What your developers DON'T know!

….and what you can do about it!

Dr. David A. Cook
Associate Professor, Department of Computer Science
College of Business
Stephen F. Austin State University
Nacogdoches, TX

Please note:

- Stephen F. Austin is NOT is Austin, Texas
  (it is in the NE corner of Texas, in the city of Nacogdoches – the oldest city in Texas)

Why this talk

- As part of the Software Engineering course I teach, I work with students on how to "succeed in the real world"

- The content of this talk comes from feedback from our Computer Science Advisory Board

They Don't Know Traditional Languages

- Most graduates have a "working" knowledge of (AT MOST) two languages
- Reason – this is all that ABET requires of CS and Comp Engineering Programs
- Typical choices
  - Java
  - C++
  - Some scripting (but usually only "cool" scripting languages)

Language Issue Side Effects

- Unable to directly work in supporting teams that work in other languages
- Unable to easily work on older projects
- Unable to do maintenance!!!
- When they say they know a dozen languages, they really have minimal experience in all but two.

They don’t know how to do Maintenance

- Legacy code and maintaining legacy code is not usually taught (other that as an abstract topic) in a curriculum.

- Developers do not know how to effectively debug, nor to plan for running a piece of code more than one time.

- They are unaware how to write code that efficiently utilizes *time* or *space.*

They haven't worked with Legacy Systems

- Which typically includes any REAL experience with any OS other that Windows
- Limited Un*x experience
- A few Mac programmers
- Little, if any, experience with older legacy OS

They are not skilled in Teamwork

- Most school projects are individual
- Most students (the good ones) really don’t work in team projects, they do the work themselves!
- They lack the ability to adequately plan and delegate work

Because they can't Plan

- Typically, start a project at the *calculated* last minute, based on intuitive performance metrics

- They lack the ability to breakdown large tasks
  - Earned Value
  - Function Points
  - LOC / methods

They often lack interpersonal skills

- Question: how can you identify an extroverted engineer?

They often lack interpersonal skills

- Question: how can you identify an extroverted engineer?

- Answer: they look at YOUR shoes when they talk to you!

Which means they might not be good at sales pitches or demonstrations

- Reasons
  - Lack of experience
  - Lack of self confidence
- Side effects
  - Inability to coherently ask for (and justify) funding
  - Inability to fully understand funding issues

And they lack Leadership Skills

- Can’t run teams
- Often don’t understand how to motivate (and not alienate) others

Other potential problems

- Dress for success
- Eat for success
- Problems with authority figures
- “Cute but charming” eccentricities

So they are not good at Presentations

- PowerPoint skills are not effectively taught in school – it's mostly learned
- Graduates cannot often coherently make an effective presentation
- They often rely on too much glamour and glitz

- Many of them are ineffective at estimating the time it takes to present slides

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”

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

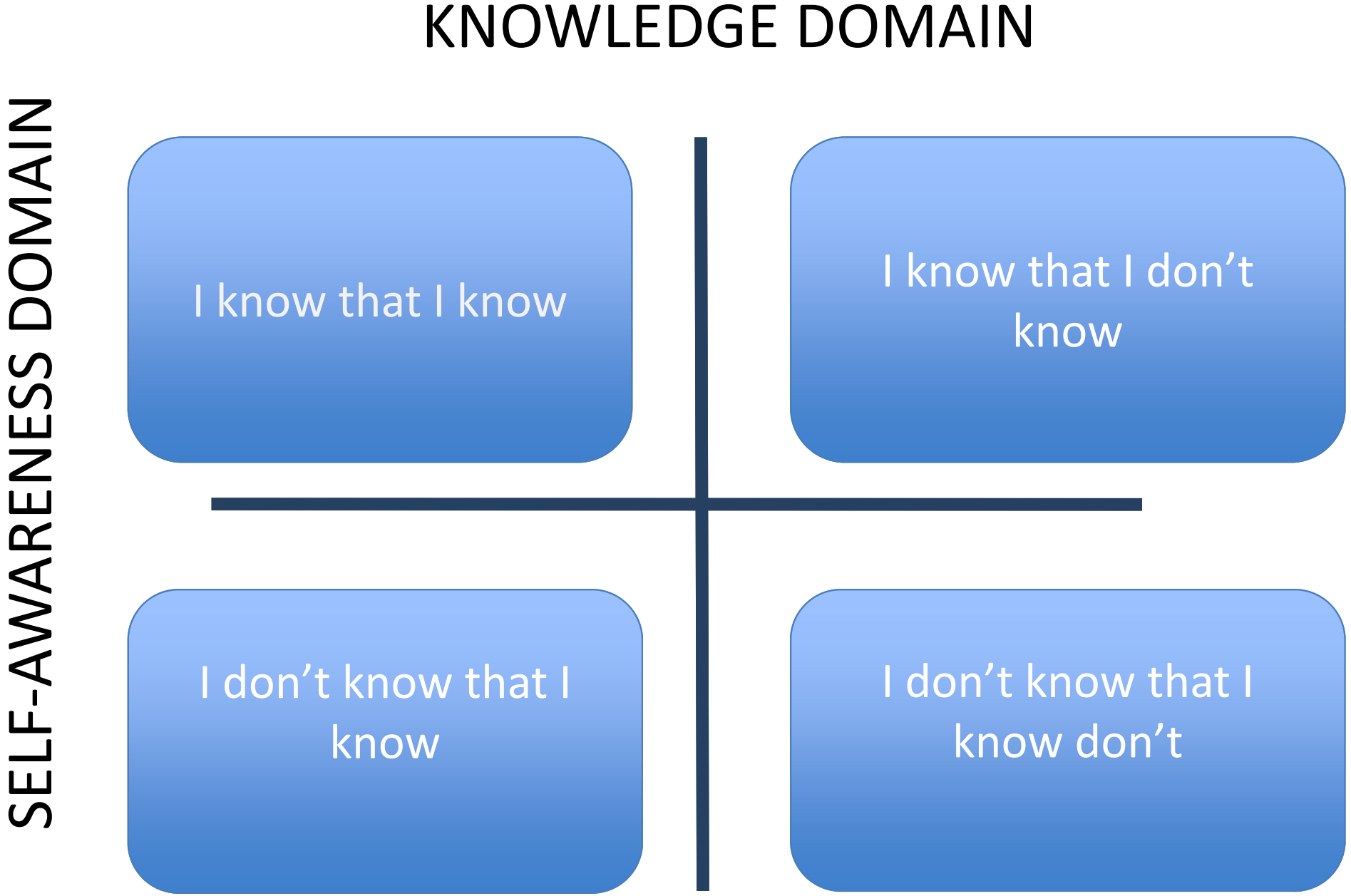
- Classes or workshops should not be limited to those who request them – they should be presented as “skill enhancers” that all “keep new developers sharp”
- New hires are often timid about asking for training

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

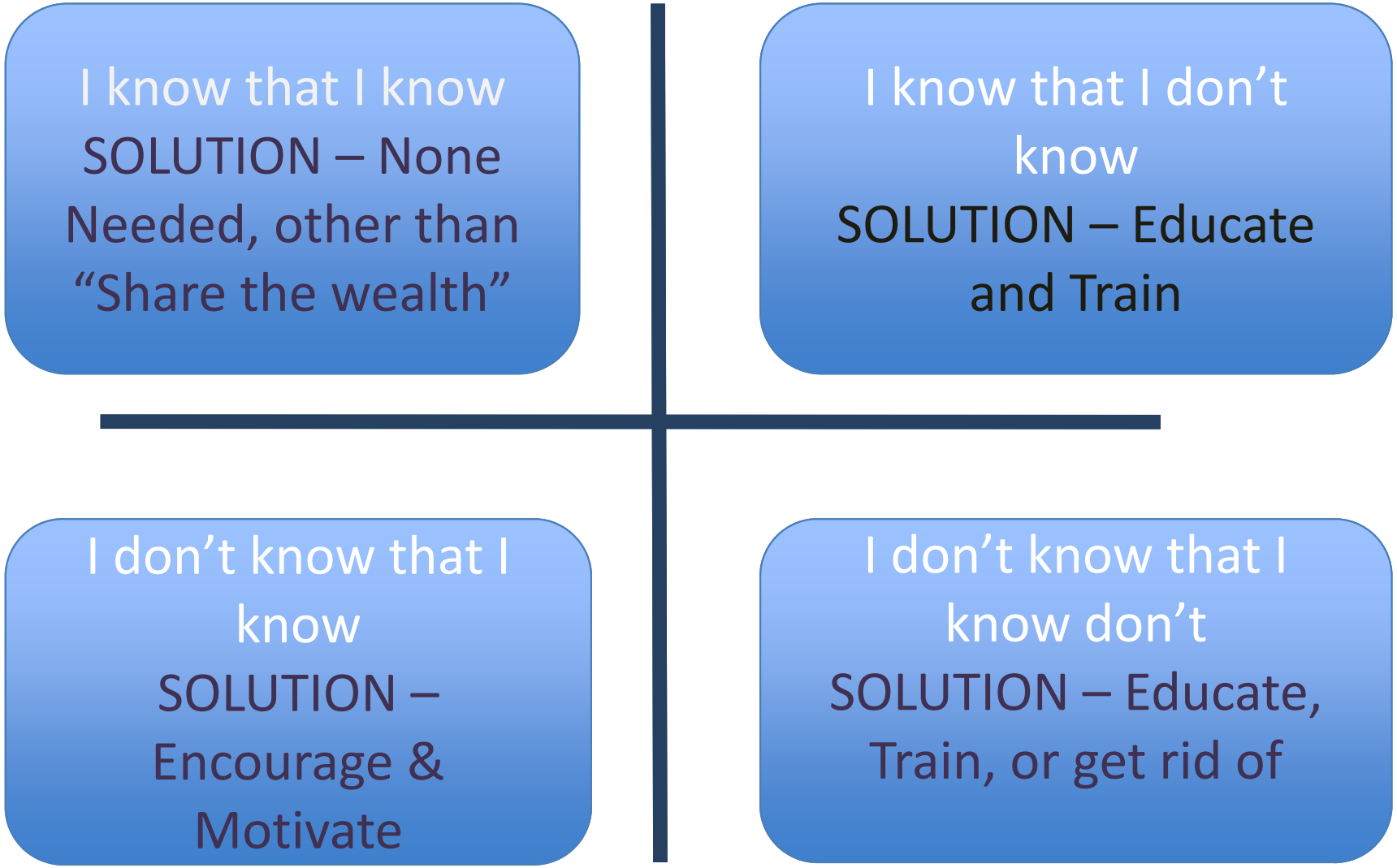
Why required skill classes?

- 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


KNOWLEDGE DOMAIN
SELF-AWARENESS DOMAIN
I know that I know
I know that I don’t know
I don’t know that I know
I don’t know that I know don’t

KNOWLEDGE DOMAIN

SELF-AWARENESS DOMAIN



Self-confidence

Interpersonal skill enhancements

- Mentors
- Constructive and non-supervisory feedback
- Opportunities to work with diverse group of developers

Final thoughts

- I used to refer to “new graduate hires” as eggs – you have to incubate them to hatch

- Now – I refer to them as “raw dough”. All of the ingredients are already incorporated. All it needs is a conducive 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