Washington University in St. Louis | Engineering

How to Write a Great Research Paper

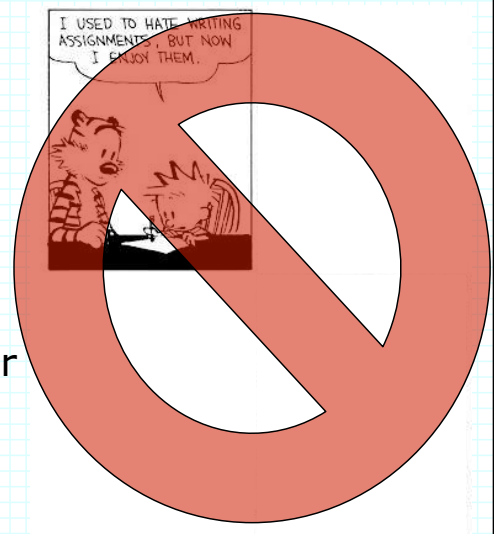
Jon Turner
Computer Science & Engineering
Washington University

www.arl.wustl.edu/~jst

Washington University in St. Louis | Engineering

Why Do We Write Research Papers?

- Why do research?
  - » to contribute to human knowledge and society
- We write to teach others what we have learned
  - » so they can use and build on our work
- And to help us organize our thoughts and structure the research
- It's NOT to impress others with how smart we are



Homicidal Psycho Jungle Cat, by Bill Waterson p. 62

Start with Good Writing

- Poorly written papers don't get read
  - » and are less likely to be published in first place
  - » papers that don't get read cannot have an impact
- Write well & others will make a point to read your work
  - » even if they are not working in your area
- Top lessons from Strunk and White
  - » choose a suitable design and hold to it
  - » omit needless words
  - » use definite, specific, concrete language
  - » revise and rewrite
- Teaching what you have learned
  - » know your readers and keep them in mind
  - » structure your writing to help them learn

A Suitable Design

- Abstract
  - What's the problem?
  - Why should the reader care?
  - What have you accomplished?
  - What are the implications?
- Introduction
  - Longer treatment of points in abstract
  - What previous work does paper build on?
    emphasize the positive, more than shortcomings
  - Pointers to rest of paper
- Body
  - » the problem
  - » your idea/approach
  - » solution/results
  - Highly variable, choose well
  - Logical progression with later sections building on what has come before
  - Anticipate readers' questions and give them answers (or tell them when you have none)
- Related Work
  - Be generous in citing others
    do not limit to work that directly influenced paper
  - Distinguish your work without bad-mouthing the competition
- Summary
  - Brief recap
  - What have you not done?
  - How might others take it further?

Omit Needless Words

- "Vigorous writing is concise. A sentence should contain no unnecessary words, a paragraph no unnecessary sentences, for the same reason that a drawing should have no unnecessary lines and a machine no unnecessary parts. This requires not that the writer make all sentences short, or avoid all detail and treat subjects only in outline, but that every word tell." – S&W p. 23
- Excess words interfere with learning process. Don't waste the reader's time
- Page limits demand that you make best use of available space

Examples (from Wikipedia)

original

- One common trend in the development of programming languages has been to add more ability to solve problems using a higher level of abstraction.
- In the case of lack of backward compatibility this can occur because the programmers have only considered coding the programs for, or testing the software, on the latest operating system they have access to or else, in isolation (no other conflicting applications running at the same time) or under 'ideal' conditions ('unlimited' memory; 'superfast' processor; latest operating system incorporating all updates, etc).

revised

- Programming languages are often designed to help programmers express computations more abstractly.
- Why is it that programs often fail to remain compatible with earlier versions? Sometimes programmers neglect to evaluate new versions in all possible operating environments. Sometimes, they fail to consider how their application interacts with others. Or, they may fail to consider the effects of limited computational resources.

Use definite, specific, concrete language

- "...the surest way to arouse and hold the reader's attention is to be specific, definite and concrete." – S&W p. 21
- Concrete language raises specific questions in readers' minds
  - » they learn as they seek answers
- Vague or overly general language is like fog
  - » obscures essentials, makes it harder for reader to engage
- Don't worry that readers will see only specific case
  - » easy for readers to appreciate the broader implications once they understand the core ideas

Examples (also, from Wikipedia)

original

- In the design of many types of programs, the choice of data structures is a primary design consideration, as experience in building large systems has shown that the difficulty of implementation and the quality and performance of the final result depends heavily on choosing the best data structure.
- In mathematics, computing, linguistics and related subjects, an algorithm is a sequence of finite instructions, often used for calculation and data processing. It is formally a type of effective method in which a list of well-defined instructions for completing a task will, when given an initial state, proceed through a well-defined series of successive states, eventually terminating in an end-state.

revised

- The right data structure can make or break a program. A table of customer data sorted by name works well for retrieving one customer's address, but is poorly suited to planning daily deliveries.
- One can sort a list of names by repeatedly exchanging adjacent out-of-order pairs. This is an example of an algorithm. More generally, an algorithm is an explicit and finite procedure for computing a desired result from given inputs.

Revise and rewrite (and do it again)

- Don't expect to get it right the first time
- Writing is an iterative process
  - » the more you write, the better you understand
  - » as you understand things better, you'll be able to express them more clearly
  - » better expression leads you to next level of understanding
- Don't be afraid of doing major surgery
  - » if ideas do not progress in a logical order, rearrange them
  - » don't hold onto a flawed organization just because you've invested a lot of time in it
- Ask others for advice and listen to it
  - » ask specific questions, not just "what did you think"

Teaching What You have Learned

- Know your readers and keep them in mind
  - » if you write it for specialists, can't expect others to read it
    - even specialists need to be told what you have learned
  - » for wider audience,
    - limit use of specialized jargon and define what you do use
    - where necessary, point to needed background
- Structure your writing to help readers learn
  - » organize material into a logical progression
  - » motivate reader to keep reading
  - » illustrate key definitions and concepts with examples
  - » proceed from specific instances to more general cases

When Should You Start Writing?

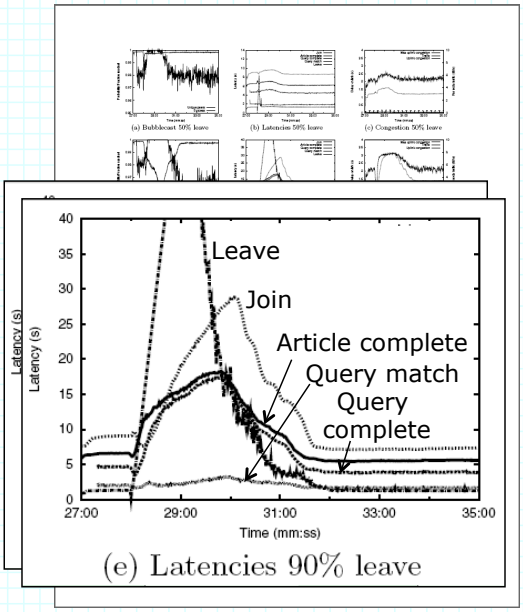
- Not two days before the conference deadline
- Write as you do the research
  - the best way to learn is to teach, and writing is teaching
  - you will discover things as you write that will change what you do in the research
  - writing forces you to choose terminology, create examples and illustrations, explain things in detail
  - the process leads to questions, which demand answers
- Write up the related work before you do research
  - you need to know and understand what's been done
  - writing about it forces you to read closely and distill essential contributions into concise form

Don't Bad-Mouth the Competition

- Neither necessary, nor smart
  - you don't have to make others look bad in order to distinguish your work from theirs
  - the competition may review your paper
  - being overly critical makes you look defensive & insecure
  - much better to emphasize how you have built upon the great work of those who came before
- When making comparisons, strive to be objective
  - don't overstate or exaggerate what you have done
  - don't select only the cases that make you look good
  - give competition the benefit of the doubt
  - acknowledge weaknesses in your approach

Presenting Performance Results

- Use charts to make points, not for decoration
- If you have nothing to say about chart, then drop it
- Chart text should be no smaller than body text
- Too many curves on a chart obscures the message
- Use labels, not legends
  - » line styles often hard to tell apart – can't count on color
  - » legends force reader to go back and forth, wasting time
- Read Tufte



For "Theory" Papers

- The traditional "theorem-proof" style
  - » eminently respectable
  - » but, can rob paper of all motivation
    - sure-fire way to lose readers
- Help the reader stay engaged
  - » take things in small steps – start with specific cases
  - » use examples and illustrations to make points
  - » provide "roadmap" to help reader perceive structure
- Develop arguments in conversational style
  - » make statement of theorem the logical conclusion
  - » moving proofs to appendix can work in systems papers, but is poor solution if analysis is central contribution

Writing for Specific Conferences

- Research communities have certain expectations
  - » understand the expectations and try to meet them
    - e.g. systems conferences have strong preference for measurements of working systems over simulation
  - » read, study, analyze papers from prior years
  - » when you must depart from community standards, make sure you have a good reason and explain it
- Dealing with rejection
  - » it happens to all of us, get used to it
    - in best conferences, 80-90% of submissions are rejected and reviewers have little time for those that clearly fall short
  - » learn from what your reviewers say
    - but also keep in mind, that reviewers can be wrong

Washington University in St. Louis
Engineering

Learning from Others' Writings

- Read papers critically
  - » not just for the research content, but for presentation
  - » learn to recognize authors who write and teach well
  - » ask what it is that makes their papers better than others
  - » pattern your writing after the best examples
- Develop an ear for English
  - » more to effective writing than following the rules
  - » read the work of good writers
    - not just technical papers
    - novels, biographies, newspapers, magazines,...

Other sources

- *How to write a great research paper,* by Simon Peyton Jones of Microsoft Research
  - http://research.microsoft.com/~simonpj/papers/giving-a-talk/writing-a-paper-slides.pdf
- *Writing Technical Articles,* by Henning Schulzrinne
  - http://www.cs.columbia.edu/~hgs/etc/writing-style.html
- *Advice on Research and Writing,* by Mark Leone
  - http://www.cs.cmu.edu/~mleone/how-to.html
- *Writing Good Software Engineering Research Papers,* by Mary Shaw
  - http://www.cs.cmu.edu/~Compose/shaw-icse03.pdf
- *The Visual Display of Quantitative Information* by Edward Tufte