Evolving tools in an Agile world

Adam Bergstein

11/3/2015 – Juniata College

Outline

- Processes
- Tools
- Review of Frameworks

Processes

Conventional wisdom and modern convention

Waterfall

- You give me requirements
- I tell you how long, how much
- I execute them

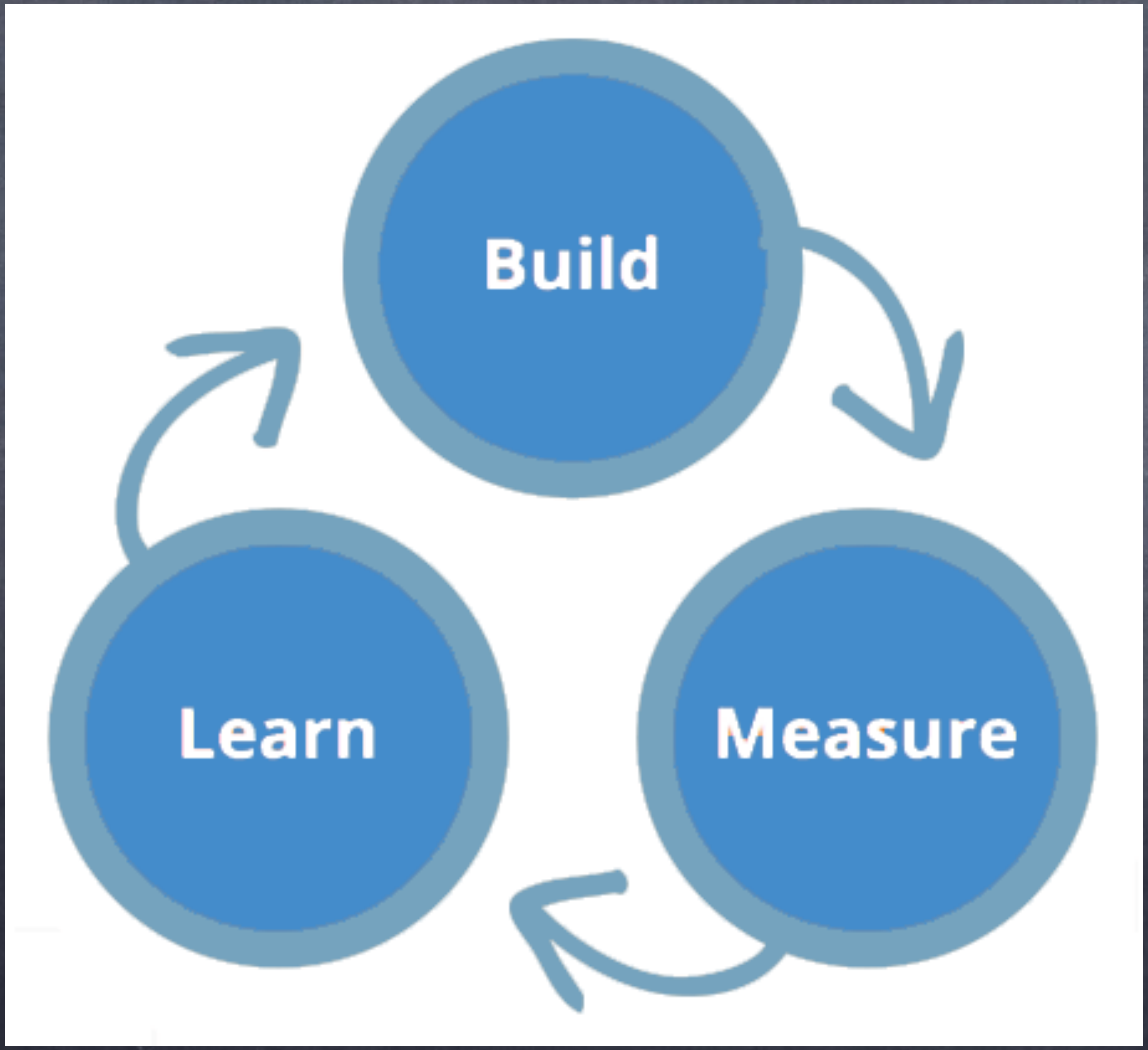
Challenges

- You translate their requirements into a technical product
- Is it right?
- There is no such thing as complete planning

Common sense

- Too much risk in completing an entire build
- Developers want more checkpoints to ensure the build is correct
- Clients want the ability to prioritize their needs as they learn them

Continuous learning



Agile

- Product owners involved in evolution of software
- Backlogs and prioritization
- Scrum every day to check in, identify next steps and blockers
- Sprints to time box work and review
- Retrospectives to reflect and learn
- Estimations and milestones
- Prototyping and rapid change

Tools

Applying lessons learned

Problems

- Custom development does not easily support rapid change
- A lack of consistency can introduce problems for a distributed team
- People should not be solving the same problems repeatedly and inconsistently

Usual suspects

- There are common tools that are process-agnostic
- Ticketing systems like JIRA to organize, estimate, and prioritize work
- Code repositories like Git or Subversion to maintain code history, release tags, etc

Tool categories

- Backend - PHP, Database, Server
- Frontend - CSS, JavaScript/jQuery
- DevOps - Development best practices, like code reviews, automated testing, etc
- Continuous integration - regular review of opportunities for improvement (technical or process), like automated deployments, ticket estimations/reporting

Backend tools

- Code frameworks are the key - many are Open Source
- Development is done in a standard and consistent way
- Assemble tools and configure them
- Limits custom development, leverage tools in framework
- Examples: Symphony (PHP), .NET (Microsoft), Ruby on Rails ()

Frontend tools

- Again… frameworks (are you seeing a pattern here?)
- Does not touch the backend, often loaded via the end user's browser
- Areas: behavioral (slideshow), visual styling (colors, fonts, layout), responsive (multi-device), and micro services (small and rapidly changing parts of a page)
- Examples: jQuery (open source JavaScript), NodeJS (open source, lightweight communication) AngularJS (Google), Bootstrap (Twitter, CSS/responsive tools)

DevOps tools

- Enforcement of best practices
- This usually changes by the adopted frameworks
- Automated code analysis tools, automated set up of environments/sandboxes, automated tests
- Drupal examples: Ansible for environment recipes, PHPCS for code review, SimpleTest for writing automated tests

Continuous integration tools

- This is more about identifying opportunities for efficiencies and ways to automate
- "Gee it would be nice if I could push a button and deploy my code"
- "Shucks, it would be nice if I could make sure my code does not break other functionality"
- Drupal examples: Drush (general commands to interact with a Drupal site), Jenkins (a server that can run remote commands), TravisCI (a tool that can run commands when mediating code requests)

Review of Frameworks

Activity

- Split into groups, you will get a topic
- Evaluate the topic and describe problems solved for 10 minutes, prepare notes to share with class
- Topics:
  - Agile (process)
  - Symfony (backend)
  - AngularJS and NodeJS (frontend)

Agile

- Incremental and iterative development
- Emerging preferences: scrum, extreme programming
- LEAN - elimination of work that adds doesn't add value
- KANBAN - matches work with capacity

Symfony

- consistency across projects
- ten years of experience, maturity
- large open source community for support
- extremely thorough documentation and cookbook of tutorials
- not confining, can use selectively
- security claims

AngularJS/NodeJS

- separates backend from frontend
- angularJS - javascript based, extends HTML, manipulates content in real time
- nodeJS - javascript, "backend" of the front-end ;)
  - callbacks for front-end microservices, very performance
  - can replace what PHP does, non blocking IO
  - lots of libraries

Questions?