Acknowledgment: The slides were prepared via valuable discussion, feedback from various colleagues and students. Our research that these talk slides were made based on has been supported in part by NSF and ARO.

How to Write Research Papers

Tao Xie

Department of Computer Science
University of Illinois at Urbana-Champaign

http://www.cs.illinois.edu/homes/taoxie/

taoxie@illinois.edu

Feb, 2006 (first version)

July, 2013 (last update)

http://people.engr.ncsu.edu/txie/publications/writepapers.pdf
https://sites.google.com/site/asergrp/advice

Caveats

- The key research contributions are the deciding factor for your paper's acceptance
  - Don't think that you should pay less attention to the "meat" in your paper
- There is no single standard way of writing research papers
  - Don't think that the writing of your paper should follow every suggestion in these subsequent slides
  - But these suggestions have strong (hopefully good) rationales; you need to understand these rationales before you (blindly) adopt any of these suggestions
  - Discuss with me (taoxie@illinois.edu) if you don't understand or disagree some points in these slides
- Quality/impact over quantity of papers

(Broader) Impact

- There are different types of impacts: research, industrial, societal/social, …
- Research impact, e.g., impact on research colleagues in various forms -- citations, inspiration, opening a new field/direction, ...
- General, fundamental, conceptual ideas (beyond a tool, implementation, infrastructure, study..) recent examples on QA
  - Godefroid/Sen et al. DART/CUTE/Concolic testing, PLDI 05/FSE 05
  - Engler et al. Coverity/Bugs as deviants, SOSP 01
  - Ernst et al. Daikon/Dynamic invariant detection, ICSE 99
  - Zeller. Delta debugging, FSE 99
- Overreaching contributions conveyed as insights

http://www.sigsoft.org/awards/ImpactAward.htm
http://www.sigsoft.org/awards/mostInfPapAwd.htm
http://academic.research.microsoft.com/CSDirectory/Paper_category_4.htm

Brief Desirable Characteristics

- Two main elements
  - Interesting idea(s) accompanying interesting claim(s)
  - claim(s) well validated with evidence
- Then how to define “interesting”?
  - Really depend on the readers’ taste but there may be general taste for a community
    - Ex: being the first in X, being non-trivial, contradicting conventional wisdoms, …
  - Can be along problem or solution space; in SE, being the first to point out a refreshing and practical problem would be much valued
  - Uniqueness, elegance, significance?

David Notkin. Software, Software Engineering and Software Engineering Research: Some Unconventional Thoughts. J. Comput. Sci. Technol. 24(2): 189-197 (2009)
David Notkin. ICSM 2006 keynote. Unconventional Views on Conventional Wisdom about Software Engineering Research.

Detailed Desirable Characteristics I

Crosscutting characteristics

- **Interesting** research, e.g., intriguing, unpredictable, surprising/unexpected
  - Ask interesting questions
  - Have interesting ideas in solution
  - Have interesting findings in evaluation

- **Novel** research, e.g., being the first
  - New problem
  - New solution
  - New findings

Detailed Desirable Characteristics II

Inspiring research

- General ideas (produced w/ research generalization, see later slides)
  - Problem formulation: general/abstract problem definition that could describe other concrete problems
  - Solution formulation: a general idea that could be used elsewhere

Detailed Desirable Characteristics III

Impactful research

- Impactful problem: a real problem with
  - high severity level: impact an case seriously
  - large scope level: impact many cases

  Mainly conveyed in the introduction section
- Impactful solution: an effective/efficient solution to well address the problem
  - E.g., many/high percentage (serious) (previously-undetected) bugs your approach finds
  - E.g., N man-hours that your approach saves
  - Great if having evidence of adoption in practice

Detailed Desirable Characteristics IV

Rigorous/accurate description

- Clear problem definition (no matter formalized or not)
  - inputs/outputs of the approach
  - requirements on the output

- Clear solution description with both algorithms and examples (don’t use only examples!)
  - reach the level of reproducible (others could reimplement your approach with enough high-level design information)

Detailed Desirable Characteristics V

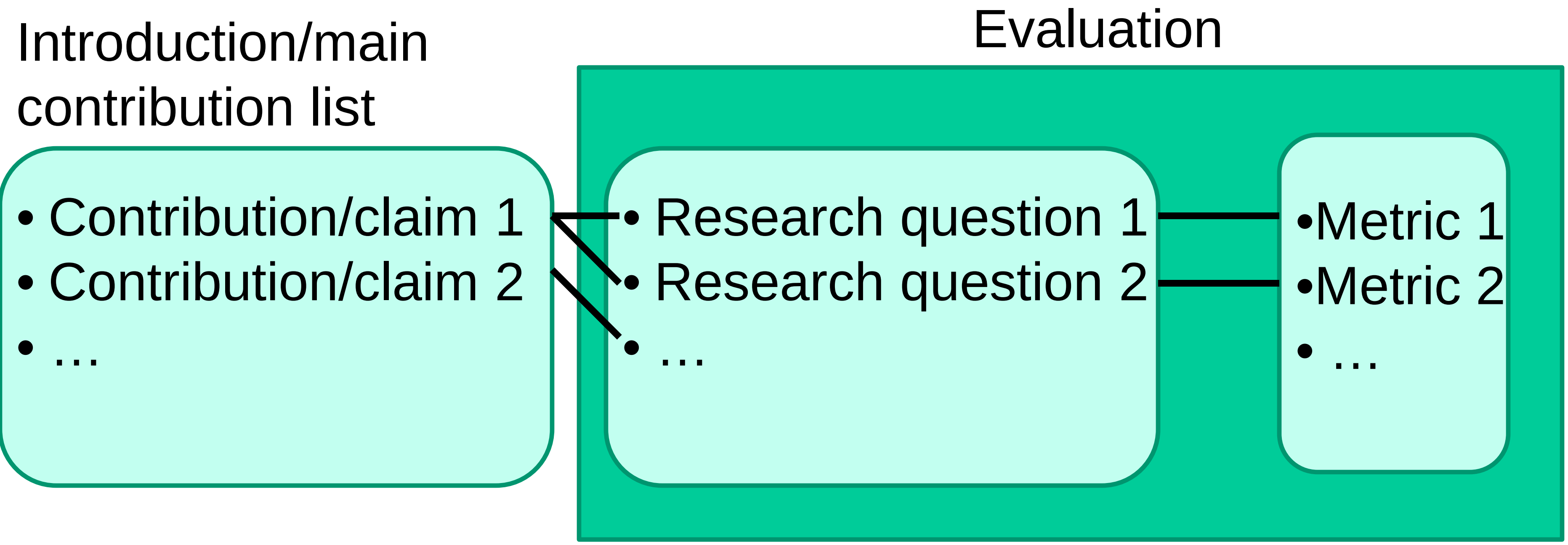
- **Significant** research (e.g., not easy problem to solve)
  - Technical challenges (see later slides)
    - problem level
    - solution level
  - Pose intellectual “stress” for whoever wants to address the problem

- **Validated** research
  - Clear and strong (empirical) evidence to validate/justify the claims

Key Questions to Double Check Your Paper

- Is the research problem significant/important?
  - NOT: a problem created/imagined by you and no one else cares about it
  - YES: a problem that people care (evidenced by concrete statistics or examples)
- Is your research solution significant or addressing technical challenges? (may be less critical for some type of work)
  - NOT: a solution that is incremental over previous work
  - NOT: a solution that is straightforward/trivial (e.g., simple adoption or slight adaption of an existing technique is not significant enough, even when you are the first one in doing so)
- Is your evaluation justifying the claimed contributions or benefits of your solution? (e.g., faster, detecting more faults, …than existing techniques if any)
  - Double check by making traceability from your claims listed in your contributions to your research questions to investigate in your evaluation

Traceability Links



- Make sure each contribution/claim is translated to (appropriate) research question(s) ➔ no unsubstantiated claims
- Make sure each question is answered with help of (appropriate) metric(s)

See GQM by Weiss/Basili
http://en.wikipedia.org/wiki/GQM

Know What Your Audience is

- Explicitly explain how your paper is relevant to the conference (or journal) you submit to (if not that obvious)
  - E.g., if ICSM, explain clearly in abstract and intro how your work is related to maintenance; if WWW, explain clearly in abstract and intro how your work is related to web; …

- Explicitly explain some basic assumptions/concepts underlying your work (even which may be obvious to your subfield but not to the conference reviewers/audience)
  - E.g., if your approach is about achieving high structural coverage of code, need to explain why achieving high structural coverage is important (e.g., related to fault detection) when you submit to WWW or even some sub-field conferences whose reviewers may not be testing experts

Justify Your Choices

- Pitfall: In intro sec, you describe that you propose a way of solutions (e.g., dynamic analysis) to address your stated problem, BUT you never discuss why alternative way of solutions (e.g., static analysis) would not be chosen
- Pitfall: In approach sec, you describe that you use a technique (e.g., hierarchical clustering) to address a sub-problem in your approach, BUT you never discuss why alterative way of techniques (e.g., partitional clustering) would not be chosen
- Pitfall: In your evaluation sec, you don’t compare the results of including or not including an important technique (e.g., filtering) claimed to be a major contribution
- Pitfall: in your evaluation sec, you don’t justify why you choose the experimental subjects or a subset of subjects used by previous work

Don’t Write Too Little or More Than Enough

- Pitfall: A student tends to write a lot of low-level implementation details, which they spent most time on; these details are of no or little interest to readers who don’t plan to reimplement the approach for the same language or using the same library/framework
  - More importantly, the space shall be devoted to high level ideas/contributions

- Pitfall: A student omits some important details of experimental setup causing readers not to be able to reproduce the experimental results

- Need balance on reproducibility and new idea/research contributions
  - E.g. solution: separation of approach and implementation sections

Formalize Just Enough

- Applicable on some type of work
- Formalization examples: formal definitions, algorithms, …
- Formalization helps
  - write clearly and force you to think and write rigorously
  - help grasp problem/solution essence, avoid shooting moving targets
- But don’t over-formalize to pose barrier for understanding – formalization is to better rather worsen understanding
- Learn how to write by reading and mimicking styles of papers (related to your work) written by PL/compiler/formal method researchers (e.g., from TACAS, POPL, PLDI, SPLASH/OOPSLA, ECOOP, CAV)
- Caveats
  - Some SE reviewers don’t like or get used to formalization
  - When you write clearly, you easily expose “holes” to reviewers ➔ formalization helps even it is not put in paper

Typical Paper Structure

- Title/Abstract
- Introduction
- Optional: Background
- Optional: Formal Problem Definition
- Related Work (alternatively put before conclusion)
- Example
- Approach/Framework
- Implementation
- Evaluation
  - Experiment/Case Studies/Experiences/Examples
- Discussion
- Conclusions (and Future work)

Title and Abstract

- Title writing pitfall:
  - Don’t put uncommon buzzwords there
    - Otherwise, bad for paper search engines or readers who would like to understand what the paper is about by reading the title
  - Be specific enough but not too specific (related to the previous bullet)
- Name your approach with a cute name (e.g., CUTE)
  - Easier for others to remember and cite
- Abstract structure: Short motivation (problem); Proposed solution; Evaluation; Evaluation results
- Abstract writing pitfall:
  - Don’t put unexplained or undefined terms whose meanings are not well known
  - Solutions: explain them; rephrase them using plain words; not get into too much detail (without mentioning them).

Introduction Structure

- Long motivation, problem to be solved, why existing solutions are not sufficient (sometimes examples help)
- Need show the problem is significant (desirable to use concrete statistics, concrete examples, or citations)
- Proposed solution (**inputs/outputs**) and key ideas (steps)
- Optional: brief mention of related work if it is very related and explain differences
- Evaluation and evaluation results
- Optional: “The paper makes the following main contributions: + bulleted items”
  - Easy for reviewers to spot out major contributions
  - Being of the “first” in something is desirable as a contribution
- Structure layout of the paper (you want to give readers high level ideas how different parts are related to each other)
  - Similar principle applied throughout the paper for subsections

Introduction –cont.

- Don’t overclaim (even throughout the paper)!
  - But it is good to put your work in a bigger picture and a larger background
  - But it is important for you emphasize the significance of the problem and your solution (esp in intro)
- Similarly don’t over-criticize other’s work (even throughout the paper)!
- If you want to claim some unjustified points, it is better to put them in conclusion or discussion section
- Even if so, be careful on wording
  - X “Our approach provides a foundation for this new field.”
  - “*We believe* our approach *can* provide a foundation…”
  - “*We believe* our approach *has a good potential* for providing a foundation …”

Introduction –cont.

- Another example: be careful on wording
  - X "Our/X's approach is the only/first one on …."
  - "*With the best of our knowledge*, our/X's approach is the only one/first on …"
  - "Our/X's approach is one of the/a few approaches …"
  - "Our/X's approach is a major/representative approach …"
- Some reviewers don't like you to claim your own approach to be "novel" (at least don't put "novel" in your paper title!) – they said novelty is to be judged by them not to be claimed by you
  - "TestEra: A Novel Framework for Automated Testing of Java Programs" → "TestEra: Specification-based Testing of Java Programs Using SAT"

Stirewalt's 5-paragraph rule on writing Introduction - 1

- Introductory paragraph: Very briefly: What is the problem and why is it **relevant** to the audience attending *THIS CONFERENCE*? Moreover, why is the problem **hard**, and what is your solution? You must be brief here. This forces you to boil down your contribution to its bare essence and communicate it directly.

http://www.cse.msu.edu/~chengb/Writing/intro-guidelines-stirewalt.txt

Stirewalt's 5-paragraph rule on writing Introduction – 2/3

- Background paragraph: Elaborate on why the problem is **hard**, critically examining prior work, trying to tease out one or two central **shortcomings** that your solution overcomes

- Transition paragraph: What keen **insight** did you apply to overcome the shortcomings of other approaches? Structure this paragraph like a syllogism: Whereas P and P => Q, infer Q.

http://www.cse.msu.edu/~chengb/Writing/intro-guidelines-stirewalt.txt

Stirewalt's 5-paragraph rule on writing Introduction – 4/5

- Details paragraph: What **technical challenges** did you have to overcome and what kinds of **validation** did you perform?

- Assessment paragraph: Assess your results and briefly state the broadly **interesting conclusions** that these results support. This may only take a couple of sentences. I usually then follow these sentences by an optional overview of the structure of the paper with interleaved section callouts.

http://www.cse.msu.edu/~chengb/Writing/intro-guidelines-stirewalt.txt

The Stanford InfoLab's patented five-point structure for Introductions

1. What is the problem?
2. Why is it **interesting** and **important**?
3. Why is it **hard**? (E.g., why do **naive approaches fail**?)
4. **Why hasn't** it been **solved** before? (Or, what's wrong with previous proposed solutions? How does mine **differ**?)
5. What are the key components of my approach and results? Also include any specific limitations.

http://infolab.stanford.edu/~widom/paper-writing.html

Problem Definition (optional)

- If your paper proposes a new problem or addresses a formalizable problem, it is good to have a section on problem definition
- Examples
  - Section 2 http://people.engr.ncsu.edu/txie/publications/issta09-ilp.pdf
  - Section 2 http://people.engr.ncsu.edu/txie/publications/icse09-carminer.p
- Such a section is useful to clearly describe the problem being addressed by the paper

Formal Problem Definition

- Define the problem that your approach intends to address
- Can be put in a section after intro/example section, serve the purpose of the example section as described later
  - When you formalize your problem, readers can have better grasp on what you are trying to address
- There you can also formally define some important concepts referred to in your approach (either in the problem space or solution space)
- Problem formalization can be a new contribution in the contribution list

Caveat: Formal Problem Definition

Example Review Comments on Not-Good-Enough Formalism

- "Section 3, the formal definition, is not very well organized. A formal definition can be useful and clarifying, but in that case ought to be crisp, clear, and elegant. To my taste your definitions are a bit messy”
- “Definition 1 is not really a definition”
- “It is also interesting to see that you don't use your formal definition in the rest of the paper.”
- “I am not sure what the formalization of XXX adds. It seems rather disconnected to the rest of the paper.”

Technical Challenges

- Why list challenges?
  - If your solution is so obvious and easy, you cannot impress readers/reviewers and justify significance
- Challenges from two levels (you can describe challenges at one or both levels)
- Problem-level challenges
  - Independently of any solution to the problem (e.g., static vs dynamic analysis), what are the challenges of addressing the problem?
- Solution-level challenges
  - For the style/direction that you will commit to (e.g., static in contrast to dynamic analysis; of cz, you need to justify why static not dynamic already here), what are the challenges of carrying out the solution to address the problem?

Simple vs. Sophisticated Solutions

- Don’t ignore simple (basic, straightforward) solutions while hunting for sophisticated solutions
  - At least try simple ones out, only when they don’t work, use the challenges/difficulties faced there to drive the hunting of more sophisticated solutions
  - Simple ones serve as baseline base in evaluation
- Often the time, students may be too proud of some clever “tricks” that they came up and had tendency of losing sight of easier, simpler solutions

“Make things as simple as possible, but not simpler.” - Einstein

Challenges ➔ Contribution Points

- Normal structure of main contribution list:
  - The overall approach
  - A list of specific techniques in the approach
  - Implementation and evaluation
  - Evaluation results
- For each specific technique in your contribution list, you shall have at least one corresponding clearly articulated technical challenge
  - If your solution/technique is so obvious and easy, you cannot impress readers/reviewers and justify significance
- Alternatively, you may articulate technical challenges just for the overall approach

Tell a Good Story in Intro

- Abstract and introduction section are very important
  - Normally a reviewer can quite accurately predict (or decide) the reject/accept decision of a paper after finishing reading the abstract and introduction section
- Need tell an interesting, intriguing, engaging story (positioned at right angle and right abstraction/scope)
  - So that readers cannot wait to see the rest of the paper
- Offer pleasant “surprise” (not boredom) to readers; exciting/interesting new things for readers to learn
  - After finishing reading the short description of the target problem, they couldn’t predict what challenges or significant issues real world setting could face [Wang et al. ICSE 09]
  - After finishing reading the description of your problem, they couldn’t predict what solutions you will provide (e.g., clever, neat ideas to address challenges)➔ attract them to read on

Suggestion Actions Against Intro/Abstract

- Iterate and improve the abstract and introduction in a small discussion group (e.g., read aloud)
- Pay attention to the logical transitions in sentences in abstract and paragraphs in introduction section (e.g., using Mind Map: http://freemind.sourceforge.net/)
- Double check that earlier stated characteristics are satisfied
    - Ex. The target problem is significant/important
    - Ex. Your solution is significant/addressing non-trivial technical challenges, and is well validated

Background and Related Work

- Differences between background and related work (c.f. my ASE journal 06 paper)

- You can organize related work with subsections or group them in several categories

- Background sometimes called Preliminaries
  - Includes notation, terminology, others’ or your previous techniques that are not part of the contributions of this paper

Related Work

- Don’t simply list related work without RELATING to your own work!
  - keywords to use: whereas, in contrast, but, however, …
  - “excuses” to use: “does not require specs”, “focus on different problems”, “complement with each other”, …
  - you can describe several similar related approaches together and compare them at once with yours
- Don’t just discuss the differences between your work with related work only in the solution space
  - Need to relate back to the effect/impact on the problem space
  - E.g., You may argue that your work uses dynamic analysis and related work uses static analysis --- but how would these two analysis types impact the problem you are addressing? Static analysis produces too many false warnings? … You need to compare them in terms of observable differences from the approaches’ user’s point of view in the problem space

Background and Related Work cont.

- Don’t make unjustified unobvious criticisms on related work if you don’t have experimental results to back you up.
  - But you can cite others’ experiments to back you up.
- Don’t overclaim your work without justification
- Don’t intentionally leave out your own very related previous papers (reviewers can find them out easily)
  - maybe even need to mention them in Introduction section and explain why the new work is different
  - reviewers often try to identify a marginal/incremental paper or a “least publishable unit (LPU)” (Google this term!)
- Put in PC members’ work if relevant

Related Work cont.

- Where to put the related work section
  - After the introduction/example section
  - Before the conclusion section
- After the introduction/example section
  - Pros: Immediately clear out reviewers’ wonder on how the work differs from previous work
  - Cons: hard to let readers to know what you are talking about before showing the approach details
    - But it may be ok to put it after the example section (see next slide)
- Before the conclusion section
  - Pros: Now reviewers’ know what your approach is about
  - Cons: reviewers keep wondering how the work differs from previous work till this point
    - But for very closely related work, you should have pointed out the differences in the introduction section

Example

- A simple example
  - Include: where it comes from; a figure listing source code; brief description
  - Throughout the paper, it is important to have illustrating examples for those places that contain “dry” descriptions of your approach
  - If you use several examples throughout the paper, you may not need a separate Example section.
- Optional/important part of the section: high level description of applying your approach on the example
  - describe **inputs/outputs** of your approach without getting into too much detail
  - very important if the later approach description involves heavy hard-to-understand formalisms
  - see my ASE 04 Rostra and TACAS 05 Symstra papers

Approach or Framework

- Generalize your work in an abstraction level, e.g., positioning it as a framework or algorithm rather than a tool
  - What you develop should be beyond your own implementation
  - Then you are in a better position when you discuss limitations of your work: Inherent limitation of the framework? Or limitation of your current particular implementation of the framework? [See my ASE journal 06 paper]
  - A workflow diagram is useful for explaining your framework
- Try to separate the ideas from (a particular) concrete implementation
  - But sometimes you have to mention it a bit and refer the readers to the implementation section.
- Explain some details with examples (even if you have illustrated your high level ideas in the example section)
  - Often still need to provide algorithm descriptions to precisely describe your approach instead of using ONLY examples to explain it
- Some reviewers don’t like it if you devote equal amount on each component of your approach/tool – read like a tool paper
  - May focus main text on the algorithms or key techniques

Implementation

- What libraries you used in your tool
  - e.g., BCEL, Daikon frontend, Soot
- Detailed implementations of each step in your framework
- List complications of implementing a certain idea and how you get around them
  - if some complications are important and general, you may move them to the framework section.
- Applicable to both approach/implementation
  - Don’t detail the entire story of how you arrived at your approach/implementation/results, unless they provide useful lessons learned to readers (even so, put them in discussion section)

Evaluation

- (Controlled) Experiment: good for tools that don't involve human interactions within the approach
  experiment writing structure:
  - Hypotheses/Questions to be answered
    - Double check your questions. Ex. "**Can** our approach perform better than a previous related approach?" ➔ "**How much better** can our approach perform than …"
  - Measures you use to answer these questions (higher better?)
  - Experiment setup: a good number of subjects, some scripts, some third-party tools or reimplemented tools for comparison
  - Independent variables+dependent variables -> metrics
  - Experimental results
    - Illustrate how to read your table/diagrams (columns, x/y axis, etc.)
    - Explain what does the curve or data mean, e.g., "We observed that …", "The experimental results show …"
    - Summarize your findings, remember to get back to answer the hypotheses and questions; it is ok to have an undecisive or negative answer based on the experimental results
    - Optional: discussion subsection; or you can put it as a separate section
  - Sometimes you may not include cost (time/memory) in your experimental results but you need to at least discuss the analysis cost
  - Threats to validity: internal, external, and construct (see my TSE 05 paper); sometimes may not need that fined-grained type classification

Evaluation cont.

- Case studies, experiences, and examples are often good for
  - approaches with human involvements [experiments can also involve humans though]
  - approaches whose results are hard to quantify with numbers (see my ICFEM 05 paper)
  - approaches you don't have a good enough number of subjects for controlled experiments
- Case studies
  - usually involve human subjects
  - often require careful preparation (tasks, questionnaires, interviews, etc.)
  - uncontrolled but just observe
  - lessons learned
- Feasibility studies: not directly assess or apply the approach on the real environment but give hints on feasibility
- Experiences/Examples
  - anecdotes; maybe just you are the one who are involved
  - You may use some wordings such as "Developers can click … to look for …"

Evaluation cont.

- Need explain evaluation results or describe your insights from the observed results rather than just describing the results
  - E.g., if some subjects' results are especially favorable or unfavorable, explain the reasons or even your hypothesis (wordings: "We suspect that …" "We hypothesize that …"). You may leave confirmation of these hypotheses to future work (e.g., on more experiments)
- Need describe "Experiment Designs"
  - E.g., factors (independent variables), treatments (one factor multiple treatments or one factor one treatment)
    C.f. "Experimental program analysis: A new program analysis paradigm." ISSTA 06
- Need hypothesis testing, t-testing especially if you want to say "A result is **significantly** better than B result"; statistically significant vs. practically significant
  - C.f. "Is mutation an appropriate tool for testing experiments?" ICSE 05

This slide made with contributions from S.C. Cheung at HKUST

Evaluation cont.

- What to be qualified as case studies? (more strict sense)
  - Must be conducted on real, uncontrolled industrial settings
  - If conducted at university settings, not qualified; then shall target at the experiment type (with a good number of samples); sometimes it may not be feasible to get a good number, can alleviate by writing if you can cite a pervious significant paper and state try your best to reach or go beyond their sample size; reviewers may be reasonable on it
- Case studies may also need hypothesis; in journals, even additionally need "rival hypothesis"
  - Different from "null hypothesis vs. alternative hypothesis" in experiments
    http://www.stats.gla.ac.uk/steps/glossary/hypothesis_testing.html
    C.f. "Statistical significance testing—a panacea for software technology experiments?" Miller JSS 04
  - E.g., Hypothesis: quality increases due to software inspection
    Rival Hypothesis: quality increases due to better working environments
  - C.f. Yin's book on Case Study Research

This slide made with contributions from S.C. Cheung at HKUST

Evaluation cont.

- In evaluation (experiments or case studies), we write

Research question (first)

Hypotheses (then) [Optional]

- Research questions
  - Abstract, general, high level
- Hypotheses
  - Concrete, specific, often answers to the research questions
- In the experimental results, need describe how the results relate back to which hypotheses and how hypotheses relate back to which research questions
- When using colored figures, make sure you describe both colors and gray-scale in text (since people may read papers in black-white copy)

This slide made based on discussion with S.C. Cheung

Evaluation cont.

- Construct a project web including the evaluation subjects, evaluation results …
  (e.g., http://research.csc.ncsu.edu/ase/projects/carminer/)
  - If tool is releasable, release your tool here (even binary form)
  - If a demo video is available, put it up here (e.g., http://osl.cs.uiuc.edu/~ksen/cute/demo.htm)
- Why? Building **trust** from reviewers in your work and your results
- When doing manual verification/inspection/confirmation of your evaluation results (e.g., confirming real defects), use >=2 persons to do so. When these persons don't have consistent decisions, they need to discuss to reach a consensus. Describe such process in the paper.
- Measure both mean and variance/deviation, not just mean

Evaluation cont.

- Evaluation on real industrial code bases is good; however, these code bases are not in the public domain and therefore, other researchers cannot reproduce the results or compare their own approaches with the approach in the paper
- Solution:
  - Include both evaluations on industrial code bases AND open source code bases
- E.g., using “benchmarks”
  - Some areas such as fault localization have “de facto” benchmarks: siemens programs, space, ….
  - Note that often using only small siemens programs is not enough (e.g., in fault localization) and need additional large benchmarks
  - Check UNL SIR: http://sir.unl.edu/portal/index.html

Evaluation cont.

- It may be risky to pick just a simple alternative solution as the baseline for comparison in your evaluation
  - Reviewers may not agree that is a fair baseline and consider it as too naive
  - Unless previous work published this baseline solution and it is believed to be the state-of-the-art one
- To alleviate the issue, pick a spectrum of the comparison bases, each of which incorporates one more technique of your proposed approach incrementally.
  - Still useful even when you already compare with a baseline solution
  - Help gain insights of the contributions of each technique in your proposed approach towards the overall effectiveness

- See Thummalapenta&Xie's ASE 2007 PARSEWeb paper, including both comparison with baseline and variant baselines

Evaluation cont.

- Some guidelines on doing/writing experiments
  - “Experimental program analysis: A new program analysis paradigm.” ISSTA 06 http://esquared.unl.edu/articles/downloadArticle.php?id=208 http://esquared.unl.edu/wikka.php?wakka=ExperimentalProgramAnalysis
  - http://www-users.cs.umn.edu/~heimdahl/ase08ds/AndrewsEvaluation.pdf
  - http://www.acm.org/crossroads/xrds7-4/empirical.html
  - http://www-static.cc.gatech.edu/~harrold/8803/Classnotes/
    - Notes of Weeks 18, 19, 20, and 21
- Some relevant papers/examples of doing/writing various types of evaluation
  - http://www.cs.washington.edu/education/courses/590n/04sp/
- Experiments vs. Case Studies
  - “Evaluating emerging software development technologies: lessons learned from assessing aspect-oriented programming” by Murphy et al. http://ieeexplore.ieee.org/search/wrapper.jsp?arnumber=799936
- A good book on case study research in general
  - “Case Study Research : Design and Methods” by Robert K. Yin
  - http://www.amazon.com/gp/product/0761925538/104-9365607-2004707?v=glance&n=283155

Evaluation cont.

- Better Empirical Science for Software Engineering, Basili and Elbaum, ICSE 06
  - http://csce.unl.edu/~elbaum/talks/PresentedICSE2006.ppt
- Preliminary guidelines for empirical research in software engineering, Kitchenham et al. TSE 02
  - http://csdl.ics.hawaii.edu/techreports/05-06/doc/Kitchenham2002.pdf
- FOSE 07: The Future of Empirical Methods in Software Engineering Research
  - http://www.simula.no/research/engineering/publications/Simula.SE.13
- Hints for Reviewing Empirical Work in Software Engineering Tichy ESE 00
  - http://www.springerlink.com/content/rr70j282h2k01960/
- Readings in Empirical Evaluation for Budding Software Engineering Researchers
  - http://csdl.ics.hawaii.edu/techreports/05-06/05-06.html
- Courses
  - http://www.cs.toronto.edu/~sme/CSC2130/index.html
  - http://www.cs.tut.fi/~pselonen/OHJ-1860/

Discussion

- Limitations and issues your approach/implementation currently cannot address
  - Optional: how are you going to address them in future work
- Other caveats (scope of your approach)
- It is often a good idea to list (obvious) limitations and discuss possible solutions for them rather than hiding them
  - Reviewers can often identify obvious limitations even if you don’t state them; then they will criticize your work on these limitations (you often don’t have a rebuttal against these criticisms in conference reviews).
  - If your paper discusses these obvious limitations as well as their potential solutions, the situation can be alleviated (it is like you have a rebuttal in your paper already before being criticized!).
- Possible applications of your approach that you haven’t validated but are convincingly feasible or effective.
- See my TACAS 05 Symstra paper

Solution Characterization

- Related to insight
- Under what situations (e.g., characteristics of the software under test) your proposed solution would achieve the best results and under what situations your proposed (e.g., characteristics of the software under test) would achieve the worst results.
  - "killer apps“/show-off vs. turn-off cases
- You may discuss your solution characterization in the discussion section and/or conclusion section
- It depends whether you want to discuss your solution characterization in the introduction

Conclusions (and Future Work)

- Often easy to write conclusions
  - nothing here should surprise readers; simply summarize your contributions and findings
  - In the introduction, “We propose a new approach …”

  vs. In the conclusions, “We have proposed a new approach …”
- You can state the broader impacts of your approach and your vision
- You can optionally describe limitations and future work here if you don’t have a discussion section for them and propose future work
- May mark your territory of your future work by saying “We are currently doing X..., and preliminary results are promising.” (http://infolab.stanford.edu/~widom/paper-writing.html)
- Acknowledgments

Example Review on a Rejected Paper

- “The paper claims to make four contributions. In terms of contribution A, XYZ et al. had proposed a similar approach to address the same problem. In terms of contribution B, the authors had already previously published a paper describing the major similar ideas of this contribution. Then I would expect that the evaluation would be the major contribution of this paper. However, unfortunately the evaluation is weak. Therefore, I wouldn't recommend accepting this paper.”

Don’t write your paper to fit the above profile

Read/Review Your Paper Like a Reviewer

- A reviewer would be very happy when a question comes in to their mind when finishing a sentence in your paper, and immediately your next sentence addresses the reviewer’s question
- You should anticipate/predict questions from reviewers at places in your paper
  - Resolve these questions immediately after these places
  - Or give reviewers head-up on where you will address them (i.e., giving them hints)
- If your answers are too far away
  - Reviewers may not read your paper carefully enough to find out the answers
  - Reviewers are not happy with heavy load of negative questions in mind along the way, even if these questions are resolved in later parts of your paper

Selected Advice from/for Empirical SE Researchers

- Slides 42-57 by Victor Basili and Sebastian Elbaum at ICSE 2006 on “Better Empirical Science for Software Engineering: How not to get your empirical study rejected: we should have followed this advice”
  - http://csce.unl.edu/~elbaum/talks/PresentedICSE2006.ppt
- Much advice there applicable in general

Research Generalization Technique: “Balloon”/“Donut”

- Adopted by Tao Xie’s research group
- *Balloon*: the process is like blowing air into a balloon
- *Donut*: the final outcome is like a donut shape (with the actual realized problem/tool as the inner circle and the applicable generalized problem/solution boundary addressed by the approach as the outer circle)
- Process: do the following for the problem/solution space separately
  - Step 1. Describe what the exact concrete problem/solution that your tool addresses/implements (assuming it is X)
  - Step 2. Ask questions like “Why X? But not an expanded scope of X?”
  - Step 3. Expand/generalize the description by answering the questions (sometimes you need to shrink if overgeneralize)
  - Goto Step 1

Example Application of “Balloon”/“Donut”

- Final Product: Xusheng Xiao, Tao Xie, Nikolai Tillmann, and Jonathan de Halleux. Precise Identification of Problems for Structural Test Generation. ICSE 2011 http://people.engr.ncsu.edu/txie/publications/icse11-covana.pdf
- Problem Space
  - Step 1. (Inner circle) Address too many false-warning issues reported by Pex
  - Step 2. Why Pex? But not dynamic symbolic execution (DSE)?
  - Step 3. Hmmm… the ideas would work for the same problem faced by DSE too
  - Step 1. Address too many false-warning issues reported by DSE
  - Step 2. Why DSE? But not symbolic execution?
  - Step 3. Hmmm.. the ideas would work for the same problem faced by symbolic execution too
  - ….
  - Outer circle: Address too many false-warning issues reported by test-generation tools that focus on structural coverage and analyze code for test generation (some techniques work for random test generation too)

Example Application of “Balloon”/“Donut”

- Final Product: Xusheng Xiao, Tao Xie, Nikolai Tillmann, and Jonathan de Halleux. Precise Identification of Problems for Structural Test Generation. ICSE 2011 http://people.engr.ncsu.edu/txie/publications/icse11-covana.pdf
- Solution Space
  - Step 1. (Inner circle) Realize issue pruning based on symbolic analysis implemented with Pex
  - Step 2. Why Pex? But not dynamic symbolic execution (DSE)?
  - Step 3. Hmmm… the ideas can be realized with general DSE
  - Step 1. Realize issue pruning based on symbolic analysis implemented with DSE
  - Step 2. Why DSE? But not symbolic execution?
  - Step 3. Hmmm … the ideas can be realized with general symbolic execution
  - ….
  - Outer circle: Realize issue pruning based on dynamic data dependence (which can be realized with many different techniques!), potentially the approach can use static data dependence but with tradeoffs between dynamic and static

Example on how to Generalize from a tool to a general approach

- Earlier version
  - Suresh Thummalapenta and Tao Xie. NEGWeb: Static Defect Detection via Searching Billions of Lines of Open Source Code. NCSU Dept CS, Technical report TR-2007-24, September 16, 2007.
  - http://people.engr.ncsu.edu/txie/publications/TR-2007-24.pdf

- Published version [ASE 09, ASE Journal 11 special issue]
  - Suresh Thummalapenta and Tao Xie. Alattin: Mining Alternative Patterns for Detecting Neglected Conditions
  - http://people.engr.ncsu.edu/txie/publications/ase09-alattin.pdf

Example Papers with Good Writing

- Papers with analysis/testing algorithms
  - Su et al. http://www.cs.ucdavis.edu/~su/publications/
  - Sen et al. http://srl.cs.berkeley.edu/~ksen/
- Papers with experiments
  - Harrold et al. http://pleuma.cc.gatech.edu/aristotle/publications.php
  - Orso et al. http://www.cc.gatech.edu/~orso/papers/index.html
- Papers with case studies
  - Murphy et al. http://people.cs.ubc.ca/~murphy/research-papers.html
  - Robillard et al. http://www.cs.mcgill.ca/~martin/papers.html
- Other papers
  - Xie et al. http://people.engr.ncsu.edu/txie/publications.htm

Example Papers with Good Writing II

- http://www.sigsoft.org/awards/disPapAwd.htm
- http://www.sigsoft.org/awards/ImpactAward.htm
- http://www.sigsoft.org/awards/mostInfPapAwd.htm

More recent highly cited papers may provide good examples

- http://academic.research.microsoft.com/CSDirectory/paper_category_4_last5.htm
- http://academic.research.microsoft.com/CSDirectory/paper_category_4_last10.htm
- http://academic.research.microsoft.com/CSDirectory/Paper_category_4.htm

More Readings

- https://sites.google.com/site/asergrp/advice
  - Mapping out a Research Agenda http://people.engr.ncsu.edu/txie/publications/researchagenda.pdf
  - Common Technical Writing Issues http://people.engr.ncsu.edu/txie/publications/writeissues.pdf
  - Research Skills http://people.engr.ncsu.edu/txie/advice/researchskills.pdf
  - Graduate Student Survival/Success Guide http://people.engr.ncsu.edu/txie/advice/gradstudentsurvival.pdf
  - Advice on Getting a Start into Research http://people.engr.ncsu.edu/txie/adviceonresearch.html

More Readings

- http://spoke.compose.cs.cmu.edu/ser04/course-info.htm
- http://www.cs.cmu.edu/~Compose/shaw-icse03.pdf
- http://infolab.stanford.edu/~widom/paper-writing.html
- http://www.cse.msu.edu/~chengb/Writing/intro-guidelines-stirewalt.txt
- http://www1.cs.columbia.edu/~kaiser/relatedwork.htm
- http://www.cs.washington.edu/homes/mernst/advice/write-technical-paper.html
- http://www.cs.washington.edu/homes/mernst/advice/review-technical-paper.html
- http://www-bsac.eecs.berkeley.edu/~muller/jmems.web/sds_editorial_june_2003.pdf
- http://www.cs.berkeley.edu/~pattrsn/talks/writingtips.html
- http://www.cs.tufts.edu/~nr/pubs/two-abstract.html

More Reading

- https://sites.google.com/site/slesesymposium/slese12.pdf by Zhendong Su
- http://avandeursen.wordpress.com/2013/07/10/research-paper-writing-recommendations/ by Arie van Deursen
- Book: Crafting Your Research Future: A Guide to Successful Master's and Ph.D. Degrees in Science & Engineering by Charles Ling and Qiang Yang
  - http://www.amazon.com/Crafting-Your-Research-Future-Engineering/dp/1608458105