A Review of **Modern Software Engineering**

Ed Snodgrass and Brandon McAllister

bjm.sh/mse

Learning Clubs

Benefits

- Learn better as a group
- Build bonds with coworkers
- Have common language and background

How They Work

1. Champion finds others to join (6-10 people)
2. Champion creates a curriculum
3. Group settles on a time/frequency to meet
4. Group meets and has discussions
5. Follow-up with a retro

Some De

The Scientific Method

“Humanity’s best approach to learning is science…”

Wikipedia says:

- Characterize
- Hypothesize
- Predict
- Experiment

What is Software Engineering?

Dave's working definition "that underpins the ideas in this book"

Software engineering is the application of an empirical, scientific approach to finding efficient, economic solutions to practical problems in software.

Design Engineering, Not Production Engineering

- We aren’t bound by the slowness of creating physical objects
- Models (or computer simulations) are used in most engineering disciplines, which are “just an approximation of the real thing.”
- “The models that we [software developers] create as software, our computer simulations of a problem, are our product.”

Engineering

Dave’s working definition

“Engineering is the application of an empirical, scientific approach to finding efficient, economic solutions to practical problems.”

Engineering != Code

- “...engineering is about applying scientific rationalism to solving problems.”
- “It is the processes, tools, and techniques. It is the ideas, philosophy, and approach that together make up an engineering discipline”
- When the Dave refers to engineering, he means “**everything that it takes to make software.** Process, tools, culture – all are part of the whole.”

Engineering vs. Craft

- “Craft is very good at creating ‘one-off’ items.”
- “... at the root, craft-based production is fundamentally low-quality. A human being, however talented, is not as accurate as a machine.”
- Precision and Scalability
- Gunsmiths
- Our opinion: Today’s gunsmiths are the folks that have a deep understanding of some API and write clever code that is not… palatable for most of us. Whereas following better practices would level up everybody.

There’s a lot of great content in this book and we don’t want to type it all out into slides so we’re moving on.

The Importance of Measurement

Stability

- Change Failure Rate
- Recover Failure Time

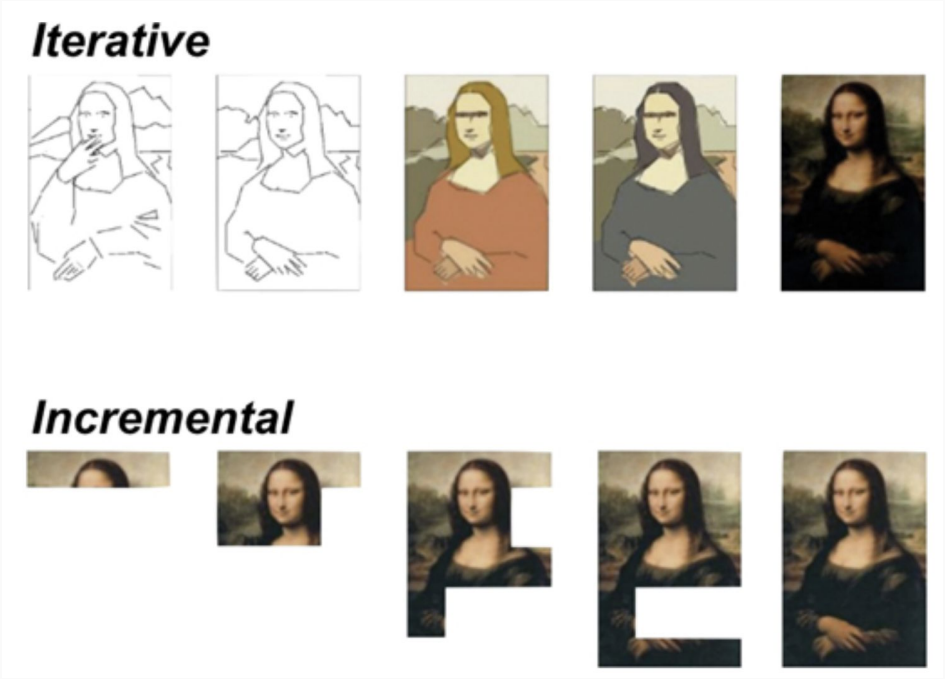
Throughput

- Lead Time
- Frequency

How to Software Engineer

Become Experts at Learning

- Iteration
- Incrementalism
- Feedback
- Empiricism
- Experimentation



Become Experts at Managing Complexity

- Modularity
- Cohesion
- Appropriate Coupling
- Separation of Concerns
- Information Hiding / Abstraction

Tools

- Testability
- Deployability
- Speed of Feedback
- Controlling the Variables
- Continuous Delivery

Summary

It would be impossible for Dave Farley to tell every individual developer how to be a better Engineer, in one book. We do hear his thoughts on how to approach software engineering in a manner that will create better software.

TDD is the answer.

Review

This book is a must read for members of software engineering teams. Ranking up there with Clean Code and Pragmatic Programmer.

If our “software engineering” practices don’t allow us to build better software faster, then they aren’t really engineering, and we should change them!

bjm.sh/mse