Software Engineering Seminar

Malte Schwerhoff
Petar Tsankov

http://lec.inf.ethz.ch/seminars/2019/ses/

Slides based on previous seminars by Markus Püschel, Martin Vechev

Learning Objects

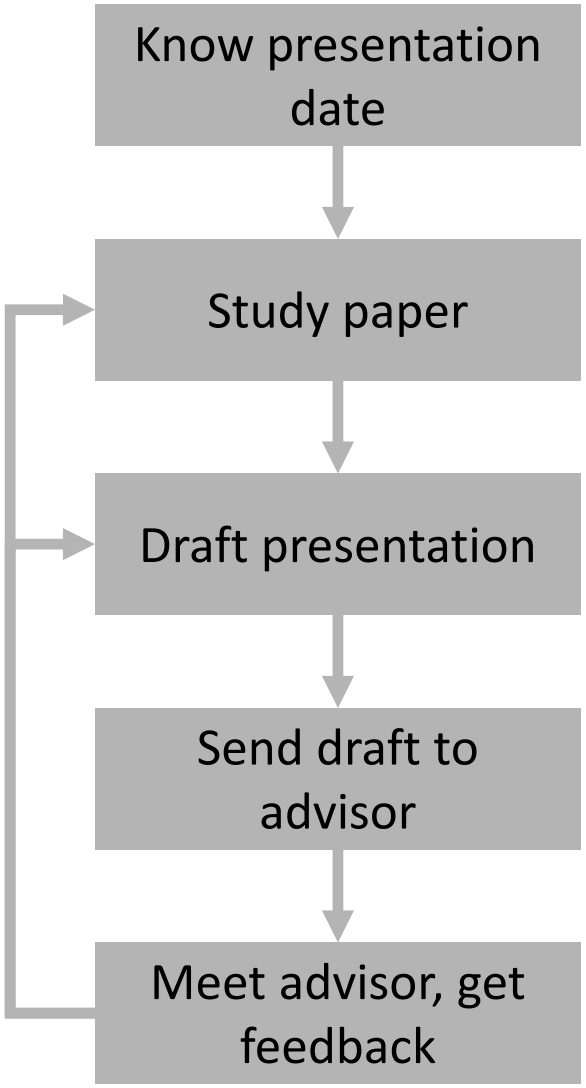
- How to present technical work
- How to read, dissect and assess research papers
- Learn about research directions in software engineering

The Team

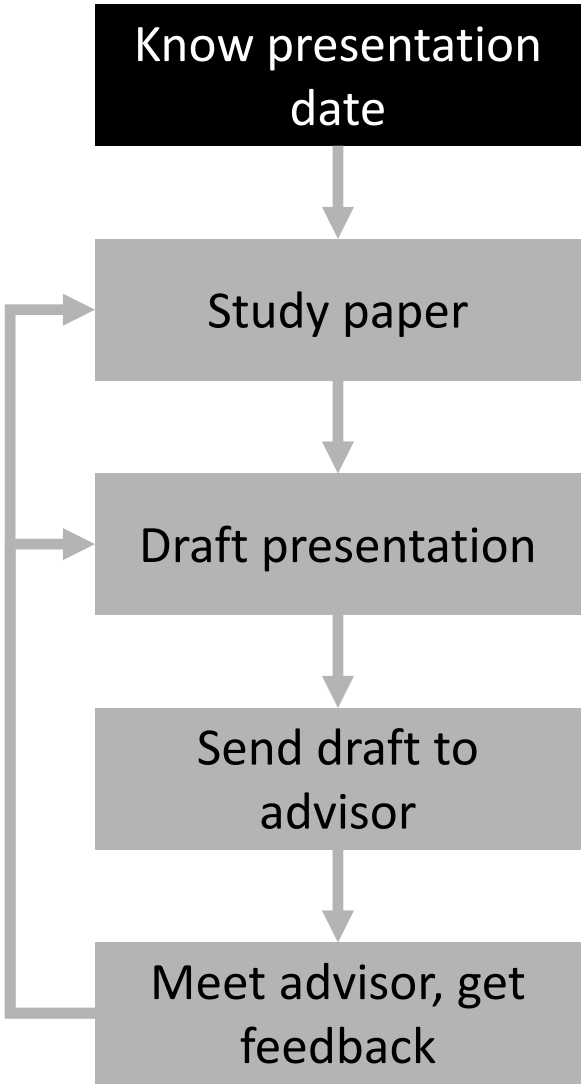
- Malte Schwerhoff, Petar Tsankov

- Teaching assistants (advisors):
  - Jingxuan He <jingxuan.he@inf.ethz.ch>
  - Pinjia He <pinjia.he@inf.ethz.ch>
  - Eilers Marco <marco.eilers@inf.ethz.ch>
  - Manuel Rigger manuel.rigger@inf.ethz.ch
  - Tyler Smith <tyler.smith@inf.ethz.ch>
  - Samuel Steffen <samuel.steffen@inf.ethz.ch>
  - Joao Rivera <hector.rivera@inf.ethz.ch>

Preparing Your Talk



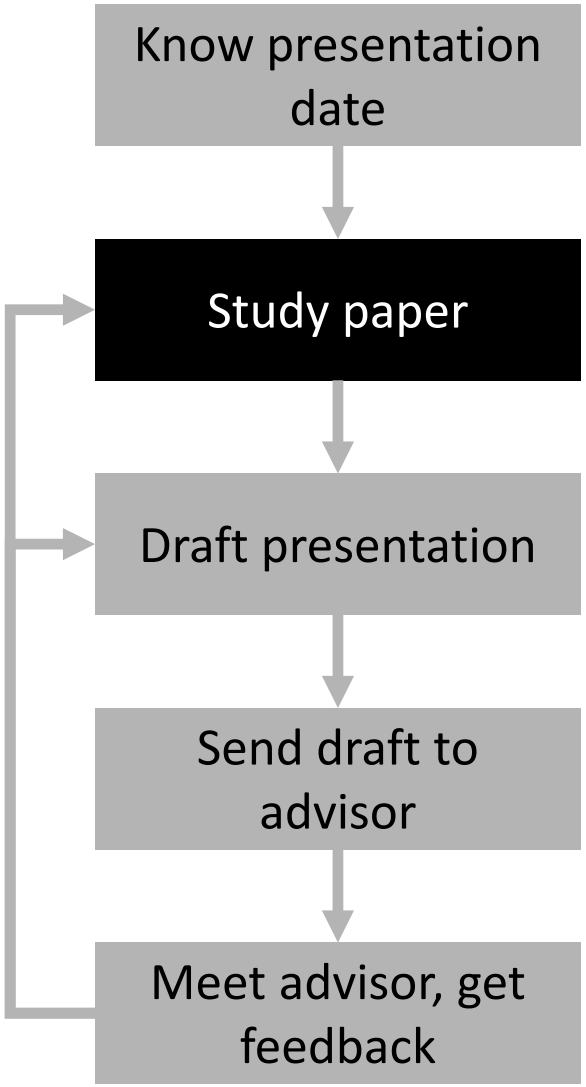
Preparing Your Talk: Know Date



Start early!

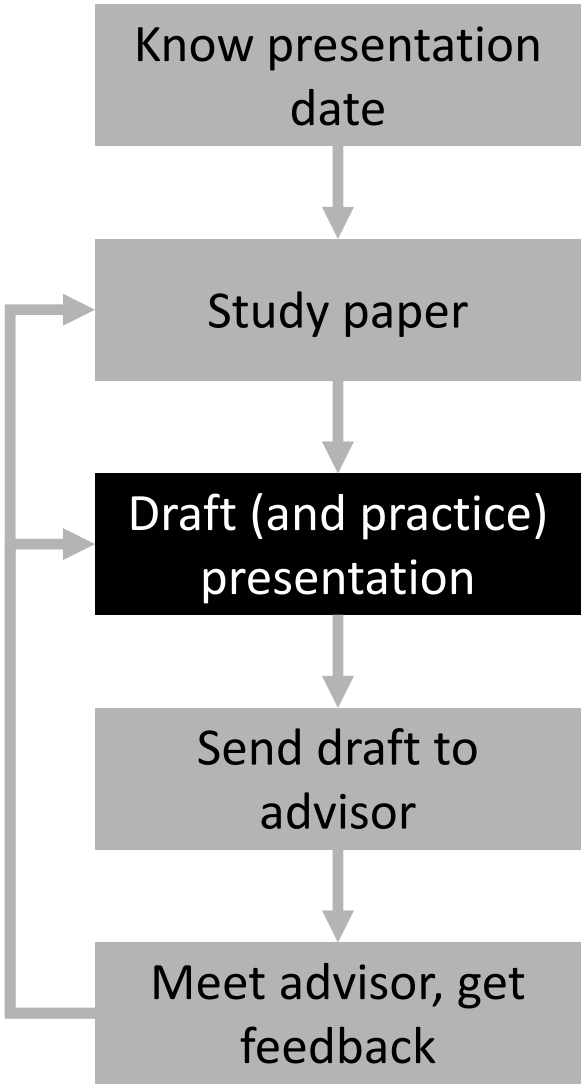
- Understanding a paper takes time
- As does preparing a good presentation

Preparing Your Talk: Study Paper



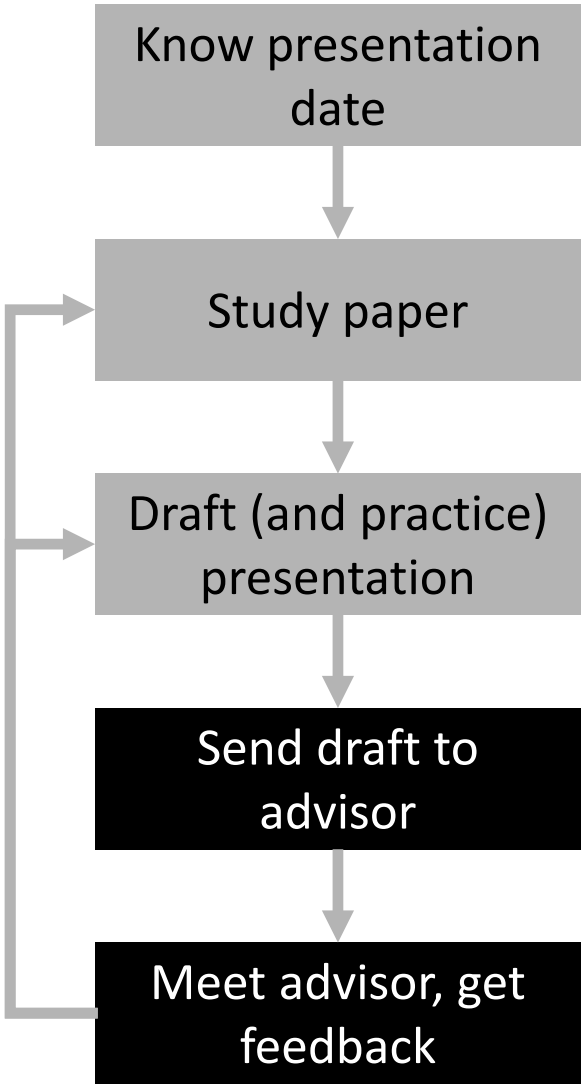
- Carefully read the paper
  - Details matter
  - Understand terminology and subtle differences
  - Briefly read cited papers, if necessary
- Be critical
  - Limitations?
  - What are the authors not telling us?
  - Potential improvements?
- Hands-on
  - Try examples on paper
  - Try tools, if available
- Consult advisor (for important questions)

Preparing Your Talk: Draft Slides



- Presentation duration: 30 minutes (plus questions)
- **General guidelines**: a presentation
  - *cannot* contain all details
  - should *intrigue* audience; get them to read the paper
- Explain *motivation* and *context*
- *Intuition* and *solution idea* before technical details …
- … but present *technical solution* as well
- Use your *own* examples (and demo tool, if possible)
- *Conclude*: results, limitations & future work, your personal assessment
- Practice!

Preparing Your Talk: Meet Advisor



- Meeting your advisor is mandatory
- Contact your advisor, and send draft, *at least 10 days* before your presentation
- *Technical questions:* consider sending alongside draft
- Ask for *general feedback*, and *specific hints* and *suggestions*
- Update presentation
- In case of *fundamental* changes: ask for a second meeting (or brief feedback via e-mail)

Presentation Style

- See these slides by Markus Püschel
- **General guidelines:** slides and speaker complement each other
- Slides:
  - Text: short; verbalise details
  - Try to *visualise* (diagrams, figures, code snippets, ...) as much as possible
  - Try to avoid text-only slides
  - Try *really hard* to not start with a text-only slide
- Presentation:
  - Check your speaking pace
  - Try to enthuse the audience
  - Be yourself :-)

Public Feedback

- *Audience* asked to *give feedback* right after your presentation
- You'll be asked if you'd rather not have this (or inform us ahead of time)

Grading

- Your presentation
  - How well did you understand the paper?
  - Did you cover the important aspects?
  - Did you use your own examples?
  - Was the presentation clear and understandable?
  - How did you handle questions from the audience?
  - Did you incorporate advisor feedback?
  - Did your presentation (roughly) last 30 minutes?
- Seminar participation:
  - Did you attend all sessions?
  - Did you ask good questions?
  - Did you provide feedback?
- We'll consider paper difficulty, order of presenters, advisor feedback

Schedule

- Two meetings per seminar session
- First two presentations on **Monday 7.10.** (no class before that)

Date	Legi	Paper	Advisor
07.10.	16-916-421	Optimizing Dynamically-Typed Object-Oriented Languages With Polymorphic Inline Caches	Manuel
07.10.	17-947-599	Is search really necessary to generate high-performance BLAS?	Tyler
14.10.	17-936-188	NetComplete: Practical Network-Wide Configuration Synthesis with Autocompletion	Sam
14.10.	17-934-258	Leveraging Rust Types for Modular Specification and Verification	Marco
21.10.	17-932-880	Towards Optimization-Safe Systems: Analyzing the Impact of Undefined Behavior	Manuel
21.10.	17-923-301	teEther: Gnawing at Ethereum to Automatically Exploit Smart Contracts	Jingxuan
28.10.	17-921-891	A Fast Analytical Model of Fully Associative Caches	Joao
28.10.	17-920-414	Compiler Validation via Equivalence Modulo Inputs	Manuel
04.11.	17-916-818	MadMax: Surviving Out-of-Gas Conditions in Ethereum Smart Contracts	Sam
04.11.	17-916-412	Optimistic Loop Optimization	Joao
11.11.	17-916-131	T-Fuzz: Fuzzing by Program Transformation	Jingxuan
11.11.	17-914-003	Programming and Proving with Distributed Protocols	Marco
18.11.	17-913-930	Effective Program Debloating via Reinforcement Learning	Jingxuan
18.11.	17-913-823	Cradle: Cross-backend validation to detect and localize bugs in deep learning libraries	Pinjia
25.11.	17-913-542	FLAME: Formal linear algebra methods environment	Tyler
25.11.	17-913-534	One VM to rule them all	Manuel
02.12.	16-941-601	REMIX: Online Detection and Repair of Cache Contention for the JVM	Joao
02.12.	16-929-036	Communication lower bounds for distributed-memory matrix multiplication	Tyler
09.12.	17-948-167	DeepXplore: Automated Whitebox Testing of Deep Learning Systems	Pinjia
09.12.	16-915-175	TensorFuzz: Debugging Neural Networks with Coverage-Guided Fuzzing	Pinjia
16.12.	16-914-871	SecCSL: Security Concurrent Separation Logic	Marco
16.12.	15-938-426	Detecting Violations of Differential Privacy	Sam

- Preliminary schedule, check website for up-to-date version

Good Luck!

&

An Interesting Seminar