About me

• 37 years industry experience
  • software engineering tools and guidance (RUP)
  • applications
  • compilers and language-based tools
  • consulting

• 10 years academic
  • professor of practice
  • software development (engineering?)
  • compilers and programming languages
  • testing
Yes, there is a crisis

And it’s here to stay
The software crisis

• Increasing the number of people will not solve the problem
• Technology may help
  • Generation
  • Domain-specific languages
• The more people who can produce software means there will be a lot more crappy software out there
  • And a lot more malware
  • And you’re going to have to throw more people at cyber-security which will worsen the crisis
• Good people will, however, create good software
  • Decide what is key to your company and put your limited supply of good people on those projects
Why don’t new grad hires work out?

• You don’t know what you want
  • “We want smart people”
  • “We want only grads from the top schools”
  • …

• You haven’t told them who you are

• You haven’t asked the right questions
  • “How do you declare a function that takes a pointer to a function with an int argument and returns a pointer to a function that takes a pointer to a character, in C?”

• Mismatched expectations
  • They don’t know what you want
  • You don’t know what they know
    • And you ask the wrong questions
CS or MIS/IT or SE

• Computer Science
  • Core disciplines (OS, Languages, Algorithms, …)
  • Cutting edge technologies
  • Should have some software engineering

• MIS and IT
  • Application oriented
  • IT infrastructure, data center management
  • Business focus (often part of the Mgmt. department)

• Software Engineering
  • Ideally, a mix of business, IT, and CS
  • Project oriented

• Don’t ignore others
  • Philosophy, music, linguistics, math
Three types I’ve experienced

Just do it my way. I know how to do it better than you.

Why is that interesting? Why should I work on this?

I don’t know exactly how, but let’s get started.
Who do you want to hire?

Programmer

Software Engineer

Computer Scientist

Technologist
Who do you want to hire?

Programmer

Software Engineer

Team member

Computer Scientist

Technologist
The education spectrum

• High school
• Vocational school beyond high school
• Two-year college
• Four-year college
  • Broad education, no specialization (e.g., technical institute)
  • Engineering, technical (applied)
  • Software engineering program
  • Engineering, technical (research focus)
• MOOC educated
ALM knowledge in the classroom

ALM

- Requirements
  - Who provides them
  - How to manage them
- Analysis & Design
  - Requirement refinement
  - Patterns, scalability, etc.
- Testing
- SCM
- Change management
- Build and release
- Defect tracking
- ...

Where do students learn ALM?

- Requirements
  - Software engineering class
  - Projects
- Analysis & Design
  - Standard CS curriculum
  - Special design class
  - Projects
- Testing
  - Software engineering class
  - Projects
- Other ALM areas
  - Software engineering curriculum
  - Projects
Software engineering courses I’ve known

- Survey
  - Almost like taking a history course
- Learn about 50 million line, 5 year, 500 person project, CMM, CMMI, UML (for the sake of UML), UP, Agile
- Well-defined projects (if any) with little change
- Well-defined roles
- Little interpersonal skill emphasis
- Little to do with reality
What can you expect?

<table>
<thead>
<tr>
<th>Area</th>
<th>Below the bar</th>
<th>At the bar</th>
<th>Above the bar</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programming</td>
<td>One language</td>
<td>2-4 languages</td>
<td>Language is not a problem</td>
</tr>
<tr>
<td>Requirements</td>
<td>Given by professor</td>
<td>Given by “customer” but very well defined</td>
<td>Come from several sources, several formats, different types</td>
</tr>
<tr>
<td>Requirements management</td>
<td>What?</td>
<td>Kept a list</td>
<td>Used one or more tools to organize, plan, and schedule</td>
</tr>
<tr>
<td>Analysis</td>
<td>What?</td>
<td>Forced UML or some other format, simple grouping into features</td>
<td>Value-based, dynamic, focus on communicating and reasoning about the problem, iterative approach</td>
</tr>
</tbody>
</table>
## What can you expect?

<table>
<thead>
<tr>
<th>Area</th>
<th>Below the bar</th>
<th>At the bar</th>
<th>Above the bar</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design</td>
<td>Given by professor</td>
<td>Use interfaces, encapsulate, …</td>
<td>Work from requirements and trace back to them, evolving design, architectural patterns, performance, …</td>
</tr>
<tr>
<td>Testing</td>
<td>What? Not my problem.</td>
<td>Unit tests, acceptance tests</td>
<td>Automated testing (with appropriate tools), TDD, BDD, acceptance tests, everyone’s responsibility</td>
</tr>
<tr>
<td>Testing tools</td>
<td>No tools</td>
<td>xUnit and friends</td>
<td>Mocking tools, test automation at all levels (Fitnesse, AgitarOne, etc.), GUI testing, able to build tools when needed static analysis</td>
</tr>
</tbody>
</table>
What can you expect?

<table>
<thead>
<tr>
<th>Area</th>
<th>Below the bar</th>
<th>At the bar</th>
<th>Above the bar</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version control</td>
<td>None, copy files</td>
<td>SVN or git or one other. No branching, etc.</td>
<td>Different tools, different models, different ways to use branches, tags, etc.</td>
</tr>
<tr>
<td>SCM</td>
<td>What?</td>
<td>One branch, one release</td>
<td>Multiple configurations for different uses. How to organize the assets.</td>
</tr>
<tr>
<td>Change and defect management</td>
<td>Not needed</td>
<td>Kept a list, email, …</td>
<td>Used one or more defect trackers (integrated with requirements and other tools?)</td>
</tr>
<tr>
<td>Documentation</td>
<td>What?</td>
<td>Code comments, some user</td>
<td>Multi-level, multi-purpose, what is enough, tools to generate, synchronize, etc.</td>
</tr>
</tbody>
</table>
## What can you expect?

<table>
<thead>
<tr>
<th>Area</th>
<th>Below the bar</th>
<th>At the bar</th>
<th>Above the bar</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build and release</td>
<td>Zip and submit</td>
<td>Organize, package, deploy</td>
<td>Set up distribution channels:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• continuous integration</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• continuous deployment</td>
</tr>
<tr>
<td>Process</td>
<td>What?</td>
<td>We use (UP, SCRUM, XP, …)</td>
<td>Tailors the process to the project. Reflection and revision. Common sense process.</td>
</tr>
<tr>
<td>Projects</td>
<td>0 to 2 small team, small scope</td>
<td>Few small with well defined capstone project, usually 1-5 person teams</td>
<td>Project-based curriculum. Projects throughout. All sizes, all levels of technical and interpersonal details</td>
</tr>
</tbody>
</table>
Even if someone is above the bar

- It’s not about tools
  - They can learn tools
- It’s not about process
  - You can teach them how your organization works
- It’s not about coding
  - But this is a requirement

Engineering is easy
People are hard

Brian Fitzpatrick, Ben Collins-Sussman, Team Geek
Companies Compete
Teams Collaborate
General approach to understanding the student

- What did you know when you entered college
- In college
  - What did you learn in class (30% <= x <= 50%)
  - What did you learn by yourself (15% <= x <= 40%)
    - How
  - What did you learn from others (25% <= x <= 60%)
    - How
- Avoid using your colloquialisms and idioms
  - Especially true if you do government contracts
  - Even terms like Agile, ALM, etc. There’s a difference between knowing a definition and actually “getting” it
My key questions to find the right person

• What was the biggest mistake you’ve ever made on a project?
  • What did you learn from it?

• Who was the most difficult person you ever had to work with?
  • Why?
  • How did you manage the situation?
  • What did you learn from the experience?

• What is the largest project you worked on?
  • People, Code, Other measurement
  • What was your major contribution?

• Any open source project experience?
Connecting

- Career fairs
  - Full-time: start in Sept. or as early as possible
  - Summer hire: no later than the beginning of the calendar year
  - Send engineers to talk to engineers
  - Usually, don’t contact specific faculty members
- Give tech talks
  - Connect with student ACM and IEEE-CS chapters
  - Call the department
- Sponsor projects
  - Senior projects
  - Class projects
THANK YOU
ANY QUESTIONS?
Give Feedback on the Sessions

1. Sign In: www.eclipsecon.org

2. Select Session Evaluate

3. Vote: +1  0  -1