Tales from Dissertationland and the Job Hunt

Jonathan Aldrich
Assistant Professor
Institute for Software Research International
School of Computer Science
Carnegie Mellon University

This talk

- Alex Potanin: Will you give advice to the Ph.D. students at the doctoral symposium?
- J: Are you sure you mean me? After all, I'm just out of grad school myself!
- J: Besides, I'm not funny.
- A: We want someone who got a good job recently enough to remember what it was like.

- Caveat emptor: this is more "recent war stories" than "wisdom from on high"
- **Advice can be very dry**
  - So I've added a few stories (and occasionally a bit of technical detail) Hope this helps!

Outline

- Dissertationland
  - Topic and area
  - Finding and solving problems
  - Along the path
  - Making it through
- The Job Hunt
  - Writing
  - Networking
  - Talks
  - Interviewing
  - Closing the deal
  - Other advice

Do you like your topic Hot or Cold?

- Hot topics
  - Advantage: everyone cares about it*

***Right now…but in 5 years, who knows?**

How a great idea ended in tragedy

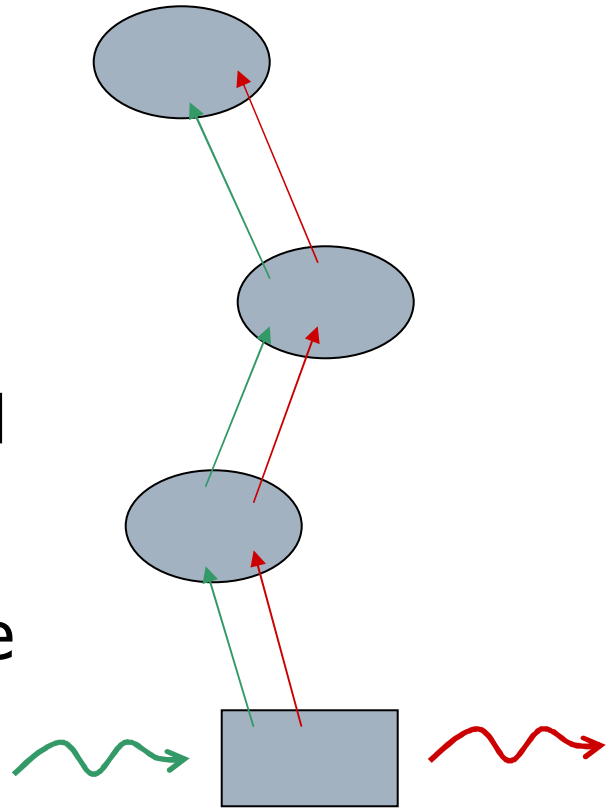
- My first real research project
  - Sychronization in Java is expensive
  - So, don't lock a lock if there can't possibly be contention
- Idea: optimize thread-local objects
  - An object is thread-local if it's not stored in the heap (or at least doesn't escape its creating method)
    - 3 OOPSLA '99 papers, 1 SAS '99 paper

How a great idea ended in tragedy

- Enhancement
  - What about objects that escape the method, but aren't shared?
    - Base case: shared global variable

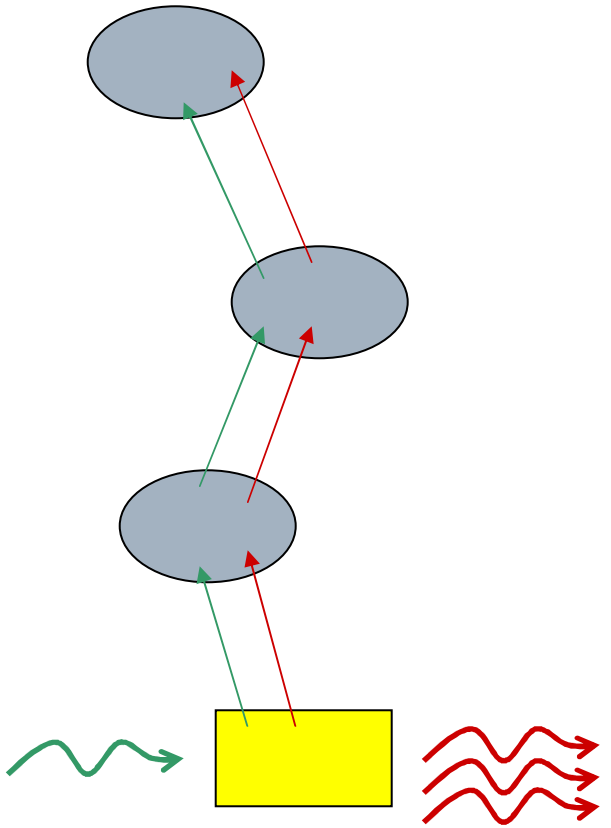
How a great idea ended in tragedy

- Enhancement
  - What about objects that escape the method, but aren't shared?
    - Base case: shared global variable
    - Recursive case: one thread writes to a shared variable, another thread reads from it
    - No other sharing possible in Java
      - treat thread & other system objects as global



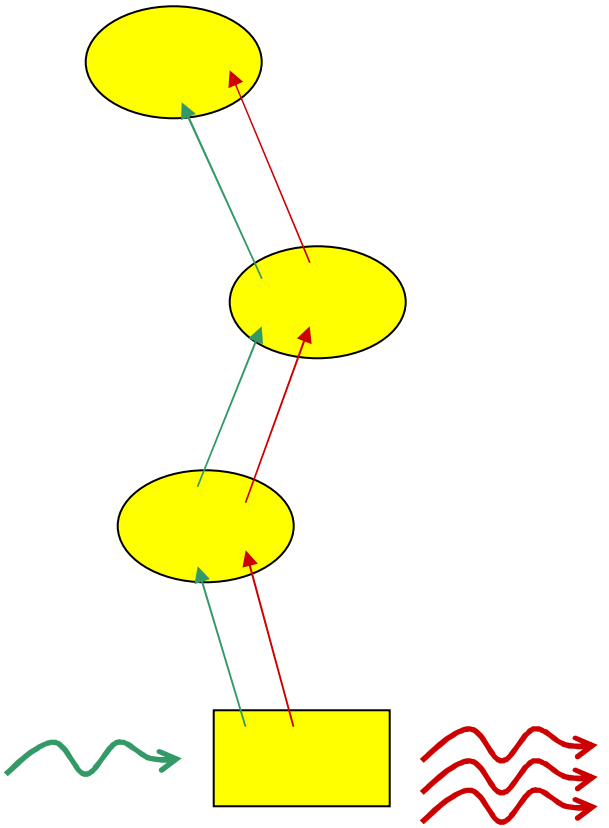
How a great idea ended in tragedy

- The algorithm
  - Compute which threads are instantiated more than once
  - Compute what code is executed by each thread
  - Compute what variables are written & read by each thread
- An object is shared if:
  - It is stored in a global variable that is read by one thread and written by another



How a great idea ended in tragedy

- The algorithm
  - Compute which threads are instantiated more than once
  - Compute what code is executed by each thread
  - Compute what variables are written & read by each thread
- An object is shared if:
  - It is stored in a global variable that is read by one thread and written by another
  - It is stored in a field of a shared object, and that field is read by one thread and written by another



How a great idea ended in tragedy

- The tragedy
  - I used off-the-shelf alias analysis to figure out what fields point to what objects
  - The scalable analyses gave poor results
  - The alias analyses that gave good results didn't scale
  - I beat my head against the wall for 6 months trying to improve the alias analysis
  - Erik Ruf published an algorithm to do the same thing, which used an alias analysis customized for the problem
  - Two rejections in a row; never published in a major conference

How a great idea ended in tragedy

- Life lessons learned
  - Hot topics are risky. You might get scooped*
    * advisors can help avoid this, but mine was on sabbatical
  - Erik Ruf is a nice guy, even though he scooped me.
  - You can still publish your work in a journal, especially if it's invited ;-)
- Technical lessons learned
  - Problem customization is crucial to alias analysis research
  - Don't pick the alias analysis problem unless you've got something amazing up your sleeve
    - E.g. John Whaley, BDD representation, PLDI '04

Do you like your topic Hot or Cold?

- Hot topics
  - Advantage: everyone cares about it
  - Disadvantage: really hard to stand out from the crowd
  - Disadvantage: easy to get scooped
- Cold topics
  - Must do *much* better at making an argument
  - My work:
    - Enforcing architecture specs: very ’90s
    - AOP & modular reasoning: different crowds
    - Prototypes (’80s) and Multiple Dispatch (’80s, ’90s)
  - Easy to convince people you’ve done something novel, hard to make them care

Split Personality

- Cross-disciplinary research
  - Apply techniques from one area to a problem in another
    - ArchJava: PL techniques applied to SE problem
  - Integrate techniques from two areas
- Advantage: easy to make a significant, novel contribution
  - ArchJava was low-hanging fruit, in a sense: one new trick—that was not even deep type theory—and a lot of implementation, validation, and proofs made it all work
  - No-one had solved the problem before because they hadn't thought of it in the way that I did

Split Personality

- Disadvantage: people from both areas can dismiss you as not relevant
  - Some in SE criticize ArchJava as being unadoptable
    - Valid in the short term
    - Have to argue that impact on timescale of language adoption is large
  - Many people in PL simply don't care
    - Architecture is too "fuzzy" for them
    - Helps to make the case that the same technology can address issues that they do care about.

Split Personality

- Disadvantage: people from both areas can dismiss you as not relevant
  - Example: AOP & modular reasoning
    - AOP crowd: you're bringing the same old modularity, we need a new modularity for AOP
      - Have to argue that old ideas are still useful, just need to be adapted
    - Hard-core PL crowd: AOP is wacko, anyway
      - Have to argue that AOP is less wacko if you have a modular reasoning property

Split Personality

- Fortunately the OO community is more open to PL/SE work than either the core PL or core SE conferences
- Interview Experiences
  - I was interviewed by schools looking only in PL, and only in SE
  - I was **not** interviewed because of a bad area match, by other schools looking only in PL, and only in SE!!!

What's Your Problem?

- Crystallizing the problem is one of the hardest things about doing research
- Common trap
  - That system is so broken, I can do it better!
    - (without a specific goal for "better")

How I wrote a nice paper on the wrong problem

- My dissertation: enforcing a software architecture specification
  - Subproblem: how to specify and check communication through shared data
- Ownership looked promising

How I wrote a nice paper on the wrong problem

- But existing ownership systems were "bad"
  - Couldn't express capturing a pointer
    - So I combined with uniqueness
  - Couldn't express iterators, events
    - So I weakened ownership guarantee to "capability-based encapsulation"
  - Hadn't been implemented, don't know how to implement actual language
    - So I implemented it and solved issues
  - No empirical evaluation of practicality
    - So I evaluated it on 4000 lines of interesting library and application code
  - Lots of annotations to write
    - So I wrote an inference system
      - Alias analysis bites again!!

How I wrote a nice paper on the wrong problem

- Did I mention I got the problem wrong?
- "Capability-based encapsulation"
  - Means you can only access an object if you can name it in the ownership system
  - Similar to OO pointers: you can access an object if you have a name (variable) for it
  - Limited improvement over Java
  - Useless for my dissertation
- Still a nice paper
  - Emphasized practicality of ownership
  - Empirical evaluation
  - Initial work on inference
  - But I hadn't focused on the real problem!

How I wrote a nice paper on the wrong problem

- What I should have done
  - Focus specifically on what property I needed to enforce architecture
  - I did this later, as I was finishing up my dissertation

Ways I've found good problems

- The literature
  - Communication integrity: comes from SE literature on architecture
- A technical property from another area
  - Karl Crary: "So, do you have any kind of abstraction result for ArchJava?"
  - J: "Abstraction? What's that?"
  - Result: ECOOP '05 paper on abstraction in the presence of advice
  - Another example: OO substitutability, applied to typestate (FSE '05)
- Reconcile two properties that seem to be in conflict
  - ECOOP '05 paper: prototypes and multiple dispatch
    - My student Lee Salzman's idea, not mine

Brain Teasers

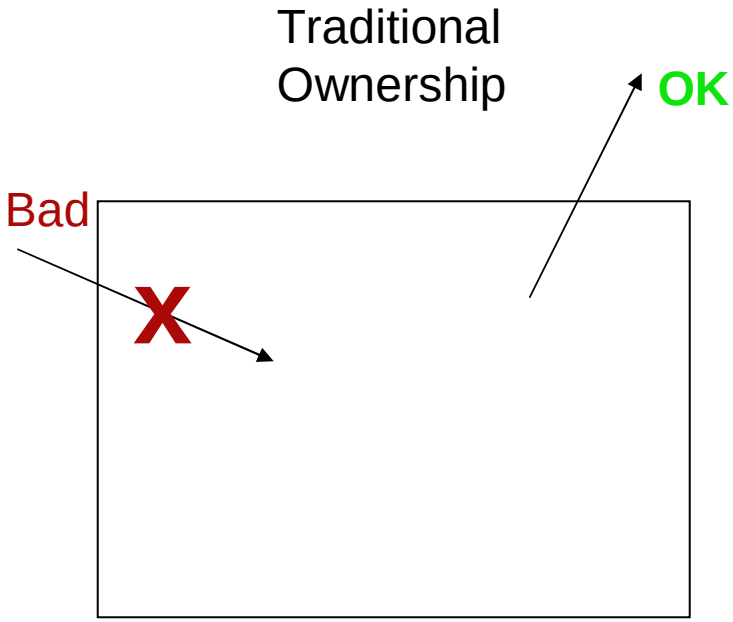
- Once you have the problem, how do you solve it?
- Often involves taking a break and looking at the solution from another angle

Brain Teasers

- I’ve only solved 2 hard problems
  - The other solutions were either obvious from the problem statement, straightforward engineering, or my students did the real thinking

Brain Teaser #1

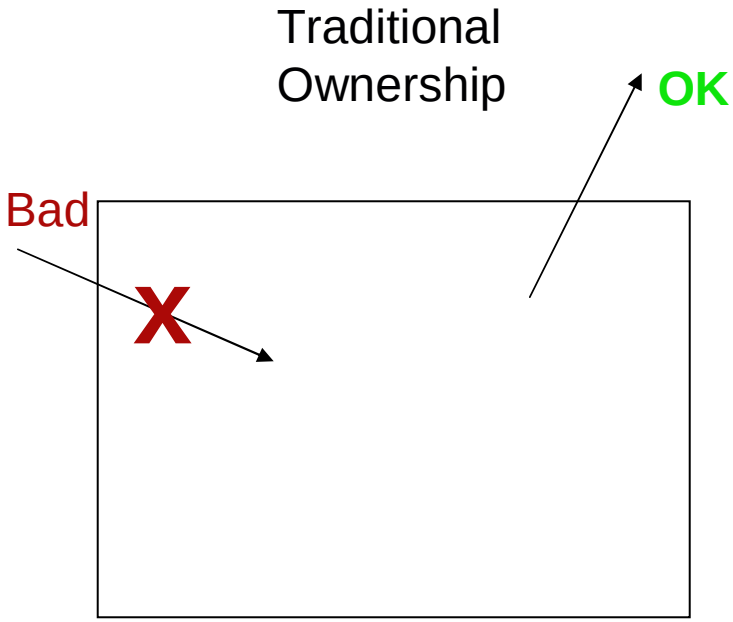
- Recasting ownership as permissions



Pointers can go out but not in

Brain Teaser #1

- Recasting ownership as permissions



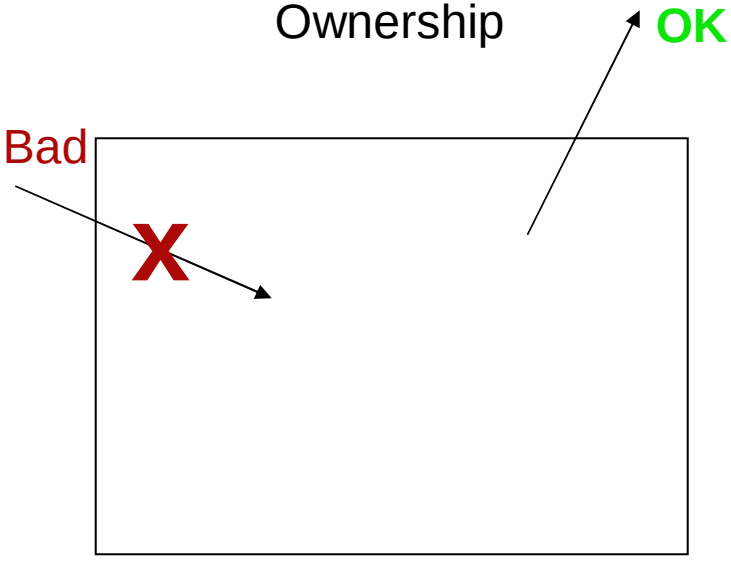
Problem: Too inflexible to check architecture!

Need ability to specify arbitrary sharing relationships
(but enforce whatever the architect specifies)

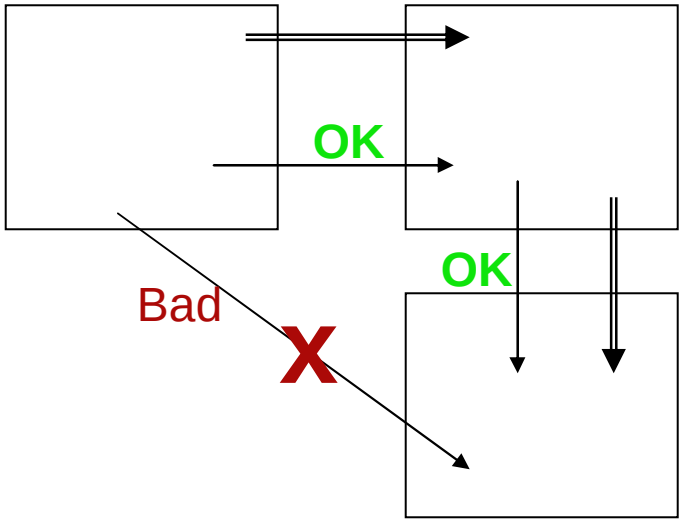
Brain Teaser #1

- Recasting ownership as permissions

Traditional Ownership



Permission-based Ownership (3am idea)



Problem: Too inflexible to check architecture!
Need ability to specify arbitrary sharing relationships (but enforce whatever the architect specifies)

Key idea: specify arbitrary access permissions
Permissions are not transitive!

Brain Teaser #2

- Modular Reasoning about Advice
  - Advice should affect declarations, not values
    - Otherwise, difficult to reason about effects—as bad as pointers
  - Treat recursive calls separately from external calls
    - What matters is module crossings

Save the World!

- Don’t try to save the world
  - There’s plenty of time when you’re a professor and you have students to help you
    - (but need to stay focused even then)
  - Agree with advisor on scope: not a bad idea to get a written agreement
- Don’t be afraid to change areas
  - If you think the new area is more compelling to others and it’s more interesting to you
  - And if you’re not too far along already
  - I did it twice!
    - Java synchronization optimization
    - AOP & modules

Serving Two Masters

- I was co-advised, and I have a co-advised student
  - *great* way to get more than one perspective
- Things to watch out for
  - Find a clear topic that both advisors are interested in
    - This was easy for me as a student, but was hard with the student I co-advise
      - We finally succeeded after nearly 2 years
  - Manage the interaction
    - Make sure they're able to work together effectively
    - Meeting separately with each advisor is often a good idea
    - Make clear that you cannot do the union of their expectations

Down the Garden Path

- Once you figure out what your thesis is, focus like a laser on it!

How I wrote a nice paper that wasn't part of my dissertation

- Architecture literature
  - Connectors are important, too!
  - What would a connector look like in ArchJava?
- Solution (ECOOP '03)
  - Method defines semantics of invocation
    - Reflective access to calls on connector
  - Method defines typechecking rules
    - Reflective access to types of ports
  - Evaluation based on a taxonomy of connectors, distributed system

How I wrote a nice paper that wasn't part of my dissertation

- Did I mention this wasn't part of my dissertation?
  - My dissertation was about architectural conformance
  - This was about reusability, abstraction, and customizable typechecking
    - No conformance story
  - Didn't make it into my dissertation writeup
- Nice paper, but it should have waited until after I graduated

Survivor!

- The hardest thing about a dissertation is finishing it
  - Write lots of papers; then you have material to work with
    - Mike Ernst: publish everything you do, even if it's in a piddly workshop
  - Knock off the hard stuff first
    - Worst thing about connectors paper: distracted me from hardest issue of dissertation
    - Didn't solve it until after my first couple of interviews
  - Don't get discouraged by problems
    - ECOOP '05 paper on aspects: rejected 3 times (nearly 4—required shepherding)
    - It's a good paper now; great reviews at ECOOP, but I had to fine-tune it (both technically as it was a new area, and sales-wise) to get it right
  - Do a bit every day: as long as you're always making progress, you'll finish eventually!

The Insane Asylum

- Becky: Tell them they need to do extracurricular stuff, too. As in, don't sit in front of the computer all day.
- J: Why? Because it helps in getting a job, or because it helps maintain your sanity?
- **B: Well, the second is not unrelated to the first, right?**

Interviewing

Writing

- Kent Beck: how to write an abstract
  - State the problem
  - State why the problem is a problem
  - Surprise the reader with your solution
  - State the consequences of your solution
- Source:
  - http://www.acm.org/sigs/sigplan/oopsla/oopsla96/how93.html

Writing

- Focus on the abstract/introduction
  - Most important part of a paper
    - Assuming you already have the ideas worked out
  - I write this first *and* last
- Outline before you write
- Get outside feedback
  - The pickier the better
  - Feedback on the technical content, the argument, and the grammar/style of writing
- Revise
  - Take one paper and focus intensely
    - Revise the argument until it's exactly right
    - Go over each sentence in detail: several minutes
    - Will take a long time, but you'll learn a lot, and next time you can do it quickly

Networking

- Give drafts of papers to people in your area
  - Offer to read their papers, too!
- Good to do summer internships elsewhere
  - So people can recommend you
  - Can also help you get a research job

What to do at ECOOP

- From David Notkin, my advisor, to his students:

"Why am I bringing you all down to the conference? In part, I just want the University of Washington to have good showing, since its good for the department. But more importantly, its good for you (1) to see the people who've written papers you've read, (2) to see what's current in software engineering research, (3) to start to build relationships with other researchers in the field, (4) to tell people what you're doing and to find out what they are doing, and (5) to find out that you're at least as smart and good as many of those researchers.

"So, you should work hard to attend lots of sessions and read lots of the papers. But it's unlikely that you'll go to every session: some will be genuinely uninteresting to you. In addition, the most important part of a conference is "schmoozing," standing in hallways talking to colleagues (satisfying most or all of the items in the previous category). You'll see me and lots of others doing this.

What to do at ECOOP

"It's scary trying to meet these "famous" people. I'll try and introduce you when I can, but I'll be pretty busy. So it's OK (actually, its more than just OK) to be a little (or a lot) pushy. If you see people you want to listen to having a conversation, feel free to move on up to them and try to listen (unless for some reason it seems like it's a personal conversation and is thus inappropriate). Sometimes they'll acknowledge you, sometimes they won't. But its worth trying to get involved in these conversations when possible. (Even listening by itself can be valuable.) Of course, the best way to get involved is to ask a question: it flatters people and makes them respond to you. And you learn something.

"Trying to have meals with folks is a really good way to meet them. Some people you already know probably know a couple of people from other places, so if they set something up, it'd be nice to try to bring another UW student or two along. (For women students, there may be a Sisters lunch one-day. It'll probably be marked on a bulletin board. Go if you can and want).

"Hang out some with each other. But don't do this exclusively, since you can do that in Seattle, but you can't schmooze with the others here. Debriefing with each other on sessions, papers, interactions with others, etc. is of value, though, and you should do this with each other on occasion."

Preparing Talks

- Preparation
  - Talk about the story with your advisor
  - Discuss an outline with your advisor
  - Go over slides with your advisor
  - Practice on your own to get it to the right time
    - Aim for 5 minutes less than the allotted time
  - Practice in public
    - Helps with nerves
    - Get feedback
  - Memorize your talk for the first few slides
    - Gets over that initial hump

Giving Talks

- One of the most important parts of your career
  - Will form main impression when you interview
  - Obvious importance for teaching
- Job talk story
  - Gave 2-3 practice talks
  - Revised twice *after* I started interviewing
- Bring talk in two formats
  - You never know when equipment will fail!
  - Old advice: slides & electronic
  - New advice: pick two of Laptop, USB, CD, Web

Where to Interview

- Distribution of quality
- Top places
  - Apply to a few—you might get lucky—but don’t count on these
- Medium schools
- A few safety places
  - You have a good chance at getting an offer

Timing

- Interview at a safety school early
  - Get practice in a low-key environment
  - Danger: they might make you an offer with a short time fuse!
    - They know they have to stretch to get you, and want to make another offer if you say no
    - Best if you have an offer you can hold on to—schools that have more hiring slots can afford to do this
- Schedule breaks
  - 3 schools is probably plenty for one trip
  - Need to relax between interviews, and also revise your talk

Preparing for an Interview

- Learn about the school beforehand
  - Who works in your area? What have they done recently that you could ask about?
- Know your work
  - Example: I was grilled on an old system that I had changed for my dissertation (and had forgotten!)
- Know your teaching
  - Example: I said I was an expert in design, but I wasn't; I was an expert in *enforcing* design. I had to backtrack and looked a bit naïve.
- Know your future ideas
  - Should have some elements that contrast with your dissertation
  - Should be plausible given your background
  - Most important: compelling & show understanding of field

My weird interview experience

- Expected: mostly get interviews and offers below a cutoff
- Seemingly random interviews
  - Ignored by several 2nd/3rd tier schools that were hiring (and interviewing others) in my area
- Got offers at the top 3 places I interviewed
  - Including CMU, ranked at a tie for #1 in the US
- My conclusion: Cold topic/two areas increased randomness in interview process
  - This worked to my advantage
  - I got to pick the top part of the distribution

Closing the Deal

- Talk with other interviewing students
- Compare notes on schools
- Compare offers
  - Schools collude; you should too!

Industry vs. Academia

- Industry
  - Must justify work to your boss
    - Typically has to be more applied
  - Less job security
  - Many fewer distractions
  - More direct impact on practice
- Academia
  - Must justify work to funding agencies
    - more work, but more flexibility
  - Tenure
  - Many responsibilities and meetings
  - Students as research multiplier
  - Teaching

Sources of Advice

- Networking on the Network
  - http://dlis.gseis.ucla.edu/people/pagre/network.html
- Mike Ernst's advice page
  - http://pag.csail.mit.edu/~mernst/advice/
- New SE faculty symposium at ICSE
  - http://www.cse.unl.edu/~grother/nsefs/nsefs03.html
- Writing Your Dissertation in Fifteen Minutes a Day. Joan Bolker.
- ECOOP doctoral symposium
- Your advisor's name here