How to Write a Great Guided Research
And why should I do it?

Roman Haas

With material from Dr. Elmar Juergens

In close cooperation with the Academic Advisors at TUM Informatics
2011 – 2017

TUM

2017 – now

CQSE

Research project “SOFIE”
thesisguide.org

- Slides
- Video
- Detailed Essays
- FAQ
Agenda

1. Motivation

2. Preparation

3. Doing the work
Guided Research

- **Guidance**
  - Advisor has research experience, helps you on your way
  - Examiner must be from TUM Informatics or affiliated with the Department of Informatics

- **Your own (small) research project**
  - Related Work
  - Implementation?
  - Proof?
  - Evaluation?

- **Document and present your work**

  ➢ Insights into real scientific work
<table>
<thead>
<tr>
<th>Guided Research</th>
<th>Master’s Thesis</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Voluntary</td>
<td>• Mandatory</td>
</tr>
<tr>
<td>• 6 months, 10 ECTS</td>
<td>• 6 months, 30 ECTS</td>
</tr>
<tr>
<td>• Effort comparable to a more labor-intensive lab course</td>
<td>• Full-Time</td>
</tr>
<tr>
<td>• Approx. 40 students/semester</td>
<td>• Approx. 100 students/semester</td>
</tr>
</tbody>
</table>
Less Formal than a Thesis

• Written document is „just“ a scientific report on your results (8-12 pages in English) which you need to send to your advisor/examinor only

• You have to present your work
  – At the chair
  – Or at a „scientific event“
There are some formalia, though...

- You have to be enrolled in a Master’s program (Informatics, Data Engineering & Analytics, Information Systems, Games Engineering)
- Registration must be done in the first lecture week
- Submission no later than the first lecture week of the next semester (6 months duration)
- Cannot be extended
- No transfer of credits, you need an internal examiner (with whom you may work together abroad)
Learning to Rank Extract Method
Refactoring Suggestions for Long Methods

• Given: A set of refactoring suggestions for long methods

• Needed: scoring function to find out which is the best

• Approach: machine learning
1 Learning to Rank Extract Method Redacting Suggestions for Long Methods

Biraj Bose* and Benjamin Huang†
† TechNical University of Munich, Lehrstuhl 8, Germany
† Stanford University, Stanford, CA, USA

1.1 Introduction

1.2 ProblemFormulation

1.3 Training and Testing

1.4 Evaluation

1.5 Related Work

1.6 Related Work

2 Conclusion

3 Acknowledgments

Figure 1: Example Method with Spacing of Statements and Example
cases

Figure 2: The learned function: a) for 40% and 60% of

Learning to rank for extract methods

Suggested by NeurIPS

1 Learning to Rank Extract Method Redacting Suggestions for Long Methods

Biraj Bose* and Benjamin Huang†
† Technical University of Munich, Lehrstuhl 8, Germany
† Stanford University, Stanford, CA, USA

1.1 Introduction

A long method is a bad habit in software engineering, and makes code-hard to read and maintain. As a result, many developers develop bad habits, such as long methods, instead of good habits, such as small, well-organized methods. This bad habit is called top-down method addiction. It is a problem because it is often difficult to refactor a long method into a smaller one. To solve this problem, we proposed a new method, LazyRefactor, that learns to rank extract methods, which takes

1.3 Training and Testing

The learning process consists of a two-step training and testing. We applied

1.4 Evaluation

1.5 Related Work

2 Conclusion

3 Acknowledgments

Figure 1: Example Method with Spacing of Statements and Example
cases

Figure 2: The learned function: a) for 40% and 60% of

Learning to rank for extract methods

Suggested by NeurIPS

1 Learning to Rank Extract Method Redacting Suggestions for Long Methods

Biraj Bose* and Benjamin Huang†
† Technical University of Munich, Lehrstuhl 8, Germany
† Stanford University, Stanford, CA, USA

1.1 Introduction

A long method is a bad habit in software engineering, and makes code-hard to read and maintain. As a result, many developers develop bad habits, such as long methods, instead of good habits, such as small, well-organized methods. This bad habit is called top-down method addiction. It is a problem because it is often difficult to refactor a long method into a smaller one. To solve this problem, we proposed a new method, LazyRefactor, that learns to rank extract methods, which takes

1.3 Training and Testing

The learning process consists of a two-step training and testing. We applied

1.4 Evaluation

1.5 Related Work

2 Conclusion

3 Acknowledgments

Figure 1: Example Method with Spacing of Statements and Example
cases

Figure 2: The learned function: a) for 40% and 60% of

Learning to rank for extract methods

Suggested by NeurIPS

1 Learning to Rank Extract Method Redacting Suggestions for Long Methods

Biraj Bose* and Benjamin Huang†
† Technical University of Munich, Lehrstuhl 8, Germany
† Stanford University, Stanford, CA, USA

1.1 Introduction

A long method is a bad habit in software engineering, and makes code-hard to read and maintain. As a result, many developers develop bad habits, such as long methods, instead of good habits, such as small, well-organized methods. This bad habit is called top-down method addiction. It is a problem because it is often difficult to refactor a long method into a smaller one. To solve this problem, we proposed a new method, LazyRefactor, that learns to rank extract methods, which takes

1.3 Training and Testing

The learning process consists of a two-step training and testing. We applied

1.4 Evaluation

1.5 Related Work

2 Conclusion

3 Acknowledgments

Figure 1: Example Method with Spacing of Statements and Example
cases

Figure 2: The learned function: a) for 40% and 60% of

Learning to rank for extract methods

Suggested by NeurIPS

1 Learning to Rank Extract Method Redacting Suggestions for Long Methods

Biraj Bose* and Benjamin Huang†
† Technical University of Munich, Lehrstuhl 8, Germany
† Stanford University, Stanford, CA, USA

1.1 Introduction

A long method is a bad habit in software engineering, and makes code-hard to read and maintain. As a result, many developers develop bad habits, such as long methods, instead of good habits, such as small, well-organized methods. This bad habit is called top-down method addiction. It is a problem because it is often difficult to refactor a long method into a smaller one. To solve this problem, we proposed a new method, LazyRefactor, that learns to rank extract methods, which takes

1.3 Training and Testing

The learning process consists of a two-step training and testing. We applied

1.4 Evaluation

1.5 Related Work

2 Conclusion

3 Acknowledgments

Figure 1: Example Method with Spacing of Statements and Example
cases

Figure 2: The learned function: a) for 40% and 60% of

Learning to rank for extract methods

Suggested by NeurIPS

1 Learning to Rank Extract Method Redacting Suggestions for Long Methods

Biraj Bose* and Benjamin Huang†
† Technical University of Munich, Lehrstuhl 8, Germany
† Stanford University, Stanford, CA, USA

1.1 Introduction

A long method is a bad habit in software engineering, and makes code-hard to read and maintain. As a result, many developers develop bad habits, such as long methods, instead of good habits, such as small, well-organized methods. This bad habit is called top-down method addiction. It is a problem because it is often difficult to refactor a long method into a smaller one. To solve this problem, we proposed a new method, LazyRefactor, that learns to rank extract methods, which takes

1.3 Training and Testing

The learning process consists of a two-step training and testing. We applied

1.4 Evaluation

1.5 Related Work

2 Conclusion

3 Acknowledgments

Figure 1: Example Method with Spacing of Statements and Example
cases

Figure 2: The learned function: a) for 40% and 60% of

Learning to rank for extract methods

Suggested by NeurIPS

1 Learning to Rank Extract Method Redacting Suggestions for Long Methods

Biraj Bose* and Benjamin Huang†
† Technical University of Munich, Lehrstuhl 8, Germany
† Stanford University, Stanford, CA, USA

1.1 Introduction

A long method is a bad habit in software engineering, and makes code-hard to read and maintain. As a result, many developers develop bad habits, such as long methods, instead of good habits, such as small, well-organized methods. This bad habit is called top-down method addiction. It is a problem because it is often difficult to refactor a long method into a smaller one. To solve this problem, we proposed a new method, LazyRefactor, that learns to rank extract methods, which takes

1.3 Training and Testing

The learning process consists of a two-step training and testing. We applied

1.4 Evaluation

1.5 Related Work

2 Conclusion

3 Acknowledgments

Figure 1: Example Method with Spacing of Statements and Example
cases

Figure 2: The learned function: a) for 40% and 60% of

Learning to rank for extract methods

Suggested by NeurIPS

1 Learning to Rank Extract Method Redacting Suggestions for Long Methods

Biraj Bose* and Benjamin Huang†
† Technical University of Munich, Lehrstuhl 8, Germany
† Stanford University, Stanford, CA, USA

1.1 Introduction

A long method is a bad habit in software engineering, and makes code-hard to read and maintain. As a result, many developers develop bad habits, such as long methods, instead of good habits, such as small, well-organized methods. This bad habit is called top-down method addiction. It is a problem because it is often difficult to refactor a long method into a smaller one. To solve this problem, we proposed a new method, LazyRefactor, that learns to rank extract methods, which takes

1.3 Training and Testing

The learning process consists of a two-step training and testing. We applied
Chronological Overview

- **Registration**
  - 01/15

- **SS 2015**
  - 04/15

- **Deadline TUM**
  - 10/15

- **Deadline SWQD**
  - 04/16

- **Camera-Ready Version**
  - 08/16

- **Deadline Slides**
  - 01/17
What is Different to Other Study Projects?

• More Freedom
  – Topic
  – Own research
  – You define schedule and pace

• Requires high level of self-organization

• Better opportunities for personal growth
Personal Conclusion

• My GR was on my „mental Stack“ during my entire studies in the Master’s program

• GR got me out of my comfort zone

• Learned a lot on research methodologies and practical application of machine learning techniques

• Working on my research topic was fun for me

• I would do it again 😊
Timo Pawelka
Automatische Erkennung der Sprache von Quelltext-Kommentaren
Bachelor’s Thesis, not published

Timo Pawelka
Elmar Juergens:
Is This Code Written in English? A Study of the Natural Language of Comments and Identifiers in Practice.
Raphael Nömmer, Roman Haas
Test Suite Minimization
Guided Research, to be published in Conference Proceedings of SWQD ‘20

Raphael Nömmer
Design and Evaluation of Regression Test Suite Minimization Techniques
Master's Thesis
Funding

Costs 1k€ – 5k€
  • Travel and accommodation costs
  • Conference fee

Funding sources (often mixed)
  • Travel Subsidies
  • Chairs
  • DAAD scholarships
  • CQSE

Decision processes take long, so organize this early!
Agenda

1. Motivation
2. Preparation
3. Doing the work
Get the Most out of your GR?! 

• GR provides the opportunity to publish scientific work at a scientific venue.

• Nevertheless, formally, you do not need to publish anything

• My recommendation: aim for a scientific publication
Submissions → Selection Procedure (10-50%) → Agenda
## Pecking Order

**Conference** 10%-25%

**Workshop** 40%-60%

### Conference Highlights

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Full Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHASE</td>
<td>11th International Workshop on Cooperative and Human Aspects of Software Engineering</td>
</tr>
<tr>
<td>CSI-SE</td>
<td>5th International Workshop on Crowd Sourcing in Software Engineering</td>
</tr>
<tr>
<td>MET</td>
<td>International Workshop on Metamorphic Testing</td>
</tr>
</tbody>
</table>

### Workshop Acronyms

- RAISE
- SEAD
- SEaCPS
- SoHeal
- MISE
- SEHS
- GE
- RoSE
- SEFAIAS
- SQUADE
- AST
- SE4COG
- FairWare
- SE4Science
- SER&IP
- SESCO
- FI
- SEEM
Several Months

Author

Organizer

Reviewer 1

Reviewer 2

Reviewer 3

Call for Paper

Paper

RFR

Review

Accept/Reject
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...
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 bad reuse by programmers, and can occur at many levels, from simple statement sequences to blocks, models, requirements or architecture today.

IWSC series of events has provided a forum for researchers worldwide to share their knowledge and experiences on the topic of clones. In particular, we expect the in-depth presentations about IWSC 2018 are here on this page.

TOPIES OF INTEREST:

Topics of interest include but not limited to:
- Use cases for clones and clone detection
- Experiences with clones and clone detection
- Types and nature of clones
- Causes and effects of cloning
- Techniques and algorithms
- Clone and clone pattern via semantic analysis
- Tools and systems for detecting clones
- Applications of clone detection
- System architecture and clone detection
- Effect of clones to software evolution
- Clone analysis in families of systems
- Measures of code similarity
- Economic and trade-off models
- Evaluation and benchmarking
- Licensing and plagiarism issues
- Clone-aware software design
- Refactoring through clone analysis
- Higher-level clones in modern software systems
- Clone evolution and variability management
- Role of clones in software evolution

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Institution</th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>Toshihiro Kamiya</td>
<td>Shimane University</td>
<td>Japan</td>
</tr>
<tr>
<td>Dqing Hou</td>
<td>Clarkson University</td>
<td>USA</td>
</tr>
<tr>
<td>Tien Nguyen</td>
<td>University of Texas at Dallas</td>
<td>USA</td>
</tr>
<tr>
<td>Nils Göde</td>
<td>CQSE GmbH</td>
<td>Germany</td>
</tr>
<tr>
<td>Jens Krinke</td>
<td>University College London</td>
<td>UK</td>
</tr>
<tr>
<td>Otavio Lemos</td>
<td>ICT-UNIFESP</td>
<td>Brazil</td>
</tr>
<tr>
<td>Manishankar Mondal</td>
<td>University of Saskatchewan</td>
<td>Canada</td>
</tr>
<tr>
<td>Ravindar Naik</td>
<td>Tata Consultancy Services</td>
<td>India</td>
</tr>
<tr>
<td>Robert Tairas</td>
<td>Vanderbilt University</td>
<td>USA</td>
</tr>
<tr>
<td>Minhaz Zibran</td>
<td>University of New Orleans</td>
<td>USA</td>
</tr>
<tr>
<td>Eunjong Choi</td>
<td>Nara Institute of Science and Technology</td>
<td>Japan</td>
</tr>
<tr>
<td>Michael Godfrey</td>
<td>University of Waterloo</td>
<td>Canada</td>
</tr>
<tr>
<td>Yoshiki Higo</td>
<td>Osaka University</td>
<td>Japan</td>
</tr>
<tr>
<td>Foutse Khomh</td>
<td>Ecole Polytechnique de Montréal</td>
<td>Canada</td>
</tr>
<tr>
<td>Nicholas A. Kraft</td>
<td>ABB Corporate Research</td>
<td>USA</td>
</tr>
<tr>
<td>Chanchal Roy</td>
<td>University of Saskatchewan</td>
<td>Canada</td>
</tr>
<tr>
<td>Hitesh Sainani</td>
<td>Microsoft</td>
<td>USA</td>
</tr>
<tr>
<td>Suresh Thummalapenta</td>
<td>Microsoft</td>
<td>USA</td>
</tr>
<tr>
<td>Xiyouin Wang</td>
<td>University of Texas at San Antonio</td>
<td>USA</td>
</tr>
<tr>
<td>Norihira Yoshida</td>
<td>Nagoya University</td>
<td>Japan</td>
</tr>
</tbody>
</table>
What If I have no Topic in Mind?

• Ask potential advisors for ideas
  – Advisor from Bachelor’s Thesis
  – Lectures
  – Seminars
  – Lab courses

• As an advisor, I do not expect
  – Students to come up with thesis topics
  – Students to apply only for documented topics

• If you have a rough idea, discuss it with potential advisors
Requirements for a GR topic

• Is there a clear problem statement?
• Can different solutions be evaluated objectively?

Why?
• Decision making while you work on it
• Easier to convince advisor
• Easier to convince program chair

Even more important for a GR than BA/MA

More info: www.thesisguide.org
What Makes a Good Guided Research Advisor

- Needs to have publishing experience
- Has already successfully published (ideally on the same workshop if you aim for a publication)
- Sources: scholar.google.com, DBLP, personal webpage.
Agenda

1. Motivation

2. Preparation

3. Doing the work
View as an Advisor

- Regular meeting
- Meeting on demand
The #icse19 technical track has received an impressive 529 submissions! Good luck to all authors and happy reviewing to the PC and PB members :-)

Tweet übersetzen
Write for the Reviewer

• Make problem statement and contribution very clear

• Use established outline: https://thesisguide.org/2014/10/13/thesis-architecture/

• Make text easily readable. This is hard and exhausting work. But you can learn it, this is no issue of talent.
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

Embrace the mess

The first draft of anything is shit.
My Personal Best Practices

- Block writing time
- Begin with outline
- Separate writing from improving
- Write complete paragraphs before improving them
- Let text „cool down“ and proof-read it later again

- There is not the one silver-bullet way of writing
Scott Berkun: How to Write an Essay in Fast Motion

How to write an essay

- Montaigne and blogging
- The stupid way they teach (but neglect that most essays suck)
- Finding ideas and starting
- Getting stuck
- Reading and revising
- Knowing when you are done

People forget writing is comprised of three things: words, sentences and paragraphs. If you know a few words, you can make a sentence. If you write a few sentences you can make a paragraph. Keep it simple and filling pages gets easy. It's when you make writing more complex that problems arise. The first lesson then is to string words together. One, two, four, ten. There is always time to make it more complex later.

There is no answer to how to start. Make an outline if you like. I often do. It's true all writing begins with ideas, but we forget ideas are like whispers in our minds all the time, and we can hear them if we have the quiet courage to listen. I keep a notebook with me all times and that's one source for writing ideas. In conversations with friends, watching TV, or waiting for the bus, I put down little ideas.

All writing begins with ideas, and that's where I generally start. I get an idea from a conversation, or a book, or while daydreaming and write it down. When I write it down I sometimes find there are several sub ideas underneath the first one that are interesting or explain the first point I started with, so I write those down as well. Sometimes this goes on for 30 seconds, other times for 5 minutes. I may come back a week later and flush some of these sets of notes out, but more often I abandon them. I have books and books of abandoned lists of ideas for things in various stages of incubation. It's a good habit to have as a writer – lots of half-baked little things lurking around. When you're bored or stuck, it's an inventory of things partially done, and that's a gift to the future me who might just need a little boost to start from next time.

If you take that list of ideas or points and put it in the blinking of a blank screen, that's my outline. I don't have it too detailed, and I don't want it too vague. Short sentences that have a clear point of view and divide the world, however neatly into two piles, are good sentences. They're personalized examples, loaded with both vision and nutrients designed for my particular mind.

Youtube: http://youtu.be/BNDEDWwZyKM
English Writing Center

• Free one-to-one consulting with native English speakers
  – GR, Thesis, Homework, CV etc.
  – Text needs not to be ready

https://www.sprachenzentrum.tum.de/sprachen/englisch/english-writing-center/
How to Spend Your Writing Time Well?

Every 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, poor writing resource allocation is thus an effective recipe to write for a bad thesis.

So how to do you do this step well? For me, writing resource allocation is a lot like allocating plate space when eating at a large buffet. For both problems, there is a similar solution strategy that is intuitive, widely applied, and reliable to produce poor results.

James Morrison
jmedits@gmail.com
1 Learning to Rank Exact Method Refactoring Suggestions for Long Methods

1.1 Introduction

A large team is a technical challenge, and it is necessary to consider that not only the quality of the team's output, but also the relationship between team members, is important. Learning to rank is an active area of research, and learning to rank is a statistically robust and objective method that can be used to assess the quality of code refactoring suggestions. In this paper, we present a learning to rank model for exact method refactoring suggestions for long methods. The learning to rank model is designed to learn the quality of the refactoring suggestions, and the model can be used to improve the quality of refactoring suggestions. The model is trained on a dataset of refactoring suggestions, and the model is evaluated on a separate dataset of refactoring suggestions. The results show that the learning to rank model can improve the quality of refactoring suggestions.

1.2 Preliminary

We define the learning to rank problem in this section. We define the learning to rank problem as a binary classification problem. In this problem, each refactoring suggestion is classified as either correct or incorrect. The learning to rank model is trained on a dataset of refactoring suggestions, and the model is evaluated on a separate dataset of refactoring suggestions. The results show that the learning to rank model can improve the quality of refactoring suggestions.

1.3 Training and Testing

The learning process consisted of two steps: training and testing. In the training phase, we used cross-validation with 10 folds, and we used the training data to train the model. In the testing phase, we used the testing data to evaluate the model.

1.4 Evaluation

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

1.4.1 Research Questions

RQ1: What are the results of the learning to rank model? To verify the results of the learning to rank model, we compared the results with the results of other refactoring suggestion models. The results show that the learning to rank model can improve the quality of refactoring suggestions.

RQ2: How stable are the learning to rank model results? To verify the stability of the learning to rank model results, we used a testing with a training set of 100,000 refactoring suggestions. The results show that the learning to rank model is stable.

RQ3: How do the learning to rank model results compare with our manual grading? To verify the results of the learning to rank model, we compared the results with the results of our manual grading. The results show that the learning to rank model can improve the quality of refactoring suggestions.

A model that achieves more than 70% accuracy in the binary classification problem is considered to be effective. Therefore, the model that achieves an accuracy of more than 70% is considered to be effective. The model that achieves an accuracy of more than 70% is considered to be effective.

1.5 Threats to Validity

Threats to validity are the factors that can affect the validity of the results. These factors can be internal or external, and they can be controlled or uncontrolled. The threats to validity are the factors that can affect the validity of the results. These factors can be controlled or uncontrolled. The threats to validity are the factors that can affect the validity of the results. These factors can be controlled or uncontrolled. The threats to validity are the factors that can affect the validity of the results. These factors can be controlled or uncontrolled. The threats to validity are the factors that can affect the validity of the results. These factors can be controlled or uncontrolled.
Prepare Presentation

https://thesisguide.org/2015/03/04/how-to-draft-your-presentation/
Presentation Differences to BA/MA

- Rehearsal talk with advisor
- Practice it in English
- Formulate starting sentences and learn them by heart
- Backup slides for questions (e.g., more details)
Conclusion

Do you want to do your own research and get to know the research community? Then a guided research is the best you can do!
http://cqse.eu/feedback-tum-talk
Thank you!

If you are interested in a guided research in the field of software analysis and testing, please let me know:

haas@cqse.eu

More Info:  
www.thesisguide.org