of a set V of nodes and a set $E \subseteq V \times V$ of edges.

[1] John Doe: All about Graphs, 1999

Footnote

Formal Definition [1]

A graph \mathcal{G} is a pair (V, E) consisting of a set V of nodes and a set $E \subseteq V \times V$ of edges.

References

- 1 John Doe: All about Graphs, 1999
- 2 ...

Reference slide

- Show references as footnotes on the same slide, or
- List all references on an additional slide at the end of the talk.

Size, Color, Animation, Videos, Live Demos, ...

- Make sure that texts, figures, tables and all other objects have an appropriate size to be easily readable by the audience.
- If fine-grained details are not relevant, do not show them or at least advice the audience to ignore them.
- Be careful with the usage of color as people might be color-blind.
- Animations should only be used for serious purposes such as helping the presenter to explain difficult concepts (e.g., step-wise descriptions of complicated algorithms).
- If you want to play a video or perform a live demo of a software during your talk, make sure that it really works under all possible circumstances without consuming too much time. Anyway, make a backup plan.

Outline



- Introduction
- Generic Structure of Scientific Presentations
- Layout & Design Principles
- The Small Things
- **Giving a Talk**
- Mastering Q & A Sessions

Some Guidelines for giving a Talk

- Practice your talk several times, but do not over-practice.
- The goal of practicing is to roughly know the content of your slides and how subsequent slides can be smoothly connected.
- Avoid to depend on any additional notes during your talk.
- The goal of practicing is *not* to completely memorize every single sentence.
- Be careful: if you over-practice your talk, you may become annoyed by it.
- Another important goal of practicing is to check the time limit.

Some Guidelines for giving a Talk

- One possible exception from the previous guideline: to feel safe, it might help to memorize the opening sentence(s) and closing sentence(s) of your talk.
- Start with a *very short* welcome note, but do not exaggerate. In most cases, someone else (e.g., the session chair) has already introduced you and announced the title of your talk.
- Clarify at the beginning of your talk whether you allow intermediate questions or comments or if you prefer to present without interruptions.

Some Guidelines for giving a Talk



- Keep in mind: Your own impression of your performance while giving a talk is always worse than it is received by the audience.
- Be authentic, find your own style and do not pretend to be someone else. Do not artificially over-tune or under-tune your performance.
 - If you have a down-to-earth and calm personality, do not try to become an entertainer during your talk. Just make sure that your talk is not too monotone.
 - If you have a lively and energetic personality, do not try to cage yourself during your talk. Just make sure that your talk is not too chaotic.

Some Guidelines for giving a Talk

- Remind yourself that the audience is not able to listen to you and to perceive slide contents (text, figures, ...) at the same time, so make enough short breaks.
- If you have too much material for the given time budget
 - ... do not just try to speak faster as this creates unnecessary pressure on you and the audience.
 - ... do not just skip slides as this makes a bad impression.
- Instead, it is better to drop some content from the talk to have a buffer of 1-2 minutes.

Emphasis = Variation

- A vital impression can be achieved by controlled variation and contrasts, not just by "bigger, faster, more...".
- You may emphasize something during your talk by varying the volume and speed of your voice as well as by your body language.
- Very important highlights can be further emphasized by making short breaks during your talk.
- However, the more things you emphasize, the less effective it will be.

Some Guidelines for giving a Talk

- Avoid any kind of meta-comments or apologies.
- Counterexamples:
 - Before the talk: "I first of all would like to apologize for the weak slides / that I feel slightly sick today / that the topic is less interesting than the previous talk", ...
 - During the talk: *"Oops, sorry for the typo on the slide"*, *"Sorry for the confusing explanation"*, ...
 - After the talk: *"Sorry for exceeding the time limit"*, ...

Some Guidelines for giving a Talk

- Interactions with the audience (polls, open questions, break-out groups etc.) may break monotonicity of otherwise purely frontal presentations.
- But: Make sure, that the intention and goal of the interaction is clear and does not lead to confusion or annoying silence.
- And: Be careful that enforced interactions do not offend people.

Some Guidelines for giving a Talk

- Be aware of your body language:
 - What to do with your hands?
 - Keep eye contact with the audience,
 - Control your breath,
 - ...
- Before the talk, make yourself familiar with the technical equipment (projector, microphone, ...)
- Use pointing devices and additional media (white-board etc.) with care and with a clear purpose.

Outline



- Introduction
- Generic Structure of Scientific Presentations
- Layout & Design Principles
- The Small Things
- Giving a Talk
- **Mastering Q & A Sessions**

Roles



- Presenter (Speaker): gives the talk and answers questions in the Q & A session (should be one single person).
- Audience: listens to the talk and asks questions in the Q & A session.
- Session chair: starts/closes the session, serves as a host, helps with technical equipment, announces the talks of the session, enforces the time limits and moderates the Q & A session.
- Discussants (optional): designated member(s) of the audiences preparing questions for a particular talk.
- Co-authors (optional): may help the presenter to answer questions.

Some Guidelines for Mastering Q & A Sessions

- Before giving an answer, shortly say "thank you" for the question and repeat the question in your own words.
- Give a concise and clear answer, do not repeat yourself or digress and always be polite.
- Think about appropriate phrases to mark the ending of answers.
- If you anticipate obvious questions, you may prepare backup slides with additional information helping you to answer those questions.
- Remind yourself that most questions are neither meant to be overly critical nor provokative, but simply ask for some clarification or express increased interest in what you presented.

Some Guidelines for Mastering Q & A Sessions

- A frequent type of question asks for a clarification of some statement, definition, example, figure etc.
- Do not just repeat your explanation from the talk, but instead try to understand (or ask back) *what* is unclear and provide an *alternative* way of explanation, an additional example, a hand-drawn figure etc.

- It frequently happens that the presenter does not understand a question or does not know a good answer.
- Do not try to just sneak out of this situation (e.g., by giving an answer that deviates from the question), but instead be honest which is not a shame!
- Examples:

"I am sorry, I do not understand your question, would you mind to rephrase it?"

"This is a very good question. I need to think about this issue in more depth. Maybe you can tell me more about this point during the coffee break."

Thank you for your attention!

Extra slides ONLY for lecturer

1. General Concepts of Science

1.2 Good Scientific Practice



https://static.uni-graz.at/fileadmin/Docservice/Bilder_820x340/csm_right-wrong__2__92e413bdb7.jpg

Overview

- Good scientific practice and research integrity
- Scientific misconduct, e.g., plagiarism
- Fake Science
- Counteracting scientific misconduct
- Research Ethics

Foundation & Base Notation

- German Constitution Art. 5 (3): “Art and science, research and education are free”

The scientific communities and organizations are self-responsible to set up proper rules and processes

- Scientific misconduct^[1], e.g., the Danish definition:
 - “Intention or gross negligence leading to fabrication of the scientific message or a false credit or emphasis given to a scientist”
- Research Integrity / Scientific Integrity^[2]
 - Deals with “best practices” or rules of professional practice of researchers to fight against scientific misconduct
- Research Ethics^[3]
 - Relates to moral issues arising during or as a result of research activities, as well as the ethical conduct of researchers.
- Fake Science
 - Fraud and falsification in science due to untrue claims and fabricated or falsified research results published with fraudulent intent.

1 https://en.wikipedia.org/wiki/Scientific_misconduct

2 https://en.wikipedia.org/wiki/Scientific_integrity

3 https://en.wikipedia.org/wiki/Research#Research_ethics

4 Andreas Hensel: Fake in der Wissenschaft? <http://bit.ly/2QQUWws>

Code of Conduct

- Code of Conduct^[1] published by the German Research Foundation (DFG) aims a
 - Fostering research integrity and establish it as an integral part of research and teaching
 - Providing a framework for safeguarding public confidence in research
 - We will focus on study specific aspects
- Background is a DFG document on scientific misconduct^[2]

The DFG documents address research process and structure and also apply to scientific Master programs at Universities!

[1] DFG: Code of Conduct - Guidelines for Safeguarding Good Research Practice, <https://doi.org/10.5281/zenodo.3923602>

[2] DFG: Safeguarding Good Scientific Practice, http://www.dfg.de/formulare/80_01/80_01_en.pdf

Scientific Misconduct

- There are three basic types of scientific misconduct (intentionally or with gross negligence)^[1]
 1. Misrepresentations
 2. Claim others' research achievements as one's own
 3. Interfere with others' research
- Misrepresentation is given if, for instance,
 - Data (incl. images, graphs etc.) and/or research findings are fabricated or falsified, e.g.
 - by suppressing and/or eliminating data and/or results obtained in the research process without disclosing this,
 - by manipulating a representation or illustration/figure in Photoshop
 - Image and corresponding statements are incongruous, e.g.
 - False assignment if measurements to an experiment
 - Making inaccurate research-related statements, e.g.
 - Pretend to properly adhere to a standard protocol/process of data acquisition



https://undsci.berkeley.edu/article/socialsideofscience_06

[1] DFG: Rules of Procedure for Dealing with Scientific Misconduct, http://www.dfg.de/formulare/80_01/80_01_en.pdf

Scientific Misconduct

- Claim others' research achievements as one's own, e.g.
 - Plagiarism, i.e., using others' content without indicating the source properly (see below)
 - Idea theft, i.e., using, sharing or publishing others' research approaches and ideas (prior to publication) w/o authorization, e.g.
 - Disclose your supervisor's concept to a company or file a patent
 - Claiming or pretending authorship or co-authorship w/o a genuine, identifiable contribution, e.g.
 - Letting someone else write your thesis or term paper
 - Falsification of research content generated by others, e.g.
 - Claiming an unreported error, shortcoming or similar w/o evidence

Scientific Misconduct

- Interference with others' research, in particular
 - Sabotage research activities, e.g.
 - Damaging, destroying or manipulating experimental setups, instrumentation, documentation, hardware, software, etc.
 - Modifying, falsifying or removing research data or documents without authorization



<https://www.business2community.com/wp-content/uploads/2020/02/Sabotage.jpg>

Quotation, Paraphrase & Plagiarism

- Plagiarism: Missing or improper reference to work of others
 - The most prevalent form of misconduct at student level
- Quotation: 1-to-1 copy of excerpts from another text
- Paraphrase: A modified (also translated) takeover of excerpts from another text w/o distortion of the meaning
- Types of plagiarism (copy w/o quotation/reference)
 - Total plagiarism: Take over completely or partially paraphrased text
 - Partial plagiarism: Take over parts of a text
 - Idea plagiarism: Take over idea w/o quoting their authors
 - Auto (self-) plagiarism: Copy one own's prior and original work

!! Scientific misconduct !!

OK!!

https://en.wikipedia.org/wiki/Plagiarism#Forms_of_academic_plagiarism

<https://de.wikipedia.org/wiki/Plagiat>

[https://web.archive.org/web/20051028114159/http://www.uni-bielefeld.de/\(de\)/philosophie/personen/beckermann/Zitieren.pdf](https://web.archive.org/web/20051028114159/http://www.uni-bielefeld.de/(de)/philosophie/personen/beckermann/Zitieren.pdf)

- Paraphrasing ranges between two extremes
 - Simply replacing some words, e.g.,
 - “A linked list is a **linear** collection of data **elements** whose order is not **given** by their physical placement in memory, but is established by pointers.”
 - “A linked list is a **sequential** collection of data **items** whose order is not **defined** by their physical placement in memory, but is established by pointers.”
 - Writing it in own words, i.e., explaining the idea anew, e.g.,
 - “A linked list stores data items in a non-sequential order in memory, while allowing linear access through pointers.”
- In Computer Science
 - Quotations are rarely used
 - Simple paraphrasing is regarded as plagiarism, even if a citation is given (as you can’t use quotes)
 - Writing in own words and adding a citation is the correct thing to do

Students Commit

1. Submitting someone's work as their own.
2. Taking passages from their own previous work without adding citations (self-plagiarism) → OK, if initial text is original from you!
3. Re-writing/paraphrase someone's work without properly citing sources.
4. Using quotations but not citing the source → minor error if citation is given in the context
5. Interweaving various sources together in the work without citing.
6. Citing some, but not all, passages that should be cited.
7. Melding together cited and uncited sections of the piece.
8. Inaccurately citing a source → minor error if source is identifiable



<https://matthewmorris.medium.com/when-students-plagiarize-f8b5680bc750>

<http://go.turnitin.com/paper/plagiarism-spectrum>

Assignment 1

Check out the first paragraph of the definition of a binary tree in Wikipedia

(https://en.wikipedia.org/wiki/Binary_tree) and write

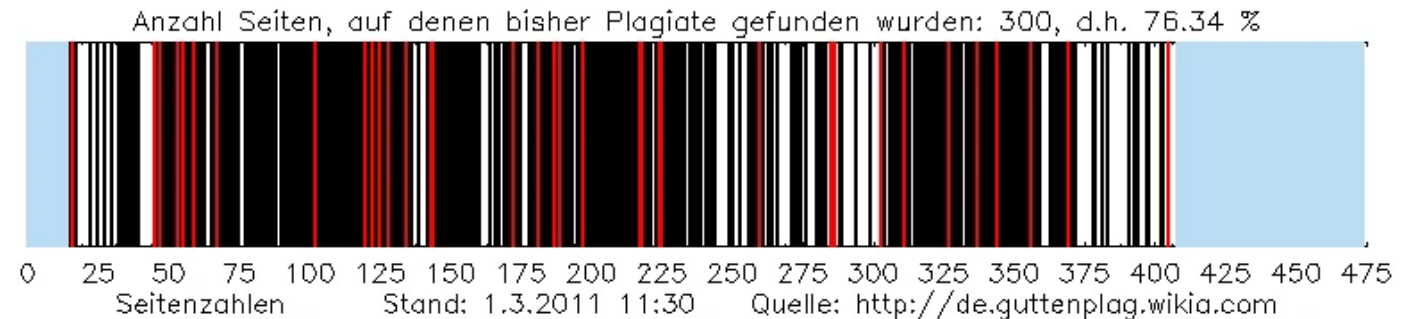
- a) A paraphrased version
- b) A version with your own words

Discuss the difference!

Prominent Examples for Scientific

Misconduct

- Many negative examples in recent years provoked a discussion on good scientific practice
- Examples:
 - Plagiarism, e.g. denied PhD degrees after evidence of severe violations of proper citation and quoting (former federal ministers Guttenberg and Schavan)



- Jan Hendrik Schön, Nano-Physicist: Published faked data about electronics of organic structures. He was regarded as upcoming Nobel Prize Winner!
- Hwang Woo-suk, Korean stem cell researcher published in Science spectacular results on the cultivation of eleven cloned human stem cell cultures, which where completely faked.

Ethics

- How can intentionally wrong results get published?
- Comprehensive discussion on pseudo-scientific publishers
 - Predatory journals and predatory publishers
 - In particular: Disregard for basic rules of scientific quality assurance
- Why do fake science journals exist at all?
 - Publisher: It is a simple business model or even financial fraud, i.e., authors pay for nothing
 - Researcher have various motivation, e.g.,
 - Publish fast w/o reviewing, i.e., being questioned on “nasty details” (see also Reviewing)
 - Get a seal of quality for a politically or otherwise inspired opinion w/o scientific evidence, e.g., „there is no anthropogenic climate change“



https://www.btr-akademie.de/media/catalog/product/w/e/webshopbild_stakeholder_2018.jpg

Ethics

- Challenges in setting up publication infra structure
 - Only science can guarantee publication quality by reviewing
 - Science is state-independent^[1]

Concepts and rules are to be developed and controlled by the scientific communities, publishers and professional organizations

- COPE (Committee on Publication Ethics)^[2]:
 - Founded in 1997 by a group of editors
 - Goal: make “ethical practices become a normal part of the publishing culture”
 - Approach “(...) influencing through education, resources and support of our members, alongside the fostering of professional debate in the wider community.”

1 see German Constitution Art. 5 (3)

2 <https://publicationethics.org/node/45216>

Ethics

- Main indicators of a predatory journal / publisher^[1,2]:
 - Journal's website contains misleading or false information, e.g.,
 - Indexing by Thompson Reuters with comparably high Impact Factor (IF) metrics
 - No ISSN or using one already assigned to another publication
 - Journal's name is the same as or easily confused with that of another journal or association
 - Extremely wide scope (“Journal of Business, Humanities and Technology”)
 - Wrong reflection of origin (“Canadian/Swiss Journal ...” w/o being located in Canada/Switzerland)
 - Peer review process is not explained, e.g.,
 - Manuscript acceptance or a very short peer review time is guaranteed.
 - Submitted manuscripts receive inadequate or no peer review.
 - Editorial board: Missing, misleading, false, or inappropriate for the journal, e.g.,
 - Editors' full names and affiliations missing
 - Editor-in-chief is also the owner/publisher
 - Editors and/or editor-in-chief are not qualified for guaranteeing quality in the journal's field

1 Predatory publishing, <https://publicationethics.org/node/45216>

2 Criteria for Determining Predatory Open-Access Publishers, <http://beallslist.net/wp-content/uploads/2019/12/criteria-2015.pdf>

Ethics

- Further indicators of a predatory journal / publisher
 - Process for identification of and dealing with allegations of research misconduct is not explained
 - Reviewing quality criteria are not described, e.g.,
 - Originality, proper exposition of state-of-the-art, ... (→ you will learn about this)
 - Publishing criteria are not described, e.g.,
 - Data sharing and reproducibility, intellectual property, conflicts of interest, handling of corrections/retractions
 - Access to and archiving of publications, e.g.,
 - No information on how papers are available to readers or on fees for libraries
 - No electronic backup and preservation of access to journal content
- There are lists of (potentially) predatory journals and publishers, such as <https://beallist.net/>

Check the following journal website and check if some of the indicators for predatory journal / publisher apply.

<http://www.borpub.com/index.php>

Dealing with Scientific Misconduct

- DFG guidelines to safeguard good scientific practice^[1]
 - 19 guideline for scientific organization
- Some of the recommendations relate to Master students as well
 - “Higher education institutions (HEI) (...) work together to define rules of good research practice, ensure that their employees are made aware of these guidelines (...), and require their employees to comply with them.”
 - “Education in the principles of good research begins at the earliest possible stage in academic teaching and research training.”
 - “The heads of HEIs (...) create the basic framework for research. They are responsible for ensuring adherence to and the promotion of good practice, and for appropriate career support for all researchers.”
 - “Researchers take into account and acknowledge the current state of research when planning a project. To identify relevant and suitable research questions, they familiarize themselves with existing research in the public domain.”

[1] DFG: Code of Conduct - Guidelines for Safeguarding Good Research Practice, <https://doi.org/10.5281/zenodo.3923602>

Level

- Further aspects to safeguard good scientific practice^[1]
 - Make research quality assurance mechanisms publicly available, e.g.
 - Equipment calibration, data selection, processing & analysis
 - Make research data and processes publicly available to allow others to reproduce and compare results, e.g.
 - Data collections acquired
 - Data analysis tools and software
 - Source code (or executables) for developed methods and algorithms
- European “Human Resources Strategy for Researchers” (HRS4R)^[2]
 - U Siegen HRS4R action plan including “Rules of good scientific practice and ethical issues” and “Working conditions”



<https://www.graduiertenakademie.uni-hannover.de/>

1 DFG: Code of Conduct - Guidelines for Safeguarding Good Research Practice, <https://doi.org/10.5281/zenodo.3923602>

2 https://www.uni-siegen.de/start/die_universitaet/ueber_uns/zertifikate/hrs4r/

Organizational Aspects – Master

Student Level

- Prior examination regulations, §39 (1) Uniform Regulations:
 - *“When handing in the paper for a seminar, a term paper, a research seminar or a thesis paper, the candidate must certify in writing that he or she has written his or her paper (...) independently and that he or she has not used any sources or aids other than those indicated and has marked citations.”*
- Current examination regulations, §18 (5) General Master Regulations:
 - *“If the candidate attempts to influence the result of his/her study or examination performance by deception, e.g., the use or carrying of unauthorized aids or the submission of plagiarism, the study or examination performance in question shall be deemed to have been graded as unsatisfactory.”*
 - That is: Students must know the concept of good scientific practice by heart

**Scientific misconduct is no minor matter, also not for students.
It leads to loss of reputation of the institute, faculty and university
At the end, good scientific practice is a matter of attitude not of control!**

Organizational Aspects – Master

Student Level

- Prior examination regulations, §39 (1) Uniform Regulations:
 - *“When handing in the paper for a seminar, a term paper, a research seminar or a thesis paper, the candidate must certify in writing that he or she has written his or her paper (...) independently and that he or she has not used any sources or aids other than those indicated and has marked citations.”*
- Current examination regulations, §18 (5) General Master Regulations:
 - *“If the candidate attempts to influence the result of his/her study or examination performance by deception, e.g., the use or carrying of unauthorized aids or the submission of plagiarism, the study or examination performance in question shall be deemed to have been graded as unsatisfactory.”*
 - That is: Students must know the concept of good scientific practice by heart

**Scientific misconduct is no minor matter, also not for students.
It leads to loss of reputation of the institute, faculty and university
At the end, good scientific practice is a matter of attitude not of control!**

Research Ethics

- Research ethics is motivated by revelation of scandals such as
 - Nazi human experimentation, e.g., freezing experiments on concentration camp prisoners
 - Tuskegee experiment with 400 Afro-Americans with untreated syphilis in the US 1932-1972
- Goal: Clear measures for the ethical governance of research
 - Ensure that people, animals and environments are not unduly harmed in research
- Ethical issues may arise in the design and implementation of research involving
 - Human and animal experimentation
 - Environmental consequences relevant for, e.g., future generations
- Basic principles of the Declaration of Helsinki
 - Respect for the individual, his right to self-determination and the right to make informed decisions regarding participation in research
 - For incompetent, physically or mentally incapable participants allowance must be given by an individual acting in the subject's best interest
 - The subject's welfare must always take precedence over the interests of science & society
 - Ethical considerations must always take precedence over laws and regulations

Research projects (also theses) potentially raising ethical issues require a permission from an ethics commission of the University

[1] https://en.wikipedia.org/wiki/Research#Research_ethics

Thank you for your attention!

Scientific Working

Malte Lochau, Model based Development
Jöran Beel, Intelligent Systems Group
Andreas Kolb, Computer Graphics & Multimedia Systems Group

2. Scientific Resources

Types of Scientific Papers (and Documents)

General Remarks

- Scientists publish scientific documents to primarily
 - present their research results to the scientific community/discipline
 - claim their results to be their original contribution/innovation
 - stimulate other researchers to professional discussions and further investigations
 - distinguish themselves in their field and gain reputation (e.g., to be eligible for getting research funds)
- Secondary motivations
 - Present and advertise themselves and their ideas to a general public
 - Receive money for this publication (mainly for textbooks)

General Concepts

- Publication channels:
 - How is the paper published?
- Paper categories:
 - What does the paper contain?
- Conditions and requirements of a publication mainly depend on the channel:
 - Underlying publication infrastructure
 - Time-line of preparation and publication
 - Reviewing process
 - Importance and reputation

Types of Publication Channels

- Main types of publication channels:
 - Journal
 - Conference, workshop & symposium
 - Book chapter
 - Book / monography
 - Technical report
 - Thesis (also seminar papers)



<https://compote.slate.com/images/ea8e3254-563d-4095-967e-fa1599366126.gif>

/Publisher

- A publisher provides infrastructure for
 - paper submission (usually a website)
 - organization of the reviewing process to assure scientific quality
 - distribution of (accepted) papers
- Publishing companies: Some prominent examples
 - Springer, e.g., “Nature”
 - Elsevier
 - Taylor & Francis
- Scientific organizations, e.g.
 - American Association for the Advancement of Science (AAAS), e.g., “Science”
 - Institute of Electrical and Electronics Engineers (IEEE)
 - Association for Computing Machinery (ACM)
- Sole document platform with no or an external reviewing process, e.g.
 - arXiv (arxiv.org)
 - University libraries



Journal

- Journals are issued regularly (bi-weekly to quarterly or bi-yearly)
- Underlying infrastructure
 - Publisher: Company (like Springer) or scientific organization (like IEEE)
- Reviewing process handled by Editorial Board:
 - Editor-in-Chief: Long-term committed and experienced researcher
 - Board members (Area-Editors) supported
- Time-line of preparation and publication
 - No fix submission time
 - Full, at-least two-stage review process
 - long time between submission & publication

Conference, Workshop, Symposium

- Conference, Workshop, Symposium: Regular (annual) meeting of domain scientists
- Rule-of-thumb definition
 - Conference: Large events with up to several 1000 participants
 - Most prestigious meetings
 - Presentation of high quality and innovative results
 - Workshop: A smaller event < 100 people
 - Presentation of intermediate results
 - More discussion and open exchange of opinions
 - Symposium: Medium event
 - Optionally: An additional open expert discussion



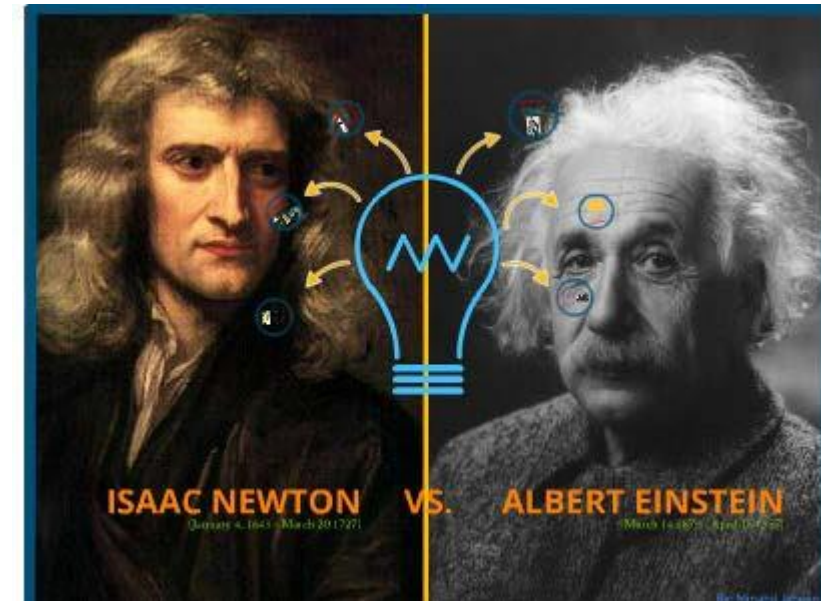
SIGGRAPH Conference 2019, Los Angeles
https://www.pdvq.it/wp-content/uploads/2019/08/ET-Audience-2_Jim.jpg

- Underlying infrastructure
 - Conferences are organized by scientific organizations or groups of scientists
 - Infrastructure: Established from the organization or from commercial publisher
 - Example: IEEE and ACM have their own, Springer offers their “Lecture Notes” series
- Accepted papers are published in proceedings or special journal issues
- Time-line of preparation and publication
 - Strict time-line with strict dates for
 - Submission deadline
 - Deadline for reviews, decision and author notification
 - Publication

Journals vs.

Conferences/Workshops/Symposia

- Importance and reputation
 - Rule of Thumb: Journals are regarded as the major publication channel, as they (can) provide a rigorous (unlimited) reviewing process
 - In some disciplines such as Computer Science, some conferences have very high reputation
 - Some conference proceedings are published as journal issues
 - But: At the end reputation counts!
 - There are strong differences between journals (e.g., fake journals)
 - Some conferences have no (real) reviewing and publication, just presentations
- Quality Measures & Rankings



https://0701.static.prezi.com/preview/v2/kh7w7q4y3fmk2frpz5fz4ml2fp6jc3sachvcdoazecfr3dnitcq_3_0.png

Book- Chapters

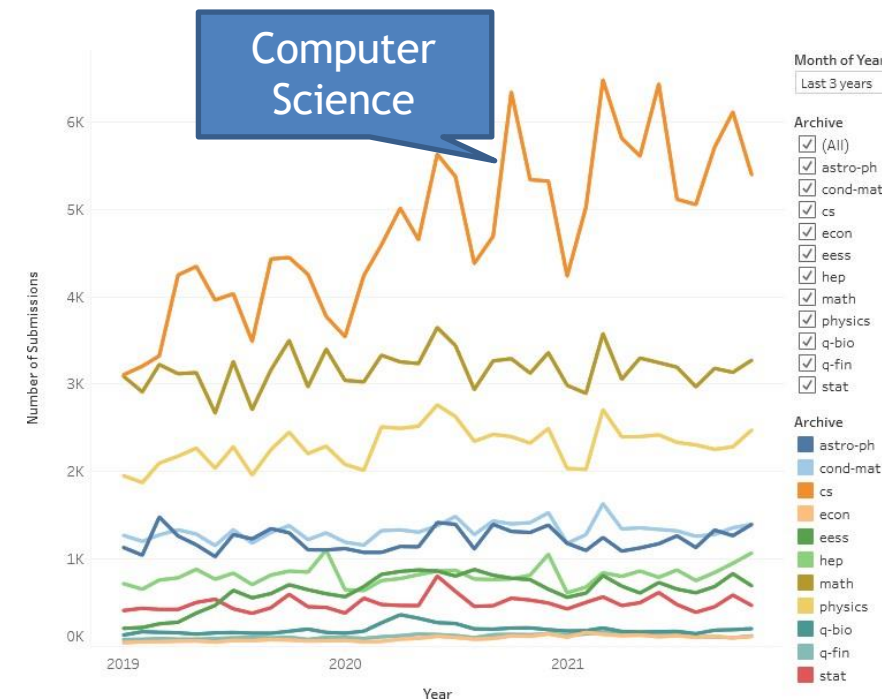
- Monographs are books written by one or several author(s) (see next)
- Here, the “book” is proposed & edited by some experts (editors)
 - Individual chapters are written by different authors
 - The editors may pre-select the authors for the chapter
- Books are triggered by specific occasion, e.g.
 - Sometimes related to a specific meeting, workshop or symposium
 - Often initiated by some scientists having a specific topic and audience in mind
- Similar publication procedure: Mainly driven by editor & publisher
 - Authors of the chapters and editors serve as reviewers for other chapters
- Time-line of preparation and publication: This is very individual and specifically tailored

Book / Monograph

- Books are wholly written by one or author or several co-authors
- Monographs are particularly important in the cultural, linguistic, and social sciences
- Main purpose of monographs
 - Summarize/structure complex subject area, condensing a large amount of literature
 - Often, books remain relevant in the scientific discussion for a long time
 - Commonly, books are regarded as vital for an academic career development
- No standard review process, but different quality-control mechanisms, e.g.,
 - The acquisitions editor invites a suitably qualified author to write a monograph, i.e., the selection of the author is part of quality control.
 - Readers appointed by the publisher act as reviewers and the development editor revises the manuscript together with the author.
 - After publication, editors may invite qualified experts to review a monograph and publish those reviews.
- Time-line of preparation and publication: This is very individual and specifically tailored

Technical Report / White Paper

- Technical Reports (TR): “Write something and put it on your own website”
 - What is a TR good for?
 - Your paper has been rejected too often, but you feel it should be accessible to other researchers
 - You have submitted a paper to a Journal
 - “claim” the field/result during reviewing process
 - Applicable for non-double-blind reviewing only!
 - Some important results are published as TR only
 - Very prominent platform for TR: arxiv.org



Monthly arXiv submissions by category

Doctoral Thesis / Dissertation

- Document in support of achieving a doctoral degree
- Quality control through a committee
 - At least two reviewers
 - Defense of the thesis and examination
- Content is usually more focused as a monography
 - Related to an initial research question
 - Structure similar to an article in a, e.g., journal or conference, but more detailed and longer
- Publication: Doctoral theses must be published (in Germany)
 - Usually handled by the University library

Discuss the following questions:

- a) What are (potential) reasons why conference / workshop / symposium proceedings are popular in Computer Science?
- b) What are (potential) reasons why monographs are not popular in Computer Science?

Paper Categories

Main Paper Categories

- Categories are not standardized and vary between disciplines
- Research Article (also technical papers): Complete description of current original research findings; >8 pages.
- Letter (also communications): Short descriptions of important current research findings with initial results, usually fast-tracked publication; ~4 pages.
- Short paper: Short descriptions of a smaller result at a conference less important than letters; 4-8 pages.
- Review article/survey: Accumulate the research results in a specific topic based on a large body of articles into a coherent narrative about the state of the art in that field.

Further Paper Categories

- Application Paper / Systems Paper: Presents the concept and compilation of an application or system intended to solve a given problem (“the contribution is to make it work”)
- Case Studies / User Study / Experience Report: Presents an empirical research study that aims to investigate an object of study in its real-world environment, potentially using questionnaires or online surveys
- Comparative Study: Compares several existing solutions using a benchmark
- Prototype / Resource / Benchmark Paper: Present novel software, benchmarks, or datasets related to a specific problem of interest to the research community
- Position Paper: Presents an (arguable) opinion of the author(s) or and institution about a current problem or issue
- Concept Paper: Often related to research projects, sometimes at the beginning of a project or prior to submission of a full proposal

Peer Review Publication Process

Roles in Publication Process

- **Publisher:** A company or professional organization, such as ACM, IEEE, that provides means for publishing papers
 - Publication channel (mainly online and digital only)
 - Infrastructure for reviewing and publication
- **Editor:** Manages the reviewing process and guarantees the quality of accepted papers
 - Selects reviewer, makes final decision
- **Reviewer:** Reads a submitted paper and suggests improvements and a decision
- **Author:** Writes the paper and its revisions, which is almost always required

Scientist

Scientist

Scientist

Peer Review Process

- Peer review process ensures quality of scientific publications
- The multi-stage process is time intensive
 - 2-5 reviews are requested
 - Peer reviewer evaluate
 - Innovation
 - Proper presentation and discussion of the state-of-the-art
 - Correctness of the presentation of the own method
 - Expressiveness of the experiments and comparison with other methods
- The process may be iterative.



<https://authorservices.wiley.com/Reviewers/journal-reviewers/what-is-peer-review/the-peer-review-process.html>

Reviewing in General

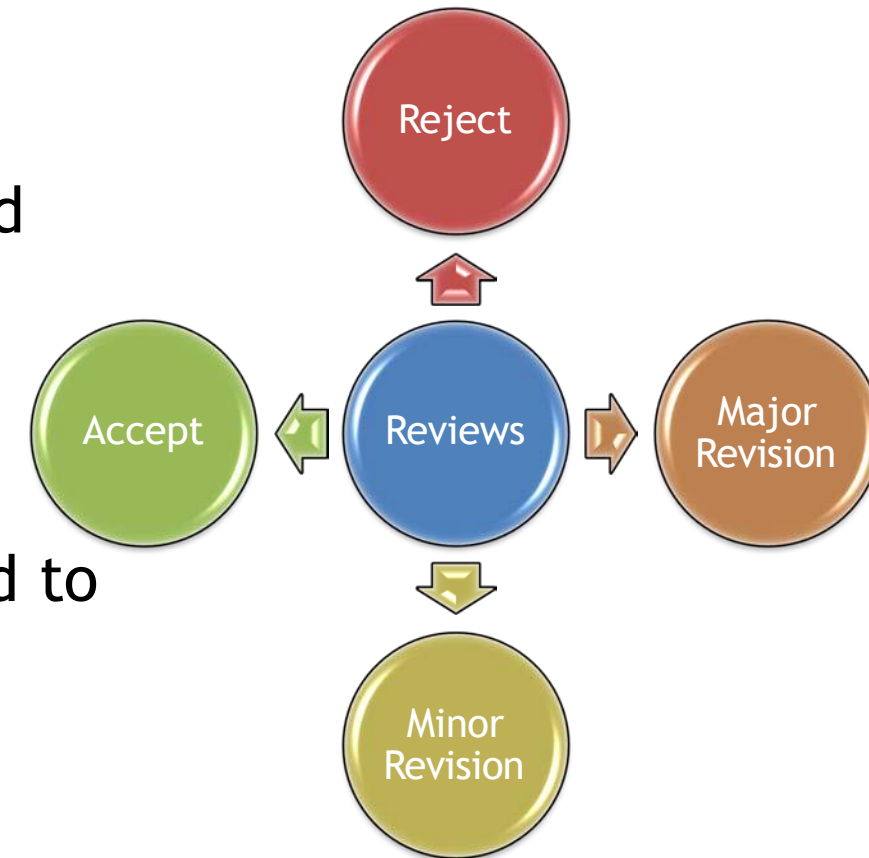
- Goals of reviewing:
 - Filtering “good” from “bad” papers
 - Improving “acceptable” papers to get better
 - Reviewing must be as objective as possible
- Objectivity
 - Single-blind: Author doesn't know reviewer (standard)
 - Double-blind: reviewing and author do not know each other (only partially established)
- Types of reviewing
 - Abstract only (rarely done in computer science)
 - Full paper

Reviewing in General

- Reviewing is channel- and quality-sensitive
- General rules of thumb:
 - The higher the reputation of the channel (and the publication), the stricter the reviewing.
 - The more reviewers, the higher the reputation
 - The stricter the reviewing, the longer the publication process (at least for Journals!)
- Acceptance Ratio
 - “# accepted paper” / “# submitted papers”
 - Rule of thumb for conferences: A “real” conference has an acceptance ratio of at max. 50-60%, i.e., half of the papers require improvements that can't be provided in the given time-frame

Journal Paper Reviewing

- Infra-Structure
 - Publisher provides technical basis (Web-based)
- Reviewers:
 - Ad hoc selection by Area-Editor due to required expertise
- Major decision types:
 - Accept “as is”: Very rarely for 1. revision
 - Minor Revision: Only marginal changes required to improve the paper
 - Major Revision: Paper needs fundamental rewriting
 - Reject: Paper is not acceptable

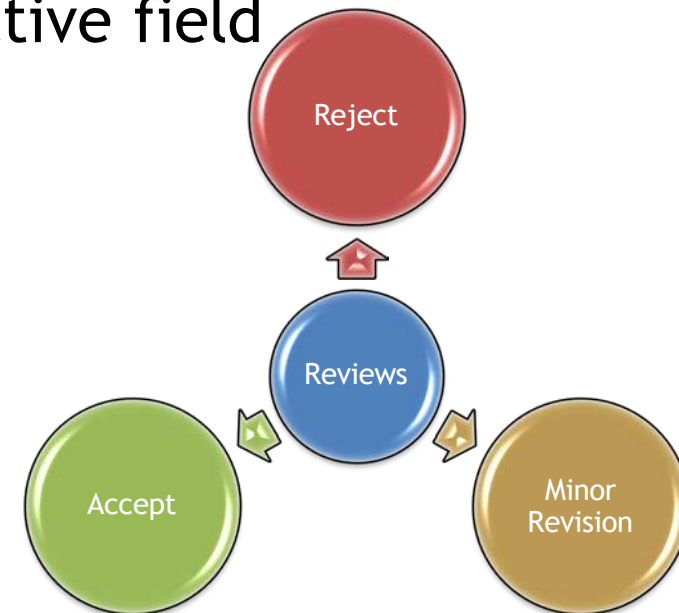


Journal Paper Reviewing

- Submission and review process:
 1. Paper submission
 2. Editor-in-Chief passes paper to Area Editor
 3. Area-Editor (single-blind) selects reviewers & passes paper on to reviewers
 4. Reviewer (single- or double-blind) prepare review
 5. Area-Editor compiles decision
- Review cycles: New revision for review result “Minor Revision” or “Major Revision” are required (or paper is withdrawn)
 - Minor Rev.: Area Editor reviews changes
 - Major Rev.: Completely new cycle starting with 3)

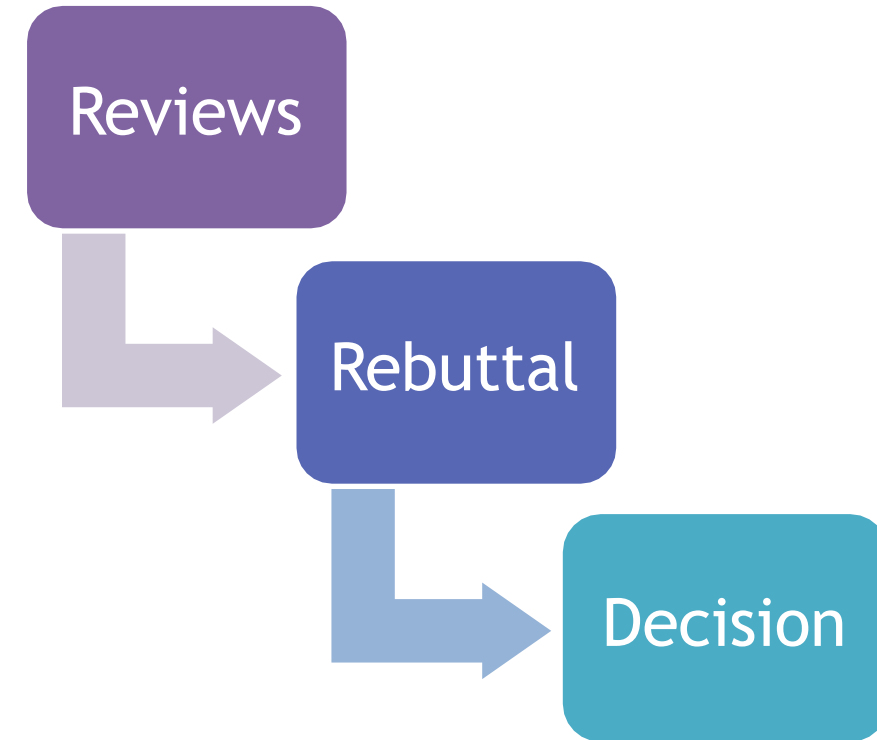
Conferences Paper Reviewing

- Paper Chair (=Editor): Selected per event by the organization
- Chair sets up a Program Committee (PC)
 - Similar to the Editorial Board of Journals
 - Members serve as reviewer and “Area Editors”, i.e. delegating reviews to others
 - PC members are usually top-researchers in the respective field
- Major decision types:
 - Accept “as is”
 - Minor Revision (“conditionally accepted”)
 - Reject



Conferences Paper Reviewing

- Submission and review process:
 1. Abstract deadline (optional)
 2. Paper submission deadline
 3. Chair distributes papers to PC members
 - PC members invite further reviewers
 - PC members single blind, reviewer double blind
 4. Reviewer prepare review
 5. PC-member compiles decision
- There are no specific review cycles, but:
 - Authors may clarify criticism with reviewers (rebuttal)
 - Reviewers often discuss with PC member to find objective decision



Quality Measures & Rankings

Relevance

- Make distinction between relevance
 - ... of content with respect to your work
 - ... of a research work to the scientific progress in the field
 - ... of a researcher/author
 - ... of a publication channel
- Relevance requires two things
 - Measure M : How is the relevance determined?
 - Data Base D : With respect to which reference data base does the relevance measure work?
- The common “atom” for relevance measures is a citation of a paper/author by another paper/author.

Impact Factor (IF)

- Impact Factor measure channels, e.g. journals
 - The IF counts the average number of citations to articles/papers published in a publication channel for a given data base and over a specific period (usually 5 years)

- Formally:

$$MM_{IF}(c) = \frac{\sum_{a \in C} \sum_{a' \in D} \sum_{a' \in C} \{a^c \text{ cites } a'\}}{\# \text{ papers in } c}$$

– c, c' pub. channel; D data base of channels; a, a' articles / papers

- IF of an article = IF of the respective channel



- IF of an author = sum of his paper's IF



→ both derived measures are rather coarse

Hirsch- or h-Index

- H-Index measures authors or channels
 - A scientist/channel has index h if h of his/her papers have at least h citations each, and the other papers have less than h citations each

- Formally:

$$c(a) = |\{a' \in D \mid a' \text{ cites } a\}|$$

$$a, a' \in D, c \in C$$

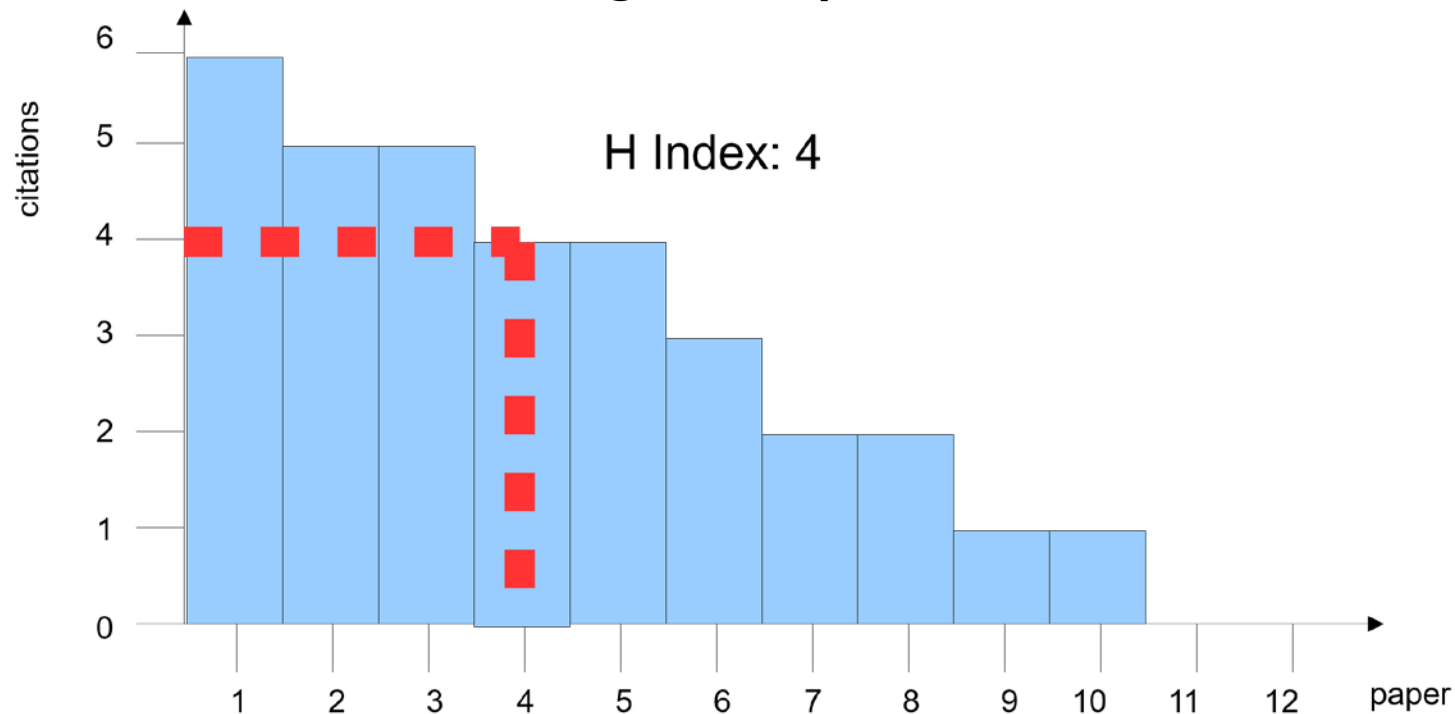
$$h(p) = \max_{i \in \mathbb{N}} \{|\{a \in A(p) \mid c(a) \geq i\}| \geq i\}$$

– p person; c pub. channel; a, a' article; A(p) articles published by p; D data base




- c(a) is the number of citation for paper a
- h-index of a channel = h-index on the basis of all papers in this channel

Hirsch- or h-Index

- Determine the number of citations for each paper
- Order the paper with descending citation count
- The H-Index is now the “largest square” that fits under the curve



Measure: Discussion

- General Motivation of IF and h-index
 - “Good paper” = “cited by many other papers” 
 - IF does not evaluate individual papers 
 - h-index mixes “many papers” & “often cited” 
- General Problems
 - All measures are context sensitive, i.e. they can not be used to compare papers across disciplines
 - Example: The average h-indices and IFs in Physics and Medicine are much higher than in Computer Science
 - Self-citations lead to distortions
 - All measures are imperfect, e.g. the “scientific quality” of the content can't be measured objectively, still they are needed

Discuss the following questions

- a) How does the average number of co-authors in a discipline influence IF and h-index?
- b) What is the relation between the IF / h-index and the acceptance ratio of a journal / conference?

Ranking

- The aforementioned measures help to identify relevant
 - Papers in your field
 - Researchers in your field
 - Publications channels in your field
- We will look at three examples
 - Google Scholar (more literature search database in chapter 3)
 - Scimago Journal & Country Rank
 - Computing Research and Education Association of Australasia (CORE) conference ranking

Google Scholar

- Freely available under scholar.google.com
- Database: “The internet”
 - Internal rating to classify scientific documents
 - Any citation in a “scientific document” counts (Master Thesis, technical report etc.)
- Scholar is a general search engine for scientific documents
 - Paper search attributes: Author, title, channel, year
- Retrieved paper information:
 - Abstract paper information
 - Link to PDF if available
 - Num. of citation for document & referring docs
- Structural and channel information, e.g.,
 - Channel rankings
 - User profiles

Google Scholar

The screenshot shows the Google Scholar homepage. A hamburger menu icon in the top left is circled in orange. An arrow points from this icon to a detailed view of the menu on the left. In the main search area, the search bar is circled in orange, with an arrow pointing to a search results page. The search results page shows a search for 'Andreas Kolb' and a highlighted author profile for 'Andreas Kolb'.

Channel search & ranking

Author profile

Initial View

Top publications

Publication	h5-index	h5-median
1. Nature	414	607
2. The New England Journal of Medicine	410	704
3. Science	391	564

Search View

Search publications

Categories

Publication	h5-index	h5-median
1. Nature	414	607
2. The New England Journal of Medicine	410	704
3. Science	391	564
4. IEEE/CVF Conference on Computer Vision and Pattern Recognition	356	583

Top 20 of Subcategory

Categories > Engineering & Computer Science > Computer Graphics

Publication	h5-index	h5-median
1. ACM Transactions on Graphics (TOG)	95	156
2. IEEE Transactions on Visualization and Computer Graphics	82	118
3. Computer Graphics Forum	60	83
4. IEEE Computer Graphics and Applications	29	41
5. The Visual Computer	29	39
6. Computers & Graphics	27	45
7. ACM Symposium on Virtual Reality Software and Technology	24	35
8. Computer Aided Geometric Design	21	31
9. ACM SIGGRAPH/Eurographics Symposium on Computer Animation	19	23
10. IEEE Symposium on Visual Analytics Science and Technology	18	32
11. IEEE Pacific Visualization Symposium	18	26
12. Computer Animation and Virtual Worlds	18	25
13. Visual Informatics	16	25
14. Symposium on Interactive 3D Graphics (SI3D)	15	22
15. International Conference on 3D Web Technology	13	17

Category / Subcategories

Categories > Engineering & Computer Science > Subcategories

Subcategories	Median
Databases & Information Systems	33
Architecture	36
Artificial Intelligence	70
Automation & Control Theory	22
Aviation & Aerospace Engineering	34
Bioinformatics & Computational Biology	57
Biomedical Technology	70
Biotechnology	36
Ceramic Engineering	35
Civil Engineering	42
Combustion & Propulsion	50
Computational Linguistics	11
Computer Graphics	45
Computer Hardware Design	13
Computer Networks & Wireless Communication	18
Computer Security & Cryptography	34
Computer Vision & Pattern Recognition	34
Computing Systems	34
Data Mining & Analysis	34
Databases & Information Systems	33
Educational Technology	33
Engineering & Computer Science (general)	33
Environmental & Geological Engineering	36
Evolutionary Computation	22
Food Science & Technology	34
Fuzzy Systems	37
Game Theory and Decision Science	70
Human Computer Interaction	36
Library & Information Science	35
Manufacturing & Machinery	35
Materials Engineering	42
Mechanical Engineering	50
Medical Informatics	11
Metallurgy	45
Microelectronics & Electronic Packaging	13
Mining & Mineral Resources	18
Multimedia	34
Nanotechnology	34
Ocean & Marine Engineering	33
Oil, Petroleum & Natural Gas	36
Operations Research	70
Plasma & Fusion	22
Power Engineering	34
Quality & Reliability	37
Radar, Positioning & Navigation	70
Remote Sensing	36
Robotics	35
Signal Processing	50
Software Systems	11
Structural Engineering	45
Sustainable Energy	13
Technology Law	18
Textile Engineering	34
Theoretical Computer Science	34
Transportation	34
Water Supply & Treatment	34
Wood Science & Technology	34

Google Scholar: Author Profile

Google Scholar



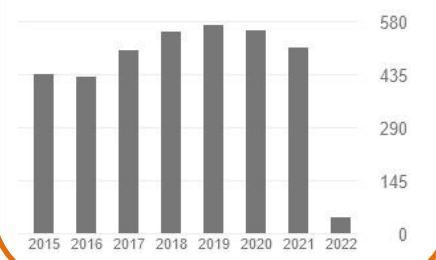
Andreas Kolb

FOLLOW

Computer Science, [University of Siegen](#)
 Verified email at uni-siegen.de - [Homepage](#)
 Computer Graphics Computer Vision Computer Animation Image Processing

Cited by [VIEW ALL](#)

	All	Since 2017
Citations	5857	2741
h-index	34	24
i10-index	79	43



Total number of citations per year

Affiliation & Research Topics

TITLE	CITED BY	YEAR
<input type="checkbox"/> Time-of-flight cameras in computer graphics A Kolb, E Barth, R Koch, R Larsen Computer Graphics Forum 29 (1), 141-159	624	2010
<input type="checkbox"/> Kinect range sensing: Structured-light versus Time-of-Flight Kinect H Sarbolandi, D Lefloch, A Kolb Computer vision and image understanding 139, 1-20	424	2015
<input type="checkbox"/> Real-time 3d reconstruction in dynamic scenes using point-based fusion M Keller, D Lefloch, M Lambers, S Izadi, T Weyrich, A Kolb 2013 International Conference on 3D Vision-3DV 2013, 1-8	369	2013
<input type="checkbox"/> SPH fluids in computer graphics M Ihmsen, J Orthmann, B Solenthaler, A Kolb, M Teschner EUROGRAPHICS 2014/S. LEFEBVRE AND M. SPAGNUOLO	274	2014
<input type="checkbox"/> Time-of-flight sensor calibration for accurate range sensing M Lindner, I Schiller, A Kolb, R Koch Computer Vision and Image Understanding 114 (12), 1318-1328	274	2010
<input type="checkbox"/> Optical techniques for 3D surface reconstruction in computer-assisted laparoscopic surgery L Maier-Hein, P Mountney, A Bartoli, H Elhawary, D Elson, A Groch, ... Medical image analysis 17 (8), 974-996	267	2013
<input type="checkbox"/> Lateral and depth calibration of PMD-distance sensors M Lindner, A Kolb International Symposium on Visual Computing, 524-533	254	2006
<input type="checkbox"/> Hardware-based simulation and collision detection for large particle systems A Kolb, L Latta, C Rezk-Salama Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware ...	215	2004

List of top-cited publications

Public access [VIEW ALL](#)

16 articles not available | 54 articles available

Based on funding mandates


Co-authors [EDIT](#)

- Martin Lambers
University of Siegen
- Reinhard Koch
Professor of Computer Science, ...
- Damien Lefloch
Research

List of Co-Authors

Scopus)

- Scopus not freely available. It covers various scientific journals, proceedings of established conferences & workshops
 - Database: Manually selected by a committee
- SCImage is a public interface to scopus allowing access to channels (<https://www.scimagojr.com/journalrank.php>)
 - We ignore other resources, e.g. country ranking here
- Categorized journals (& some conferences) into Q1 (top) - Q4 (low)
- Most conferences are not ranked, but some data might be given
 - Cites per paper per year
 - SJR index: Weighted average citations over 3 years
- Search for channels via name or category

also developed by scimago:  SCIMAGO INSTITUTIONS RANKINGS

SJR Scimago Journal & Country Rank





Home Journal Rankings Country Rankings Viz Tools Help About Us

All subject areas All subject categories All regions / countries All types 2020

Only Open Access Journals Only SciELO Journals Only WoS Journals [?] Display journals with at least Citable Docs. (3years) **Apply**

Download data

1 - 50 of 32958


Title	Type	↓ SJR	H index	Total Docs. (2020)	Total Docs. (3years)	Total Refs. (2020)	Total Cites (3years)	Citable Docs. (3years)	Cites / Doc. (2years)	Ref. / Doc. (2020)	
1 Ca-A Cancer Journal for Clinicians	journal	62.937 Q1	168	47	119	3452	15499	80	126.34	73.45	
2 MMWR Recommendations and Reports [?]	journal	40.949 Q1	143	10	9	1292	492	9	50.00	129.20	
3 Nature Reviews Molecular Cell Biology	journal	37.461 Q1	431	115	338	8439	10844	167	32.83	73.38	
4 Quarterly Journal of Economics	journal	34.573 Q1	259	40	110	2733	1945	109	16.00	68.33	

CORE Conference Ranking

- CORE ranks journals and conferences in computing disciplines (<http://portal.core.edu.au/conf-ranks>)
- Ranking is done by the CORE Executive Committee according to
 - A* - flagship conference, a leading venue in a discipline area
 - A - excellent conference, and highly respected in a discipline area
 - B - good to very good conference, and well regarded in a discipline area
 - C - other ranked conference venues that meet minimum standards
 - (Australasian - A conference with main audience in Australia & New Zealand
 - Unranked - A conference for which no ranking decision has been made
 - (National - A conference which is run primarily in a single country, with Chairs from that country, and which is not sufficiently well known to be ranked)
 - (Regional - Similar to National but may cover a region crossing national borders)

CORE Conference Ranking

Logged in as Kolb Andreas (andreas.kolb@uni-siegen.de)
[Logout](#)



CORE Conference Portal
Computing Research & Education

[CORE homepage](#) | [CORE rankings page](#) | [Frequently asked questions](#)

Search by: All Source: CORE2021

Showing results 1 - 50 of 970

CORE2021 Summary:

- A* - 7.2% of 805 ranked venues
- A - 16.02% of 805 ranked venues
- B - 37.27% of 805 ranked venues
- Australasian B - 1.61% of 805 ranked venues
- C - 36.15% of 805 ranked venues
- Australasian C - 1.74% of 805 ranked venues
- Other - 165 total

Title	Acronym	Source	Rank	DBLP	hasData?	Primary FoR	Comments	Average Rating
National Conference of the American Association for Artificial Intelligence	AAAI	CORE2021	A*	view	No	4602	1	5.0
International Joint Conference on Autonomous Agents and Multiagent Systems (previously the International Conference on Multiagent Systems, ICMAS, changed in 2000)	AAMAS	CORE2021	A*	view	Yes	4602	2	5.0
Association of Computational Linguistics	ACL	CORE2021	A*	view	No	4602	0	N/A
ACM Multimedia	ACMMM	CORE2021	A*	view	No	4603	2	N/A
Automated Software Engineering Conference	ASE	CORE2021	A*	view	Yes	4612	2	5.0
Architectural Support for Programming Languages and Operating Systems	ASPLOS	CORE2021	A*	view	No	4612	0	N/A
Computer Aided Verification	CAV	CORE2021	A*	view	Yes	4612	0	N/A
ACM Conference on Computer and Communications Security	CCS	CORE2021	A*	view	Yes	4604	0	N/A
International Conference on Human Factors in Computing Systems	CHI	CORE2021	A*	view	Yes	4608	0	N/A
Conference on Learning Theory	COLT	CORE2021	A*	view	Yes	4611	1	N/A
Advances in Cryptology	CRYPTO	CORE2021	A*	view	Yes	4604	0	N/A
IEEE Conference on Computer Vision and Pattern Recognition	CVPR	CORE2021	A*	view	Yes	4603	1	5.0
ACM Conference on Economics and Computation	EC	CORE2021	A*	view	Yes	4602	2	5.0
European Conference on Computer Vision	ECCV	CORE2021	A*	view	Yes	4603	4	5.0
European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (duplicate was listed as ESEC, removed from DB)	ESEC/FSE	CORE2021	A*	view	Yes	4612	6	5.0
International Conference on the Theory and Application of Cryptographic Techniques	EuroCrypt	CORE2021	A*	view	Yes	4604	0	N/A
IEEE Symposium on Foundations of Computer Science	FOCS	CORE2021	A*	view	No	4613	0	N/A
International Symposium on High Performance Computer Architecture	HPCA	CORE2021	A*	view	No	4606	0	N/A
International Conference on Automated Planning and Scheduling	ICAPS	CORE2021	A*	view	Yes	4602	0	N/A

Toggle between journals and conferences

Assignment 3

Determine and compare the ranking / classification of the following journal in google scholar, SCImago and CORE

- a) Computer Graphics Forum
- b) ACM Transactions on Graphics
- c) Computer Graphics International (Conference)

Discuss your results!

Thank you for your attention!