Research Topics in Software Engineering

Zhendong Su

Autumn Semester 2018

(based on Prof. Peter Müller's slides)

ETH zürich

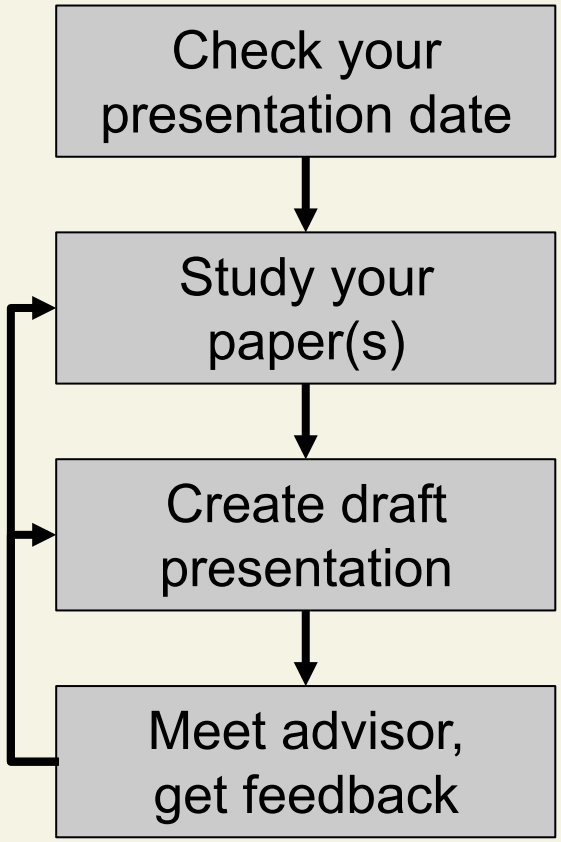
Objectives

- Learn to present technical work
- Learn to understand & evaluate research papers
- Learn several key research directions in the area

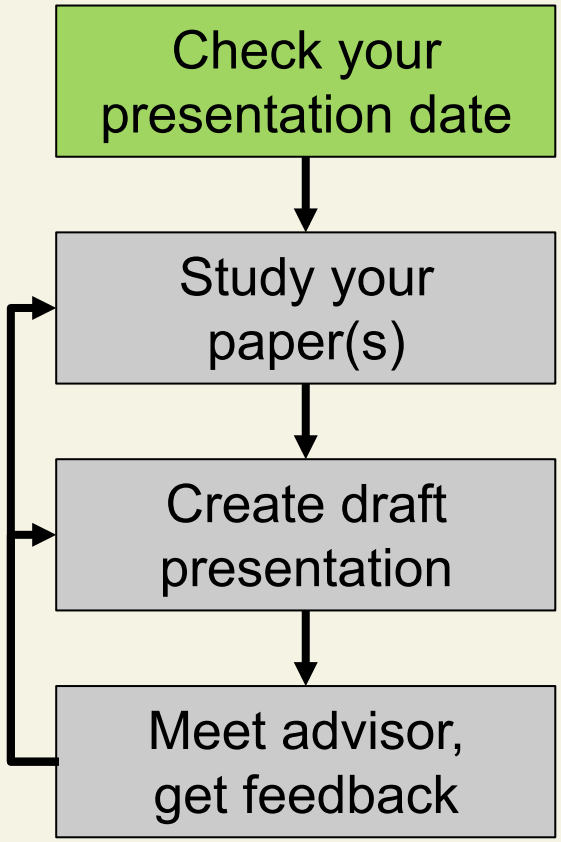
Objectives

- Learn to present technical work
- Learn to understand & evaluate research papers
- Learn several key research directions in the area
- Have a good deal of fun doing so

Preparing a Talk

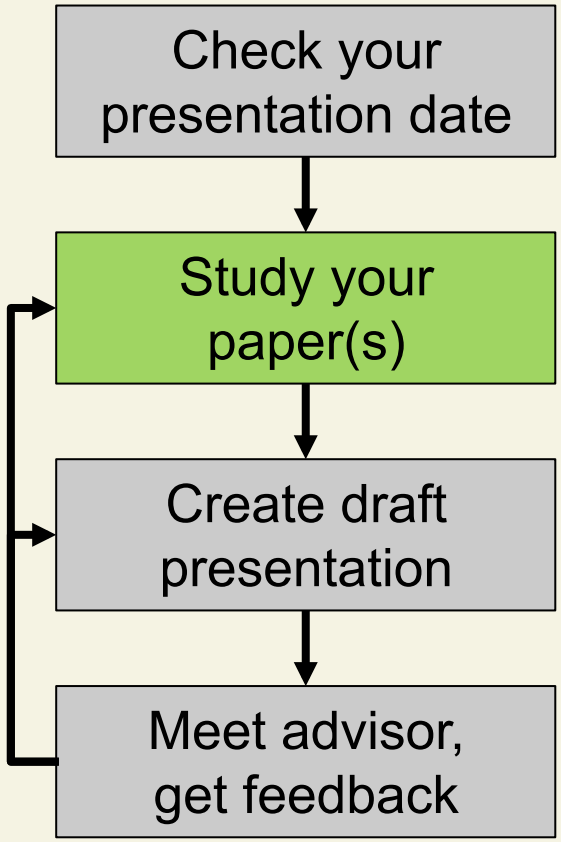


Preparing a Talk: **Start Early**



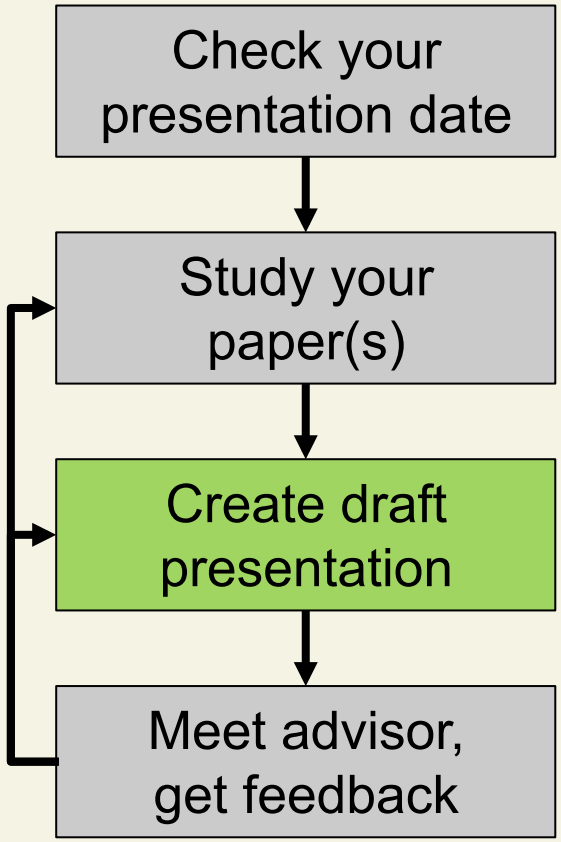
- Preparing a good presentation takes time
- So, please start early!

Preparing a Talk: **Study Paper**



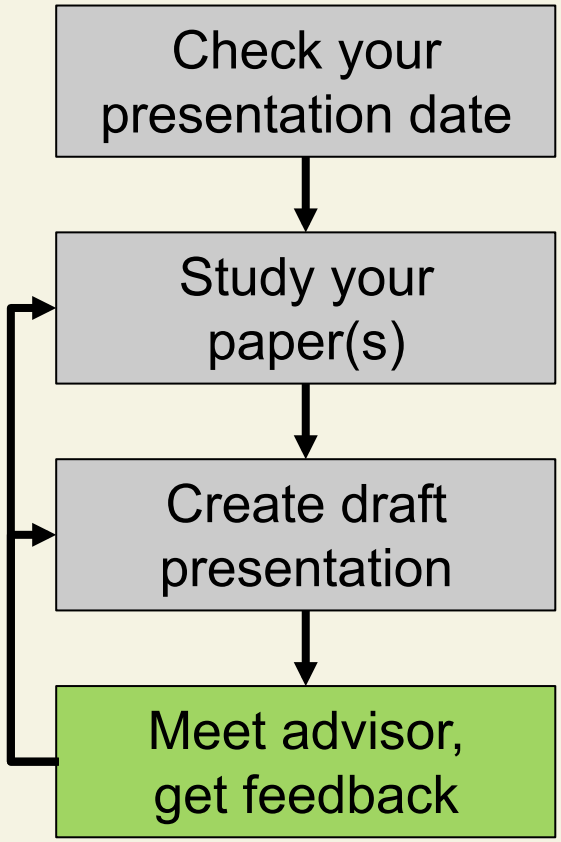
- 3 ‘C’s of reading
  - Carefully: look up terms, possibly read cited papers
  - Critically: find limitations, flaws
  - Creatively: think of improvements
- Try examples by hand
- Try tools if available
- Consult with TA if questions

Preparing a Talk: **Create Draft**



- Explain the work's motivation
- Clearly present the technical solution and results
  - Use own examples, not the ones in the paper
  - Include a demo if appropriate
- Discuss limitations or improvements
- Focus on key concepts
  - Do not present all the details

Preparing a Talk: Get Feedback



- Prepare for the meeting
  - Schedule early
  - Send slides in advance
  - Write down questions
- Do address feedback
  - Take notes
- Meeting is mandatory!
  - At least 1 week before the talk

Grading

- Quality of your presentation
  - How well did you understand the material?
  - How well did you present it?
  - How well did you answer the questions?
- Participation
  - Did you ask good questions?
  - Did you attend all sessions?
- We will take into account
  - the difficulty of the paper
  - suggestions you received from your TA advisor
  - time you had to prepare

Feedback

- Discuss your talk's strengths/weaknesses in-class
  - Let us know upfront if you prefer us not to
- Arrange a meeting with your TA to get feedback

Schedule

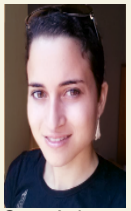
- Meet once per week with ~2 presentations each time
  - Next meeting on October 11
- Detailed schedule will be published online shortly
  - https://people.inf.ethz.ch/suz/teaching/263-2100.html
  - Including names of TA advisors

Workshop on Dependable and Secure Software Systems

Oct 19-20, 2018

Alumni Pavillon, Rämistrasse 101, ETH Zürich

ETH zürich



Sara Achour
MIT

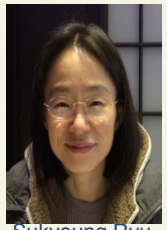
Marc Brockschmidt
MSR Cambridge

Matt Fredrikson
Carnegie Mellon

Chris Hawblitzel
MSR Redmond

Mike Hicks
University of Maryland

Suresh Jagannathan
Purdue University



Sukyoung Ryu
KAIST

Yannis Smaragdakis
University of Athens

Nikhil Swami
MSR Redmond

Andreas Zeller
Saarland University

Xiangyu Zhang
Purdue University

Harry Xu
UCLA

Peter Müller
ETH, co-organizer

Zhendong Su
ETH, co-organizer

Martin Vechev
ETH, co-organizer

- ✓ Big Data Program Analysis
- ✓ SMT Solver Tactics
- ✓ Machine Learning for Analysis
- ✓ Compilers for Analog Hardware
- ✓ Practical Verification for Systems
- ✓ Safety of Deep Learning
- ✓ Abstract Interpretation
- ✓ Security Analysis
- ✓ Blockchain Semantics

More information and registration: http://www.sri.inf.ethz.ch/workshop2018

Your Talk: **Timing**

- 30 min for talk
  - 1.5 ~ 2 min per slide
- 15 min for Q&A and discussion

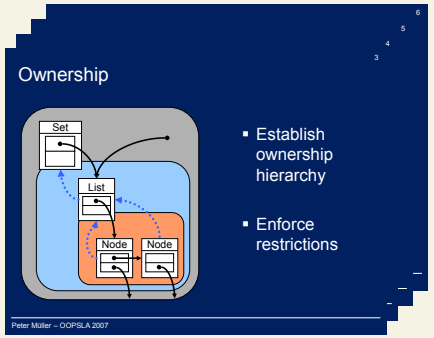
- The pace of talk is important
  - Too fast, the audience cannot follow
  - Too slow, people can get bored

- Practice your talk
  - Checkpoint after ~10 minutes

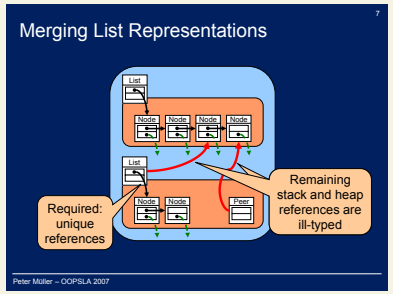
Your Talk: **Structure**

Title slide

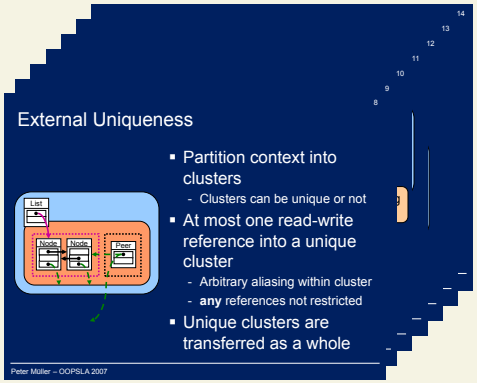
Splash



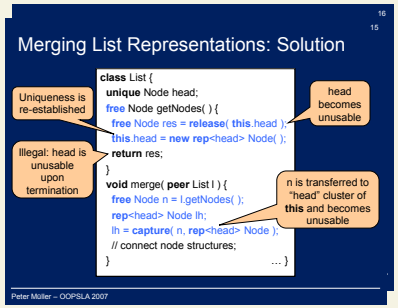
Motivation, background



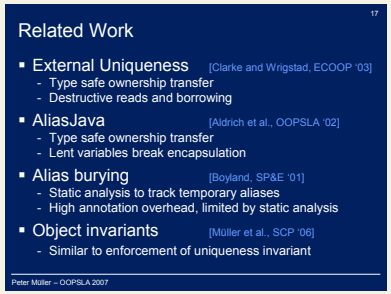
Problem



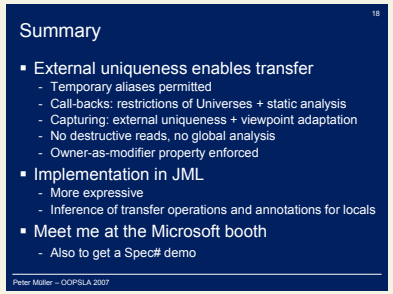
Solution



Evaluation, experiments, demo



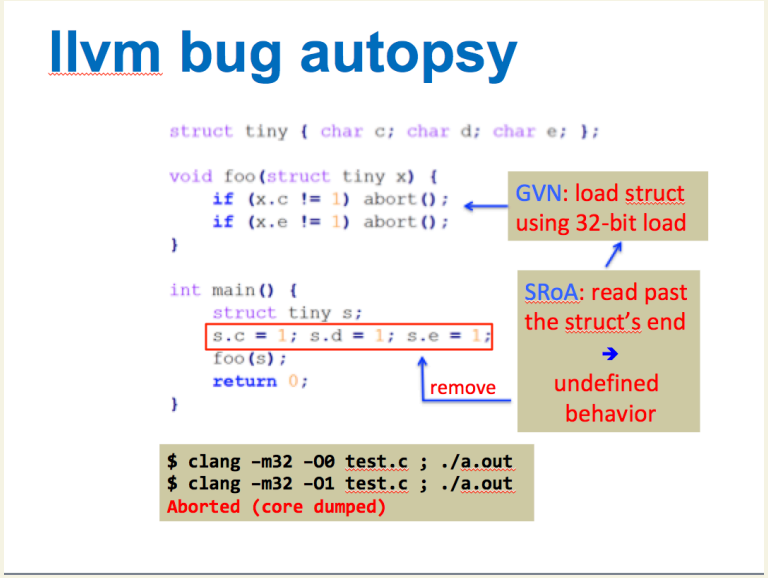
Related work



Summary, conclusion

Your Talk: **Examples**

- Examples are crucial for understanding
  - Both yours and the audience's
  - Prepare your own examples
- Try to find a running example
  - For motivation, problem, solution
  - Explain in detail (takes time)
- Reduce code example to the absolute necessary
  - Most people hate reading code
  - Use visualizations



Your Talk: **Design**

descriptive title

large font (> 20pt)

don't overload slide

visuals

llvm bug autopsy

```
struct tiny { char c; char d; char e; };

void foo(struct tiny x) {
    if (x.c != 1) abort();
    if (x.e != 1) abort();
}

int main() {
    struct tiny s;
    s.c = 1; s.d = 1; s.e = 1;
    foo(s);
    return 0;
}
```

GVN: load struct using 32-bit load

SRoA: read past the struct's end → undefined behavior

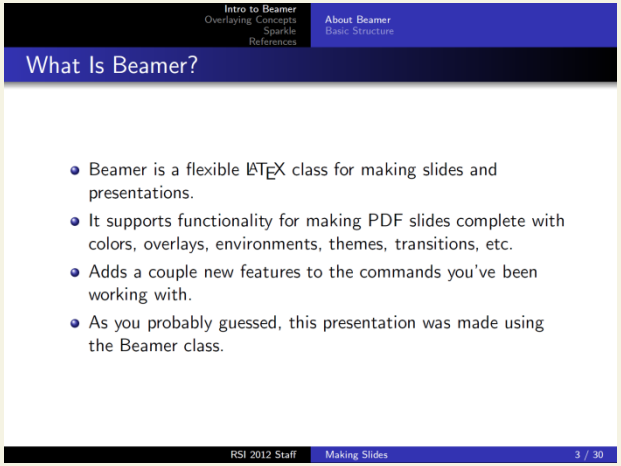
remove

```
$ clang -m32 -O0 test.c ; ./a.out
$ clang -m32 -O1 test.c ; ./a.out
Aborted (core dumped)
```

Powerpoint vs. Latex

- Powerpoint
  - Visualizations and animations are easy
  - Don’t over-do it!

- Latex
  - Visualizations and animations are painful
  - Don’t under-do it!



Your Talk: **Avoid Frequent Mistakes**

- Don’t try to present all details
  - Focus on the key messages:
    motivation, problem, main idea, main result
- Don’t stare at the screen or your laptop
  - Look at the audience
- Come prepared
  - Study paper in depth
  - Rehearse your talk

References

Markus Püschel's small guide on giving talks

http://www.inf.ethz.ch/personal/markusp/teaching/guides/guide-presentations.pdf

Highly recommended!

Paper Pool for Fall 2018

1. DART: Directed Automated Random Testing. PLDI 2005.
2. Towards Optimal Concolic Testing. ICSE 2018.
3. Coverage-based Greybox Fuzzing as Markov Chain. CCS 2016.
4. PerfFuzz: Automatically Generating Pathological Inputs. ISSTA 2018.
5. Fairness testing: testing software for discrimination. ESEC/FSE 2017.
6. Efficient Sampling of SAT Solutions for Testing. ICSE 2018.
7. Compiler Validation via Equivalence Modulo Inputs. PLDI 2014.
8. Finding missed compiler optimizations by differential testing. CC 2018.
9. Provably correct peephole optimizations with Alive. PLDI 2015.
10. Automatically improving accuracy for floating point expressions. PLDI 2015.
11. Achieving high coverage for floating-point code via unconstrained programming. PLDI 2017.
12. A Comprehensive Study of Real-World Numerical Bug Characteristics. ASE 2017.
13. Just-in-Time Static Analysis. ISSTA 2017.
14. Pinpoint: Fast and Precise Sparse Value Flow Analysis for Million Lines of Code. PLDI 2018.
15. Securify: Practical Security Analysis of Smart Contracts. CCS 2018.
16. DeepXplore: Automated Whitebox Testing of Deep Learning Systems. SOSP 2017.
17. AI2: Safety & Robustness Certification of Neural Networks with Abstract Interpretation. S&P 2018.
18. Formal Security Analysis of Neural Networks using Symbolic Intervals. USENIX Security 2018.
19. Programmatic and Direct Manipulation, Together at Last. PLDI 2016.
20. Automatic patch generation by learning correct code. POPL 2016.
21. Neural Sketch Learning for Conditional Program Generation. ICLR 2018.
22. Debugging reinvented. ICSE 2008.
23. COZ: Finding Code that Counts with Causal Profiling. SOSP 2015.
24. Towards optimization-safe systems: analyzing the impact of undefined behavior. SOSP 2013.
25. What You Get is What You C: Controlling Side Effects in Mainstream C Compilers. S&P 2018.
26. Into the Depths of C: Elaborating the De Facto Standards. PLDI 2016.
27. Bringing the web up to speed with WebAssembly. PLDI 2017.