

Syncing with Technology: What are our new engineers being taught?

Dr. David A. Cook
Department of Computer Science
Stephen F. Austin State University
Nacogdoches, TX 75962

cookda@sfasu.edu



Why this talk

- I consult with organizations that hire new graduates
- These graduates come with different degrees
 - Engineering (usually electrical)
 - Computer Science
 - Computer and Information Systems
 - Information Technology
- Much of this this talk comes from feedback from our Advisory Board and also from interviewing corporations that have had recent hires

Skills for large-scale development

- Team Dynamics
- Software Processes and basic programming skills
- Software Lifecycles
 - Traditional
 - Agile
- Requirements Engineering
- Software Design

Let's start with the basics

- Most graduates have a “working” knowledge of (AT MOST) two languages
 - For a loose definition of “working”
- This assumes that they come from the CS/CIS/IT side of the house
- Many Engineers know only one language

Let's start with the basics

- Typical choices
 - Java (usually on Windows)
 - C++ (mostly on Windows)
 - Visual Basic (which means Windows only)
- Problem – most colleges and universities have fallen in love with Object-Oriented languages
- Outcome – the students know classes and methods, but have problem with basic programming skills

Language Issues

- Students have learned to rely upon rich class-libraries with exotic methods for everything
- They can't perform tasks such as parsing arrays, working with multi-dimensional arrays, sorting, or searching
- Many are unable to declare and use pointers

Language Problems

- It takes months to become useful in new languages, and may require years to become proficient in writing code
- Unable to contribute effectively on legacy code
- Unable to do maintenance!!!

To combat this problem

- Be wary of new hires who claim to be experienced in multiple languages
- Plan on giving them time to learn features of languages
- Give them a mentor, and consider formal language training (in-house)
- Be very afraid of self-taught self-proclaimed experts. Teach new hires to share code and skills, and learn from others

Software Processes

- Most new hires are unfamiliar with basic debugging and testing techniques
- They do not even know enough to know they need help
- Consider workshops to teach these important skills

Software Processes

- Most new hires are used to code that is run once, then discarded.
- “Student code” mentality – little documentation, little design and planning. They don’t understand coding for maintenance!
- It requires a different mindset to effectively write code that runs 24/7, and will be around for years and years.



They have few team skills

- Unable to design or code well in a team
 - Not used to being “cog” when all they can see is a wheel
- Unused to testing/debugging/reading other developers’ code
- Solution – force team skills upon them. During hiring interviews, evaluate them as potential team members

Because they lack software engineering skills ...

- They are unfamiliar with planning
- Most are unaware of the “time sink” of meetings
- They are more deadline driven (and tend to overestimate their ability to slam code out at the last minute)

This is not as bad as you think!

- Shift from milestones to “inch-pebbles”
- Shift them to Agile methodologies, where short-term deadlines are more appropriate
- Agile methods force new hires into planning and team interaction



Requirements

- DO NOT expect new hires to be able to gather or organize requirements
- They are used to having assignments handed out to them
- They often learn well from “focused failure” – in a non-threatening manner

Lack of formal software processes..

- Often leads to code-focused mentality
- Few know how to manage complexity, cohesiveness and coupling
- Only a few are used to anything other than simple module design (which is mostly useless, anyway)

To combat this..

- Introduce them to
 - Architectural Design
 - Interface Design
 - Data Design (although this is an area that some are VERY good at!)

Why it's not as bad as you think!

- Note that “Law Schools” are not “lawyer schools” – they teach the “theory of the law”, not how to be an attorney.
- Medical Schools make graduates an “M.D.”, but still require 4 years of internship to become a “Practicing Doctor”.
- Hire for the “near future” if you use new hires. If you need immediate skills, be prepared to hire more experienced (and more expensive) developers.

What you can do!

- Assume that new developers straight out of college will need some time to become both technically and interpersonally proficient.
- Take steps to meet their needs on both of these topics

Technical skills

- Assume that pride (and insecurity) will keep new developers from asking for help.
- Classes or workshops should not be limited to those who request them – they should be presented as “skill enhancers” that all “keep new developers sharp”.
- New hires are often timid about asking for training.

Two “must haves”

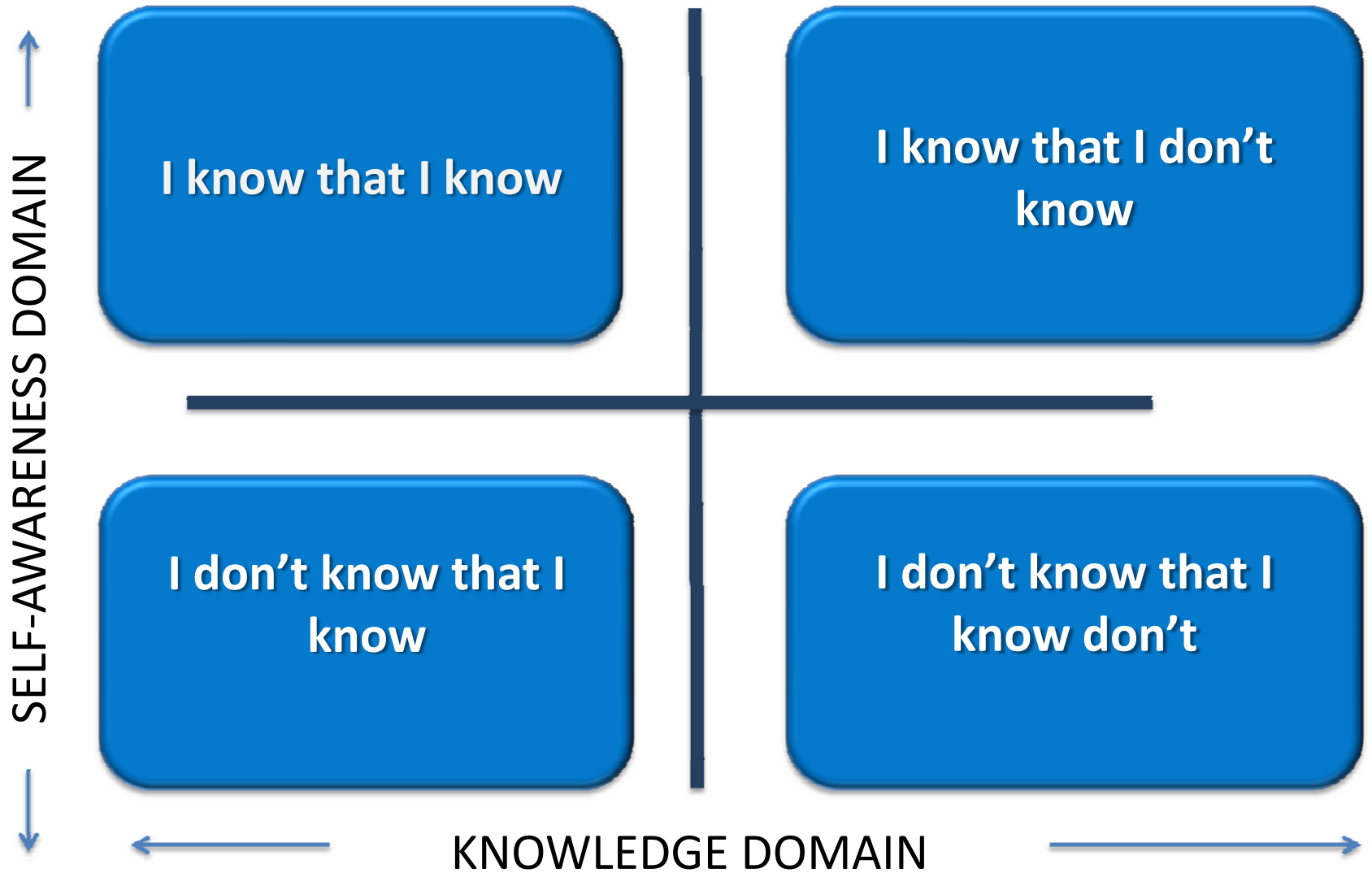
- A mentor program that is separate from the supervisory chain
 - The mentor should not even work for the same supervisor
- A peer review program that is not optional
 - Education/Training
 - Knowledge
 - Skill Enhancement
 - Sharing of ideas

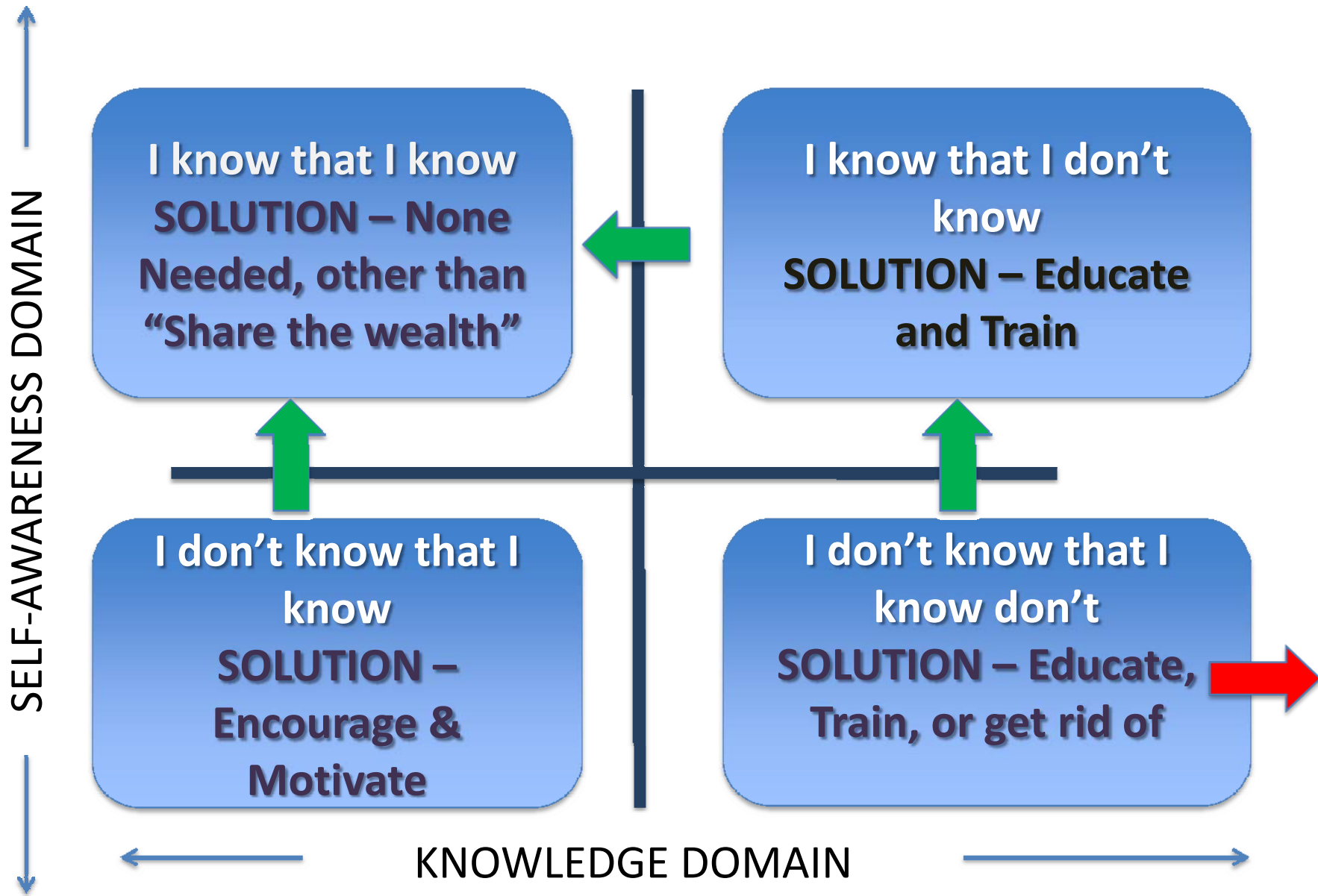
Why required skill classes?

- It's the old "If all you have is a hammer – all problems look like nails" problem
- Many new graduates do not know what they don't know



Cook - What your developers don't know





Combat the “newness” with

- Mentors
- Constructive and non-supervisory feedback
- Opportunities to work with diverse group of developers
- Constructive teamwork – with engineers who can effectively mentor, train, teach, and share, NOT just give busywork!

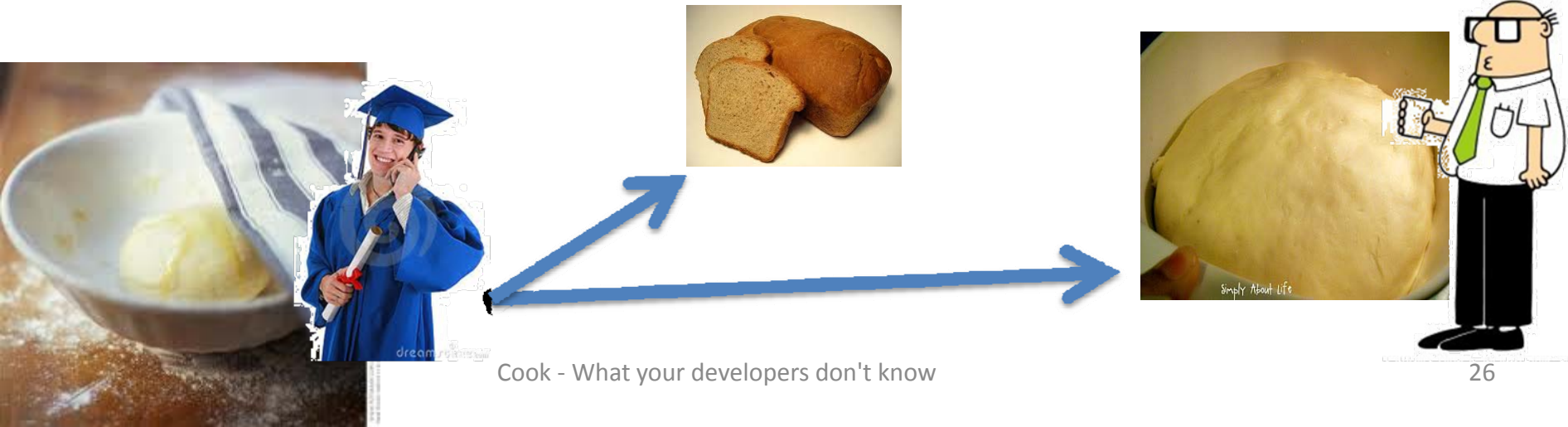
Also....

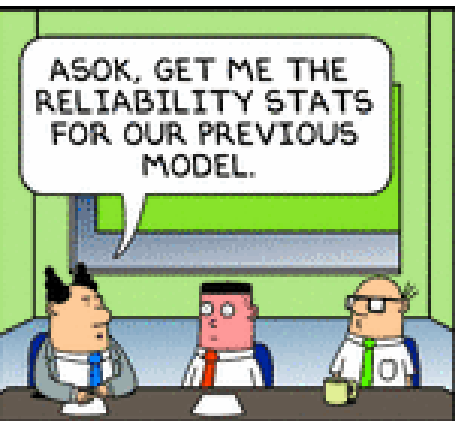
- Remember to leverage the skills of the new hires. Most have a new skill that older, more “experienced” developers might not have. It can be a two-way street.



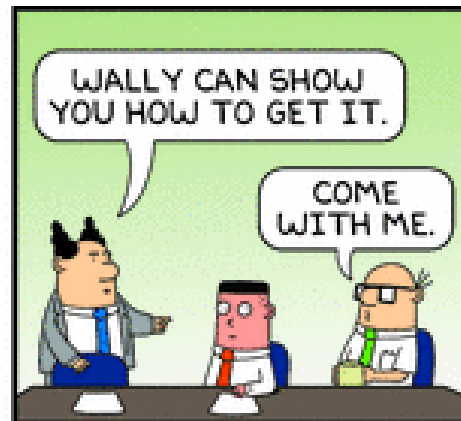
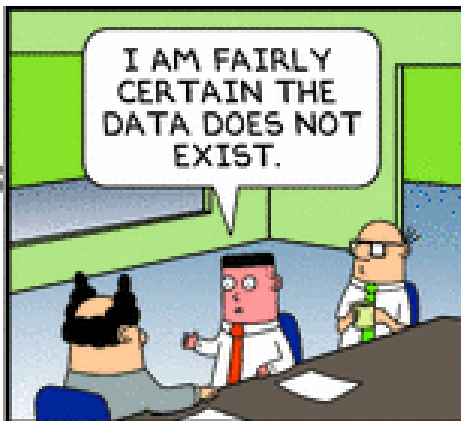
Final thoughts

- Think of new hires as “raw dough”. All of the ingredients are already incorporated. All it needs is a warm, supportive environment.
- If there is not enough “heat”, it stays dough. On the other hand, if there is too much “heat”, it rises too quickly and you get poor results

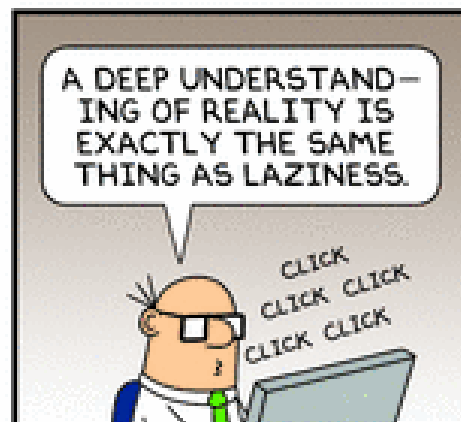
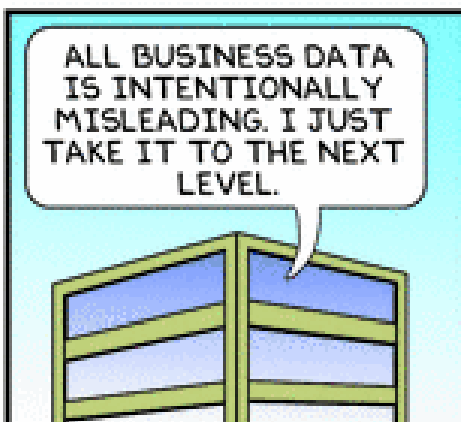
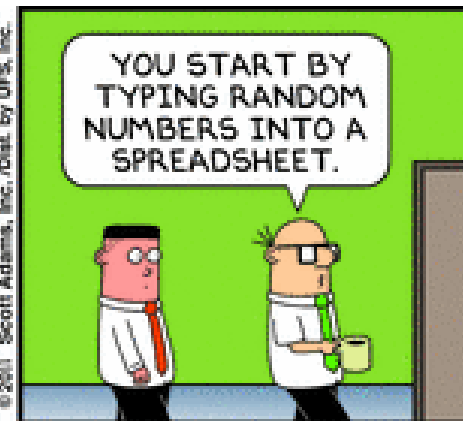




DilbertCartoons@gmail.com



© 2011 Scott Adams, Inc. Dist. by UFS, Inc.



www.dilbert.com
3/28/11



Questions or comments??

Dr. David A. Cook
Department of Computer Science
Stephen F. Austin State University
Nacogdoches, TX 75962

cookda@sfasu.edu



Note: Please consider this talk the conclusion of a presentation I did last year, “What your developers don’t know”. Email me for a .pdf if you would like a copy