

Research Methods in computer science

Fall 2013

Lecture 20

Omprakash Gnawali
November 5, 2013

Agenda

Conference Plans

Oral presentations

Paper Introduction and Related Work

Feedback

Three essential components

Summary

Strength

Weakness

Should cover style and substance

Presentation Order

Behrang Mehrparvar

Cheng Wang

Daxiao Liu

Dong Han

Jeremy A Kemp

Li Wei

Qiang Li

Rengan Xu

SeyyedHessamAldin

MohammadMoradi

Shengrong Yin

Tao Feng

Xianping Zhou

Xiaonan Tian

n-2 and n-1 gives
feedback for n

Take notes for
HW10

How to write a great research paper

Simon Peyton Jones

Microsoft Research, Cambridge

Writing papers is a skill

- Many papers are badly written
- Good writing is a skill you can learn
- It's a skill that is worth learning:
 - You will get more brownie points (more papers accepted etc)
 - Your ideas will have more impact
 - You will have better ideas

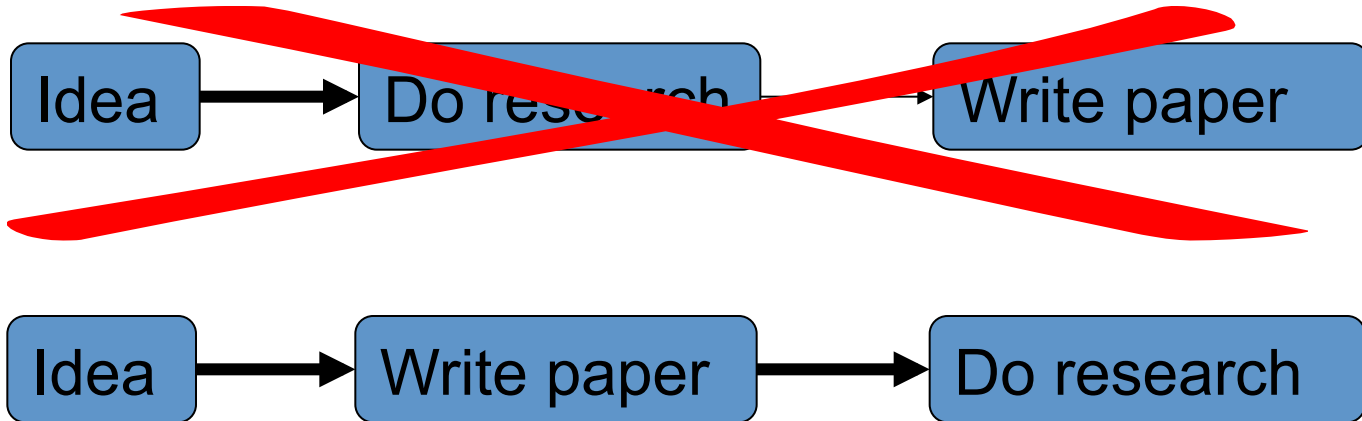
Increasing importance



Writing papers: model 1



Writing papers: model 2



- Forces us to be clear, focused
- Crystallises what we don't understand
- Opens the way to dialogue with others: reality check, critique, and collaboration

Do not be intimidated

Fallacy You need to have a fantastic idea before you can write a paper. (Everyone else seems to.)

Write a paper,
and give a talk, about

any idea,

no matter how weedy and insignificant it may
seem to you

Do not be intimidated

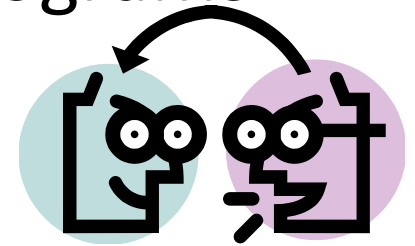
Write a paper, and give a talk, about any idea, no matter how insignificant it may seem to you

- **Writing the paper is how you develop the idea in the first place**
- It usually turns out to be more interesting and challenging than it seemed at first

The purpose of your paper

Papers communicate ideas

- Your goal: to infect the mind of your reader with **your idea**, like a virus
- Papers are far more durable than programs (think Mozart)



The greatest ideas are (literally)
worthless if you keep them to yourself

The Idea

Idea

A re-usable insight,
useful to the reader

- Figure out what your idea is
- Make certain that the reader is in no doubt what the idea is. Be 100% explicit:
 - “The main idea of this paper is....”
 - “In this section we present the main contributions of the paper.”
- Many papers contain good ideas, but do not distil what they are.

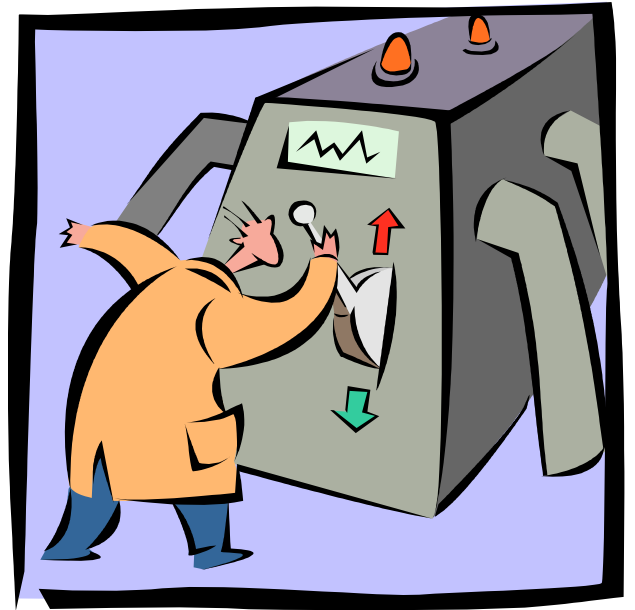
One ping

- Your paper should have just one “ping”: one clear, sharp idea
- Read your paper again: can you hear the “ping”?
- You may not know exactly what the ping is when you start writing; but you must know when you finish
- If you have lots of ideas, write lots of papers

Thanks to Joe Touch for “one ping”

The purpose of your paper is not...

To describe
the WizWoz
system



- Your reader does not have a WizWoz
- She is primarily interested in re-usable brain-stuff, not executable artefacts

Your narrative flow

- Here is a problem
- It's an interesting problem
- It's an unsolved problem
- **Here is my idea**
- My idea works (details, data)
- Here's how my idea compares to other people's approaches

I wish I knew how to solve that!

I see how that works. Ingenious!



Structure (conference paper)

- Title (1000 readers)
- Abstract (4 sentences, 100 readers)
- Introduction (1 page, 100 readers)
- The problem (1 page, 10 readers)
- My idea (2 pages, 10 readers)
- The details (5 pages, 3 readers)
- Related work (1-2 pages, 10 readers)
- Conclusions and further work (0.5 pages)

The abstract

- I usually write the abstract last
- Used by program committee members to decide which papers to read
- Four sentences [Kent Beck]
 1. State the problem
 2. Say why it's an interesting problem
 3. Say what your solution achieves
 4. Say what follows from your solution

Abstract - Example

Many papers are badly written and hard to understand. This is a pity, because their good ideas may go unappreciated. Following simple guidelines can dramatically improve the quality of your papers. Your work will be used more, and the feedback you get from others will in turn improve your research

Structure

- Abstract (4 sentences)
- **Introduction** (1 page)
- The problem (1 page)
- My idea (2 pages)
- The details (5 pages)
- Related work (1-2 pages)
- Conclusions and further work (0.5 pages)

The introduction (1 page)

- 1. Describe the problem**
- 2. State your contributions**

...and that is all

ONE PAGE!

Describe the problem

1 Introduction

There are two basic ways to implement function application in a higher-order language, when the function is unknown: the *push/enter* model or the *eval/apply* model [11]. To illustrate the difference, consider the higher-order function **zipWith**, which zips together two lists, using a function **k** to combine corresponding list elements:

```
zipWith :: (a->b->c) -> [a] -> [b] -> [c]
zipWith k []      []      = []
zipWith k (x:xs) (y:ys) = k x y : zipWith xs ys
```

Here **k** is an *unknown function*, passed as an argument; global flow analysis aside, the compiler does not know what function **k** is bound to. How should the compiler deal with the call **k x y** in the body of **zipWith**? It can't blithely apply **k** to two arguments, because **k** might in reality take just one argument and compute for a while before returning a function that consumes the next argument; or **k** might take three arguments, so that the result of the **zipWith** is a list of functions.

Use an example to introduce the problem

State your contributions

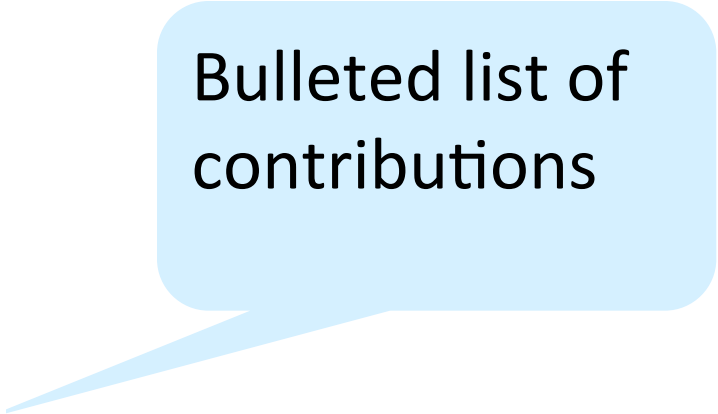
- Write the list of contributions first
- The list of contributions drives the entire paper: the paper substantiates the claims you have made
- Reader thinks “gosh, if they can really deliver this, that’s be exciting; I’d better read on”

State your contributions

Which of the two is best in practice? The trouble is that the evaluation model has a pervasive effect on the implementation, so it is too much work to implement both and pick the best. Historically, compilers for strict languages (using call-by-value) have tended to use `eval/apply`, while those for lazy languages (using call-by-need) have often used `push/enter`, but this is 90% historical accident — either approach will work in both settings. In practice, implementors choose one of the two approaches based on a qualitative assessment of the trade-offs. In this paper we put the choice on a firmer basis:

- We explain precisely what the two models are, in a common notational framework (Section 4). Surprisingly, this has not been done before.
- The choice of evaluation model affects many other design choices in subtle but pervasive ways. We identify and discuss these effects in Sections 5 and 6, and contrast them in Section 7. There are lots of nitty-gritty details here, for which we make no apology — they were far from obvious to us, and articulating these details is one of our main contributions.

In terms of its impact on compiler and run-time system complexity, `eval/apply` seems decisively superior, principally because `push/enter` requires a stack like no other: stack-walking



Bulleted list of contributions

Do not leave the reader to guess what your contributions are!

Contributions should be refutable

| 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 answers to these questions form the fundamental contributions of this paper:

- Detailed examination of where energy goes reveals that over 50% of the electricity is spent on computing. PC's account for 17% of the bill despite the fact that their utilization is very low. Networking equipment comes at 3.5% and shows no temporal changes despite variations in traffic load.
- Data analysis shows that estimating savings based on a few isolated desktop measurements is prone to errors due to the wide spread of PC power draws. Assuming that a day of power is representative and using it to calculate yearly values can be off by as much as 20%.
- Our deployment and data studies expose the relative importance of device coverage versus duration of deployment. Once a deployment is past the first month of data collection, one must prioritize the 'what to measure' question over the time scale of the study.

Some examples

Introduction – Another take

1. What is the problem?
2. Why is it interesting and important?
3. Why is it hard?
4. Why hasn't it been solved before?
5. What are the key components of my approach and results?

Courtesy Jennifer Widom

No “rest of this paper is...”

- Not:

“The rest of this paper is structured as follows. Section 2 introduces the problem. Section 3 ... Finally, Section 8 concludes”.
- Instead, **use forward references from the narrative in the introduction.**

The introduction (including the contributions) should survey the whole paper, and therefore forward reference every important part.

Related work

Fallacy

To make my work look good, I have to make other people's work look bad

The truth: credit is not like money

Giving credit to others does not diminish the credit you get from your paper

- Warmly acknowledge people who have helped you
- Be generous to the competition. “In his inspiring paper [Foo98] Foogle shows.... We develop his foundation in the following ways...”
- Acknowledge weaknesses in your approach

Credit is not like money

Failing to give credit to others can
kill your paper

If you imply that an idea is yours, and the referee knows it is not, then either

- You don't know that it's an old idea (bad)
- You do know, but are pretending it's yours (very bad)

Related Work

- Collect papers
 - Identify related papers and their related papers and their related papers...
 - ACM/IEEE libraries
 - Google/Google Scholar
- Organize the papers
 - Use some structure (tables, graphs, etc.)
- Relate to each work/group you mention

Related Work - Common Structure

Sub sections (3-4)

Discuss 1-5 papers per subsection

It is ok to have “Other” subsection

Try to have subsection titles but at least have separate paragraphs

Low-power Sensornet Research

| Approaches | Systems |
|--------------------|--|
| Application | AppSleep, TinyDB |
| Network | FPS, SPAN, GAF, ASCENT, MT, ETX, Energy Routing Metrics |
| MAC | S-MAC, T-MAC, B-MAC, STEM, X-MAC, SCP-MAC, PAMAS, Piconet, Dozer, Koala |
| OS | TinyOS Scheduler |
| Hardware | Trio and Prometheus |

Prior Work

