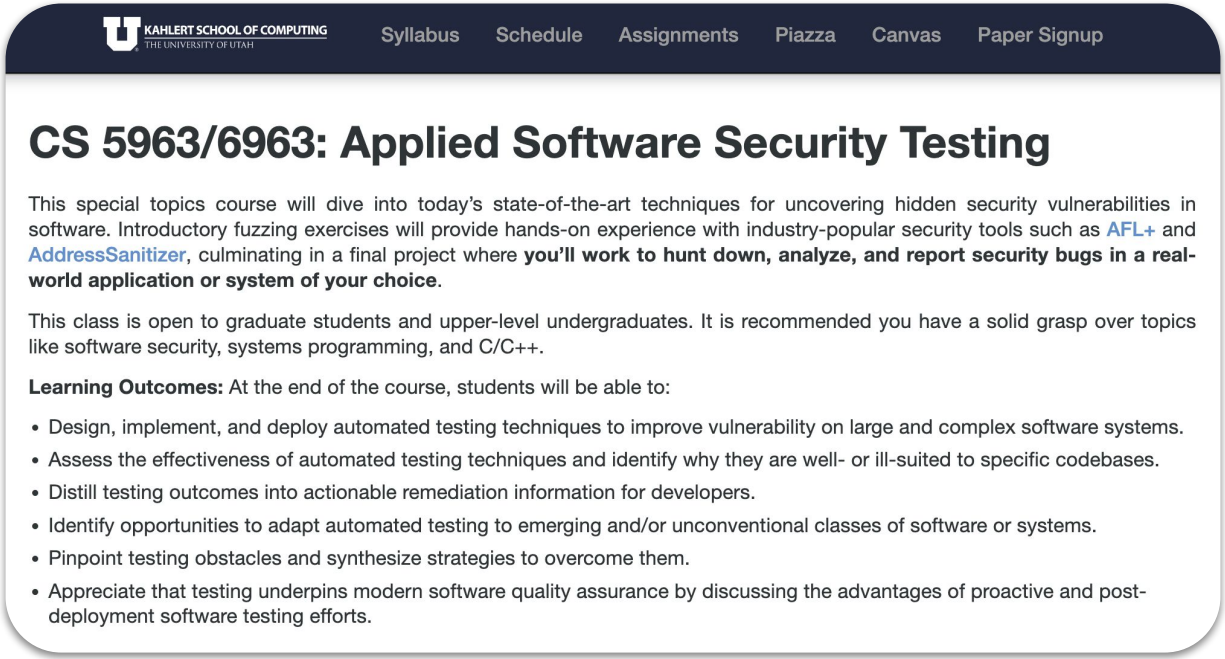
Week 2: Lecture B

Research 101: Writing

Wednesday, January 17, 2024

Recap: Course Website

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



Recap: Course Resources

Course website ..... assignments, schedule, slides, paper signup

Piazza ..... questions, discussion, announcements

Canvas ..... homework submission, course gradebook

Instructor email (snagy@cs.utah.edu**)** ..... administrative issues

Recap: Lateness Policy

- Assignments will be posted on course website
  - See cs.utah.edu/~snagy/courses/cs5963/assignments
- Due by **11:59 PM** on the specified deadline date
  - Late assignments will **not** be accepted
- If you are sick / traveling / abducted by aliens…
  - Try to keep me posted and we will figure something out

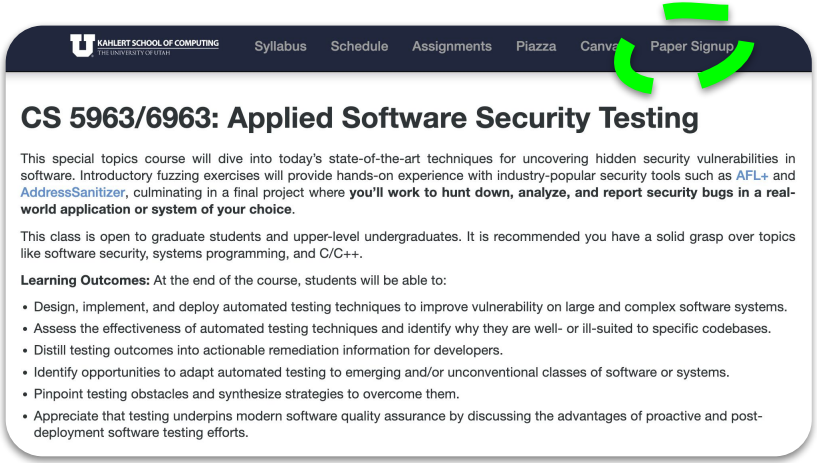
Recap: Course Materials

- No textbook is required for this course
- Some excellent resources on fuzzing are:
  - **The Fuzzing Book** by Zeller, Gopinath, Böhme, Fraser, and Holler
  - **Fuzzing Against the Machine** by Antonio Nappa and Blazquez
- Other general computer security textbooks:
  - **Introduction to Computer Security** by Goodrich and Tamassia
  - **Security Engineering** by Ross Anderson
- These are are linked on the course syllabus
  - cs.utah.edu/~snagy/courses/cs5963/

Recap: No Exams

Recap: Paper Presentations

- **Signup sheet** available on course website (must use **UofU gcloud** account)
  - **38 fuzzing papers** from top venues in security, software engineering, and some workshops
  - Choose one paper by **Monday, January 22**



KAHLERT SCHOOL OF COMPUTING
THE UNIVERSITY OF UTAH

Syllabus | Schedule | Assignments | Piazza | Canvas | Paper Signup

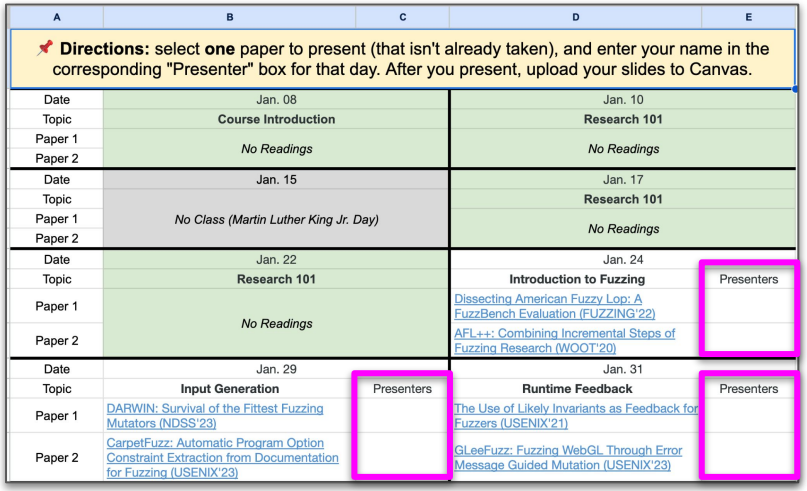
CS 5963/6963: Applied Software Security Testing

This special topics course will dive into today's state-of-the-art techniques for uncovering hidden security vulnerabilities in software. Introductory fuzzing exercises will provide hands-on experience with industry-popular security tools such as AFL+ and AddressSanitizer, culminating in a final project where **you'll work to hunt down, analyze, and report security bugs in a real-world application or system of your choice**.

This class is open to graduate students and upper-level undergraduates. It is recommended you have a solid grasp over topics like software security, systems programming, and C/C++.

Learning Outcomes: At the end of the course, students will be able to:

- Design, implement, and deploy automated testing techniques to improve vulnerability on large and complex software systems.
- Assess the effectiveness of automated testing techniques and identify why they are well- or ill-suited to specific codebases.
- Distill testing outcomes into actionable remediation information for developers.
- Identify opportunities to adapt automated testing to emerging and/or unconventional classes of software or systems.
- Pinpoint testing obstacles and synthesize strategies to overcome them.
- Appreciate that testing underpins modern software quality assurance by discussing the advantages of proactive and post-deployment software testing efforts.



📌 **Directions:** select **one** paper to present (that isn't already taken), and enter your name in the corresponding "Presenter" box for that day. After you present, upload your slides to Canvas.

	A	B	C	D	E
1	📌 **Directions:** select **one** paper to present (that isn't already taken), and enter your name in the corresponding "Presenter" box for that day. After you present, upload your slides to Canvas.				
2	Date	Jan. 08		Jan. 10	
3	Topic	**Course Introduction**		**Research 101**	
4	Paper 1	*No Readings*		*No Readings*	
5	Paper 2				
6	Date	Jan. 15		Jan. 17	
7	Topic	*No Class (Martin Luther King Jr. Day)*		**Research 101**	
8	Paper 1			*No Readings*	
9	Paper 2				
10	Date	Jan. 22		Jan. 24	
11	Topic	**Research 101**		**Introduction to Fuzzing**	Presenters
12	Paper 1	*No Readings*		Dissecting American Fuzzy Lop: A FuzzBench Evaluation (FUZZING'22)	
13	Paper 2			AFL++: Combining Incremental Steps of Fuzzing Research (WOOT'20)	
14	Date	Jan. 29		Jan. 31	
15	Topic	**Input Generation**	Presenters	**Runtime Feedback**	Presenters
16	Paper 1	DARWIN: Survival of the Fittest Fuzzing Mutators (NDSS'23)		The Use of Likely Invariants as Feedback for Fuzzers (USENIX'21)	
17	Paper 2	CarpetFuzz: Automatic Program Option Constraint Extraction from Documentation for Fuzzing (USENIX'23)		GLeeFuzz: Fuzzing WebGL Through Error Message Guided Mutation (USENIX'23)	

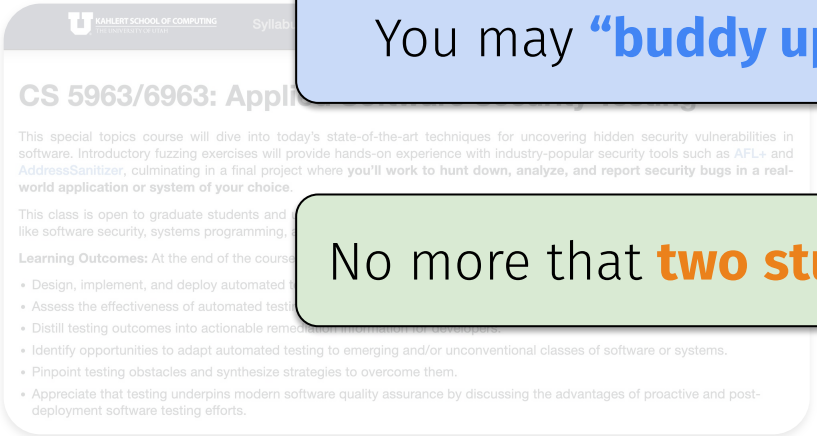
Recap: Paper Presentations

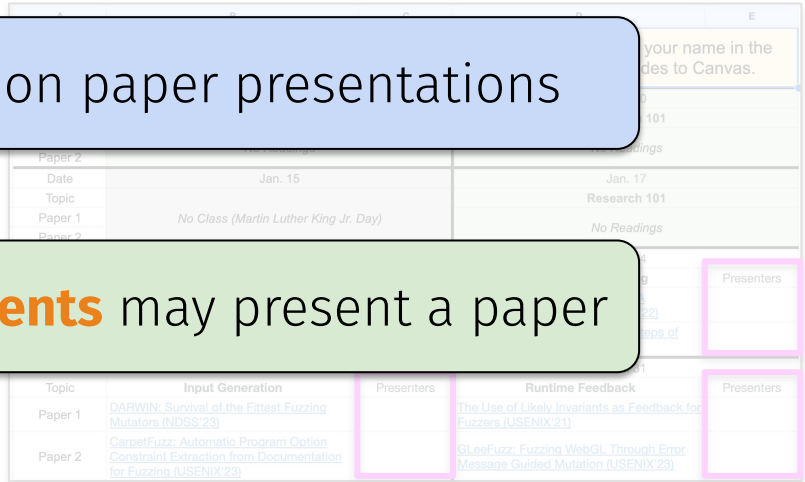
- Signup she… …ccount)
  - 38 fuzzi… …orkshops
  - Choose one paper by Monday, January 22

Enrollment has **exceeded** the number of papers

You may **“buddy up”** on paper presentations

No more that **two students** may present a paper

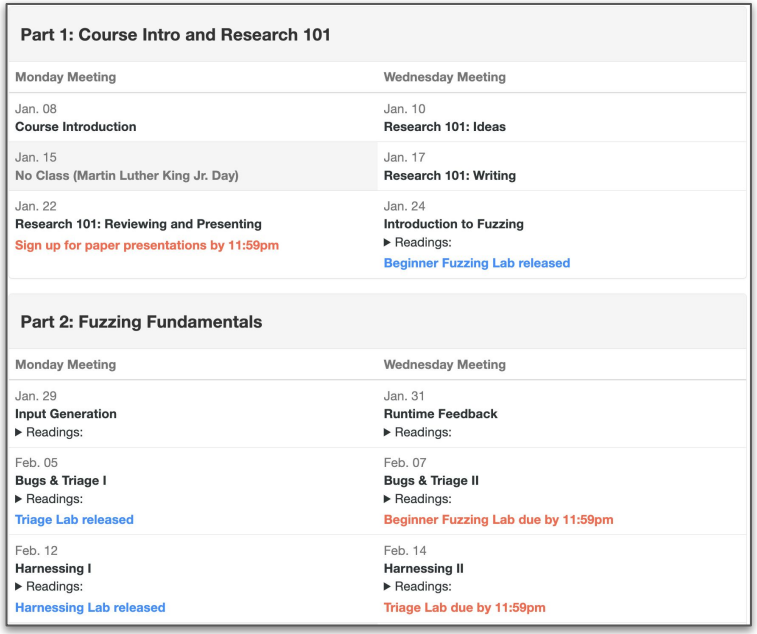




Recap: Key Dates

- **Jan. 22** **Select one paper to present**
- **Feb. 07** Lab 1 due
- **Feb. 14** Lab 2 due
- **Feb. 19** No class (President's Day)
- **Feb. 28** Lab 3 due
- **Feb. 28** **5-minute project proposals**
- **Mar. 04 & 06** No class (Spring Break)
- **Apr. 17 & 22** **Final project presentations**

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



Part 1: Course Intro and Research 101

Monday Meeting	Wednesday Meeting
Jan. 08 **Course Introduction**	Jan. 10 **Research 101: Ideas**
Jan. 15 **No Class (Martin Luther King Jr. Day)**	Jan. 17 **Research 101: Writing**
Jan. 22 **Research 101: Reviewing and Presenting** Sign up for paper presentations by 11:59pm	Jan. 24 **Introduction to Fuzzing** ▸ Readings: Beginner Fuzzing Lab released

Part 2: Fuzzing Fundamentals

Monday Meeting	Wednesday Meeting
Jan. 29 **Input Generation** ▸ Readings:	Jan. 31 **Runtime Feedback** ▸ Readings:
Feb. 05 **Bugs & Triage I** ▸ Readings: Triage Lab released	Feb. 07 **Bugs & Triage II** ▸ Readings: Beginner Fuzzing Lab due by 11:59pm
Feb. 12 **Harnessing I** ▸ Readings: Harnessing Lab released	Feb. 14 **Harnessing II** ▸ Readings: Triage Lab due by 11:59pm

Questions?

Writing:
The Communication of Research

Why write papers?

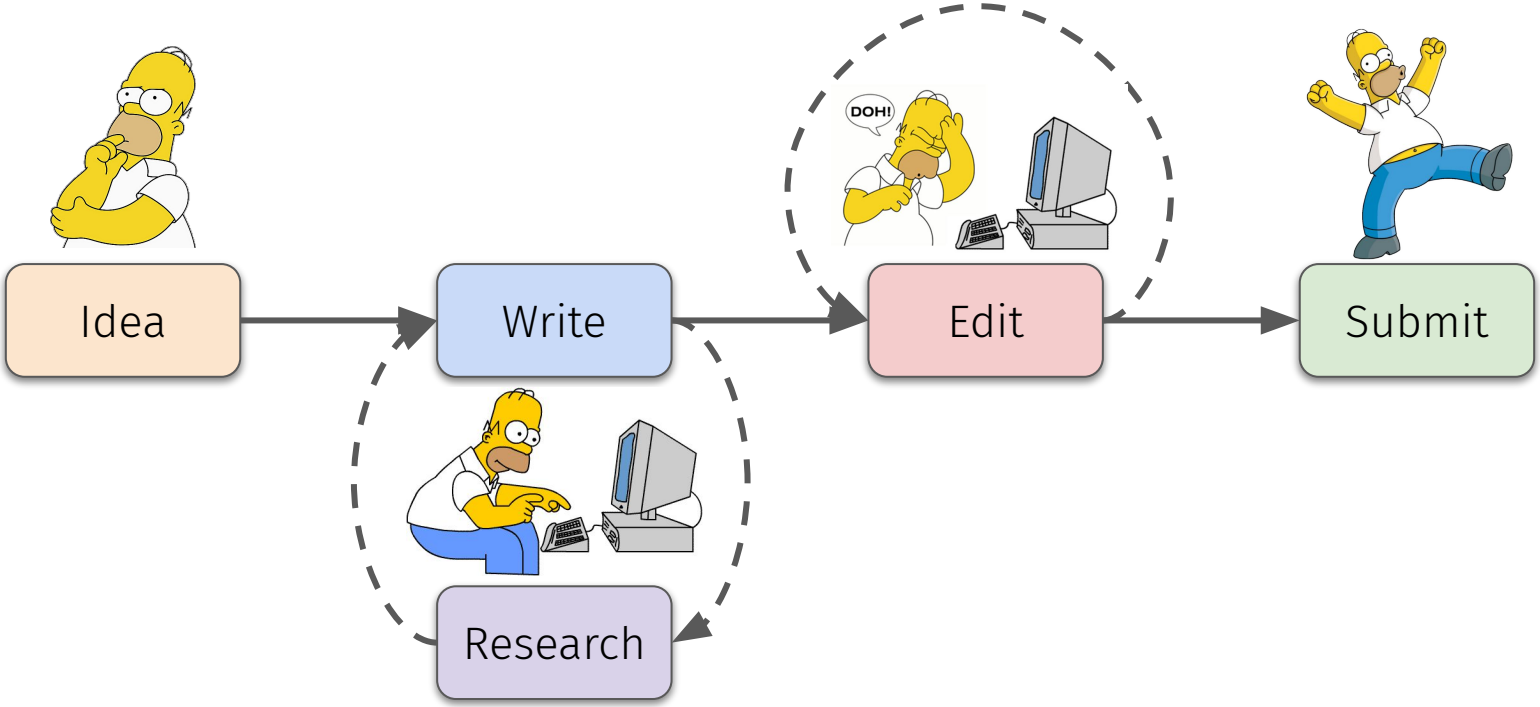
If you don’t publically document your work, then it does not exist (beyond you)

Why write papers?

- Document and communicate **what you did**
- Convince others that **you actually did it**
- Convince others that what you did **actually matters**
- **Because you won't get a Ph.D. without it**

Writing Papers

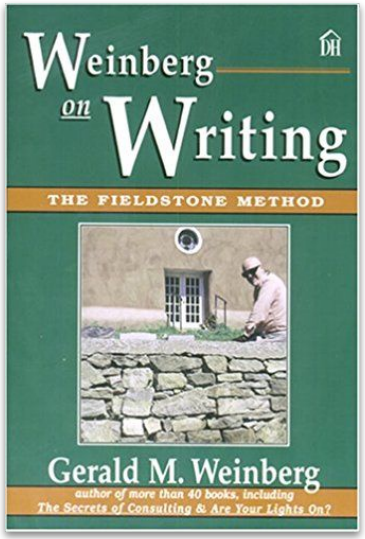
Writing papers is a process...



The Fieldstone Writing Method

- Start putting words on paper **as early as possible**
  - Writing near a deadline is really hard
  - Finalizing **“the pitch” is an iterative process**

- Write as you go along
  - It is **easier** to talk about a problem you are currently solving
  - It is **harder** to remember all problems you solved on short notice
  - It is **easier** to revise and remove than to create from scratch



Before you start: The Tagline

- **What is your paper's tagline?**
- At most **two sentences** (15 seconds in an elevator)
- Rest of paper must gracefully support the tagline

"There can only be ONE (paper tagline)"

Titles

- Highlight **what you do** and **distinguishing properties**
  - Objective is not to make you look smart (e.g., big words)
- Common distinguishing adjectives:
  - Automatic
  - Low-overhead
  - Dynamic
  - Reconfigurable
  - **Find a favorite thesaurus**
- Disambiguate the core message

The Title Rule

Paper titles should be fun or catchy, and ultimately memorable.

"You Autocomplete Me: ..."

"Fuzzing Hardware like Software"

Who's Calling? Characterizing Robocalls... "

"Who Left Open the Cookie Jar? A Evaluation of Third-Party Cookies"

"Users Really Do Plug in USB Drives They Find"

The (other) Title Rule

Your paper title and system name should be Google-able.

"ParmeSAN : Sanitizer-guided Fuzzing..."

"Fuzzing@Home : Distributed Fuzzing..."

"ExcelLint : Finding Spreadsheet Errors..."

"Favocado : Fuzzing Binding Code..."

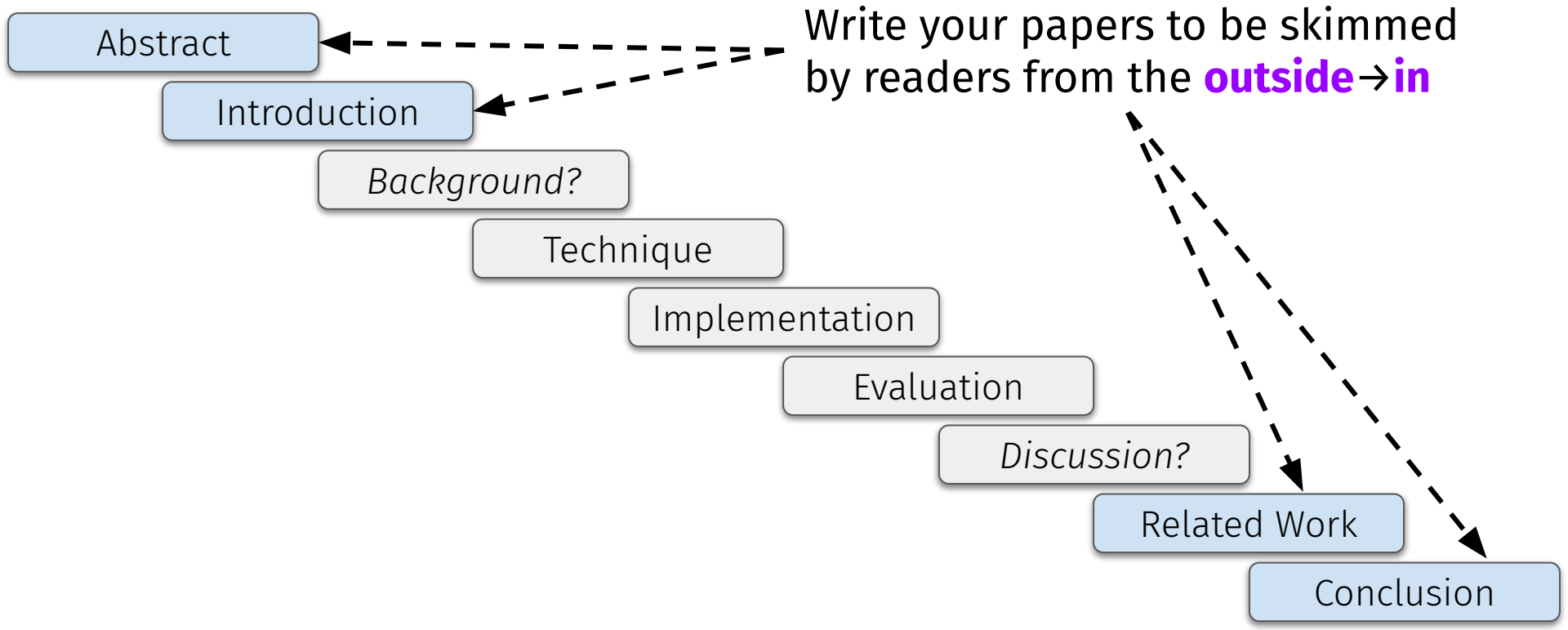
Paper Outline

- Abstract
- Introduction
- *Background?*
- Technique
- Implementation
- Evaluation
- *Discussion?*
- Related Work
- Conclusion

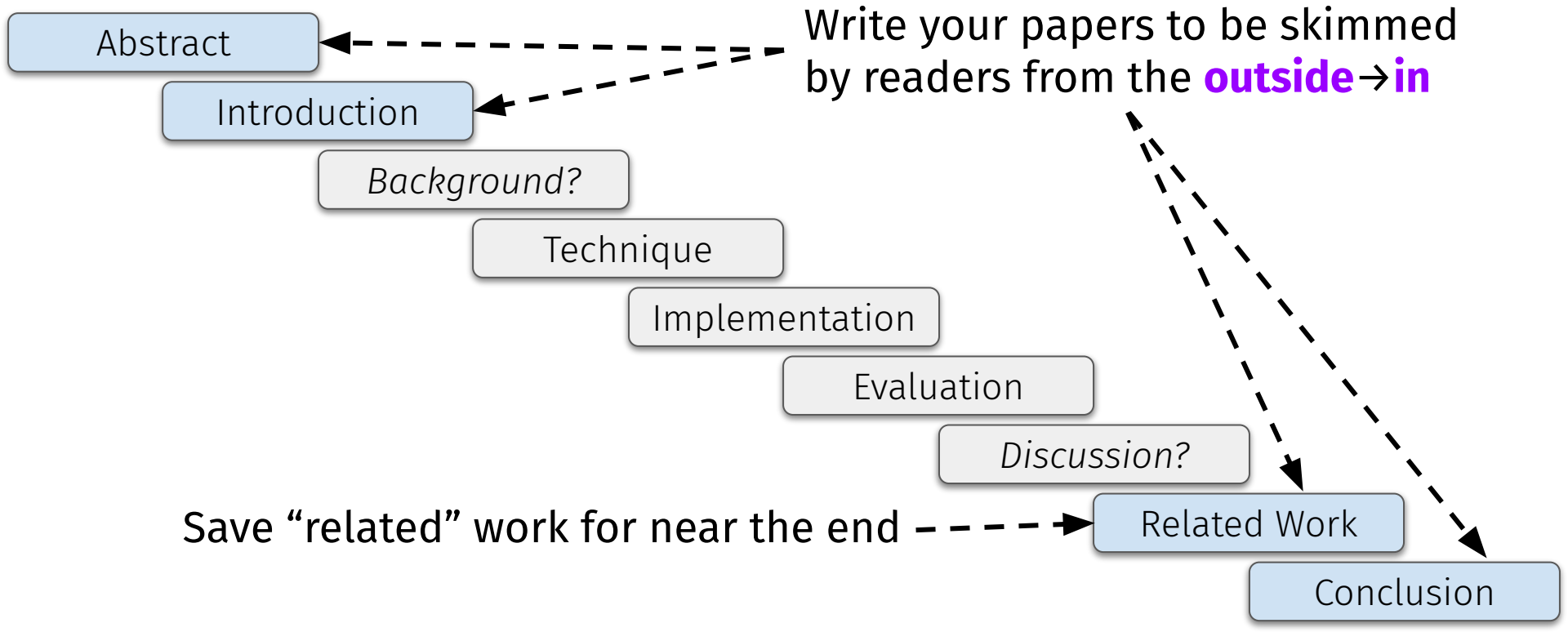
The “Makes Sense” Rule

A research paper should make sense just from reading its **introduction**, **conclusion**, and the **captions** to all figures and tables.

Paper Outline



Paper Outline



Structure of (a good) Paper

Abstract

- **Write as if it's a standalone document**
  - Very high level and concise description
- First paragraph
  - High-level problem and motivation
  - Bridge sentence: **what's the gap?**
- Second paragraph
  - Description of insights and approach
  - Sum up your experiment and results
  - Use **formatting** to your *advantage*

Abstract

- **Shark Tank:** study the “art” of the pitch

- Good pitches
  - Concise, correct, and high-level
  - Idea hasn’t been done before
  - Something that matters to society
  - Proof is in the pudding (i.e., results)

- Bad pitches
  - Not concise, incorrect, or too technical
  - Limited impact or already been done
  - Outcomes bad or not measurable

SHARK TANK

Choose Clarity over Complexity

Once, David Goodstein, a colleague of the Nobel-Prize-winning theoretical physicist Richard Feynman, said "Rich, explain to me, so that I can understand it, why spin one-half particles obey Fermi-Dirac statistics?"

Feynman looked at Goodstein and said, "I'll prepare a freshman lecture on it."

The physicist went away to compose his lesson, but a few days later came back to his colleague, "I couldn't do it," Feynman said, "I couldn't reduce it to the freshman level. That means we don't really understand it."

https://archinect.com/news/article/150171511/complexity-over-clarity-what-happens-when-we-try-to-sound-too-smart

Papers Should Tell a Story

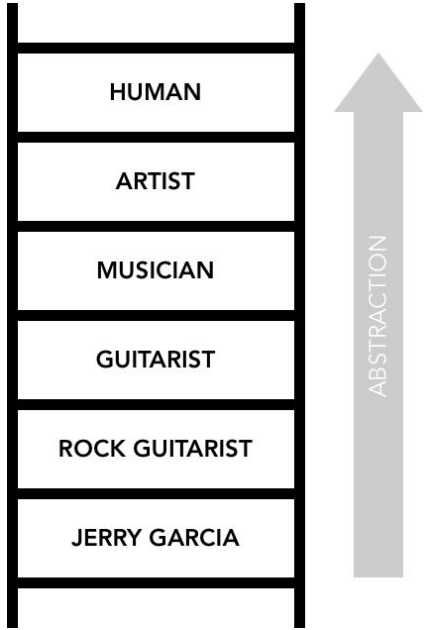
Telling a story in a technical paper is not like Shakespeare or writing Dante's Inferno, but like creating a character in a movie. The goal is logical connectedness. We have all been to movies where a character did something that didn't make sense or where we said, "I'd never do that." We have all experienced plot holes in movies. When writing a technical paper, our goals is to tell a story without plot holes. We NEVER want a reviewer to say, "Why did they do it that way? This doesn't make sense. This is unclear, what are the authors hiding?" This motto applies to writing the design section just as much as the evaluation section. It even applies to the intro where our goal is to convince reviewers that our problem is important, challenging, that our approach follows given previous work, and that our approach is effective.

The “Get to the Point” Rule

Don't write a mystery novel; give the reader the important information up front.

Walk the "Abstraction Ladder"

- **High level:** the abstract concepts, layman's terms
  - Don't include low-level details or terms here
  - Be succinct yet correct
  - Assume audience is **clueless**
- **Low level:** technical details, specialized terms
  - Assume audience is **knowledgeable**
- Work your description from the high-level to the low-level, then back up



Introduction

- Accept/reject decisions often made here
- 4–6 paragraphs
  - Motivation
  - Problem
  - What others have done and the gap
  - What you do
  - How you implement and evaluate
  - Results
  - List of contributions
- Don’t waste space with a paper outline

Introduction

- Don't spend too much space addressing the work of others
  - It detracts from the presentation of **your work**
- Address works **reviewers** will most likely relate to your work
- A string of references signals to readers that your work is a small boat in a sea of precious work
- A good intro generally takes 1–1.5 pages
- Only append a citation once for each context you use it in

The Heilmeier Catechism

- **What are you trying to do?** Articulate objectives using absolutely no jargon.
- **How is it done today**, and **what are the limits** of current practice?
- **What is new in your approach** and why do you think it will be successful?
- **Who cares?** If you are successful, what difference will it make?
- What are the **risks**?
- How much will it **cost**?
- How **long** will it take?
- What are the mid-term and final **“exams”** to check for success?

DARPA

The Novelty Rule

Don't claim to be first—even if you are—because that pisses reviewers off and they can always find a paper that—from 10,000 feet away—looks similar to yours.

Let others make those claims.

Corollary to The Novelty Rule

Don’t tell readers what to think
(e.g., our approach is **simple**, **clever**, **novel**, **awesome**, the **best ever**).

Let others make those claims.

Be Concrete and Explicit

NO!	YES!
We describe the WizWoz system. It is really cool.	We give the syntax and semantics of a language that supports concurrent processes (Section 3). Its innovative features are...
We study its properties	We prove that the type system is sound, and that type checking is decidable (Section 4)
We have used WizWoz in practice	We have built a GUI toolkit in WizWoz, and used it to implement a text editor (Section 5). The result is half the length of the Java version.

The Algorithm Rule

Your technique is better explained **in words**.

Algorithm 1: The UnTracer algorithm integrated in AFL.

Input: P: the target program
Data: b: a basic block
B: a set of basic blocks
i: an AFL-generated test case
Φ: the set of all coverage-increasing test cases

```
1 AFL_SETUP()
  // Instrument oracle and tracer binaries
2 P_O ← INSTORACLE(P)
3 P_T ← INSTTRACER(P)
  // Find and modify all of oracle's blocks
4 B = ∅
5 B ← GETBASICBLOCKS(P)
```

Related Work

- Not a place to disparage previous work
- Not a place to show your breadth of knowledge
- **Tell a story**
  - How the problem has progressed throughout history
  - How ideas relate to and build off each other
- Keep it to a few lines per paper
- End sections with **how your work fits in**

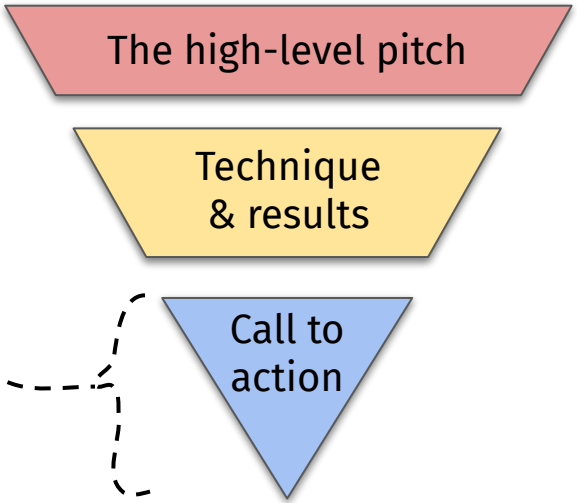
Related Work

- Delay an in-depth literature review
  - **Don't try to learn everything at once**
  - Read one paper per week during system building
  - Curate, organize, and annotate a bibliography
- What papers are reviewers likely to think of when they read yours?
- Refer to papers by how they are best known (not always by author)
  - **Example:** "SystemName shows…" **instead of** "Simpson et al. shows…"
- Sentences should be complete if you were to remove citations
  - **Example:** "SystemName [1] shows…" **instead of** "[1] shows…"

Conclusion

- Reverse pyramid: **tell them what you did**
  - **Start:** specific evaluation results
  - **End:** area and societal meaning

- High-level implications of your results
  - Recommendations
  - New opportunities
  - Future directions

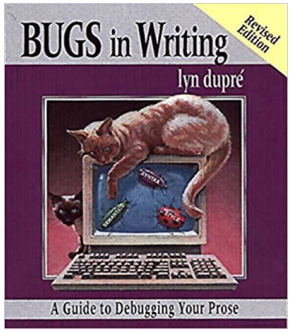


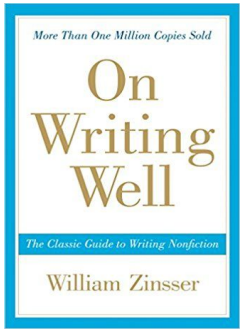
Refining your Writing

Grammar

- Avoid past tense; use **present tense**
  - “We implement” instead of “We implemented”
- Avoid passive voice; **use active voice**
  - “We analyze functions…” instead of “Functions are analyzed…”
- Avoid **contractions**
  - “do not” instead of “don’t”
- Avoid **wiggle words**
  - Would, could, should, maybe, possibly, can
- Avoid abstract; **be concrete**
  - “A 300 pound elephant” instead of “A large elephant”
- Do not tell the reader what to do; **tell what you did**
  - “First, you need to find the six least-connected components… ”

Grammar Resources





grammarly

UNIVERSITY WRITING CENTER
J.WILLARD MARRIOTT LIBRARY
SECOND FLOOR, RM 2701
295 S 1500 E
SALT LAKE CITY, UT 84112
(801) 587-9122

Graphics

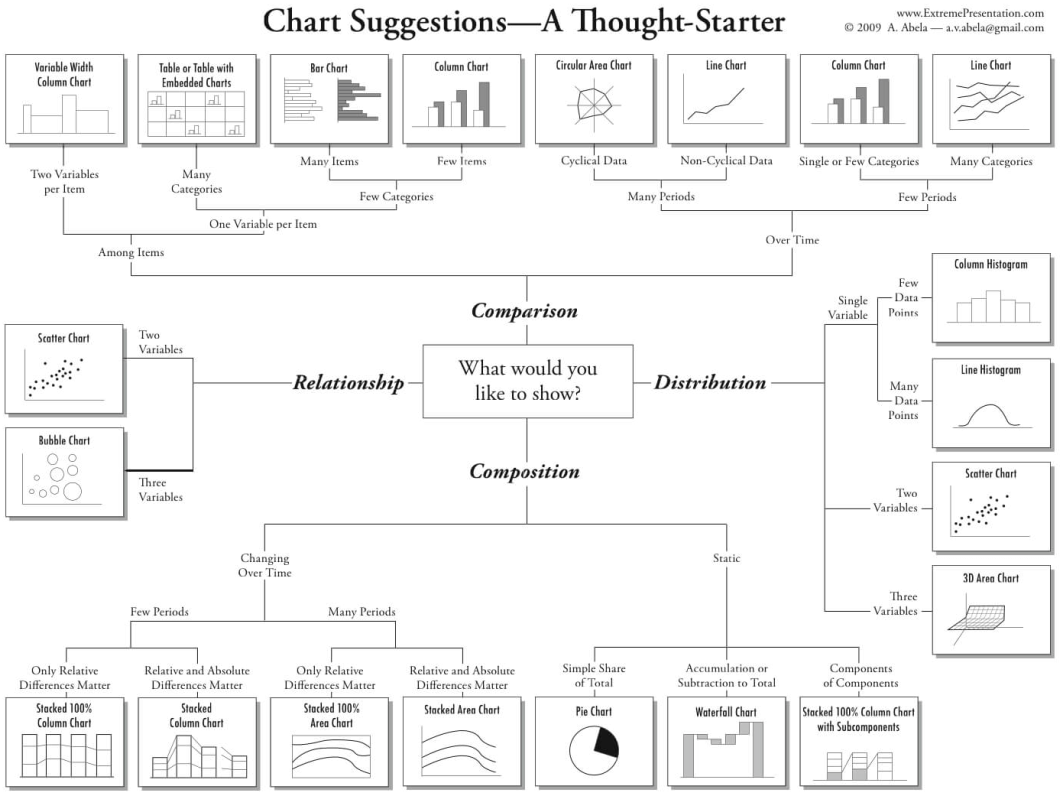
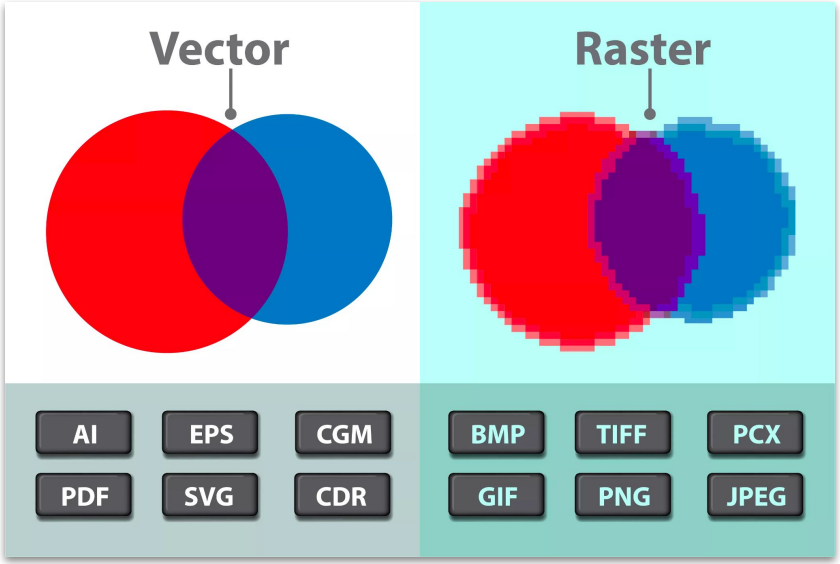


Chart Suggestions—A Thought-Starter
www.ExtremePresentation.com
© 2009 A. Abela — a.v.abela@gmail.com
Variable Width Column Chart
Table or Table with Embedded Charts
Bar Chart
Column Chart
Circular Area Chart
Line Chart
Column Chart
Line Chart
Two Variables per Item
Many Categories
Many Items
Few Items
Few Categories
Cyclical Data
Non-Cyclical Data
Single or Few Categories
Many Categories
One Variable per Item
Many Periods
Few Periods
Among Items
Over Time
Comparison
Scatter Chart
Two Variables
Relationship
What would you like to show?
Distribution
Single Variable
Few Data Points
Column Histogram
Many Data Points
Line Histogram
Bubble Chart
Three Variables
Composition
Two Variables
Scatter Chart
Changing Over Time
Static
Three Variables
3D Area Chart
Few Periods
Many Periods
Only Relative Differences Matter
Relative and Absolute Differences Matter
Only Relative Differences Matter
Relative and Absolute Differences Matter
Simple Share of Total
Accumulation or Subtraction to Total
Components of Components
Stacked 100% Column Chart
Stacked Column Chart
Stacked 100% Area Chart
Stacked Area Chart
Pie Chart
Waterfall Chart
Stacked 100% Column Chart with Subcomponents

The Graphics Rule

Make graphics in vector formats, not rasters.



Graphics Resources

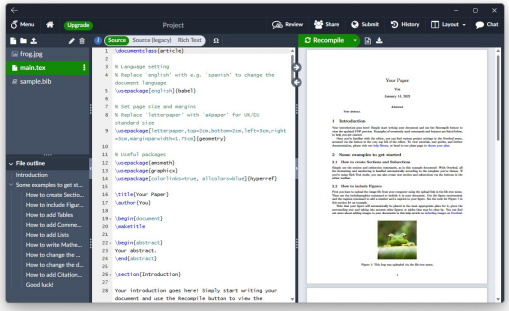
- **Charts and graphs**
  - Python matplotlib
  - Seaborn for enhanced charts

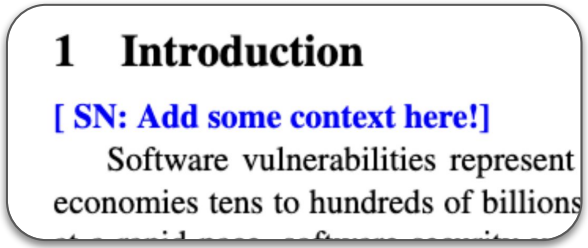
- **System diagrams**
  - Draw.io
  - Powerpoint
  - Inkscape (advanced)

- **Image search engines**
  - Vecteezy
  - Google image search
  - Various clipart websites

Paper Editing

- Use **version tracking** (e.g., GitHub, Overleaf)
- Create macros to leave **in-lined suggestions**
- PDF **diffing utilities** (essential for revisions)
  - `latexpand`: combine many `.tex` files into a single file
  - `latexdiff`: markup changes between two `.tex` files
  - Build a PDF diff from the resulting `.tex` file





Perform an Adversarial Review

- Identify the hurdles to **believing** your paper
  - Remove hurdles through proof or citation
  - Sometimes a citation is stronger than a self-contained proof
  - **Goal: minimize hurdles**

Perform an Adversarial Review

- Identify the hurdles to **believing** your paper
  - Remove hurdles through proof or citation
  - Sometimes a citation is stronger than a self-contained proof
  - **Goal: minimize hurdles**
- Where and how will a reader **get confused**?
- What will a competitor **disagree with**?
- Are all my claims supported by **reference** or through **experimentation**?

Maintain a Paper Template

- Essential sections
  - With notes on how to write those sections
- Default set of packages
- Useful commands
  - Macros (e.g., `\NameOfOurCoolSystem`)
  - Inlined comment macros (great for collaboration)
- Examples of common insets
  - Figures
  - Tables
  - Code snippets

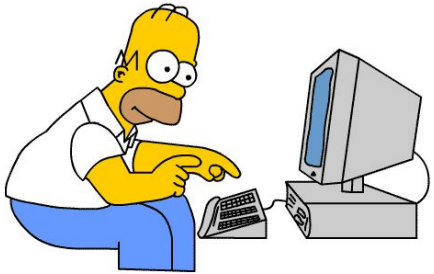
Collaborative Research: SaTC: CORE: Medium: ...

Overview:
In this project, we propose...
1. *In what ways...?*
2. *What challenges...?*
3. *Can we...?*

Intellectual Merit:
Offering to extend principled scientific techniques...
1. **Thing 1:**
2. **Thing 2:**
3. **Thing 3:**

Broader Impacts:
The research outcomes of this project will be...

Keywords: *Software*



Miscellaneous Tips

- **Introduction:** goal, intuition, reasoning, and takeaways are critical to a story
- For each paragraph:
  - At this point in the paper, **what does the reader know**?
  - What **one point** does this paragraph need to make?
- **Evaluations:** include analysis with description
  - Do not reiterate what readers can see for themselves (e.g., 50% overhead)
- **Implementations are not ideas**
  - Ideas should be **general** (e.g., implementable on other systems)
  - Implementations are **narrow embodiments** of the general idea

Maintain a Lexicon of Cool Words

- Decompose
- Expedition, Frontier
- Side-channel, Out-of-band
- Offline, Online
- Dynamic, Static
- Continuum
- Artifact
- Transient
- Intermittent
- Taxonomy, Orthogonal, Tradeoff space
- Forward error correction, Backward error correction
- Towards (in a title especially)
- Overcoming, Suggests, Asymmetry

The Final Rule

One must always leave something for reviewers to say.

Questions?

Next time on CS 5963/6963...

Research 101: Presenting and Reviewing