Software Engineering
Lecture 3
Engineering Practices

Gregory S. DeLozier, Ph.D.
Kent State University
Jan 25, 2017

Best Practices

"A **best practice** is a method or technique that has consistently shown results superior to those achieved with other means, and that is used as a benchmark."

- wikipedia

Best Practices

- Behaviors and strategies that are known to work.
  - Lots of different areas: PMBOK, SWEBOK(*), etc.

- Identifying these comes from
  - Experience
  - Experimentation

- They can be restraining or liberating, depending
  - Allows cooperation
  - Can prevent creativity
  - One best practice: reconsider your practices, often

Barely Sufficient Practices

- "Barely Sufficient Software Engineering"
- What constitutes a minimum set of best practices?

    - Following them should lead to success
    - Not following them exposes common problems
    - Available across a wide range of contexts.
    - Adaptable to a large range of situations.

10 Practices to Improve...

- 10 practices identified.
  - Well, actually 11

- From Heroux and Willinbreng

- The PDF is in the /papers directory on GitHub

Manage source (the basics)

- GitHub is great
- Put it somewhere safe

Use issue-tracking software

GitHub has this.
Use for requirements, features, and bugs
Demo time...

Manage source (branches and tags)

Learn how to use branches for development
Learn how to merge correctly
Make tags to identify critical points

Use mailing lists (or something)

- I like Slack a lot
- There are other options, of course

Checklists for repeated processes

- Checklists go in your document repository
- Automation is a particularly nice checklist format

Source-centric documentation

- Use documents that connect (or are in) source code
- Make them simple to keep up
- Extract and publish using automation

- REST APIs are a good example
- Commands and options are another

Use configuration management

- Tools to set up your environment
- Tools to set up your deployment

- We will cover this in more detail
- Don't set things up by hand. Please.

Write tests first, run often

- Test Driven Development
- Behavior Driven Development
- Frankly, any kind of repeatable testing is helpful

Program the tough stuff together

- Pair Programming
  - Programming is about creating a mental model
  - Shares the complexity of the cognitive model
  - Frequently one person can 'navigate'
  - Use when it's needed

- Code Reviews
  - And security reviews, etc…
  - Anything challenging should be reviewed
  - Reviews spread risk

Use a formal release process

- Many failures happen during release
- Accidental release may not match business expectations
- Frequently there are legal issues

Perform continuous improvement

- Continually examine performance
- In particular analyze failures
- Allow improvements to guide practice evolution

Reading

- https://en.wikipedia.org/wiki/Best_practice
- http://www.sandia.gov/~maherou/docs/BarelySufficientSoftwareEngineering.pdf