Wie schreibe ich mein Guided Research?

Und vor allem: Warum sollte ich?

Dr. Elmar Juergens, Roman Haas

In enger Abstimmung mit Dr. Angelika Reiser

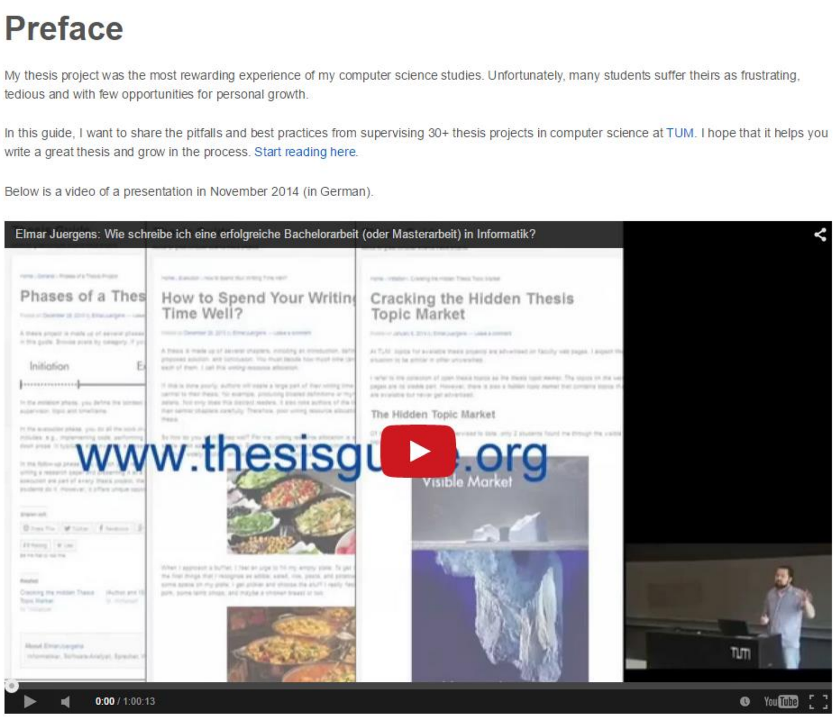
2000 - 2006

2006 - heute

2009 - heute

Grundlage: www.thesisguide.org

- Folien
- Video
- Detaillierte Essays
- FAQ



Presentation on Master‘s Thesis (English): 12.6., 18 Uhr, Interims 2

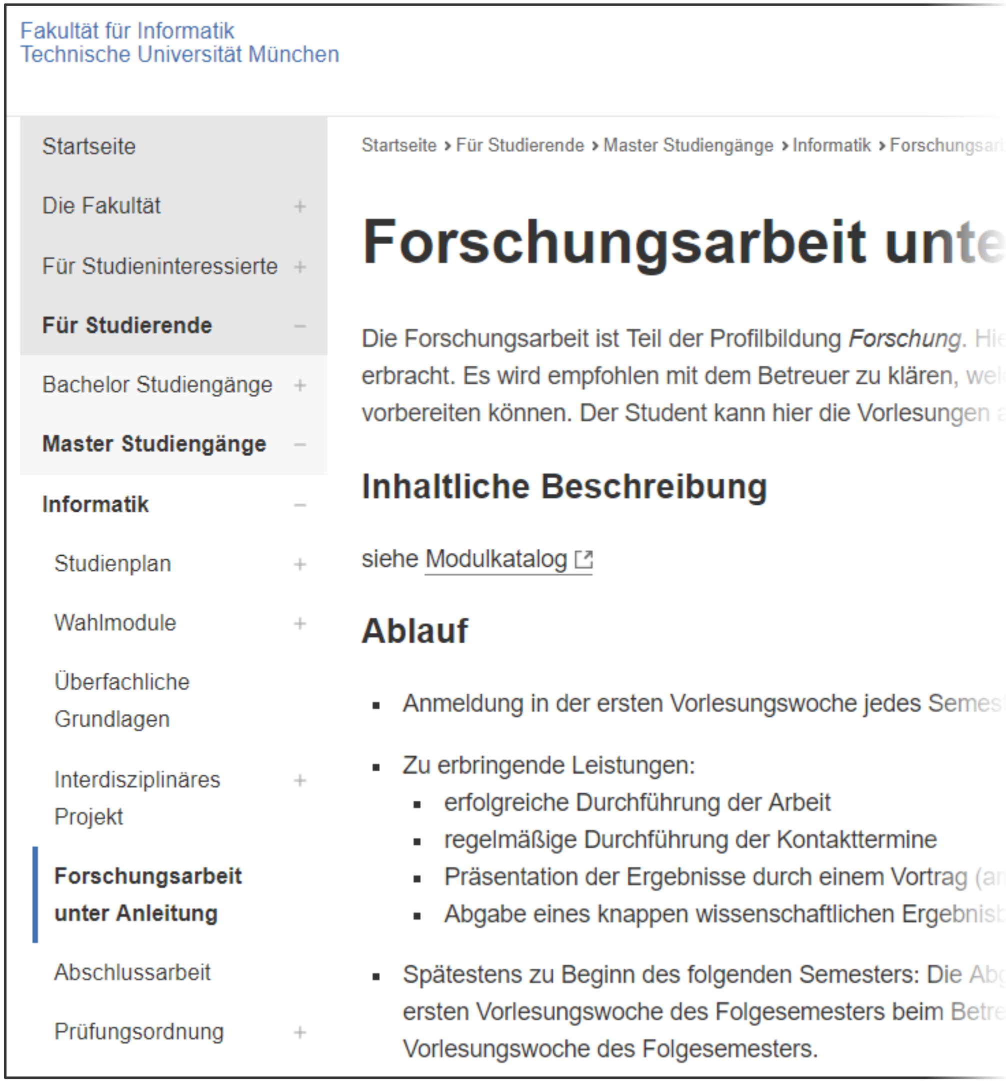
Vortrag Bachelorarbeit (Deutsch): 3.7., 18 Uhr, HS2

Agenda

1. Motivation
2. Anbahnung
3. Durchführung

Guided Research

- Eigene Forschungs-arbeit & Vortrag
- Läuft genau ein Semester
- Erstreckt sich möglicherweise in die Semesterferien


Fakultät für Informatik
Technische Universität München

Startseite
Die Fakultät +
Für Studieninteressierte +
Für Studierende –
Bachelor Studiengänge +
Master Studiengänge –
Informatik –
Studienplan +
Wahlmodule +
Überfachliche Grundlagen
Interdisziplinäres Projekt +
Forschungsarbeit unter Anleitung
Abschlussarbeit
Prüfungsordnung +

Startseite › Für Studierende › Master Studiengänge › Informatik › Forschungsar

Forschungsarbeit unte

Die Forschungsarbeit ist Teil der Profilbildung *Forschung*. Hi
erbracht. Es wird empfohlen mit dem Betreuer zu klären, we
vorbereiten können. Der Student kann hier die Vorlesungen

Inhaltliche Beschreibung

siehe Modulkatalog

Ablauf

- Anmeldung in der ersten Vorlesungswoche jedes Semes
- Zu erbringende Leistungen:
  - erfolgreiche Durchführung der Arbeit
  - regelmäßige Durchführung der Kontakttermine
  - Präsentation der Ergebnisse durch einem Vortrag (a
  - Abgabe eines knappen wissenschaftlichen Ergebnis
- Spätestens zu Beginn des folgenden Semesters: Die Ab
ersten Vorlesungswoche des Folgesemesters beim Betr
Vorlesungswoche des Folgesemesters.

https://www.in.tum.de/de/fuer-studierende/master-studiengaenge/informatik/forschungsarbeit-unter-anleitung/

Formalitäten

- Informatik, Data Engineering&Analytics Wirtschaftsinformatik, Informatik: Games Engineering, Biomedical Computing

- Anmeldung in 1. Vorlesungswoche
- Abgabe Ende Semester
- Nicht verlängerbar
- Man muss schon im Master eingeschrieben sein
- Keine Anerkennung aus dem Ausland (man kann aber mit TUM-Betreuer auch im Ausland schreiben)

Guided Research

- Freiwillig
- 10 ECTS (aber nicht weniger Arbeit)
- Konferenzvortrag oft deutlich nach Abgabe
- ca. 40/Semester

Bachelorarbeit

- Verpflichtend
- 15 ECTS
- Mit Abgabe & Vortrag abgeschlossen
- ca. 200/Semester?

2011 - 2017

2017 - heute

2014: BA
2015: GR
2016-2017: MA

Learning to Rank Extract Method Refactoring Suggestions for Long Methods

- Gegeben: Eine Menge von Refactoring-Vorschlägen für lange Methoden
- Gesucht: Ordnung der Vorschläge
- Ansatz: Machine Learning

Resultat

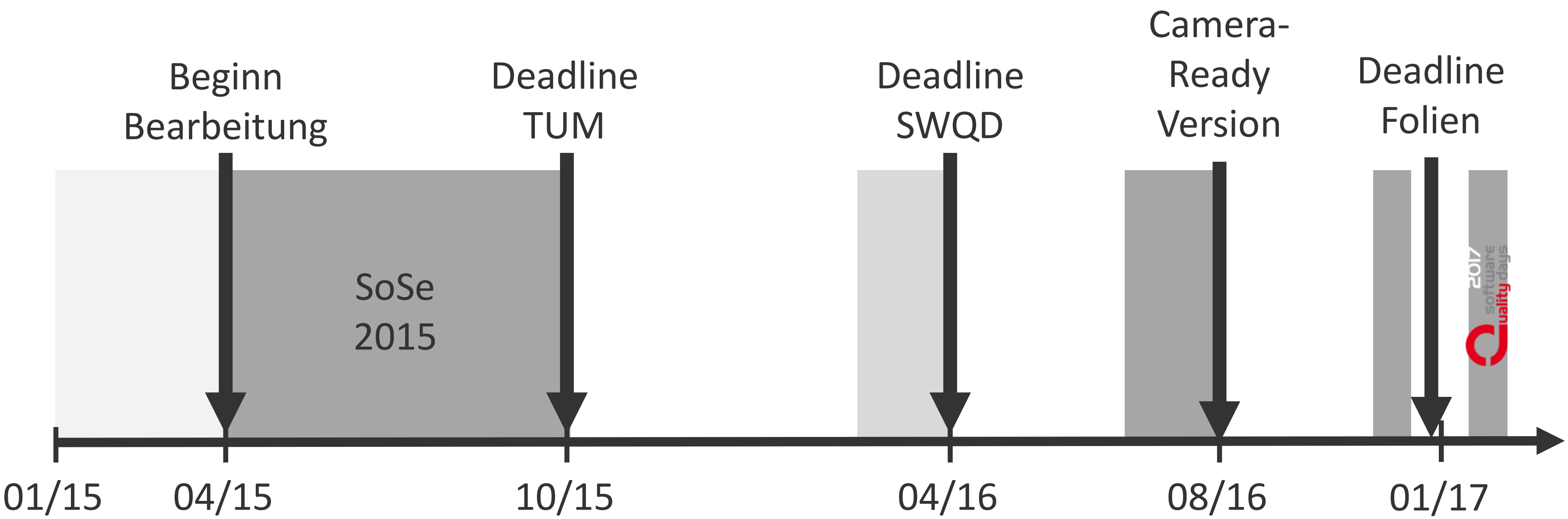

Learning to Rank Extract Method Refactoring Suggestions for Long Methods
Roman Haas and Benjamin Hummel
1.1 Introduction
1.2 Fundamentals
1.2.2 Training and Testing


1.3 Approach
1.3.1 Extract Method Refactoring Candidates
1.3.2 Scoring Function
1.3.3 Training and Test Data Generation
1.4 Evaluation
1.4.1 Research Questions
1.4.3 Results
1.4.4 Discussion
1.5 Threats to Validity
1.6 Related Work
1.7 Conclusion and Future Work
Acknowledgements

Konferenz Tag 1 - 18.01.2017

	Track A	Track B	Track C	Scientific-Track	Solution Provider Forum I	Solution Provider Forum II
		hibition area / Kaffeepause & Networking im Ausstellungsbereich				
		nfug!...aber ist Form von nalyse wirklich Über die	Continuous Integration für Mobile Apps (Zühlke Engineering (Austria)	Improve your software models with search-based techniques	Ranorex in the Agile World (Ranorex GmbH, Graz) Deutsch , Einsteiger	Testumgebungen auf einen Klick - zeitgemäßes Testumgebungsmanagement als Herausforderung und
		Deutsch , Fortgeschrittene		(FUJITSU Enabling Software Englisch , Fortgeschrittene		
14:05				Learning to Rank Extract Method Refactoring Suggestions for Long Methods (Technical University of Munich, Garching, DE) Englisch , Fortgeschrittene		
14:25						
14:35	Kontinuierliche Architekturanalyse (Software Quality Lab, Linz) Deutsch , Einsteiger	Strukturierte Tests bei defizitärer Dokumentation - Wie man zwei Fliegen mit einer Klappe schlägt (SRC Security Research & Consulting GmbH, Bonn, DE) Deutsch , Fortgeschrittene	Continuous Delivery - Feel your Quality - Every Day (Automic Software GmbH (CA Technologies), Wien, AT) (Automic Software GmbH, Wien, AT) Englisch , Fortgeschrittene	A portfolio of internal quality metrics for software architects (University of Gothenburg, Gothenburg, SE) Englisch , Fortgeschrittene	Scrum in Embedded Systems (Software Quality Lab GmbH, Linz, AT) (ENGEL AUSTRIA GmbH, Schwertberg, AT) Deutsch , Fortgeschrittene	Zertifizierung Quality Engineer für das Internet der Dinge (iSQI GmbH, Potsdam, DE) Englisch , Experte
14:55				Validating converted Java Code via Symbolic Execution (ZT Prentner-IT, Wien, AT) Englisch , Fortgeschrittene		

Zeitlicher Überblick



Delta zu Alternativen im Masterstudium

- Mehr Freiheiten
  - Themenrichtung
  - Eigene Forschung mit abgestimmter Methodik
  - Eigenes Tempo und eigener Zeitplan

- Höhere Anforderungen an Selbstorganisation

- Mehr Möglichkeiten für persönliches Wachstum

Persönliches Fazit

- GR war über das gesamte Masterstudium auf meinem „mentalen Stack“
- GR hat mich aus meiner Komfortzone geholt
- Ich habe richtige Forschung betrieben
- Ich habe die Forschungscommunity kennengelernt
- Ich würde das GR nochmal machen

Finanzierung

Zu deckende Kosten: 1k€ bis 5k€

- Anreise und Übernachtung
- Konferenzgebühr

Finanzierungsquellen (oft Mischfinanzierung)

- Reisekostenzuschuss der Fakultät
- Lehrstühle
- DAAD Stipendien
- CQSE

Entscheidungsprozesse dauern oft lange -> Früh kümmern

Agenda

1. Motivation
2. **Anbahnung**
3. Durchführung

Konferenz Tag 1 - 18.01.2017

	Track A	Track B	Track C	Scientific-Track	Solution Provider Forum I	Solution Provider Forum II
		...hibition area / Kaffeepause & Networking im Ausstellungsbereich				
		...nfug!...aber ist ... Form von ...nalyse wirklich ... Über die	Continuous Integration für Mobile Apps (Zühlke Engineering (Austria)	Improve your software models with search-based techniques	Ranorex in the Agile World (Ranorex GmbH, Graz) Deutsch , Einsteiger	Testumgebungen auf einen Klick - zeitgemäßes Testumgebungsmanagement als Herausforderung und ... ...e Design und ...Wien, AT) ...e Design und ...Wien, AT) ...hrittene
						...ments ... eine ...etzung mit ...ia ... Kornwestheim,
				(FUJITSU Enabling Software Englisch , Fortgeschrittene		
14:05				Learning to Rank Extract Method Refactoring Suggestions for Long Methods (Technical University of Munich, Garching, DE) Englisch , Fortgeschrittene		
14:25						
14:35	Kontinuierliche Architekturanalyse (Software Quality Lab, Linz) Deutsch , Einsteiger	Strukturierte Tests bei defizitärer Dokumentation - Wie man zwei Fliegen mit einer Klappe schlägt (SRC Security Research & Consulting GmbH, Bonn, DE) Deutsch , Fortgeschrittene	Continuous Delivery - Feel your Quality - Every Day (Automic Software GmbH (CA Technologies), Wien, AT) (Automic Software GmbH, Wien, AT) Englisch , Fortgeschrittene	A portfolio of internal quality metrics for software architects (University of Gothenburg, Gothenburg, SE) Englisch , Fortgeschrittene	Scrum in Embedded Systems (Software Quality Lab GmbH, Linz, AT) (ENGEL AUSTRIA GmbH, Schwertberg, AT) Deutsch , Fortgeschrittene	Zertifizierung Quality Engineer für das Internet der Dinge (iSQI GmbH, Potsdam, DE) Englisch , Experte
14:55				Validating converted Java Code via Symbolic Execution (ZT Prentner-IT, Wien, AT) Englisch , Fortgeschrittene		

Einreichungen

Auswahlverfahren

Agenda

Hackordnung

ICSE 40th INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING
MAY 27 - JUNE 3 2018
GOTHENBURG, SWEDEN

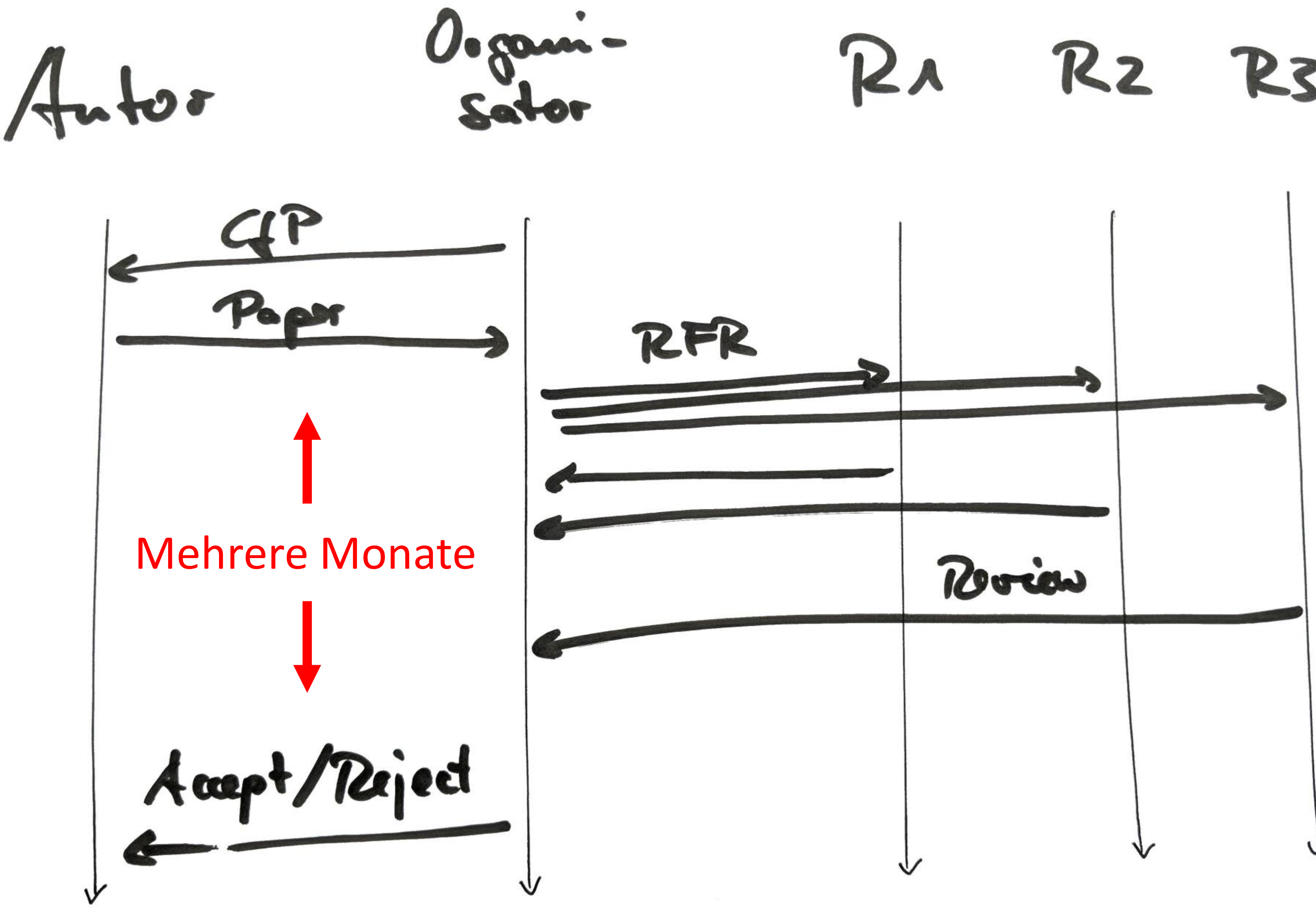
Konferenz
10%-25%

Acronym	Full Name	Date
CHASE	11th International Workshop on Cooperative and Human Aspects of Software Engineering	27-May
CSI-SE	5th International Workshop on Crowd Sourcing in Software Engineering	27-May
MET	International Workshop on Metamorphic Testing	27-May

RAISE	SoHeal	MiSE	GE	SQUADE	SE4COG	SER&IP	SE4Science
SEAD	WETSEB	SEHS	RoSE	AST	FairWare	SESoS	RET
SEsCPS	GREENS	CESI	SEFAIAS	SBST	RCoSE	GI	SEEM

Workshop
40%-60%

Ziel: Einreichung auf Workshops



Autor
Organi-
sator
R1
R2
R3
CfP
Paper
RFR
Mehrere Monate
Review
Accept/Reject

IWSC 2018 Registration Objectives Program Call for Papers Important Dates Keynote Team Previous Editions

Call for Papers

12th International Workshop on Software Clones (IWSC 2018)

Co-located with the 25th IEEE International Conference on Software Analysis, Evolution, and Reengineering (SANER 2018)
March 20, 2018, Campobasso, Italy

Software clones are often a result of copying and pasting as an act of ad-hoc reuse by programmers, and can occur at many levels, from simple
statement sequences to blocks,
models, requirements or architec
today.

IWSC series of events has provid
IWSC aims to bring researchers a
particular, we expect the in-depth
about IWSC 2018 are here on thi

TOPICS OF INTEREST:

Topics of interest include but not

- Use cases for clones and c
- Experiences with clones ar
- Types and nature of clones
- Causes and effects of clon
- Techniques and algorithms
- Clone and clone pattern vis
- Tools and systems for dete
- Applications of clone detec
- System architecture and cl
- Effect of clones to system
- Clone analysis in families c
- Measures of code similarit
- Economic and trade-off mo
- Evaluation and benchmark
- Licensing and plagiarism is
- Clone-aware software desi
- Refactoring through clone
- Higher-level clones in mod
- Clone evolution and variati
- Role of clones in software

PAPERS SOUGHT:

Each paper will be reviewed by at least three members of the program committee following **a full double-blind process. Authors must adhere to SANER's double blind guidelines - http://saner.unimol.it/restrack**. The following types of papers are sought:

- Full papers (7 pages maximum)
- Position papers (2 pages maximum)
- Tool demonstration papers (4 pages maximum)

SUBMISSION:

Papers must conform to the IEEE proceedings paper format guidelines.If the paper is accepted, at least one author must attend the workshop and present the paper. Accepted papers will be published in the IEEE Xplore Digital Library along with the SANER proceedings.

All submissions must be in PDF and must be submitted online by the deadline via the IWSC 2018 EasyChair conference management system.

Submit your papers here >>> EasyChair<<<

IMPORTANT DATES:

- Abstract submission deadline: January 19, 2018 AoE
- Paper submission deadline: January 26, 2018 AoE
- Notifications: February 16, 2018
- Camera Ready deadline: ** February 22, 2018 **
- Workshop day: March 20 2018

GENERAL CHAIR:

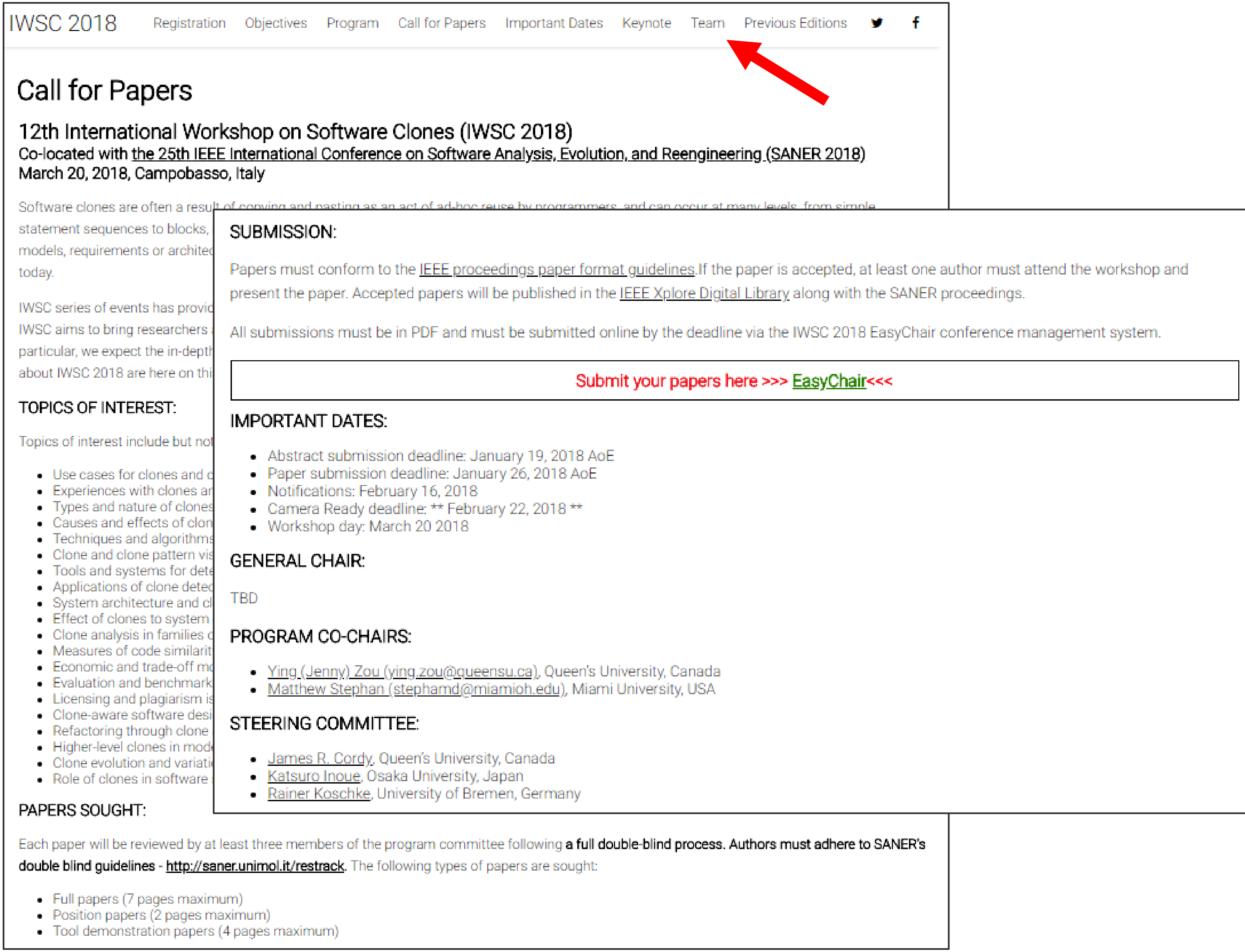
TBD

PROGRAM CO-CHAIRS:

- Ying (Jenny) Zou (ying.zou@queensu.ca), Queen's University, Canada
- Matthew Stephan (stephamd@miamioh.edu), Miami University, USA

STEERING COMMITTEE:

- James R. Cordy, Queen's University, Canada
- Katsuro Inoue, Osaka University, Japan
- Rainer Koschke, University of Bremen, Germany



IWSC 2018 Registration Objectives Program Call for Papers Important Dates Keynote Team Previous Editions

Call for Papers

12th International Workshop on Software Clones (IWSC 2018)

Co-located with the 25th IEEE International Conference on Software Analysis, Evolution, and Reengineering (SANER 2018)
March 20, 2018, Campobasso, Italy

Software clones are often a result of copying and pasting as an act of ad-hoc reuse by programmers, and can occur at many levels, from simple statement sequences to blocks, models, requirements or architec today.

IWSC series of events has provid IWSC aims to bring researchers particular, we expect the in-depth about IWSC 2018 are here on thi

TOPICS OF INTEREST:

Topics of interest include but not

- Use cases for clones and c
- Experiences with clones ar
- Types and nature of clones
- Causes and effects of clon
- Techniques and algorithms
- Clone and clone pattern vis
- Tools and systems for dete
- Applications of clone detec
- System architecture and cl
- Effect of clones to system
- Clone analysis in families c
- Measures of code similarit
- Economic and trade-off mo
- Evaluation and benchmark
- Licensing and plagiarism is
- Clone-aware software desi
- Refactoring through clone
- Higher-level clones in mode
- Clone evolution and variati
- Role of clones in software

SUBMISSION:

Papers must conform to the IEEE proceedings paper format guidelines.If the paper is accepted, at least one author must attend the workshop and present the paper. Accepted papers will be published in the IEEE Xplore Digital Library along with the SANER proceedings.

All submissions must be in PDF and must be submitted online by the deadline via the IWSC 2018 EasyChair conference management system.

Submit your papers here >>> EasyChair<<<

IMPORTANT DATES:

- Abstract submission deadline: January 19, 2018 AoE
- Paper submission deadline: January 26, 2018 AoE
- Notifications: February 16, 2018
- Camera Ready deadline: ** February 22, 2018 **
- Workshop day: March 20 2018

GENERAL CHAIR:

TBD

PROGRAM CO-CHAIRS:

- Ying (Jenny) Zou (ying.zou@queensu.ca), Queen's University, Canada
- Matthew Stephan (stephamd@miamioh.edu), Miami University, USA

STEERING COMMITTEE:

- James R. Cordy, Queen's University, Canada
- Katsuro Inoue, Osaka University, Japan
- Rainer Koschke, University of Bremen, Germany

PAPERS SOUGHT:

Each paper will be reviewed by at least three members of the program committee following **a full double-blind process. Authors must adhere to SANER's double blind guidelines - http://saner.unimol.it/restrack**. The following types of papers are sought:

- Full papers (7 pages maximum)
- Position papers (2 pages maximum)
- Tool demonstration papers (4 pages maximum)

Program Committee

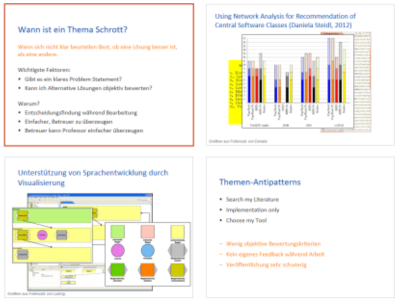
Name	Institiution	Country
Toshihiro Kamiya	Shimane University	Japan
Daqing Hou	Clarkson University	USA
Tien Nguyen	University of Texas at Dallas	USA
Nils Göde	CQSE GmbH	Germany
Jens Krinke	University College London	UK
Otavio Lemos	ICT-UNIFESP	Brazil
Manishankar Mondal	University of Saskatchewan	Canada
Ravindra Naik	Tata Consultancy Services	India
Robert Tairas	Vanderbilt University	USA
Minhaz Zibran	University of New Orleans	USA
Eunjong Choi	Nara Institute of Science and Technology	Japan
Michael Godfrey	University of Waterloo	Canada
Yoshiki Higo	Osaka University	Japan
Foutse Khomh	Ecole Polytechnique de Montréal	Canada
Nicholas A. Kraft	ABB Corporate Research	USA
Chanchal Roy	University of Saskatchewan	Canada
Hitesh Sajnani	Microsoft	USA
Suresh Thummalapenta	Microsoft	USA
Xioyin Wang	University of Texas at San Antonio	USA
Norihiro Yoshida	Nagoya University	Japan

Anforderungen an Thema

- Gibt es ein klares Problem Statement?
- Kann ich Alternative Lösungen objektiv bewerten?

Warum?

- Entscheidungsfindung während Bearbeitung
- Einfacher, Betreuer zu überzeugen
- Einfacher, PC zu überzeugen

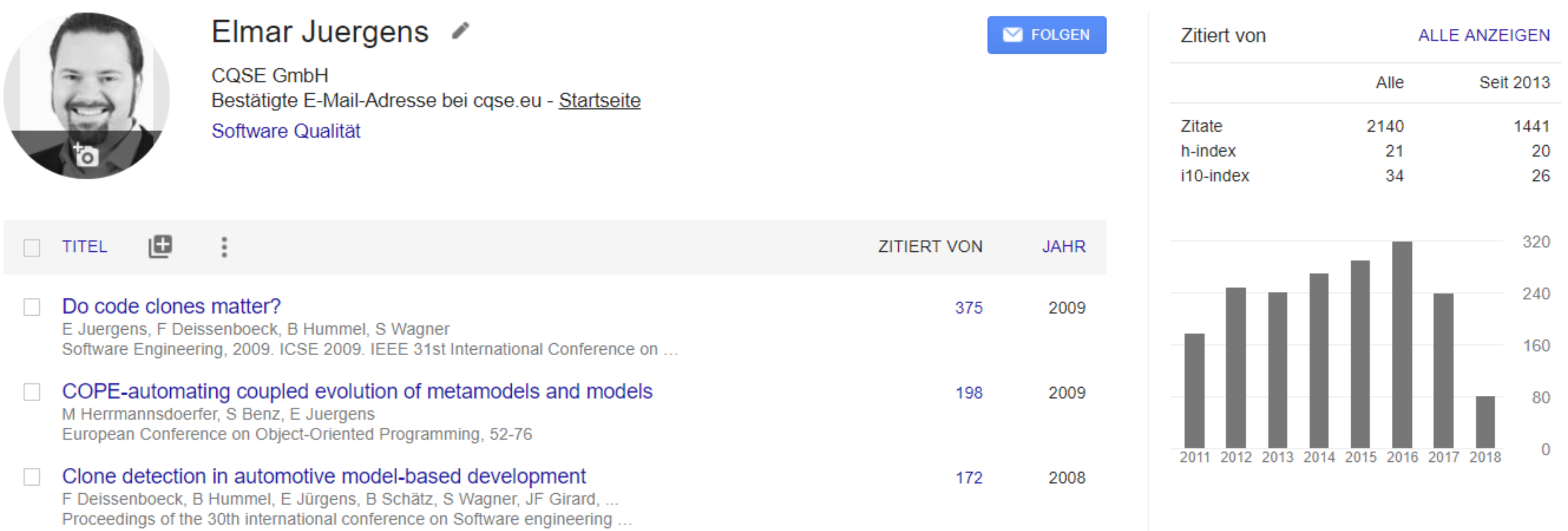


Noch wichtiger für GR, als für BA oder MA.

Mehr Infos: www.thesisguide.org

Anforderungen an Betreuer

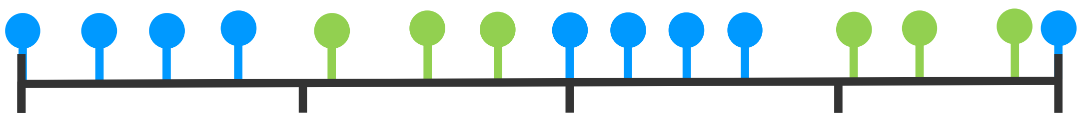
- Veröffentlichungserfahrung notwendig
- Idealerweise auf geplantem Workshop
- Quellen: scholar.google.com, DBLP, persönlich Webseite.



Agenda

1. Motivation
2. Anbahnung
3. **Durchführung**

Sicht eines BA/MA-Betreuers



 Regelmäßiges Treffen

 Treffen nach Bedarf

1	Izautino P. Oliveira and Renata Souza	Stories for New Products: A Research on the Use of Storytelling in Requirements Identification in Software Engineering	Sep 03, 04:18
2	Hans Jochen Scholl, William Menten-Weil and Timothy S. Carlson	Artifact Evaluation with TEDSrate	Sep 19, 21:10
3	Celal Ziftci and Jim Reardon	Who Broke the Build? Automatically Identifying Changes That Induce Test Failures In Continuous Integration at Google Scale	Sep 27, 20:32
4	Uma Viswanath and Ramya Shyama Palakodati	Measuring Leanness in a lean software organization: A model to gauge the lean implementation using the 3 dimensional approaches of Process, Product, and People	Oct 05, 04:09
5	Sofia Modesto and Miguel Mira Da Silva	Gamification to Increase Scrum Adoption	Oct 14, 15:51
6	Guillermo Rodriguez, Alvaro Soria and Marcelo Campo	A Case-based Reasoning Approach to Reuse Quality-driven Design Alternatives in Service-Oriented Architectures	Oct 14, 19:08
7	Andrea Arcuri	Back After 5 Years In Industry As Software Engineer / Tester: We Need Usable Automated Test Generation	Oct 15, 16:58
8	Christof Ebert and Michael Weyrich	Architecture Evolution for the Internet of Things	Oct 16, 05:59
9	Andrew Ko	A Three-Year Participant Observation of Software Startup Software Evolution	Oct 17, 22:08
10	Torvald Mårtensson, Pär Hammarström and Jan Bosch	Continuous Integration Is Not About Build Systems	Oct 19, 19:43
11	Junjie Wang, Qiang Cui, Song Wang and Qing Wang	Domain Adaptation for Test Report Classification in Crowdsourced Testing	Oct 20, 08:09
12	Fernando Pinciroli, Jose Luis Barros Justo and Raymundo Forradellas	Aspect-Oriented Business Process Modeling Approaches: An assessment of AOP4ST	Oct 20, 13:32
14	Paul L. Li, Andrew J. Ko and Andrew Begel	Expert Non-Software-Engineers's Perspectives on Why Software Engineering Teams Succeed or Fail	Oct 20, 16:10
15	Balbir Barn, Souvik Barat, Tony Clark and Vinay Kulkarni	Reviewing the Software Engineering Nexus of Current Research, Practice, and Future Prospects	Oct 21, 14:00
16	Saad Mubeen, Mikael Sjödin, Harold Lawson, John Lundbäck, Mattias Gålnander and Kurt-Lennart Lundbäck	Provisioning of Predictable Embedded Software in the Vehicle Industry: The Rubus Approach	Oct 22, 19:47
17	Christopher Theisen, Brendan Murphy, Kim Herzig and Laurie Williams	Risk-Based Attack Surface Approximation: How Much Data is Enough?	Oct 23, 03:16
18	Daniel Russo, Paolo Ciancarini, Tommaso Falasconi and Massimo Tomasi	Software Quality Concerns in the Italian Bank Sector: the Emergence of a Meta-Quality Dimension	Oct 23, 19:51
19	Akond Rahman, Asif Partho, David Meder and Laurie Williams	Which Factors Influence Practitioners' Usage of Build Automation Tools?	Oct 24, 00:43
20	Cuiyun Gao, Yichuan Man, Hui Xu, Jieming Zhu, Yangfan Zhou and Micheal R. Lyu	Assist Developers in Mobile Advertising via User Reviews and Case Studies	Oct 24, 17:49
21	Zhitao Hou, Hongyu Zhang, Haidong Zhang and Dongmei Zhang	MetroEyes: A Visual Analytics System for Exploring Multi-Dimensional Data	Oct 25, 00:49
22	S M Sohan, Craig Anslow and Frank Maurer	Automated Example Oriented REST API Documentation at Cisco	Oct 25, 05:34
23	Terese Besker, Antonio Martini and Jan Bosch	Time to Pay Up - Technical Debt From a Software Quality Perspective	Oct 25, 08:53
24	Dale Blue, Orna Raz, Rachel Tzoref-Brill, Paul Wojciak and Marcel Zalmanovici	Novel applications of combinatorial testing in validating software design	Oct 25, 12:40
25	Jingzheng Wu, Shen Liu, Shouling Ji, Mutian Yang, Yanjun Wu, Yongji Wang and Tianyue Luo	Exception Beyond Exception: Crashing Android System by Trapping in "uncaughtException"	Oct 25, 14:28
26	Charles Weir, Awais Rashid and James Noble	Dialectical Security: Challenging the Developers of Mobile and IoT Software	Oct 25, 16:38
27	Samuel Marks and Andrew White	Code-generation driven development	Oct 25, 18:35
28	Daniel Izquierdo-Cortazar, Nelson Sekitoleko, Jesus M. Gonzalez-Barahona and Lars Kurth	Using Metrics to Track Code Review Performance: the Xen Case	Oct 25, 18:45
29	Zakariya Dehlawi and Andrew J. Ko	Predicting the Diffusion of Software Security Activities	Oct 25, 19:02
30	Marcos Kalinowski, Pablo Curty, Aline Paes, Alexandre Ferreira, Rodrigo Spínola, Daniel Méndez Fernández, Michael Felderer and Stefan Wagner	Supporting Defect Causal Analysis in Practice with Cross-Company Data on Causes of Requirements Engineering Problems	Oct 25, 20:07
31	Steven D. Fraser and Dennis Mancl	"No Silver Bullet" Revisited: Panel Session	Oct 25, 20:14
32	Trishank Kuppusamy, Vladimir Diaz and Justin Cappos	Mercury: Bandwidth-Effective Prevention of Rollback Attacks Against Community Repositories	Oct 25, 20:52
33	Mojdeh Golagha, Alexander Pretschner, Dominik Fisch and Roman Nagy	Reducing Failure Analysis Time: An Industrial Evaluation	Oct 25, 21:41
34	Atif Memon, Zebao Gao, Bao Nguyen, Sanjeev Dhanda, Eric Nickell, Rob Siemborski and John Micco	Taming Google-Scale Continuous Testing	Oct 25, 23:03
35	Arun Kalyanasundaram, Judith Bishop and James Herbsleb	Industrial Open Source Project Decisions, Best Practices and Community Engagement: A Case Study at Microsoft	Oct 26, 01:40
36	Bargav Jayaraman, Anurag Dwarakanath, Breno D. Cruz and Collin McMillan	A Deep Learning approach for the Multi-lingual identification of Vagueness	Oct 26, 02:19
37	Jingzheng Wu, Sizhe Zhao, Shouling Ji, Mutian Yang, Tianyue Luo, Yanjun Wu and Yongji Wang	MAD-API: Detection, Correction and Explanation of API Misuses in Android Applications	Oct 26, 03:13
38	Remo Eckert, Sathya Kay Meyer and Matthias Stuermer	Capability Maturity Model of Inner Source Implementation	Oct 26, 05:17
39	Khaled Alnawasreh, Patrizio Pelliccione, Zhenxiao Hao, Mårten Rånge and Antonia Bertolino	Online Robustness Testing of Distributed Embedded Systems: an Industrial Approach	Oct 26, 06:56
40	Yulai Zhou, Patrizio Pelliccione, Johan Haraldsson and Mafjiul Islam	Improving Robustness of AUTOSAR Software Components with Design by Contract: A study within Volvo AB	Oct 26, 07:03
41	Henrik Edholm, Mikaela Lidström, Jan-Philipp Steghöfer and Håkan Burden	Crunch Time: The Reasons and Effects of Unpaid Overtime in the Games Industry	Oct 26, 07:26
42	Jacob Krüger, Andy Kenner, Christopher Kruczek and Thomas Leich	Modularizing Conditional Compilation: An Automatic Minimal-Invasive Approach	Oct 26, 07:42
43	Katja Kevic, Brendan Murphy, Laurie Williams and Jennifer Beckmann	Characterizing Experimentation in Continuous Deployment: a Case Study on Bing	Oct 26, 09:23
44	Jakub Misek and Filip Zavoral	Binding semantic tree of dynamic languages to static language constructs	Oct 26, 10:04
45	Ivica Crnkovic and Anna Börjesson Sandberg	Meeting Industry – Academia Research Collaboration Challenges with agile methodologies	Oct 26, 10:21
46	Eero Laukkanen, Maria Paasivaara, Juha Itkonen, Casper Lassenius and Teemu Arvonen	Towards Continuous Delivery by Reducing the Feature Freeze Period: A Case Study	Oct 26, 11:02
47	Pete Rotella, Cody Peeples and Mark-David McLaughlin	Prioritizing Security Bug Fixes: A Novel Text Analytics Approach	Oct 26, 11:03
48	Mohamad Kassab, Jooyoung Lee, Manuel Mazzara, Giancarlo Succi and Rasul Tumyrkin	Software Quality – Traditional vs. Agile: an Empirical Investigation	Oct 26, 11:47
49	Kumar Abhinav, Alpana Dubey, Sakshi Jain, Gurdeep Virdi, Alex Kass and Manish Mehta	CrowdAdvisor: A Framework for Worker Assessment in Crowdsourcing	Oct 26, 11:55
50	Lingfeng Bao, Zhenchang Xing, Xin Xia, David Lo and Shanping Li	Who Will Leave the Company? A Large-Scale Industry Study of Developer Turnover by Mining Monthly Work Report	Oct 26, 13:05
51	Padmalata Nistala and Kesav Vithal Nori	Towards A Software Product Quality Taxonomy to Elicit Quality Requirements	Oct 26, 13:17
52	Christoph Seidl, Thorsten Berger, Christoph Elsner and Klaus-Benedikt Schultis	Challenges and Solutions for Opening Small and Medium-Scale Industrial Software Platforms	Oct 26, 13:31
53	Jayati Deshmukh, Annervaz K M, Sanjay Podder, Shubhashis Sengupta and Neville Dubash	A Deep Learning Approach for Accurate Duplicate Bug Detection	Oct 26, 14:11
54	Ayse Tosun, Ozgur Turkgulu, Dogan Razon, Hamza Yusuf Aydemir and Arda Gureller	Predicting defects using test execution logs in an industrial setting	Oct 26, 14:14
55	Raffaele Ciriello, Alexander Richter and Gerhard Schwabe	When Prototyping Meets Storytelling: Practices and Malpractices in Innovating Software Firms	Oct 26, 14:26
56	Rebekka Wohlrab, Patrizio Pelliccione, Eric Knauss and Mats Larsson	Agility in Automotive: Continuous Engineering of Systems Engineering Artifacts	Oct 26, 14:48
57	Yingxia Wei, Rui Wang and Yu Jiang	From Off-line Towards Real-time : A Runtime Verification Approach for Robot Systems	Oct 26, 15:55
58	Kee-Choon Kwon, Jang-Soo Lee and Eunkyoung Jee	Application of Safety Case for Digital Reactor Protection System in Nuclear Power Plants	Oct 26, 16:04
59	Ulrik Eklund and Christian Berger	Scaling Agile Development in Mechatronic Organizations – A Comparative Case Study	Oct 26, 16:12
60	Jürgen Cito, Fábio Oliveira, Philipp Leitner, Priya Nagpurkar and Harald Gall	Context-Based Analytics - Establishing Explicit Links between Runtime Traces and Source Code	Oct 26, 16:32
61	Francesco Sorrentino	Elastic Partitioning: A Tool for Testing Scalable Distributed Systems	Oct 26, 17:43
62	Hennie Huijgens, Leandro Minku, Chris Lokan and Arie van Deursen	Effort versus Cost in Software Development: A Comparison of Two Industrial Data Sets	Oct 26, 18:56
63	Ma. Laura Caliusco, Emiliano Reynares, Néstor Reynoso, Juan Echagüe, Santiago Sosa and Agustín Martínez	Ontology-Driven Information Systems at the Oil & Gas Domain: An Experience Report	Oct 26, 19:19
64	Simon Harrer, Matthias Geiger, Vincenzo Ferme, Cesare Pautasso, Jörg Lenhard, Marigianna Skouradaki and Frank Leymann	Lessons Learned in Evaluating Workflow Management Systems -- "What you Expect and What you Get"	Oct 26, 19:36
65	Helena Holmström Olsson and Jan Bosch	So Much Data; So Little Value A multi-case study on improving the impact of data-driven development practices	Oct 26, 19:39
66	Franz Zieris and Lutz Prechelt	Pair Programming Feasibility Critically Depends on Task Difficulty	Oct 26, 20:09
67	Abram Hindle and Curtis Onuczko	Stopping Duplicate Bug Reports before they start with Continuous Querying for Bug Reports	Oct 26, 20:18
68	Denae Ford, Thomas Zimmermann, Christian Bird and Nachiappan Nagappan	Personas in Practice: Adapting Knowledge Worker Actions to Software Engineers	Oct 26, 20:23

1	Izautino P. Oliveira and Renata Souza	Stories for New Products: A Research on the Use of Storytelling in Requirements Identification in Software Engineering	Sep 03, 04:18
2	Hans Jochen Scholl, William Menten-Weil and Timothy S. Carlson	Artifact Evaluation with TEDSrate	Sep 19, 21:10
3	Celal Ziftci and Jim Reardon	Who Broke the Build? Automatically Identifying Changes That Induce Test Failures In Continuous Integration at Google Scale	Sep 27, 20:32
4	Uma Viswanath and Ramya Shyama Palakodati	Measuring Leanness in a lean software organization: A model to gauge the lean implementation using the 3 dimensional approaches of Process, Product, and People	Oct 05, 04:09
5	Sofia Modesto and Miguel Mira Da Silva	Gamification to Increase Scrum Adoption	Oct 14, 15:51
6	Guillermo Rodriguez, Alvaro Soria and Marcelo Campo	A Case-based Reasoning Approach to Reuse Quality-driven Design Alternatives in Service-Oriented Architectures	Oct 14, 19:08
7	Andrea Arcuri	Back After 5 Years In Industry As Software Engineer / Tester: We Need Usable Automated Test Generation	Oct 15, 16:58
8	Christof Ebert and Michael Weyrich	Architecture Evolution for the Internet of Things	Oct 16, 05:59
9	Andrew Ko	A Three-Year Participant Observation of Software Startup Software Evolution	Oct 17, 22:08
10	Torvald Mårtensson, Pär Hammarström and Jan Bosch	Continuous Integration Is Not About Build Systems	Oct 19, 19:43
11	Junjie Wang, Qiang Cui, Song Wang and Qing Wang	Domain Adaptation for Test Report Classification in Crowdsourced Testing	Oct 20, 08:09
12	Fernando Pinciroli, Jose Luis Barros Justo and Raymundo Forradellas	Aspect-Oriented Business Process Modeling Approaches: An assessment of AOP4ST	Oct 20, 13:32
14	Paul L. Li, Andrew J. Ko and Andrew Begel	Expert Non-Software-Engineers's Perspectives on Why Software Engineering Teams Succeed or Fail	Oct 20, 16:10
15	Balbir Barn, Souvik Barat, Tony Clark and Vinay Kulkarni	Reviewing the Software Engineering Nexus of Current Research, Practice, and Future Prospects	Oct 21, 14:00
16	Saad Mubeen, Mikael Sjödin, Harold Lawson, John Lundbäck, Mattias Gålnander and Kurt-Lennart Lundbäck	Provisioning of Predictable Embedded Software in the Vehicle Industry: The Rubus Approach	Oct 22, 19:47
17	Christopher Theisen, Brendan Murphy, Kim Herzig and Laurie Williams	Risk-Based Attack Surface Approximation: How Much Data is Enough?	Oct 23, 03:16
18	Daniel Russo, Paolo Ciancarini, Tommaso Falasconi and Massimo Tomasi	Software Quality Concerns in the Italian Bank Sector: the Emergence of a Meta-Quality Dimension	Oct 23, 19:51
19	Akond Rahman, Asif Partho, David Meder and Laurie Williams	Which Factors Influence Practitioners' Usage of Build Automation Tools?	Oct 24, 00:43
20	Cuiyun Gao, Yichuan Man, Hui Xu, Jieming Zhu, Yangfan Zhou and Micheal R. Lyu	Assist Developers in Mobile Advertising via User Reviews and Case Studies	Oct 24, 17:49
21	Zhitao Hou, Hongyu Zhang, Haidong Zhang and Dongmei Zhang	MetroEyes: A Visual Analytics System for Exploring Multi-Dimensional Data	Oct 25, 00:49
22	S M Sohan, Craig Anslow and Frank Maurer	Automated Example Oriented REST API Documentation at Cisco	Oct 25, 05:34
23	Terese Besker, Antonio Martini and Jan Bosch	Time to Pay Up - Technical Debt From a Software Quality Perspective	Oct 25, 08:53
24	Dale Blue, Orna Raz, Rachel Tzoref-Brill, Paul Wojciak and Marcel Zalmanovici	Novel applications of combinatorial testing in validating software design	Oct 25, 12:40
25	Jingzheng Wu, Shen Liu, Shouling Ji, Mutian Yang, Yanjun Wu, Yongji Wang and Tianyue Luo	Exception Beyond Exception: Crashing Android System by Trapping in "uncaughtException"	Oct 25, 14:28
26	Charles Weir, Awais Rashid and James Noble	Dialectical Security: Challenging the Developers of Mobile and IoT Software	Oct 25, 16:38
27	Samuel Marks and Andrew White	Code-generation driven development	Oct 25, 18:35
28	Daniel Izquierdo-Cortazar, Nelson Sekitoleko, Jesus M. Gonzalez-Barahona and Lars Kurth	Using Metrics to Track Code Review Performance: the Xen Case	Oct 25, 18:45
29	Zakariya Dehlawi and Andrew J. Ko	Predicting the Diffusion of Software Security Activities	Oct 25, 19:02
30	Marcos Kalinowski, Pablo Curty, Aline Paes, Alexandre Ferreira, Rodrigo Spinola, Daniel Méndez Fernández, Michael Felderer and Stefan Wagner	Supporting Defect Causal Analysis in Practice with Cross-Company Data on Causes of Requirements Engineering Problems	Oct 25, 20:07
31	Steven D. Fraser and Dennis Mancl	"No Silver Bullet" Revisited: Panel Session	Oct 25, 20:14
32	Trishank Kuppusamy, Vladimir Diaz and Justin Cappos	Mercury: Bandwidth-Effective Prevention of Rollback Attacks Against Community Repositories	Oct 25, 20:52
33	Mojdeh Golagha, Alexander Pretschner, Dominik Fisch and Roman Nagy	Reducing Failure Analysis Time: An Industrial Evaluation	Oct 25, 21:41
34	Atif Memon, Zebao Gao, Bao Nguyen, Sanjeev Dhanda, Eric Nickell, Rob Siemborski and John Micco	Taming Google-Scale Continuous Testing	Oct 25, 23:03
35	Arun Kalyanasundaram, Judith Bishop and James Herbsleb	Industrial Open Source Project Decisions, Best Practices and Community Engagement: A Case Study at Microsoft	Oct 26, 01:40
36	Bargav Jayaraman, Anurag Dwarakanath, Breno D. Cruz and Collin McMillan	A Deep Learning approach for the Multi-lingual identification of Vagueness	Oct 26, 02:19
37	Jingzheng Wu, Sizhe Zhao, Shouling Ji, Mutian Yang, Tianyue Luo, Yanjun Wu and Yongji Wang	MAD-API: Detection, Correction and Explanation of API Misuses in Android Applications	Oct 26, 03:13
38	Remo Eckert, Sathya Kay Meyer and Matthias Stuermer	Capability Maturity Model of Inner Source Implementation	Oct 26, 05:17
39	Khaled Alnawasreh, Patrizio Pelliccione, Zhenxiao Hao, Mårten Rånge and Antonia Bertolino	Online Robustness Testing of Distributed Embedded Systems: an Industrial Approach	Oct 26, 06:56
40	Yulai Zhou, Patrizio Pelliccione, Johan Haraldsson and Mafjiul Islam	Improving Robustness of AUTOSAR Software Components with Design by Contract: A study within Volvo AB	Oct 26, 07:03
41	Henrik Edholm, Mikaela Lidström, Jan-Philipp Steghöfer and Håkan Burden	Crunch Time: The Reasons and Effects of Unpaid Overtime in the Games Industry	Oct 26, 07:26
42	Jacob Krüger, Andy Kenner, Christopher Kruczek and Thomas Leich	Modularizing Conditional Compilation: An Automatic Minimal-Invasive Approach	Oct 26, 07:42
43	Katja Kevic, Brendan Murphy, Laurie Williams and Jennifer		
44	Jakub Misek and Filip Zavoral		
45	Ivica Crnkovic and Anna Börjesson Sandberg		
46	Eero Laukkanen, Maria Paasivaara, Juha Itkonen, Casper L		
47	Pete Rotella, Cody Peeples and Mark-David McLaughlin		
48	Mohamad Kassab, Jooyoung Lee, Manuel Mazzara, Giancar		
49	Kumar Abhinav, Alpana Dubey, Sakshi Jain, Gurdeep Virdi,		
50	Lingfeng Bao, Zhenchang Xing, Xin Xia, David Lo and Shar		
51	Padmalata Nistala and Kesav Vithal Nori		
52	Christoph Seidl, Thorsten Berger, Christoph Elsner and Kla		
53	Jayati Deshmukh, Annervaz K M, Sanjay Podder, Shubhasl		
54	Ayse Tosun, Ozgur Turkgulu, Dogan Razon, Hamza Yusuf A		
55	Raffaele Ciriello, Alexander Richter and Gerhard Schwabe		
56	Rebekka Wohlrab, Patrizio Pelliccione, Eric Knauss and Mat		
57	Yingxia Wei, Rui Wang and Yu Jiang		
58	Kee-Choon Kwon, Jang-Soo Lee and Eunkyoung Jee		
59	Ulrik Eklund and Christian Berger		
60	Jürgen Cito, Fábio Oliveira, Philipp Leitner, Priya Nagpurka		
61	Francesco Sorrentino		
62	Hennie Huijgens, Leandro Minku, Chris Lokan and Arie var		
63	Ma. Laura Caliusco, Emiliano Reynares, Néstor Reynoso, J		
64	Simon Harrer, Matthias Geiger, Vincenzo Ferme, Cesare Pa		
65	Helena Holmström Olsson and Jan Bosch		
66	Franz Zieris and Lutz Prechelt		
67	Abram Hindle and Curtis Onuczko		
68	Denae Ford, Thomas Zimmermann, Christian Bird and Nac		

69	Parastoo Mohagheghi and Magne Jørgensen	Success factors, challenges and lessons learned: An empirical study of software projects in the public sector	Oct 26, 20:34
70	Maximiliano Agustin Mascheroni and Emanuel Irrazábal	A Framework for Continuous Testing of RESTful Web Services	Oct 26, 20:39
71	Alessia Knauss, Jan Schroeder, Christian Berger and Henrik Eriksson	Paving the Roadway for Safety of Automated Vehicles: An Empirical Study on Testing Challenges	Oct 26, 20:44
72	Luis Marino	An scalable and adaptable maturity model for product development teams	Oct 26, 20:49
73	Lukas Alperowitz, Andrea Marie Weintraud, Stefan Christoph Kofler and Bernd Bruegge	Continuous Prototyping: Unified Application Delivery from Early Design to Code	Oct 26, 20:59
74	Carmelo Ardito, Maria Teresa Baldassarre, Danilo Caivano and Rosa Lanzilotti	Towards Converging Agile to Human Centered Design: an action research study	Oct 26, 21:17
75	Jan Schroeder, Christian Berger, Alessia Knauss, Harri Preenja, Mohammad Ali, Miroslaw Staron and Thomas Herpel	Prediction of Software Model Growth in Practice	Oct 26, 21:46
76	Carmine Vassallo, Fiorella Zampetti, Daniele Romano, Sebastiano Panichella, Massimiliano Di Penta and Andy Zaidman	What Types of Build Failures Stop Continuous Delivery? An Empirical Study at ING NL	Oct 26, 21:48
77	Man Zhang, Tao Yue, Shaukat Ali, Bran Selic, Oscar Okariz, Roland Norgren, Karmele Intxausti and Santiago Charramendieta	Specifying Uncertainty in Use Case Models in Industrial Settings	Oct 26, 21:57
78	Litsietsi Montsi	AN ADAPTIVE PROCESS FRAMEWORK FOR ARCHITECTING REAL-TIME BIG DATA SYSTEMS	Oct 26, 22:00
79	Rob van der Valk, Patrizio Pelliccione, Patricia Lago, Rogardt Heldal, Eric Knauss and Jacob Juul	Continuous Delivery in the Automotive Ecosystem: Transparency Trade-offs in Software Value-Chains	Oct 26, 22:01
80	Razieh Saremi, Ye Yang, Gunther Ruhe and Dave Messinger	Leveraging Crowdsourcing For Team Elasticity: An Empirical Evaluation at TopCoder	Oct 26, 22:41
81	Tse-Hsun Chen, Mark D. Syer, Weiyi Shang, Zhen Ming Jiang, Ahmed E. Hassan, Mohamed Nasser and Parminder Flora	Analytics-Driven Load Testing: An Industrial Experience Report on Load Testing of Large-Scale Systems	Oct 26, 23:31
82	Mohammad Moein Almasi, Hadi Hemmati, Gordon Fraser, Andrea Arcuri and Janis Benefelds	An Industrial Evaluation of Unit Test Generation: Finding Real Faults in a Financial Application	Oct 27, 00:13
83	Xiao Liu, Yufei Jiang and Dinghao Wu	A Lightweight Verification Framework for Regular Expressions	Oct 27, 00:13
84	Truong Ho-Quang, Regina Hebig, Gregorio Robles, Michel R.V. Chaudron and Miguel Angel Fernandez	Practices and Perceptions of UML Use in Open Source Projects	Oct 27, 00:32
85	Eliana C. M. Ishikawa, Luan B. Beluzzo, Simone N. Matos and Guataçara Dos S. Junior	Collabora: A collaborative architecture for evaluating individuals participation during the development of activities	Oct 27, 00:46
87	George Marsicano, Fabio Silva and Diana Valença	Daily Meetings in Agile Teams: A multiple case study	Oct 27, 02:30
88	Michael Lescisin, Qusay Mahmoud and Anca Cioraca	SFCI: A Tool for Security Focused Continuous Integration	Oct 27, 02:31
89	Zhi Zhang, Na Robby, John Hatcliff, Yannick Moy and Pierre Courtieu	Focused Certification of an Industrial Compilation and Static Verification Toolchain	Oct 27, 03:17
90	Josiane Kroll, Shai Friboim and Hadi Hemmati	An Empirical Study of Search-Based Task Scheduling in Global Software Development	Oct 27, 03:52
91	Darya Melicher	Pair Programming: An Experience Report	Oct 27, 04:32
92	Shreya Kumar and Charles Wallace	Patterns of Identity and Interaction in an Evolving Agile Workplace	Oct 27, 04:37
93	Yu Jiang, Han Liu, Houbing Song, Xun Jiao, Yue Gao, Ming Gu and Jiaguang Sun	A Lightweight Model-Driven Design Environment for the Engineering Practice of Train Controller	Oct 27, 05:44
94	Toni James, Matthias Galster, Kelly Blincoe and Grant Miller	How do Female and Male Software Professionals Work with Others? Insights from the Trenches	Oct 27, 05:50
95	Cornel Barna, Marin Litoiu, Marios Fokaefs, Mark Shtern and Joe Wigglesworth	On Developing Linear Quadratic Performance Controllers for Cloud Applications	Oct 27, 06:21
96	Wing Lam, Siwakorn Srisakaokul, Blake Bassett, Peyman Mahdian, Tao Xie, Pratap Lakshman and Jonathan de Halleux	A Characteristic Study of Parameterized Unit Tests in .NET Open Source Projects	Oct 27, 07:29
97	Haibing Zheng, Dengfeng Li, Xia Zeng, Beihai Liang, Wujie Zheng, Yuetang Deng, Wing Lam, Wei Yang and Tao Xie	Automated Test Input Generation for Android: Towards Getting There in an Industrial Case	Oct 27, 07:58
98	Jorge Mendes, Jácome Cunha, Francisco Duarte, Gregor Engels, João Saraiva and Stefan Sauer	Systematic Spreadsheet Construction Processes	Oct 27, 09:54
99	Hajer Ayed, Benoit Vanderose and Naji Habra	Agile Cultural Challenges in Europe and Asia: Insights from Practitioners	Oct 27, 09:55
100	Rachit Singhal, Xi Zheng, Min Fu, Rahul Ramdas, Umaima Akram Khan, Xiao Liu, Armin Anvari, Babanpreet Kaur, Akash Chhetri and Liming Zhu	Using IaaS or Not?	Oct 27, 10:15
101	Michael de Jong, Arie van Deursen and Anthony Cleve	Zero-Downtime SQL Database Schema Evolution for Continuous Deployment	Oct 27, 10:39
102	Yingnong Dang, Dongmei Zhang, Song Ge, Ray Huang, Chengyun Chu and Tao Xie	Transferring Code-Clone Detection and Analysis to Practice	Oct 27, 10:41
103	Ken Power and Kieran Conboy	Six Key Metrics for Understanding Flow and Impediments in Software Engineering	Oct 27, 10:50
104	Alaaeddin Swidan, Felienne Hermans and Efthimia Aivaloglou	What Factors Affect Spreadsheet Performance? An Analysis of 4 Datasets	Oct 27, 11:04
105	Roberto Oliveira, Leonardo Sousa, Rafael Mello, Natasha Valentim, Adriana Lopes, Tayana Conte, Alessandro Garcia, Edson Oliveira and Carlos Lucena	Collaborative Identification of Code Smells: A Multi-case Study	Oct 27, 11:13
107	Yan Zhang and Lejian Liao	Group Behavior Aggregator for Realtime Simulation of Cyberspace Situation Awareness	Oct 27, 12:02
108	Monika Solanki	Exploiting ontologies for harnessing, aligning and integrating software design requirements and implementation issues: The PoolParty Case Study	Oct 28, 03:41
109	Anagha Jamthe, Pranjal Ambardekar and Mandar Chincholkar	Predicting Defect Resolution Time using Cosine Similarity	Oct 31, 15:24

ICSE 2017 SEIP (PC member)

Submissions | Reviews | Status | Bidding | Events | ICSE 2017 SEIP | Premium | News | EasyChair

Reviews of Submissions Assigned to Me

Review submission or updates have now been disabled. Please contact chairs if you believe they should be enabled.

#	Submission	Details	paper	Show reviews	Contact subreviewer
2	Hans Jochen Scholl, William Menten-Weil and Timothy S. Carlson. *Artifact Evaluation with TEDSrate*				
35	Arun Kalyanasundaram, Judith Bishop and James Herbsleb. *Industrial Open Source Project Decisions, Best Practices and Community Engagement: A Case Study at Microsoft*				
95	Cornel Barna, Marin Litoiu, Marios Fokaefs, Mark Shtern and Joe Wigglesworth. *On Developing Linear Quadratic Performance Controllers for Cloud Applications*				
102	Yingnong Dang, Dongmei Zhang, Song Ge, Ray Huang, Chengyun Chu and Tao Xie. *Transferring Code-Clone Detection and Analysis to Practice*				

Reviews and Comments

PC member:	Guenther Ruhe
Reviewer:	Didar Al Alam <didar522@gmail.com>
Time:	Nov 27, 23:35
Overall Evaluation:	**2:** (accept - I support acceptance)
Potential Impact to Industry:	**4:** (High - Work impacting industry)
Real World Focus:	**4:** (Excellent - 100% real world focus)
Reviewer's confidence:	**5:** (expert)

Review:

1. This paper presents multiple case stu
making and best practices for open sou

2. Points for the paper:
• Context of the paper and purpose of t
• Of strong interest for (large) organiza
• Well organized background study. Aut
explained and the whole paper is struct
• Comparison of projects from two diffe
aware of generalizing results across pro
• The paper revealed findings related to
firm to open source projects. The paper
• Findings and practices are supported

Points against the paper:
• Overall, the paper would benefit from
• Archival data from Github is mentione
• The abstract should present a summa
• Decision points and trade-off decision
alternatives? Utility function(s)? Who m
• For projects with multiple repositories
discuss under "Threats to validity" how
• It seems, the data is collected from G
change over time as well.
• For some findings, some of the examp
present the percentage of projects follo
• Each section consists of a set of findir
hard to identify the key message. Read
• In section iv-c, the authors discussed
cons of both approaches.
• In section iv authors discussed decisi
conditions.
• Authors identified a number of metric
• Measures like download counts, numb
is identified when the data is filtered ba
get affected in case of considering one
• To measure popularity of a project, a
popularity. Or should we consider them
• Some of the best practices are applica
This information should be added for ea
• Key findings and messages of the pap
discussion, example and practices. It is
• Minor:
o DEE is first used reader does not kno
o Justifications for not recording intervi
on a decision.

4. Suggested improvements:
- Improve messaging by highlighting sc
- Provide more concrete data
- Address issues listed under 3.

Confidential remarks for the program committee: Potentially interesting paper if some mc

PC member:	Christof Ebert
Time:	Dec 06, 14:29
Overall Evaluation:	**-1:** (weak reject - reject, but could ac
Potential Impact to Industry:	**2:** (Low - Not expected much impact)
Real World Focus:	**3:** (Good - Enough real world focus)
Reviewer's confidence:	**4:** (high)

Review:

1) The authors present a case study a
promised benefits from open source. (
trade-offs and best practices.

2) Points for the paper
- The paper is relevant as it presents i
- The research approach using intervi
- Many observations are discussed ste

3) Points against the paper
- While the paper is strong in digging
- Topics such as security etc. are treat
code more secure vs. the reasonable
- I also find it interesting to read that

- treat above findings in (3)
- change presentation to have three parts, namely (1) study se
and best practices to be used in other companies, e.g. how to l

Confidential remarks for the program committee: Good topic, but weak presentation.

R

PC member:	Elmar Jurgens
Time:	Dec 22, 10:02
Overall Evaluation:	**-2:** (reject - I support rejection)
Potential Impact to Industry:	**2:** (Low - Not expected much impact)
Real World Focus:	**3:** (Good - Enough real world focus)
Reviewer's confidence:	**4:** (high)

Review:

The paper presents a study based on interviews and data extracted from
research division and from a product division at Microsoft. Most of the de

My main concern with the paper is the lack of a strong problem statemer
community in terms of success of external engagement and later on intr
does a good job at iterating different motives for open-sourcing code, wh
reader which specific goal such a general measurement as DDE has and
measure behaves differently, depending on project life cycle phase or rel
unclear, if the behavior of DEE is problematic or not.
In a nutshell, the paper deals too much with what is easy to measure, nc

Points in favor:
- Our community would benefit from a better understanding of the facto
- Direct developer involvement

Points against:
- Most of the projects are from the research division. As the paper states
research projects. For the in practice track, this is not a good fit to me.
- There is very little information that I as a practitioner who faces these
- The introduction reads too much like a related work / survey and too li
to me, the study and DEE are hard to interpret.

Minor points:
- On page 1, the acronym DEE is used before it is defined on page 2. Thi
- In page 2, the paper states that the projects were "carefully selected".
unclear to me why the fact that "project owners [that] are more concern
this study. If they are "more" concerned, they are probably not represen
- Why would a deeper knowledge of user demographics help developers
conversations with the user base to be a much better means to understa
- On p5, "external contributions" explicitly contain contributions that "no
number of commits authored by external developers" and thus probably

Further points
I was personally very excited to start to read this paper, since the title hi
that the majority of the paper is about the research, not the product divi

Confidential remarks for the program committee: To me, this is a research track paper, not a SEIP paper. For the most par
something. For me as a practitioner, the paper lacks a specific problem a

Comment 1

PC member: Natalia Juristo

Comment:
Hi folks,

We are starting the discussion phase.

Guenther, since you are the one with a different view here, could you ple
and tell us whether they modify your view?

Thanks,

Natalia

Time: Dec 28, 17:11

Comment 2

PC member: Guenther Ruhe

Comment:
Yes, they do. My former concerns already have been that the paper does
an industrial perspective. I agree with the related comment made by Rev
attractive TOPIC and the good presentation.

I modified my evaluation.

Thanks,
Guenther

Time: Dec 28, 19:17

R

PC member:	Guenther Ruhe
Reviewer:	Didar Al Alam <didar522@gmail.com>
Time:	Dec 28, 19:18
Overall Evaluation:	**-1:** (weak reject - reject, but could accept)
Potential Impact to Industry:	**3:** (Enough - Work could impact industry)
Real World Focus:	**3:** (Good - Enough real world focus)
Reviewer's	

Thanks,

Natalia

Time: Dec 28, 17:11

Comment 2

PC member: Guenther Ruhe

Comment:
Yes, they do. My former concerns already have been that the paper does not offer too much on tangible take-ways from an industrial perspective. I agree with the related comment made by Reviewer 3. The main reason I scored 2 was the attractive TOPIC and the good presentation.

I modified my evaluation.

Thanks,
Guenther

Time: Dec 28, 19:17

Review 1

PC member:	Guenther Ruhe
Reviewer:	Didar Al Alam <didar522@gmail.com>
Time:	Dec 28, 19:18
Overall Evaluation:	**-1:** (weak reject - reject, but could accept)
Potential Impact to Industry:	**3:** (Enough - Work could impact industry)
Real World Focus:	**3:** (Good - Enough real world focus)
Reviewer's confidence:	**5:** (expert)

Review

Review:

1. This paper presents multiple case study at Microsoft. It examines the challenges of open sourcing industrial projects. The authors analyzed decision making and best practices for open sourcing projects. They also compare challenges and practices from projects under different divisions.

2. Points for the paper:
• Context of the paper and purpose of the study are well explained.
• Of strong interest for (large) organizations following hybrid closed and open source development.
• Well organized background study. Authors identified existing gaps in literature and mapped with contributions of the paper. Goal of the study is well explained and the whole paper is structured around the goals.
• Comparison of projects from two different divisions. It helps to understand the commonalities and differences between divisions. It also makes reader aware of generalizing results across projects.
• The paper revealed findings related to assessing DEE in industrial OSS projects. Extracted best practices will provide a road map for any industrial software firm to open source projects. The paper does not provide decision support, but helps to understand the decision process.
• Findings and practices are supported by example projects, interview statements.

Points against the paper:
• Overall, the paper would benefit from some higher degree of specific. For example, when talking about trade-offs, what are the dimensions of it?
• Archival data from Github is mentioned in the abstract. Wish there would be more details on that process and the data.
• The abstract should present a summary of findings. Reader does not have any clue of actual findings until they reach the end of the introduction.
• Decision points and trade-off decisions are mentioned several time in the paper. However, the decision scenario is not made explicit. What are the decision alternatives? Utility function(s)? Who makes the decision? Based on what?
• For projects with multiple repositories, authors considered only one (to "avoid complications"). This looks like a strong simplification. The author should discuss under "Threats to validity" how this decision impacts the study.
• It seems, the data is collected from GitHub over time. Why forks, stars and watchers are considered as static data instead of temporal. These values change over time as well.
• For some findings, some of the examples presented show completely different behaviour. Along with reporting these behaviours, authors should also present the percentage of projects follow a certain behaviour and which is applicable under what type of condition.
• Each section consists of a set of findings or practices. Authors should list the key findings in each section. With all the examples, stories and discussion, it is hard to identify the key message. Reader may get lost and miss important information easily.
• In section iv-c, the authors discussed converting industrial projects to OSS being done early vs late. For a fair comparison, authors should discuss pros and cons of both approaches.
• In section iv authors discussed decision making under different conditions. It would be good to suggest a certain option that works better under specific conditions.
• Authors identified a number of metrics (from practice) for DEE calculation. Any recommendation on their application?
• Measures like download counts, number of page views in GitHub, and number of logins give an estimate of the user activity. However, size of the user base is identified when the data is filtered based on IP (Internet Protocol) addresses and other unique identifiers. Author should explain, how DEE calculations will get affected in case of considering one vs the other.
• To measure popularity of a project, a number of metrics are presented. How all these metrics will come together to calculate an overall value of the popularity. Or should we consider them independently?
• Some of the best practices are applicable for both industrial and non-industrial OSS projects. Some are only applicable to industrial open source projects. This information should be added for each practice.
• Key findings and messages of the paper should be summarized or visualized. The study is comprehensive. A large list of findings is presented with discussion, example and practices. It is hard for the reader to keep track of the content or identify all the findings and their applicability.
• Minor:
o DEE is first used reader does not know what DEE stands for. It was first defined one section later.
o Justifications for not recording interview is not clear. Recording has its positive arguments as well. Author should discuss both pros and cons before focusing on a decision.

4. Suggested improvements:
- Improve messaging by highlighting some concrete and operational findings.
- Provide more concrete data
- Address issues listed under 3.

Confidential remarks for the program committee: Potentially interesting paper if some more details and concrete numbers and findings would be synthesized from all the writing.

Comment 3

PC member: Natalia Juristo

Comment:
Ok. We reject this paper then.

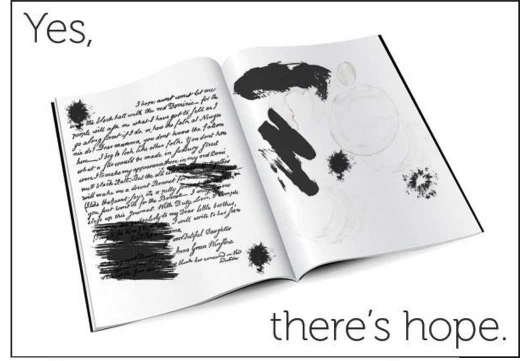
Thanks!

Natalia

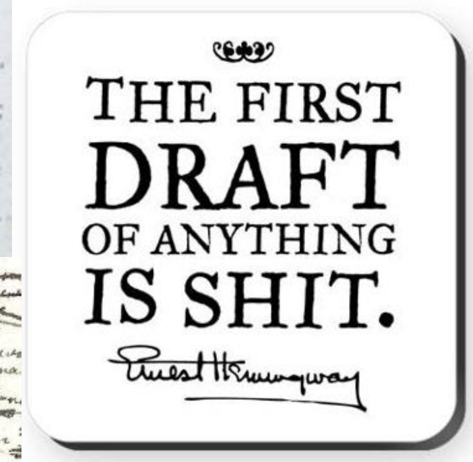
Für den Reviewer schreiben

- Problem-Statement und Contribution herausarbeiten
- Etablierte Gliederung verwenden: https://thesisguide.org/2014/10/13/thesis-architecture/
- Text einfach lesbar machen. Das ist hart und anstrengend. Aber planbar und erlernbar, keine Talentfrage.


SHITTY FIRST DRAFTS
ALL GOOD WRITERS START WITH SHITTY FIRST DRAFTS. THIS IS HOW THEY END UP WITH GOOD SECOND DRAFTS AND TERRIFIC THIRD DRAFTS
ANNE LAMOTT


Yes,
there's hope.


First drafts don't have to be perfect. They just have to be written.

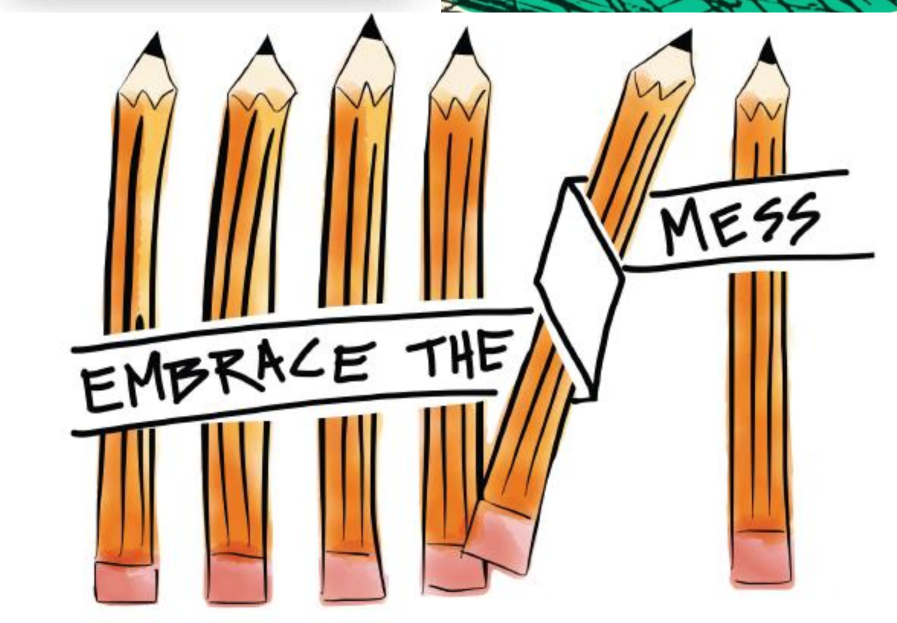

THE FIRST DRAFT OF ANYTHING IS SHIT.
Ernest Hemingway


SHITTY FIRST DRAFTS


Shitty First Drafts


SHITTY FIRST DRAFTS

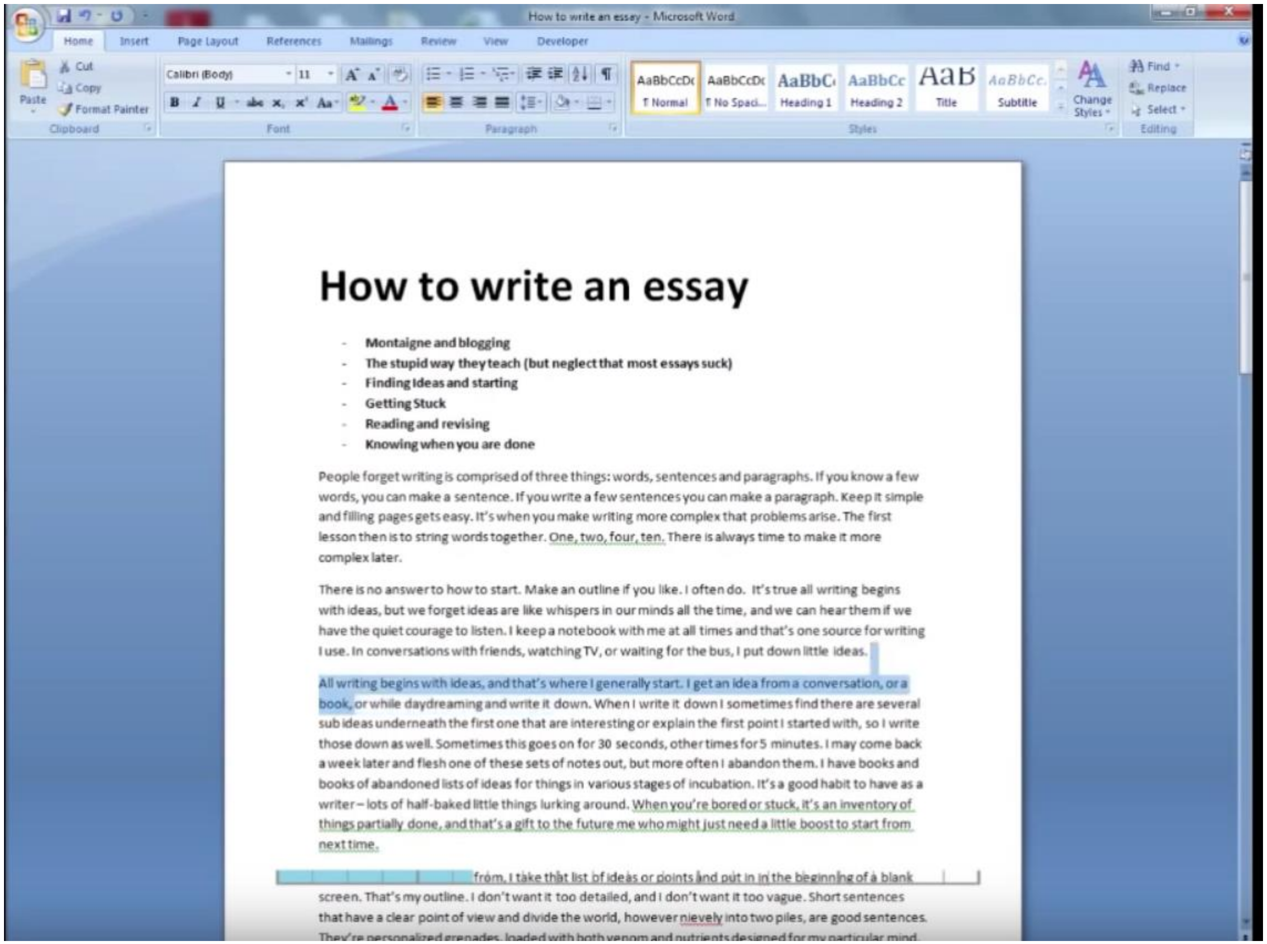

THE FIRST DRAFT OF anything IS SHIT
- ERNEST HEMINGWAY
LENGSTORF.COM


EMBRACE THE MESS

Was mir am meisten bringt

- Schreibzeit blocken
- Outline zuerst
- Schreiben und verbessern voneinander trennen
- Kompletten Absatz schreiben, bevor ich irgendwas verbessere
- Text „abkühlen lassen“ und dann *nochmal* Korrekturlesen. Bei mir am besten mind. 1 Tag später.
- Es gibt nicht die eine „richtige“ Art zu schreiben, die für alle gleich gut funktioniert.

Scott Berkun: Essay-Schreiben im Zeitraffer



Youtube: http://youtu.be/BNDEDWwZyKM

English Writing Center

- Kostenlose 45-minütige Einzelsessions mit englischsprachigen Muttersprachlern

- Beratung zum Schreiben englischsprachiger Texte
  - GR, Abschlussarbeit, Hausaufgabe, Lebenslauf etc.
  - auch wenn der Text noch nicht fertig ist

www.tum.de/writing-center

Professioneller Lektor

How to Spend Your Writing Time Well?

~~Every~~ A thesis ~~comprises~~ is made up of several chapters, ~~such as~~ including an introduction, definitions, related work, proposed solution, and conclusion. You must decide how much time (and pages) to spend on each of them. I call this *writing resource allocation*.

If this step is done poorly, authors will waste a large part of their writing time on chapters that are not central to their thesis; for example, producing bloated definitions or~~,~~ ~~a~~ myriad ~~of~~ irrelevant technical details ~~or other waste~~. Not only does this distract readers, it also ~~inevitably~~ robs authors of the time they need to write their central chapters carefully. Therefore, p~~P~~oor writing resource allocation is ~~thus an effective~~ recipe ~~to write~~ for a bad thesis.

So h~~H~~ow ~~to~~ do you do ~~it~~ this step well? For me, writing resource allocation is a lot like allocating plate space when eating at a ~~large~~ buffet. ~~For b~~Both problems have ~~, there is~~ a similar solution strategy that is intuitive, widely applied, and reliable ~~to produce poor results~~.~~:~~

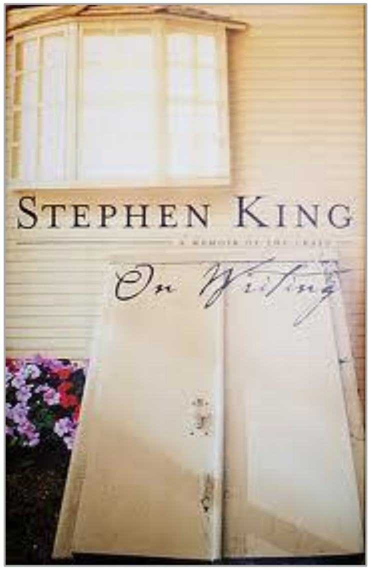
When I approach a buffet, I feel an urge to fill my empty plate. To get quick results, I put
pasta, and ~~P~~potatoes.
I get ~~more picky~~pickier
~~f P~~pork, some ~~L~~lamb
about to leave with a full
d be a fool to leave it out! I
ce for the ~~S~~scallops and
~~everybody~~ anyone that not

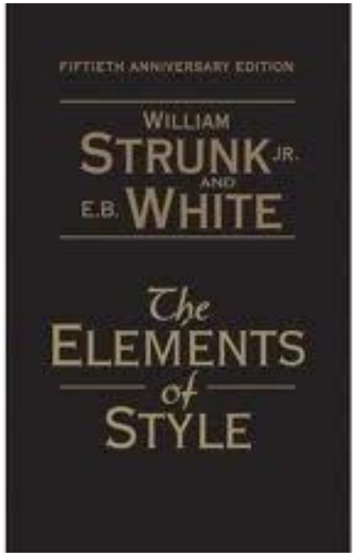
greedy allocation strategy~~.~~
well.~~:~~

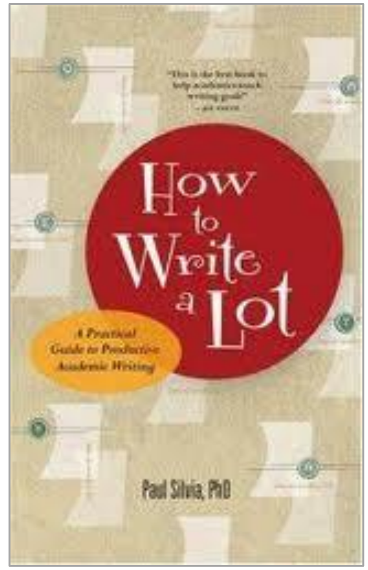
pty pages. To get quick
s is often the introduction
tions. ~~Its~~ It can be
~~, in~~ adding three pages
just heard a course about,
ay. So it should be in the

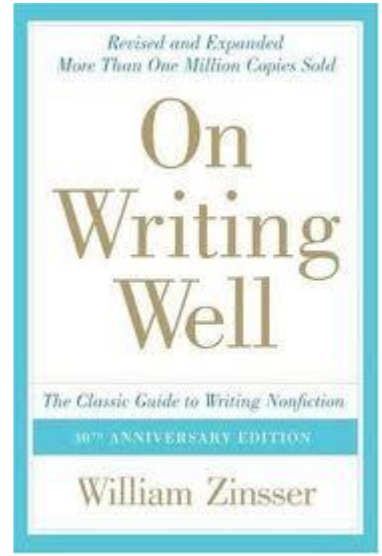
dy strategy when it
comes to the important chapters. ~~You are hard pressed for time when you~~ When it comes time to write the contribution and evaluation, or whichever chapters matter most, you find yourself pressed for time. To make things worse, the really interesting ideas often come only after you have been immersed into a topic for a while; that is, ~~.~~ ~~This is~~ at the end of your writing time. Just ~~as~~ like with ~~for~~ the ~~really tasting~~ tastiest buffet items, there is no space left. They end up ~~are~~ either being left out~~,~~ or they

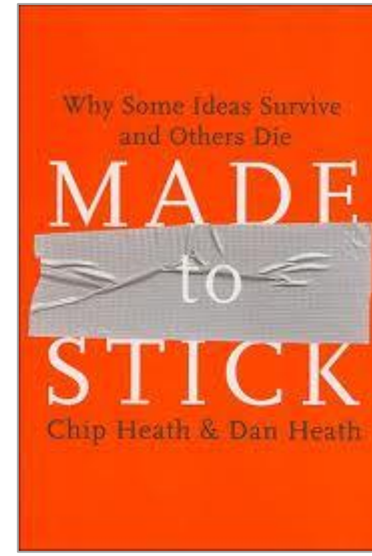
James Morrison
jmedits@gmail.com

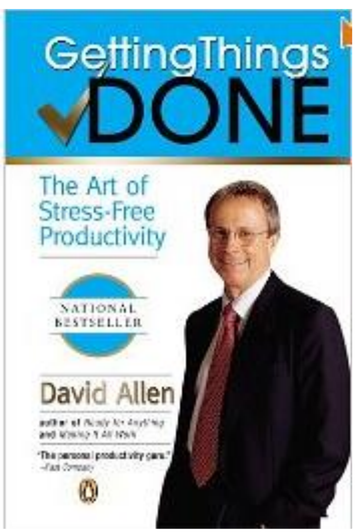

STEPHEN KING
On Writing


FIFTIETH ANNIVERSARY EDITION
WILLIAM STRUNK JR. AND E.B. WHITE
The ELEMENTS of STYLE

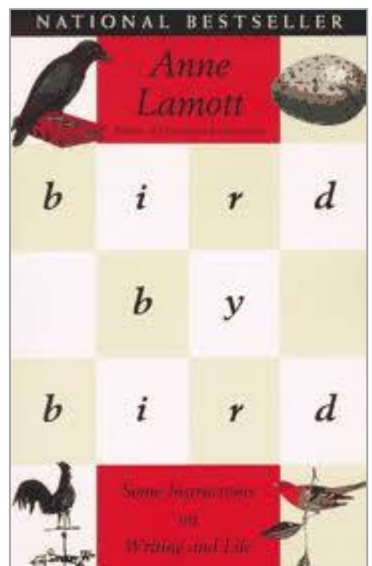

How to Write a Lot
A Practical Guide to Productive Academic Writing
Paul Silvia, PhD

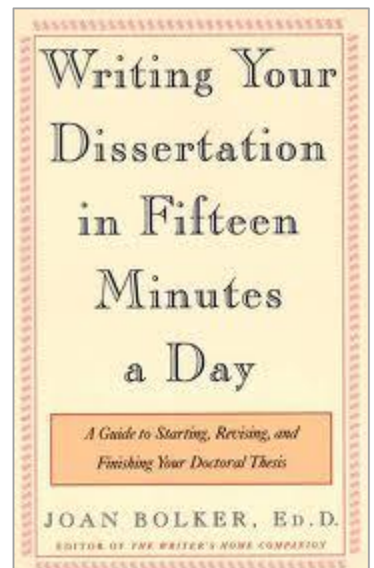

Revised and Expanded
More Than One Million Copies Sold
On Writing Well
The Classic Guide to Writing Nonfiction
30TH ANNIVERSARY EDITION
William Zinsser

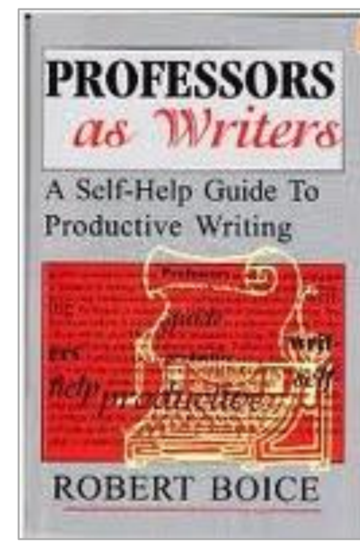

Why Some Ideas Survive and Others Die
MADE to STICK
Chip Heath & Dan Heath

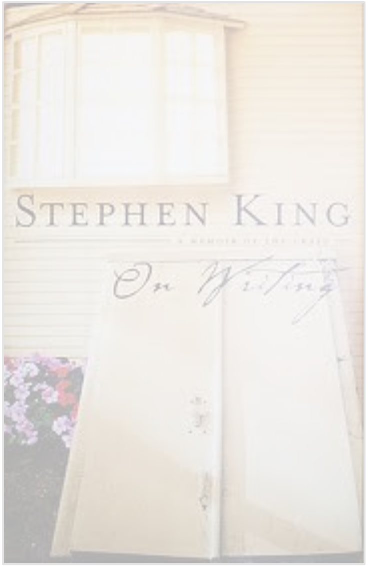

Getting Things DONE
The Art of Stress-Free Productivity
NATIONAL BESTSELLER
David Allen

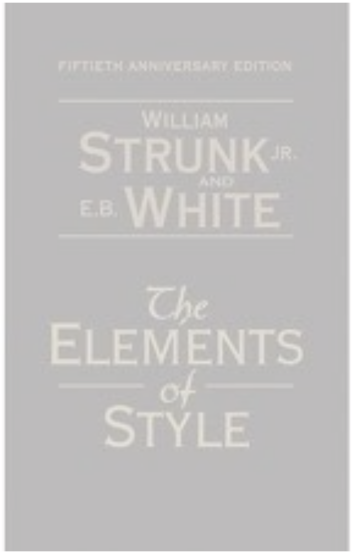

THE TRUTH ABOUT
GETTING THINGS DONE
"You need to believe it in order to achieve it..."
MARK FRITZ

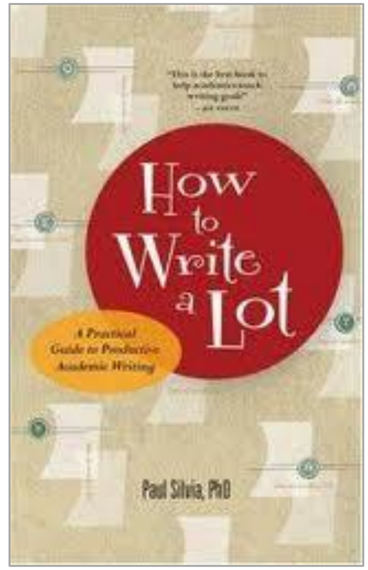

NATIONAL BESTSELLER
Anne Lamott
bird by bird


Writing Your Dissertation in Fifteen Minutes a Day
A Guide to Starting, Revising, and Finishing Your Doctoral Thesis
JOAN BOLKER, Ed.D.


PROFESSORS as Writers
A Self-Help Guide To Productive Writing
ROBERT BOICE

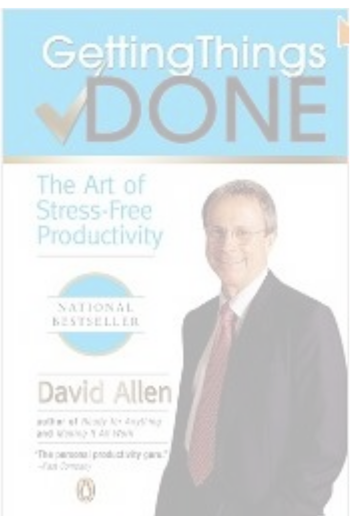

Stephen King
On Writing


FIFTIETH ANNIVERSARY EDITION
William Strunk Jr. and E.B. White
The Elements of Style


How to Write a Lot
A Practical Guide to Productive Academic Writing
Paul Silvia, PhD

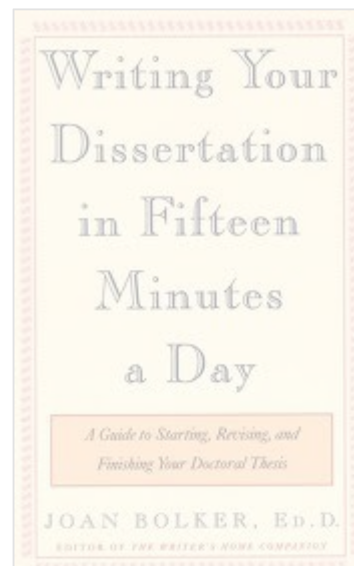
Revised and Expanded
More Than One Million Copies Sold
On Writing Well
The Classic Guide to Writing Nonfiction
30th ANNIVERSARY EDITION
William Zinsser

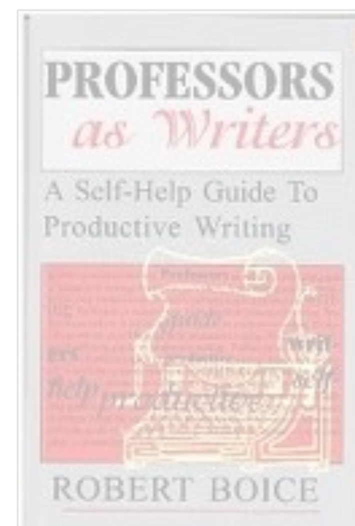

Why Some Ideas Survive and Others Die
MADE to STICK
Chip Heath & Dan Heath

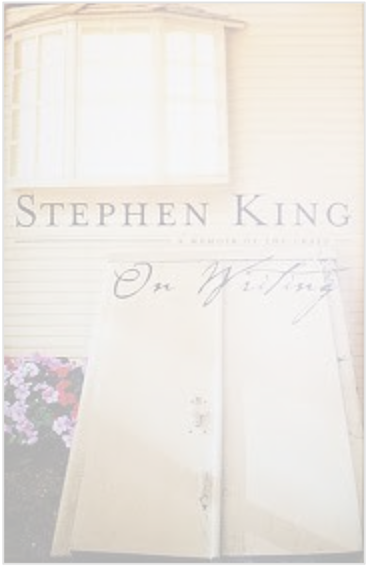

Getting Things DONE
The Art of Stress-Free Productivity
NATIONAL BESTSELLER
David Allen

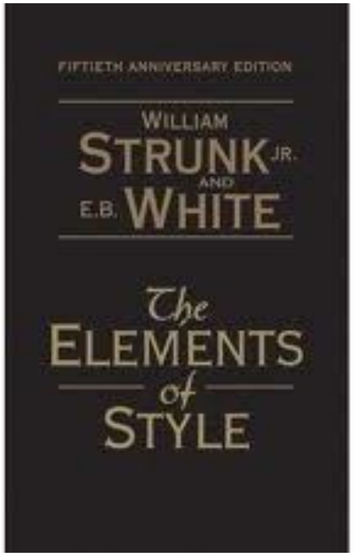

THE TRUTH ABOUT
GETTING THINGS DONE
"You need to believe it in order to achieve it..."
MARK FRITZ

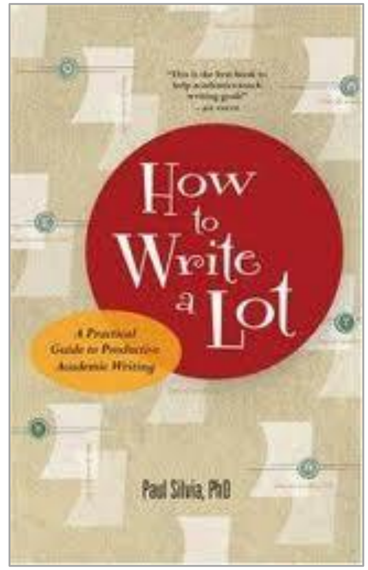

NATIONAL BESTSELLER
Anne Lamott
b i r d
b y
b i r d

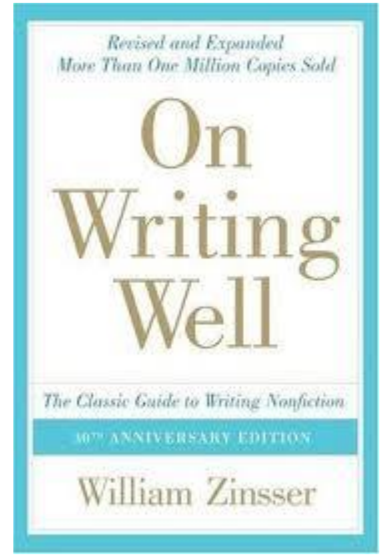

Writing Your Dissertation in Fifteen Minutes a Day
A Guide to Starting, Revising, and Finishing Your Doctoral Thesis
JOAN BOLKER, Ed.D.


PROFESSORS as Writers
A Self-Help Guide To Productive Writing
ROBERT BOICE

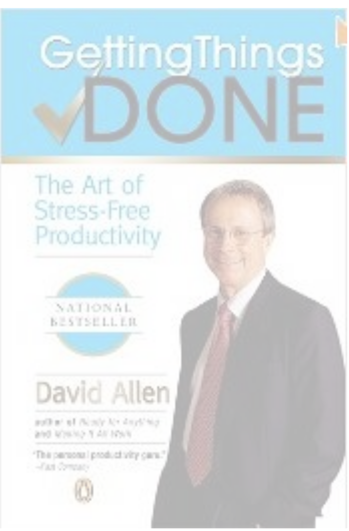

STEPHEN KING
On Writing


FIFTIETH ANNIVERSARY EDITION
WILLIAM STRUNK JR. AND E.B. WHITE
The ELEMENTS of STYLE


How to Write a Lot
A Practical Guide to Productive Academic Writing
Paul Silvia, PhD

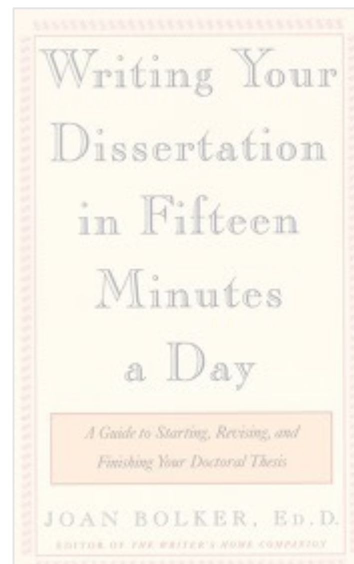

Revised and Expanded
More Than One Million Copies Sold
On Writing Well
The Classic Guide to Writing Nonfiction
30TH ANNIVERSARY EDITION
William Zinsser

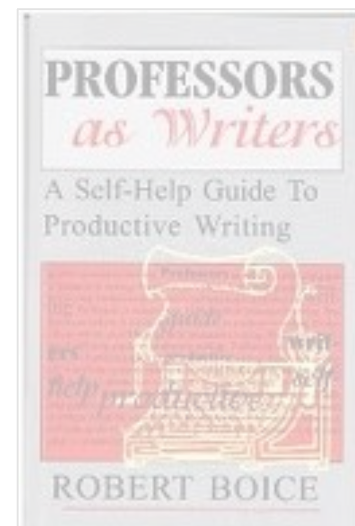

Why Some Ideas Survive and Others Die
MADE to STICK
Chip Heath & Dan Heath


Getting Things DONE
The Art of Stress-Free Productivity
NATIONAL BESTSELLER
David Allen


THE TRUTH ABOUT
GETTING THINGS DONE
"You need to believe it in order to achieve it..."
MARK FRITZ


NATIONAL BESTSELLER
Anne Lamott
b i r d
b y
b i r d


Writing Your Dissertation in Fifteen Minutes a Day
A Guide to Starting, Revising, and Finishing Your Doctoral Thesis
JOAN BOLKER, Ed.D.


PROFESSORS as Writers
A Self-Help Guide To Productive Writing
ROBERT BOICE

1

Learning to Rank Extract Method Refactoring Suggestions for Long Methods

Roman Haas[1] and Benjamin Hummel[2]

[1] Technical University of Munich, Lichtenbergstr. 8, Garching, Germany roman.haas@tum.de

[2] CQSE GmbH, Lichtenbergstr. 8, Garching, Germany hummel@cqse.eu

Summary. Extract method refactoring is a common way to shorten long methods in software development. It improves code readability, reduces complexity, and is one of the most frequently used refactorings. Nevertheless, sometimes developers refrain from applying it because identifying an appropriate set of statements that can be extracted into a new method is error-prone and time-consuming.

In a previous work, we presented a method that could be used to automatically derive extract method refactoring suggestions for long Java methods, that generated useful suggestions for developers. The approach relies on a scoring function that ranks all valid refactoring possibilities (that is, all *candidates*) to identify suitable candidates for an extract method refactoring that could be suggested to developers. Even though the evaluation has shown that the suggestions are useful for developers, there is a lack of understanding of the scoring function. In this paper, we present research on the single scoring features, and their importance for the ranking capability. In addition, we evaluate the ranking capability of the suggested scoring function, and derive a better and less complex one using learning to rank techniques.

Key words: Learning to Rank, Refactoring Suggestion, Extract Method Refactoring, Long Method

1.1 Introduction

A long method is a bad smell in software systems [2], and makes code harder to read, understand and test. A straight-forward way of shortening long methods is to extract parts of them into a new method. This procedure is called 'extract method refactoring', and is the most often used refactoring in practice [20].

The process of extracting a method can be partially automated by using modern development environments, such as Eclipse IDE or IntelliJ IDEA, that can put a set of extractable statements into a new method. However, developers still need to find this set of statements by themselves, which takes

rienced developers sometimes select statements that cannot be extracted (for example, when several output parameters are required, but are not supported by the programming language) [12].

The refactoring process can be improved by suggesting to developers which statements could be extracted into a new method. The literature presents several approaches that can be used to find extract method refactorings. In a previous work, we suggested a method that could be used to automatically find good extract method refactoring candidates for long Java methods [3]. Our first prototype, which was derived from manual experiments on several open source systems, implemented a scoring function to rank refactoring candidates. The result of our evaluation has shown that this first prototype finds suggestions that are followed by experienced developers. The results of our first prototype have been implemented in an industrial software quality analysis tool.

Problem statement. The scoring function is an essential part of our approach to derive extract method refactoring suggestions for long methods. It is decisive for the quality of our suggestions, and also important for the complexity of the implementation of the refactoring suggester. However, it is currently unclear how good the scoring function actually performs in ranking refactoring suggestions and how much complexity will be needed to obtain useful suggestions. Therefore, in order to enhance our work, we need a deeper understanding of the scoring function.

Contribution. We do further research on the scoring function of our approach to derive extract method refactoring suggestions for long Java methods. We use learning to rank techniques in order to learn which features of the scoring function are relevant, to get meaningful refactoring suggestions, and to keep the scoring function as simple as possible. In addition, we evaluate the ranking performance of our previous scoring function, and compare it with the new scoring function that we learned. For the machine learning setting, we use 177 training and testing data sets that we obtained from 13 well-known open source systems by manually ranking five to nine randomly selected valid refactoring candidates.

In this paper, we show how we derived better extract method refactoring suggestions than in our previous work using learning to rank tools.

1.2 Fundamentals

We use learning to rank techniques to obtain a scoring function that is able to rank extract method refactoring candidates, and use normalized discounted cumulative gain (NDCG) metrics to evaluate the ranking performance. In this section, we explain the techniques, tools and metrics that we use in this paper.

Learning to rank refers to machine learning techniques for training the model in a ranking task [4].

There are several learning to rank approaches, where the pairwise and the listwise approach usually perform better than common pointwise regression approaches [8]. The pairwise approach learns by comparing two training objects and their given ranks ('ground truth'), whereas in our case the listwise approach learns from the list of all given rankings of refactoring suggestions for a long method. Liu et al. [8] pointed out that the pairwise and the listwise approaches usually perform better than the pointwise approach. Therefore, we do not rely on a pointwise approach but use pairwise and listwise learning to rank tools.

Qin et al. [15] constructed a benchmark collection for research on several learning to rank tools on the Learning To Rank (LETOR) data set. Their results support the hypothesis that pointwise approaches perform badly compared with pairwise and listwise approaches. In addition, listwise approaches often perform better than pairwise. However, *SVM-rank*, a pairwise learning to rank tool by Tsochantaridis et al. [18], performs quite well and the first experiments on our data set showed that *SVM-rank* may lead us to interesting results. We set the parameter -c to 0.5 and the parameter -# to 5,000 as a trade-off between time consumption and learning performance.

Beside SVM-rank, we used a listwise learning to rank tool, *ListMLE* by Xia et al. [21]. In their evaluation, they showed that ListMLE performs better than ListNet by Cao et al. [1], which was also considered to be good by Qin et al.. Lan et al. [7] improved the learning capability of ListMLE, but did not provide binaries or source code; so we were unable to use the improved version.

ListMLE needs to be assigned a tolerance rate and a learning rate. In a series of experiments we performed, we found that the optimal ranking performance on our data set was with a tolerance rate of 0.001 and a learning rate of 1E-15.

1.2.2 Training and Testing

The learning process consisted of two steps: training and testing. We applied cross-validation [16] with 10 sets, that is, we split our learning data into 10 sets of (nearly) equal size. We performed 10 iterations using these sets, where nine of the sets were considered to be training data and one set was used as test data.

Test data is used to evaluate the ranking performance of the learned scoring function by comparing the grade of a refactoring candidate determined by the learned scoring function with its grade given by the learning data. We use NDCG metric to compare different scoring functions and their performances.

which is described in more detail by Järvelin and Kekäläinen [5], and measures the goodness of the ranking list (obtained by the application of the scoring function). Mistakes in the top-most ranks have a bigger impact on the DCG measure value. This is useful and important to us because we will not suggest all possible refactoring candidates, but only the highest-ranked ones. Given a long method, m_i, with refactoring candidates, C_i, suppose that π_i is the ranking list on C_i and y_i, the set of manually determined grades, then, the DCG at position k is defined as $DCG(k) = \sum_{j:\pi_i(j)\le k} G(j)D(\pi_i(j))$, where $G(\cdot)$ is an exponential gain function, $D(\cdot)$ is a position discount function, and $\pi_i(j)$ is the position of refactoring candidate, $c_{i,j}$, in π_i. We set $G(j) = 2^{y_{i,j}} - 1$ and $D(\pi_i(j)) = \frac{1}{\log_2(1+\pi_i(j))}$. To normalize the DCG, and to make it comparable with measures of other long methods, we divide this DCG by the DCG that a perfect ranking would have obtained. Therefore, the NDCG for a candidate ranking will always be in $[0,1]$, where the NDCG of 1 can only be obtained by perfect rankings. In our evaluation, we consider the NDCG value of the last position so that all ranks are taken into account. See Hang [4] for further details.

1.3 Approach

We discuss our approach to improve the scoring function in order to find the best suggestions for extract method refactoring.

1.3.1 Extract Method Refactoring Candidates

In our previous work [3], we presented an approach to derive extract method refactoring suggestions automatically for long methods. The main steps are: generating valid extract method refactoring candidates, ranking the candidates, and pruning the candidate list.

In the following, a *refactoring candidate* is a sequence of statements that can be extracted from a method into a new method. The *remainder* is the method that contains all the statements from the original method after applying the refactoring, plus the call of the extracted method. The suggested refactorings will help to improve the readability of the code and reduce its complexity, because these are main reasons for developers to initiate code refactoring [6].

We derived refactoring candidates from the control and data flow graph of a method using the Continuous Quality Assessment Toolkit (ConQAT[3]) open source software. We filtered out all invalid candidates, that is those that violate preconditions that need to be fulfilled for extract method refactoring (for details, see [12]). The second step of our approach was to rank the valid

[3] www.conqat.org

by filtering out very similar candidates, in order to obtain essentially different suggestions.

In the present paper, we focus on the ranking of candidates, and especially on the scoring function that defines that ranking.

1.3.2 Scoring Function

We aimed for an optimized scoring function that is capable of ranking extract method refactoring candidates, so that top-most ranked candidates are most likely to be chosen by developers for an extract method refactoring. The scoring function is a linear function that calculates the dot product of a coefficient vector, c, and a feature value vector, f, for each candidate. Candidates are arranged in decreasing order of their score.

In this paper, we use a basis of 20 features for the scoring function. In the following, we give a short overview about the features. There are three categories of features: complexity-related features, parameters, and structural information.

We illustrate the feature values with reference to two example refactoring candidates (C_1 and C_2) that were chosen from the example method given in Figure 1.1. The gray area shows the nesting area, which is defined below. The white numbers specify the nesting depth of the corresponding statement.

Fig. 1.1: Example Method with Nesting Area of Statements And Example Candidates

Table 1.1: Features and Values in Example

#	Feature	Type	C_1 L. 9 – 19	C_2 L. 10 – 18
1	LoC Red (abs)	int	9	9
2	Token Red (abs)	int	33	43
3	Stmt Red (abs)	int	5	6
4	LoC Red (rel)	double	0.42	0.43
5	Token Red (rel)	double	0.38	0.47
6	Stmt Red (rel)	double	0.38	0.46
7	Nest Depth Red	int	0	1
8	Nest Area Red	int	1	0
9	# Read Var	int	4	4
10	# Written Var	int	1	1
11	# Used Var	int	4	4
12	# Input Param	int	2	3
13	# Output Param	int	0	0
14	∃ Introd Com	bool	1	0
15	# Introd Com	int	2	0
16	∃ Concl Com	bool	0	0
17	# Concl Com	int	0	0
18	Same T Before	bool	0	0
19	Same T After	bool	0	0
20	# Branch Stmt	int	3	2

Complexity-related features

We mainly focused on reducing complexity and increasing readability. For complexity indicators, we used length, nesting and data flow information. For

reduction of the method length (with respect to the longest method after the refactoring). We considered length based on the number of lines of code (LoC), on the number of tokens, and on the number of statements – all of them as both absolute values and relative to the original method length.

We consider highly nested methods as more complex than moderately nested ones, and we use two features to represent the reduction of nesting: reduction of nesting depth and reduction of nesting area. The nesting area of a method with statements S_1 to S_n, each having a nesting depth of d_{S_i}, is defined to be $\sum_{i=1}^{n} d_{S_i}$. The idea of nesting area comes from the area alongside the single statements of pretty printed code (see the gray areas in Figure 1.1).

Dataflow information can also indicate complexity. We have features representing the number variables that are read, written or read and written.

Parameters

We considered the number of input and output parameters as an indicator of data coupling between the original and the extracted methods, which we want to keep low using our suggestions. The more parameters that are needed for a set of statements to be extracted from a method, the more the statements will depend on the rest of the original method.

Structural information

Finally, we have some features that represent structural aspects of the code. A design principle for code is that methods should process only one thing [9]. Methods that follow this principle are easier to understand. As developers often put blank lines or comments between blocks of code that process something else, we have features representing the existence and the number of blank or commented lines at their beginning, or at their end. Additionally, for first statement of the candidate, we check to see whether the type of the preceding is the same; and for the last statement, we check to see whether the type of the following statement is the same. Our last feature considers a structural complexity indicator – the number of branching statements in the candidate.

1.3.3 Training and Test Data Generation

To be able to learn a scoring function, we need training and test data. We derived this data by manually ranking approximately 1,000 extract method refactoring suggestions. To obtain this learning data, we selected 13 Java open source systems from various domains, and of different sizes. We consider a method to be 'long' if it has more than 40 LoC. From each project we randomly selected 15 long methods. For each method, we randomly selected valid refactoring candidates, where the number of candidates depended on the method length.

into the code. Therefore, in the pruning step of our approach, we usually filter out candidates that need more than three input parameters, thus avoiding the 'long parameter list' mentioned by Fowler [2]. To avoid learning that too many input parameters are bad, we considered only candidates that had less than four input parameters.

We ranked the selected candidates manually with respect to complexity reduction and readability improvement. The higher the ranking we gave a candidate, the better the suggestion was for us.

Some of the randomly selected methods were not suitable for an extract method refactoring. That was most commonly the case when the code would not benefit from the extract method, but from other refactorings. In addition, for some methods, we could not derive a meaningful ranking because there were only very weak candidates. That is why we did not use 18 of the 195 randomly selected long methods to learn our scoring function.[4]

1.4 Evaluation

In this section, we present and evaluate the results from the learning procedure.

1.4.1 Research Questions

RQ1: What are the results of the learning tools? In order to get a scoring function that is capable of ranking the extract method refactoring candidates, we decided to use two learning to rank tools that implement different approaches, and that had performed well in previous studies.

RQ2: How stable are the learned scoring functions? To be able to derive implications for a real-world scoring function, the coefficients of the learned scoring function should not vary a lot during the 10-fold cross evaluation procedure.

RQ3: Can the scoring function be simplified? For practical reasons, it is useful to have a scoring function with a limited number of features. Additionally, reducing the search space may increase the performance of the learning to rank tools – resulting in better scoring functions.

RQ4: How does the learned scoring function compare with our manually determined one? In our previous work, we derived a scoring function by manual experiments. Now we can use our learning data set to evaluate the ranking performance of the previously defined scoring function, and to compare it with the learned one.

[4] On http://in.tum.de/~haas/l2r_emrc_data.zip we provide our rankings and the corresponding code bases from which we generated the refactoring candidates.

To answer RQ1 and RQ2, we used the learning to rank tools SVM-rank and ListMLE to perform a 10-fold cross validation on our training and test data set of 177 long methods, and a total of 1,185 refactoring candidates. We illustrate the stability of the single coefficients by using box plots that show how the coefficients are distributed over the ten iterations of the 10-fold cross validation.

To answer RQ3, we simplified the learned scoring function by omitting features, where the selection criterion for the omitted features is preservation of the ranking capability of the scoring function. Our initial feature set contained six different measures of length. For the sake of simplicity, we would like to have only one measure of length in our scoring function. To find out which measure best fits in with our training set, we re-ran the validation procedure (again using ListMLE and SVM-rank), but this time with only one length measurement, using each of the length measurements one at a time. We continued with the feature set reduction until only one feature was left.

1.4.3 Results

The following paragraphs answer the research questions.

RQ1: What are the results of the learning tools?

Figures 1.2 and 1.3 show the results of the 10-fold cross validation for ListMLE and for SVM-rank, respectively. For each single feature, i, there is a box plot of the corresponding coefficient, c_i.

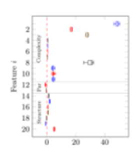


Fig. 1.2: Learning Result From ListMLE With All Features

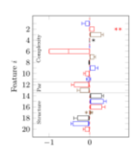


Fig. 1.3: Learning Result From SVM-rank With All Features

0.873, whereas for SVM-rank it is 0.790. Therefore, the scoring function found by ListMLE performed better than the scoring function found by SVM-rank.

Table 1.2: Coefficients of Variation for Learned Coefficients

	ListMLE	SVM-rank
AVG CV	0.0887	22.522
Min CV	0.0053	0.8970
Max CV	0.5767	451.2

RQ2: How stable are the learned scoring functions?

Table 1.2 shows the average, minimum and maximum coefficients of variation (CV) for the learned coefficients for ListMLE and for SVM-rank. Small CVs indicate that in relative terms the results from the single runs in the 10-cross fold procedure did not vary a lot, whereas big CVs indicate big differences between the learned coefficients. As the CVs of the single features from ListMLE are much smaller than those of SVM-rank, the coefficients of ListMLE are much more stable compared with SVM-rank. SVM-rank shows coefficients with a big variance between the single iterations of the validation process; that is, despite the heavy overlapping of the training sets, the learned coefficients vary a lot and can hardly be generalized.

RQ3: Can the scoring function be simplified?

Figure 1.4 shows a plot of the averaged NDCG measure for all 12 runs. Remember that we actually had three length measures, and we considered the absolute and the relative values for all of them. As the reduction of the number of statements led to a higher NDCG for ListMLE (which outperformed SVM-rank with respect to NDCG), we chose to use it as our length measure. In practice, that seems sensible since, while LoC also count empty and commented lines, the number of statements only counts real code.

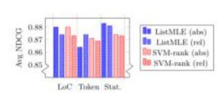


Fig. 1.4: Averaged NDCG When Considering Only One Length Measure

on the ranking performance and removed it in the next iteration. A scoring function that only considered the number of input parameters and length and nesting area reduction still had an average NDCG of 0.885.

RQ4: How does the learned scoring function compare with our manually determined one?

The scoring function that we presented in [3] achieved a NDCG of 0.891, which is better than the best scoring function learned in this evaluation.

1.4.4 Discussion

Our results show that, in the initial run of the learning to rank tools, features indicating a reduction of complexity are much more relevant for the ranking, and therefore have a comparatively high impact. Furthermore, the stability of ListMLE is higher on our data set than the stability of SVM-rank. For SVM-rank there is a big variance in the learned coefficients, which might also be a reason for the comparatively lower performance measure values.

The results for RQ3 show that it is possible to achieve a great simplification without big reductions in the ranking performance. The biggest influences on the ranking performance were the reduction of the number of statements, the reduction of nesting area (both are complexity indicators), and the number of input parameters.

Manual improvement As already mentioned, the learned scoring functions did not outperform the manually determined scoring function from our previous work. Obviously, the learning tools were not able to find optimal coefficients for the features. To improve the scoring function from our previous work, we did manual experiments that were influenced by the results of ListMLE and SVM-rank, and tested the results using the whole learning data set.

We were able to find several scoring functions that had only a handful of features and a better ranking performance than our scoring function from previous work (column 'Previous' in Table 1.3). In addition to the three most important features that we obtained in the answer to RQ3 (features #3, #7, #10), we also took the comment features (#14-17) into consideration. The main differences between the previous scoring function and the manually improved one from this paper are the length reduction measure, the omission of nesting depth, and the number of output parameters.

By taking the results of ListMLE and SVM-rank into consideration, we were able to find a coefficient vector such that the scoring function achieved a NDCG of 0.894 (see Table 1.3). That means that we were able to find a better ranking function when we combined the findings of our previous work with the learned coefficients from this paper.

#	Feature \ Fct	Previous	Learned	Improved
1	LoC (abs)	0.036	-	-
3	Stmt (abs)	-	0.681	0.066
7	Nesting Depth	0.362	-	-
8	Nesting Area	0.724	0.731	0.895
12	# Input P.	-0.362	-0.024	-0.331
13	# Output P.	-0.362	-	-
14	∃ Introd Com.	0.181	-	0.166
15	# Introd Com.	0.181	-	0.166
16	∃ Concl Com.	0.090	-	0.166
17	# Concl Com.	0.090	-	0.052
	AVG NDCG	0.891	0.885	0.894

1.5 Threats to Validity

Learning from data sources that are either too similar or too small means that there is a chance that no generalization of the results is possible. To have enough data to enable us to learn a scoring function that can rank extract method refactoring candidates, we chose 13 Java open source projects from various domains and from each project we randomly selected 15 long methods. We manually reviewed the long methods, and filtered out those that were not appropriate for the extract method. From the 177 remaining long methods, we randomly chose five to nine valid refactoring suggestions, depending on the method length. We ensured that our learning data did not contain any code clones to avoid learning from redundant data.

The manual ranking was performed by a single individual, which is a threat to validity since there is no commonly agreed way on how to shorten a long method, and therefore no single ranking criterion exists. The ranking was done very carefully, with the aim of reducing the complexity and increasing the readability and understandability of the code as much as possible; so, the scoring function should provide a ranking such that we can make further refactoring suggestions with the same aim.

We relied on two learning to rank tools, which represents another threat to validity. The learned scoring functions heavily depend on the tool. As the learned scoring function varies, it is necessary to have an independent way of evaluating the ranking performance of the learned scoring functions. We used the widely used measure NDCG to evaluate the scoring functions, and applied a 10-fold cross validation approach to obtain a meaningful evaluation of the ranking performance of the learned scoring function.

A threat to external validity is the fact that we derived our learning data from 13 open source Java systems. Therefore, results are not necessarily generalizable.

1.6 Related Work

In our previous work [3], we presented an automatic approach to derive extract method refactoring suggestions for long methods. We obtained valid

All valid refactoring candidates were ranked by a manually-determined scoring function that aims to reduce code complexity and increase readability. In the present work, we have put the scoring function on more solid ground by learning a scoring function from many long methods, and manually ranked refactoring suggestions.

In the literature, there are several approaches that learn to suggest the most beneficial refactorings – usually for code clones. Wang and Godfrey [19] propose an automated approach to recommend clones for refactoring by training a decision-tree based classifier, C4.5. They use 15 features for decision-tree model training, where four consider the cloning relationship, four the context of the clone, and seven relate to the code of the clone. In the present paper, we have used a similar approach, but with a different aim: instead of clones, we have focused on long methods.

Mondal et al. [10] rank clones for refactoring through mining association rules. Their idea is that clones that are often changed together to maintain a similar functionality are worthy candidates for refactoring. Their prototype tool, *MARC*, identifies clones that are often changed together in a similar way, and mines association rules among these. A major result of their evaluation on thirteen software systems is that clones that are highly ranked by MARC are important refactoring possibilities. We used learning to rank techniques to find a scoring function that is capable of ranking extract method refactoring candidates from long methods.

1.7 Conclusion and Future Work

In this paper, we have presented an approach to derive a scoring function that is able to rank extract method refactoring suggestions by applying learning to rank tools. The scoring function can be used to automatically rank extract method refactoring candidates, and thus present a set of best refactoring suggestions to developers. The resulting scoring function needs less parameters than previous scoring functions but has a better ranking performance.

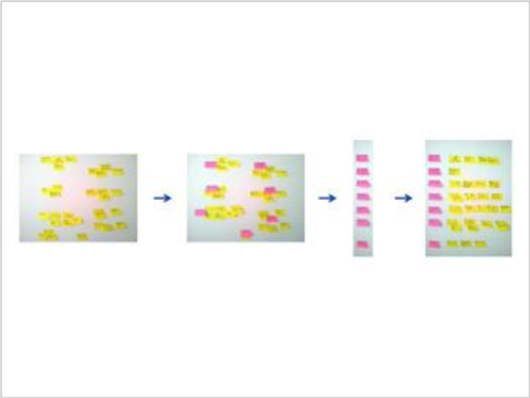
In the future, we would like to suggest sets of refactorings, especially those that remove clones from the code.

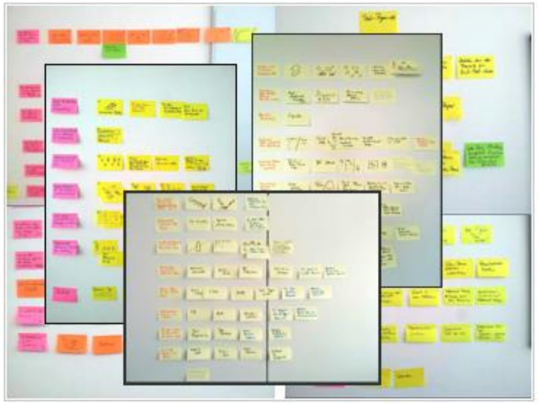
We would also like to find out whether the scoring function provides good suggestions for object-oriented programming languages other than Java and whether other features need to be considered in that case.

Acknowledgments

Thanks to the anonymous reviewers for their helpful feedback. This work was partially funded by the German Federal Ministry of Education and Research (BMBF), grant "Q-Effekt, 01IS15003A". The responsibility for this article lies with the authors.

Vortrag Vorbereiten





https://thesisguide.org/2015/03/04/how-to-draft-your-presentation/

Vortragsplanung: Delta BA/MA

- Probevortrag vor Betreuer
- Vortrag auf Englisch üben
- Einstiegssätze aufschreiben und auswendig lernen.
- Backup Folien für mögliche Fragen

Forschungsarbeiten @ CQSE

- Mi., 27.06., 17 Uhr im Gate
- Agenda: Ablauf einer Forschungsarbeit @ CQSE
  - Analyse-Implementierung
  - Studie
  - Betreuung
  - Pitch aktueller Themen
- Hinterher Pizza und Bier ☺

Anmeldung: https://forschungsarbeiten-cqse.eventbrite.de

Fazit

Willst Du selbst Forschen und die Forschungscommunity kennenlernen? Dann mach ein Guided Research.

Danke!

Bei Interesse einfach bei uns melden:

juergens@cqse.eu

haas@cqse.eu

Am 26.6. um 17 Uhr: Forschungsarbeiten @ CQSE

Mehr Infos: www.thesisguide.org