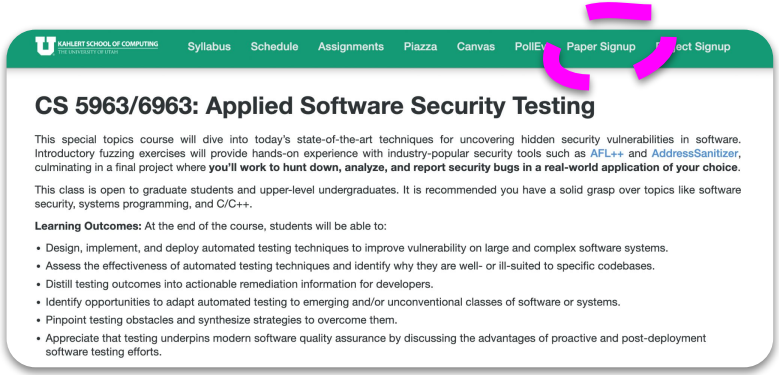
Week 2: Lecture B

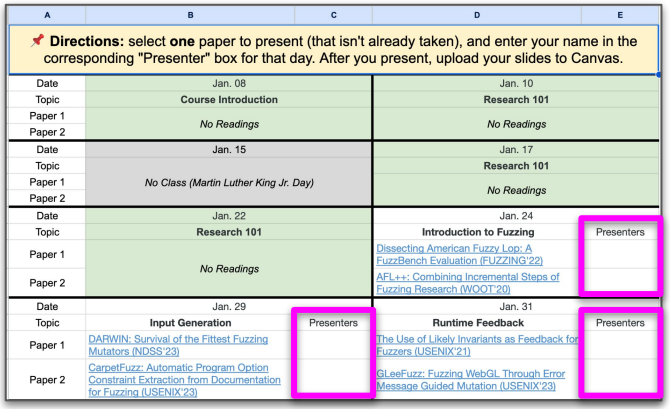
Research 101: Reviewing & Presenting

Wednesday, January 15, 2025

Recap: Paper Presentations

- **Signup sheet** available on course website (must use **UofU gcloud** account)
  - **40-45 fuzzing papers** from top venues in security, software engineering, and PL
- Choose one paper by **Wednesday, January 15**
  - **CS 5963 students:** must team-up in pairs of **two**
  - **CS 6963 students:** must present paper **solo**





Recap: Paper Presentations

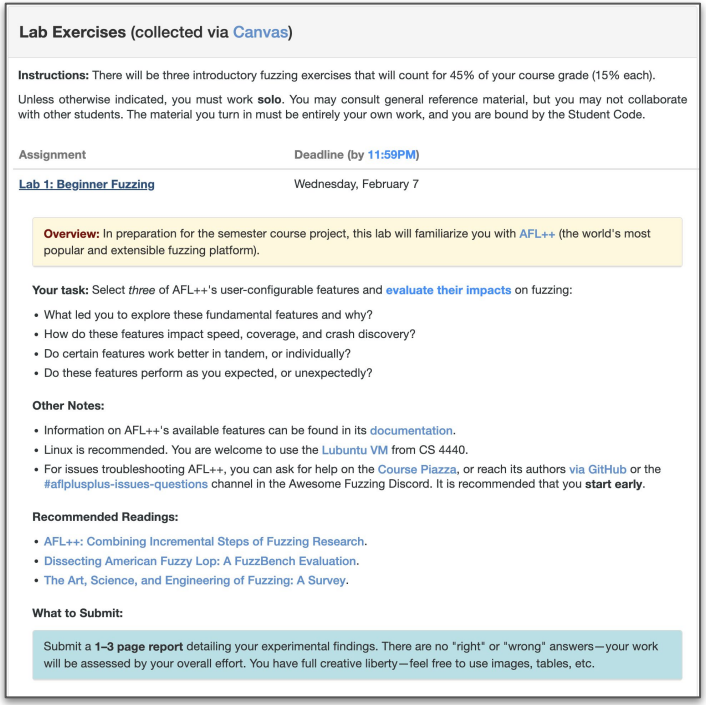
- **Signup sheet** available on course website
  - **40-45 fuzzing papers** from top venues in security, software engineering, and PL
- Choose one paper by **Wednesday, January 15**
  - **CS 5963 students:** must team-up in groups of 2
  - **CS 6963 students:** must present paper solo

(must use **UofU gcloud** account)

Lab 1: Beginner Fuzzing

Lab 1: Beginner Fuzzing

- See **Assignments** tab on course website
  - Click the drop-down link for Lab 1
- Deadline: **Wednesday, January 29**
  - Submit on **Canvas** by **11:59 PM MST**
  - Late assignments **are not accepted**



Lab Exercises (collected via Canvas)

Instructions: There will be three introductory fuzzing exercises that will count for 45% of your course grade (15% each).

Unless otherwise indicated, you must work **solo**. You may consult general reference material, but you may not collaborate with other students. The material you turn in must be entirely your own work, and you are bound by the Student Code.

Assignment	Deadline (by 11:59PM)
Lab 1: Beginner Fuzzing	Wednesday, February 7

Overview: In preparation for the semester course project, this lab will familiarize you with AFL++ (the world's most popular and extensible fuzzing platform).

Your task: Select *three* of AFL++'s user-configurable features and **evaluate their impacts** on fuzzing:

- What led you to explore these fundamental features and why?
- How do these features impact speed, coverage, and crash discovery?
- Do certain features work better in tandem, or individually?
- Do these features perform as you expected, or unexpectedly?

Other Notes:

- Information on AFL++'s available features can be found in its documentation.
- Linux is recommended. You are welcome to use the Lubuntu VM from CS 4440.
- For issues troubleshooting AFL++, you can ask for help on the Course Piazza, or reach its authors via GitHub or the #aflplusplus-issues-questions channel in the Awesome Fuzzing Discord. It is recommended that you **start early**.

Recommended Readings:

- AFL++: Combining Incremental Steps of Fuzzing Research.
- Dissecting American Fuzzy Lop: A FuzzBench Evaluation.
- The Art, Science, and Engineering of Fuzzing: A Survey.

What to Submit:

Submit a **1–3 page report** detailing your experimental findings. There are no "right" or "wrong" answers—your work will be assessed by your overall effort. You have full creative liberty—feel free to use images, tables, etc.

Lab 1: Beginner Fuzzing

- **Assignment:** familiarize yourself with AFL++
  - Read its documentation in **docs/**
- **Pick three features, try them out, and discuss your findings**
  - E.g., impacts on code coverage, speed, crash discovery
  - What insights do you have?
  - Why did one feature work better than another?
- **Deliverable:** a **1–3 page report** detailing your findings
  - Feel free to make it your own (e.g., pictures, text, etc.)
- Need a Linux environment
  - Use the **CS 4440 Lubuntu VM** if you don't have one

Lab 1: Introduction to Fuzzing

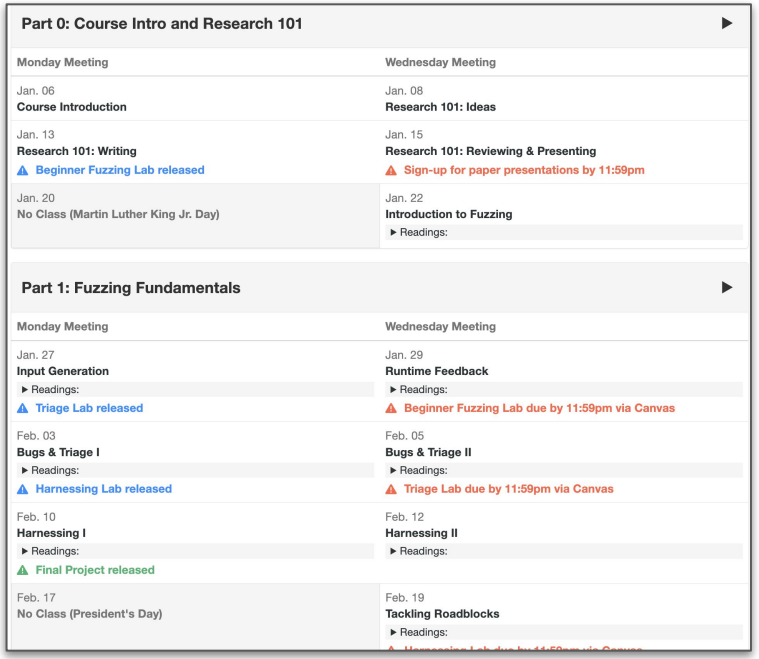
- Primary goal: **prepare you for the semester project**

- Other goals:
  - Give you experience with industry-standard tools
  - Put you in the “research” mindset
  - Improve your debugging skills

Recap: Key Dates

- **Jan. 15** — **Select one paper to present**
- **Jan. 20** — No class (MLK Jr. Day)
- **Jan. 29** — Lab 1 due
- **Feb. 05** — Lab 2 due
- **Feb. 17** — No class (President's Day)
- **Feb. 19** — Lab 3 due
- **Feb. 26** — **5-minute project pitches**
- **Mar. 10 & 12** — No class (Spring Break)
- **Apr. 14 & 21** — **Final project presentations**

cs.utah.edu/~snagy/courses/cs5963/schedule



Part 0: Course Intro and Research 101

Monday Meeting	Wednesday Meeting
Jan. 06 **Course Introduction**	Jan. 08 **Research 101: Ideas**
Jan. 13 **Research 101: Writing** Beginner Fuzzing Lab released	Jan. 15 **Research 101: Reviewing & Presenting** Sign-up for paper presentations by 11:59pm
Jan. 20 **No Class (Martin Luther King Jr. Day)**	Jan. 22 **Introduction to Fuzzing** ▸ Readings:

Part 1: Fuzzing Fundamentals

Monday Meeting	Wednesday Meeting
Jan. 27 **Input Generation** ▸ Readings: Triage Lab released	Jan. 29 **Runtime Feedback** ▸ Readings: Beginner Fuzzing Lab due by 11:59pm via Canvas
Feb. 03 **Bugs & Triage I** ▸ Readings: Harnessing Lab released	Feb. 05 **Bugs & Triage II** ▸ Readings: Triage Lab due by 11:59pm via Canvas
Feb. 10 **Harnessing I** ▸ Readings: Final Project released	Feb. 12 **Harnessing II** ▸ Readings:
Feb. 17 **No Class (President's Day)**	Feb. 19 **Tackling Roadblocks** ▸ Readings:

Questions?

Research Reviewing

Why review research?

Reviewers are the essential gatekeepers that make our research system work

Why review research?

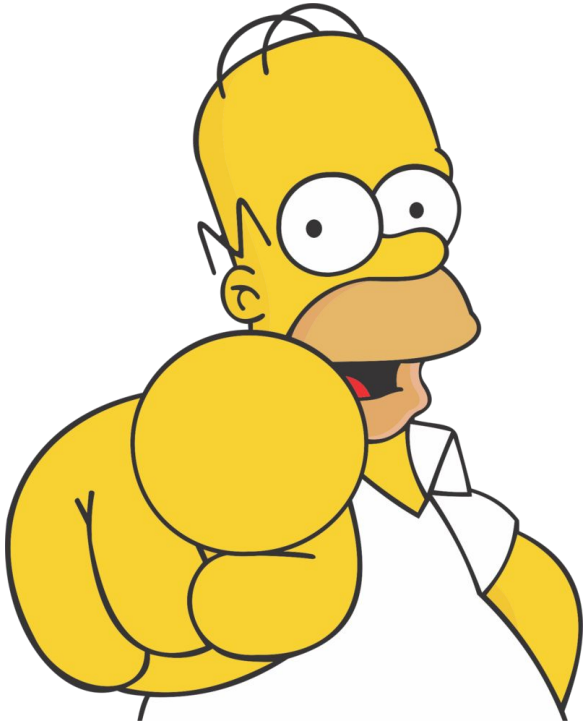
- **Idea #1:** accept **everything**
  - No way to keep up
  - Risk (more) flawed results
  - Each reader must gauge what a paper's value is
  - All work stays at a local maximum—no advances
  - How do we identify/reward/encourage **the best**?

Why review research?

- **Idea #1:** accept **everything**
  - No way to keep up
  - Risk (more) flawed results
  - Each reader must gauge what a paper's value is
  - All work stays at a local maximum—no advances
  - How do we identify/reward/encourage **the best**?

- **Idea #2:** accept **nothing**
  - Science stalls

Why am I covering this here?



Reviewing is a process...

1. Does the problem they're solving **matter**?
   - Will it matter in the future?
2. Are their claimed contributions **enough**?
   - Is the work incremental?
   - Are they throwing too many things at the wall?
3. Are their claimed contributions **supported**?
   - Design decisions
   - Evaluation results
   - Motivating experiments

What makes a problem important?

- Timely
  - E.g., Meltdown and Spectre
  - In-browser crypto mining malware
- An obvious “next step”
- Contests a common assumption
- Must be **surprising** in some way
- Opens a new and realistic line of research

Are the claimed contributions enough?

- Are they actually **new**?
- Do the contributions **push the area** forward?
- Do they **open a new area** of investigation?

Does the system support the contributions?

- Watch out for a design **bait-and-switch**
  - Intro mentions **X**, but the authors implement **Y**
  - But **Y != X** in meaningful ways

Does the system support the contributions?

- Watch out for a design **bait-and-switch**
  - Intro mentions **X**, but the authors implement **Y**
  - But **Y != X** in meaningful ways

- **Evaluation** funny business
  - Do the authors evaluate **what they design**?
  - E.g., they fabricate **a physical chip**, but use **simulations** in all experiments

Does the system support the contributions?

- Watch out for a design **bait-and-switch**
  - Intro mentions **X**, but the authors implement **Y**
  - But **Y != X** in meaningful ways
- **Evaluation** funny business
  - Do the authors evaluate **what they design**?
  - E.g., they fabricate **a physical chip**, but use **simulations** in all experiments
- Is the evaluation **fair**?
  - We use these benchmarks [which behave in a way that suits our system]
  - We allocate 1 GB of memory [our competitor is memory limited]
  - Watch out for how **randomness** can influence results

Fixable, but grave sins...

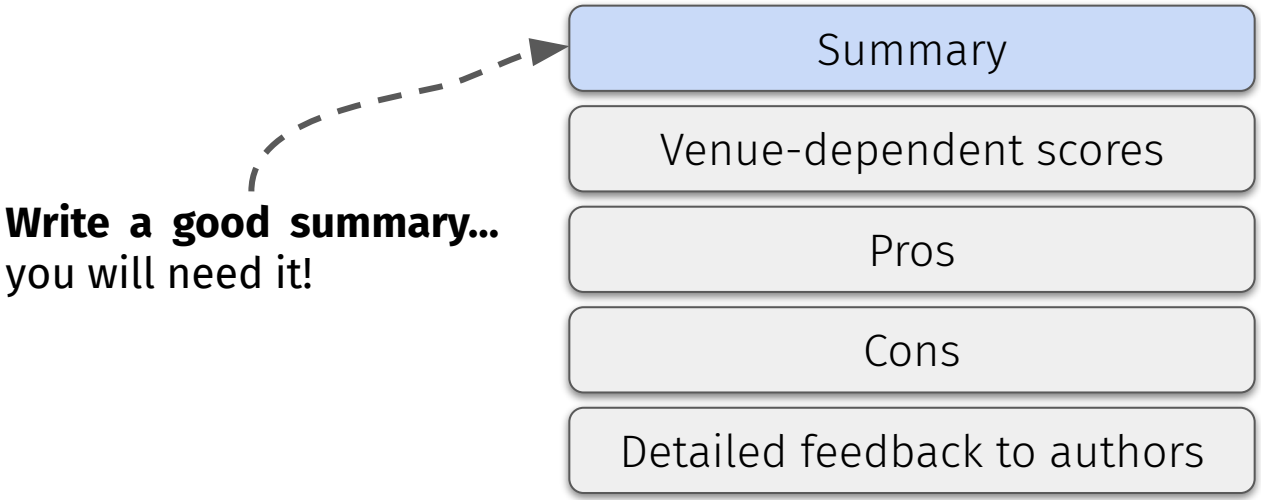
- Must identify all **stated** and **unstated** assumptions
  - This can break a paper
  - Or be easily fixable in a revision

- Are all **assertions** made in the paper supported by data?
  - Prevent future papers from citing an unsupported statement in this paper
  - “The Dobber method is superior to the Fastly method.”

What goes into a review...

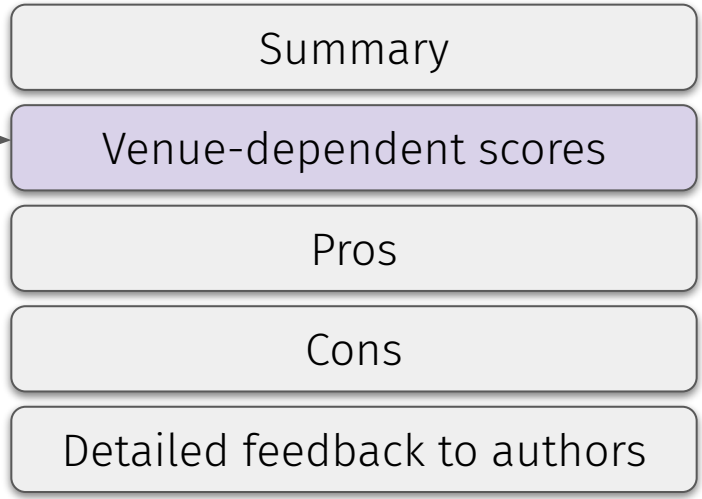
- Summary
- Venue-dependent scores
- Pros
- Cons
- Detailed feedback to authors

What goes into a review...



Write a good summary...
you will need it!

What goes into a review...



- **Reject:** just stop
- **Weak reject:** the paper isn't not ready yet
- **Weak accept:** it's ready but not compelling
- **Accept:** will argue for

“The Bar” depends on the Venue

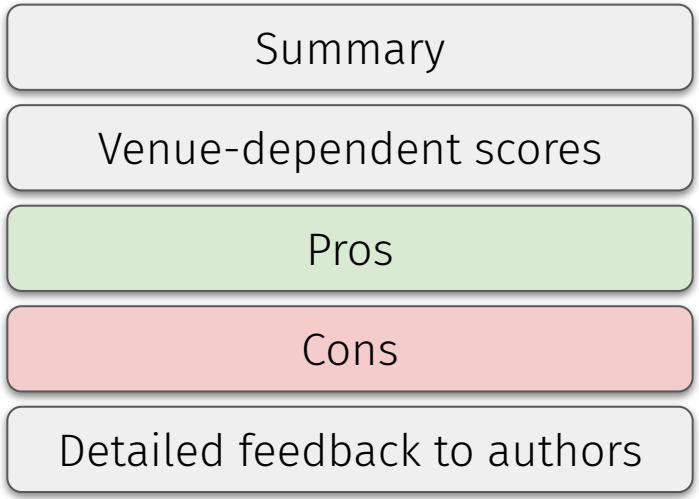
7. Overall, how good is it? What do you recommend?

Can you put the paper into one of these categories?

1. Major results - very significant. (fewer than 1% of all papers written.)
2. Good, solid, interesting work; a definite contribution. (fewer than 10% of the papers you will see.)
3. Minor, but positive, contribution to knowledge. (perhaps 10% to 30% of the papers submitted.)
4. Elegant and technically correct but useless. This category includes sophisticated analyses of flying pigs, as mentioned above.
5. Neither elegant nor useful, but not actually wrong.
6. Wrong and misleading.
7. The paper is so badly written that a technical evaluation is impossible.

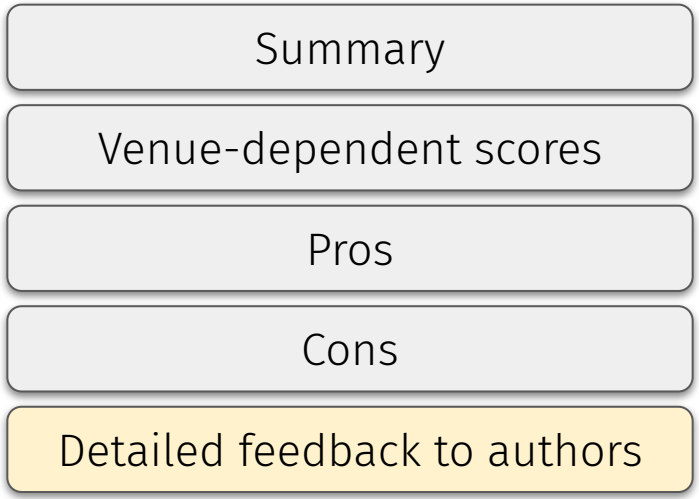
Source: The Task of the Referee, A. J. Smith

What goes into a review...



Main points that other PC members will read

What goes into a review...



Be helpful, but don't go out of your way for blatantly unpublishable papers

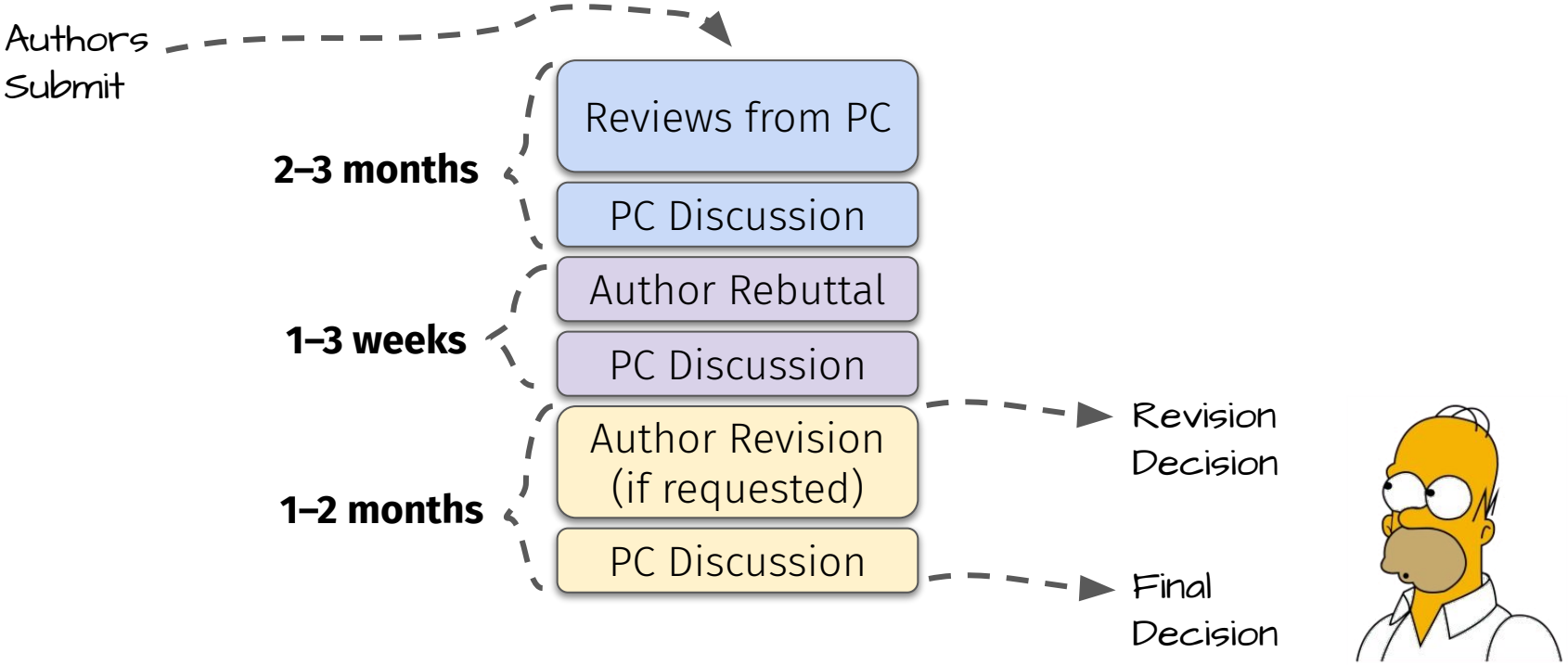
Things good reviewers do…

- **Be constructive, concrete, and courteous**
- Spend time with borderline papers
- Help authors improve their paper
  - “Use this tool”
  - “Evaluate your system this way”
  - “Pitch your contribution this way”
  - “You can fix your system by doing…”
  - “Cover this related work [1], it relates to your paper this way”
- Don’t just say something exists… point to it!

Things bad reviewers say...

- “This is the worst paper I’ve read”
- “I can do it better, so let’s reject it”
- “Here is every single grammar error”
- “They didn't work on the ‘right’ problem”
- “I don’t like this inconsequential low-level detail”
- “Here’s a review I wrote for a previous version of the paper”
- “They didn’t cite these non-peer-reviewed works” (e.g., arXiv)
- “It’s too similar to these other works that I didn’t actually read”

Conference Reviewing Cycle



A final note...

It is easier to be a detractor than to be a champion; **be a champion**!

Questions?

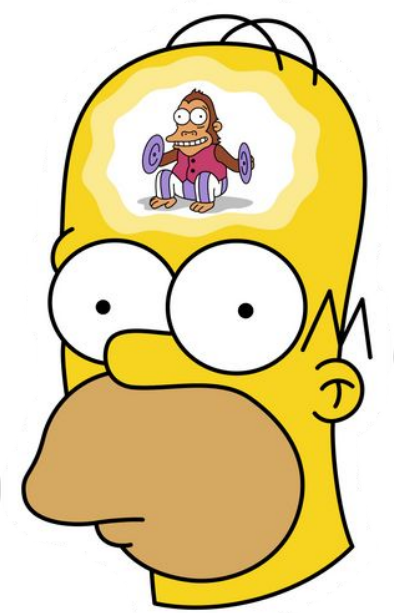
Presenting Research

Why present your work?

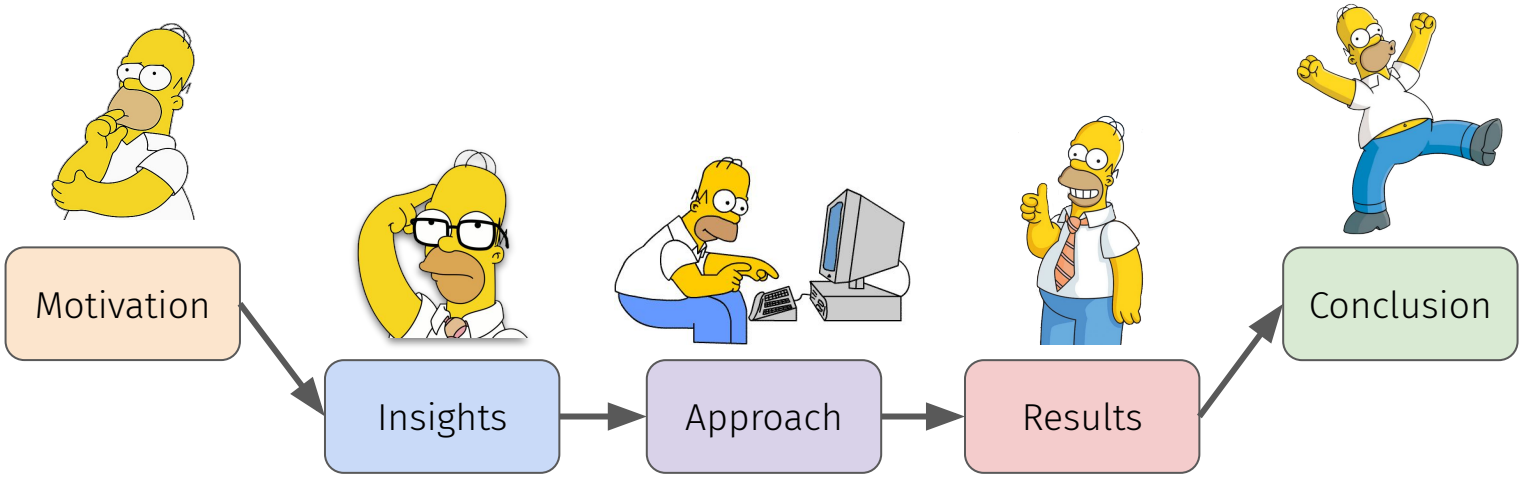
If you don’t publicize your work, then how will people know they should read it?

Why present your work?

- Document and **communicate what you did**
- Convince others that **they should go deeper**
  - Read your work
  - Fund your work
  - Build off your work
  - Hire you to do more work
- Facilitate others **spreading your message**
  - Reading groups and seminars
  - The Twittersphere
- **Not to show how smart you are!**



Building good presentations is a process



Before you start: The Tagline

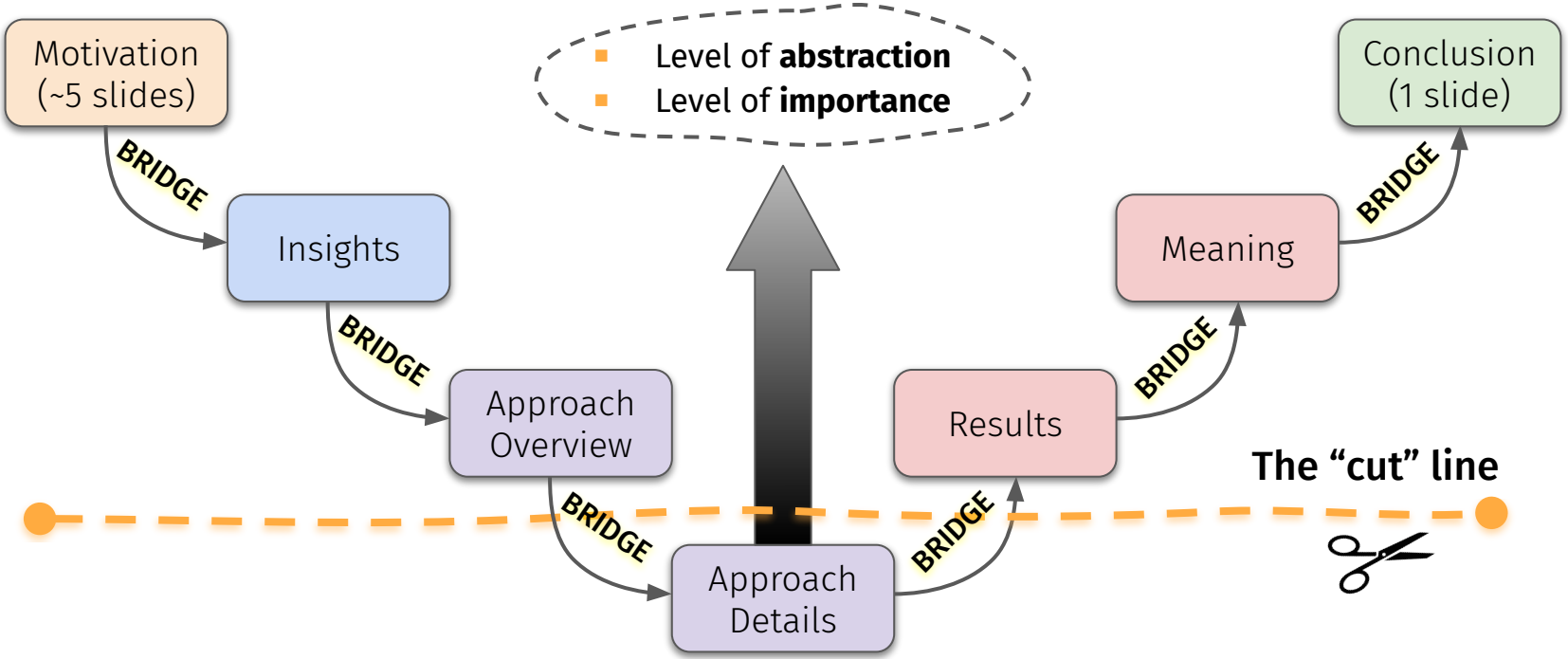
- **What is your talk's tagline?**
- What idea will the entire audience **understand**?
- Reiterate it throughout your talk!

"There can only be ONE (talk tagline)"

Know thy Audience

- What is their **background**?
  - **Expert:** someone who knows all the jargon
  - **Non-expert:** clueless (e.g., your non-CS friends)
  - **Tailor your technical jargon accordingly**
- Why should they **care**?
- What are they **expecting**?
  - How long of a talk
  - What level of quality

Structure presentations to be cut for time...



Tell, tell, tell...

1. Tell them what you **will tell** them
2. **Tell them**
3. Tell them what you **told them**

- **Bridge:** tell them what you **told them**, and what you will tell **next**

I'll tell you my FAVORITE kind of donut....

The frosted ones!

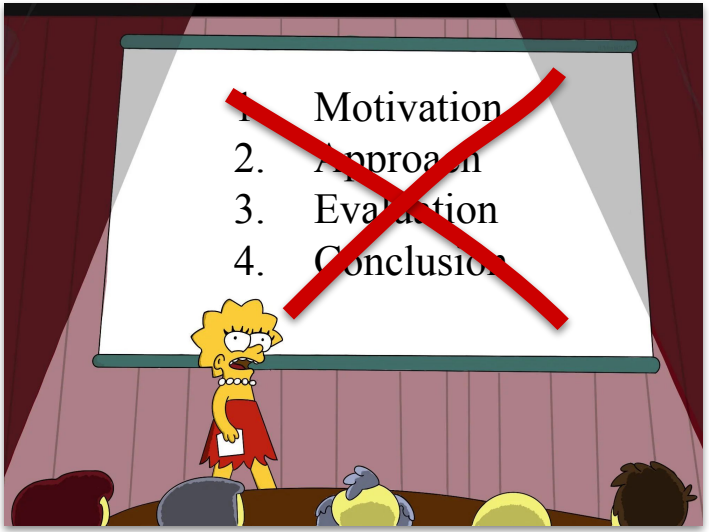
So now that I've told you my favorite donuts...

I'll tell you about my favorite beverage!

Outline Creatively

- Does your audience require an **outline**?
  - **Short talks**: no outline
  - **Longer talks**: use an outline

- Developing outlines
  - **Tell, tell, tell** can be an outline
  - **Bridges** offer a localized outline
  - **Don't just list your section titles!**



Evaluations Must Tell a Story

- What question are you answering and why?
- How did you setup your experiment?
- What are the **important** results?
  - What did you expect to happen?
  - Draw attention to key/interesting results
  - **Don't just reuse your paper's results**
  - **Always explain how to interpret charts**

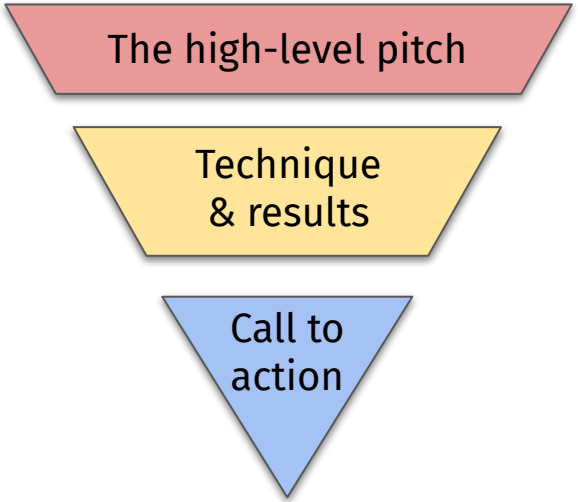
Evaluations Must Tell a Story

- What do the results suggest?
  - E.g., “Improvement over Conventional Testing”
  - **Incorporate this in slide titles**
- **Bridge evaluation questions**
  - E.g., “We know X and Y… but what about Z?”
- Order questions by importance
  - E.g., “Does it work?” before “How fast is it?”

Conclude with a statement bigger than your work!

- Tell them what you told them

- Make a **call-to-action** statement
  - What do your results **make possible**?
  - What **impact** on the world do your results have?
  - What **new research** will stem from your work?



Mention only key related work—but be aware of it *all*

- You don’t have time for a related work slide
  - Most conference presentations are 10—20 minutes max
  - If you must, add it as a backup slide
- Mention related work in the opening and along the way
  - Mention important authors (or tools) by name
  - **Be positive about prior work**
- Don’t worry about mentioning every piece of related work
  - That’s what the paper is for
  - As your talk gets deeper, focus only on the key related work

Backup slides are to be seen—not heard!

- Flipping around in your slides looks bad
  - Avoid going backwards through your presentation
  - Sometimes the audience will ask you to

- Have backup slides, but **avoid using them**
  - Treat them as you would paper appendix sections
  - Be aware that **they will end up in the final PDF**

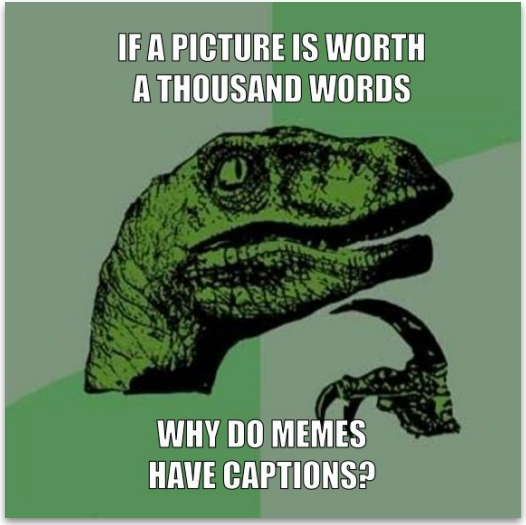
Presentation Tips

Examples help audiences understand!

- Introduce a simple running example
  - Gradually add complexity
  - Refer to it for each new point
- Have a central motif for your presentation
- Make sure your example is **correct**
  - Critical to your audience's mental model

Slides only support your talk!

- You give the talk—**slides are just visual support**
- Humans read words on a slide to themselves
  - … while you are trying to talk to them
- Humans remember **pictures** better than text
  - Higher-quality graphics = higher-quality presentation
- If you must have text, be concise!
  - Like paragraphs, **each slide should make *one* point**



Text must infer meaning!

- Use font differences to communicate meaning and association
  - **Bold** and underline = important
  - Larger is more important than smaller
  - **Red** = bad, **green** = good
  - `Monospace` font = this is code;
  - Call-out boxes draw attention
- Be consistent!
- Avoid font sizes smaller than 14pt

Use presentation guardrails!

- Each slide must have **one clear and concise reason** for existing
  - Keeps the talk on track
  - Less memorization for you
  - Easier for the audience to follow
  - Easier to edit and cut
- Designate specific slides as **time checkpoints**
  - E.g., “at 5:00 minutes be on slide 6”
  - Use a stopwatch (e.g., your phone) to make sure you’re on track
- **Know when to cut content for time**

Design your slides to be “flattened”...

- Your slides will be published as a PDF
- Compress your images!
  - No one wants a 200mb PDF
- PDFs don’t support animation
  - Animations get flattened onto a single slide
  - Can hide content
  - Solution: **split animations into multiple slides**

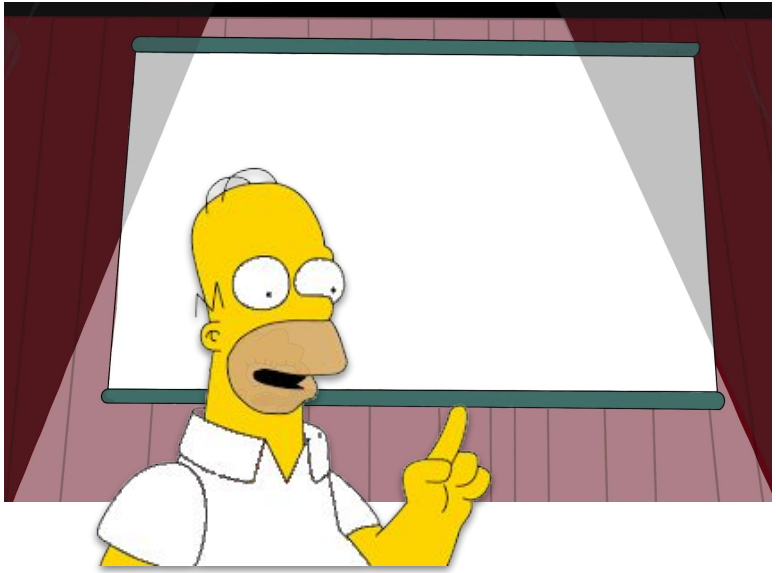
Always number your slides!

- Make references to your slides easy
  - Slide feedback
  - Audience questions

Presenter Tips

Ditch the podium—be passionate!

Ditch the laser—get into your slides!

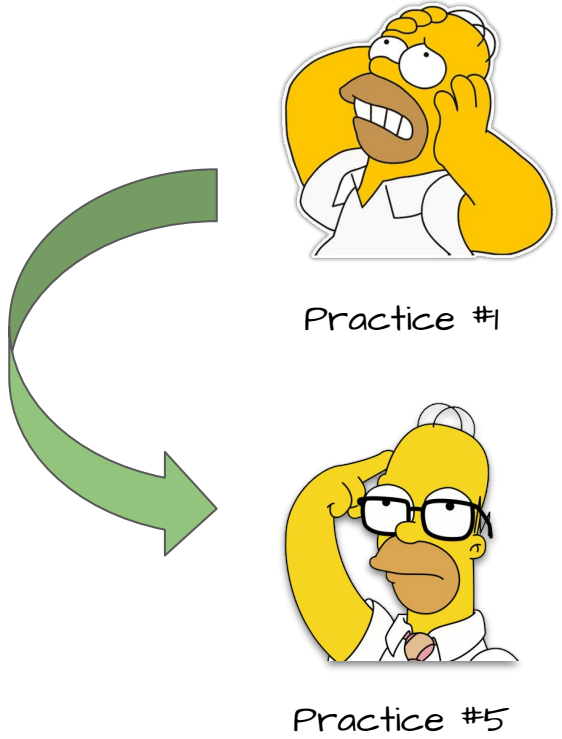


Fielding Questions

- Practice answering **questions you'd expect to see**
- Answering the audience's questions is a dance
  - Some questions are intentionally adversarial
  - **Repeat the question** and ask if your understanding is correct
  - Better yet: **rephrase it to "better" question**, and answer that
  - Always **be friendly!**
- **Confidence comes with practice**

Practice, practice, practice!

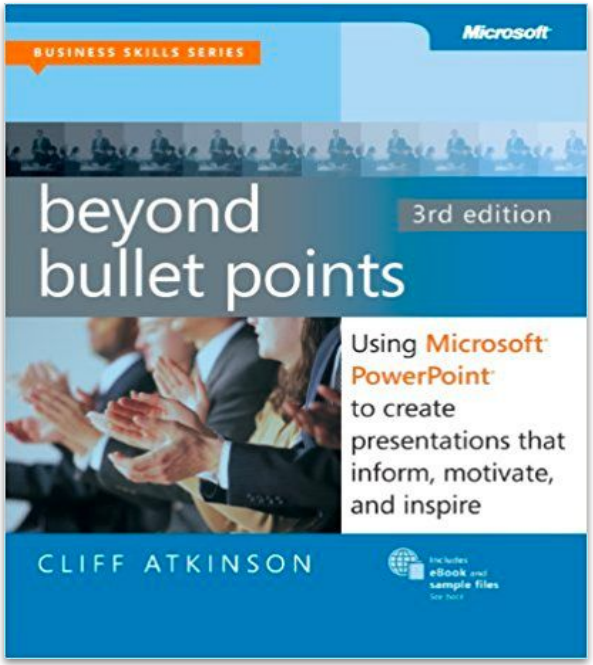
- Like any good performance, **memorize your lines**!
  - Create a **short** script and read through it several times
- When you're ready, ditch the script
  - I try to memorize one slide at a time
- Practice with different audiences
  - Your lab, reading groups, friends
- **Repeat!**

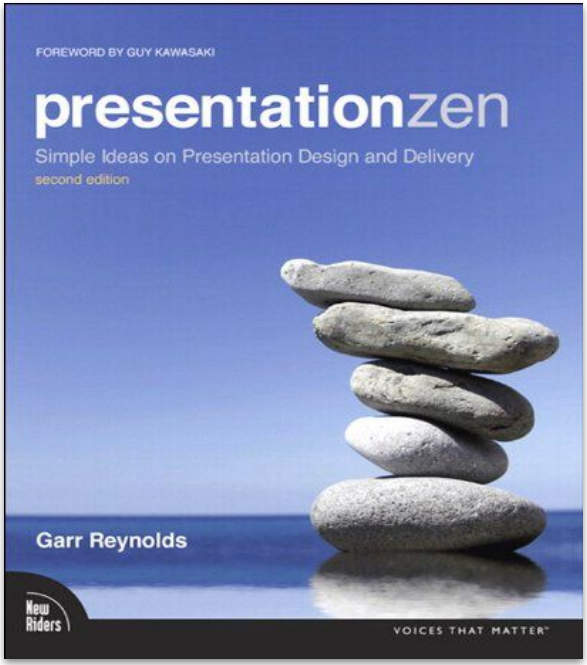


Advertise yourself!

- Introduction: **what you're seeking**
  - "I'm on the job market this year"
  - "I'm seeking internships this summer"
  - **Ask the session chair to mention this**
- Conclusion: **relevant links**
  - Link to your prototype's source code
  - QR code to link to your website
  - Your Twitter handle
  - **Remind the audience what you're seeking**

Presentation Resources





Presentation Resources

- Great talk on technical presentations:
  - https://www.youtube.com/watch?v=Unzc731iCUY
- Tips from hucksters:
  - https://www.youtube.com/watch?v=vC5cmW8O3L8
- Telling a story:
  - https://www.youtube.com/watch?v=YDXNJBmuV4Q
- How to start:
  - https://www.youtube.com/watch?v=w82a1FT5o88

Questions?

Next time on CS 5963/6963...

Introduction to Fuzzing