Funability

Chad Michel

Why did we get into software development?

- Rapid return on our effort
- Work on tough problems
- Build tools that people use
- Enriching our lives
- Building something innovative
- Impacting people's lives
- Saving money / creating wealth
- Automating complex activities

Problem is…

… the journey many of us are on to seek fulfillment of those goals has required enduring a lot of “pain” along the way.

Cool office spaces is not enough

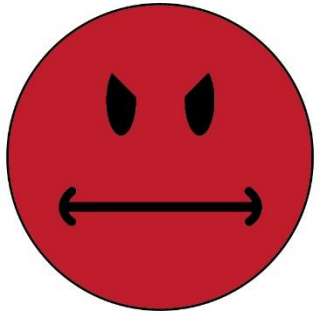
The pain…

- Swim through 5 layers of inheritance
- 12 hour product releases
- 6 weeks of stabilization
- Estimates <> reality
- Dreading project status reviews
- Hours wading through code to determine how something works
- Hope and prayers during releases

- Edge of seat waiting for support calls about system down
- Looking for new projects to avoid maintaining ugly code
- Test environment cumbersome and shared
- Estimates driven by deadlines vs reality
- Silos of design philosophy throughout the system
- …

What we want…

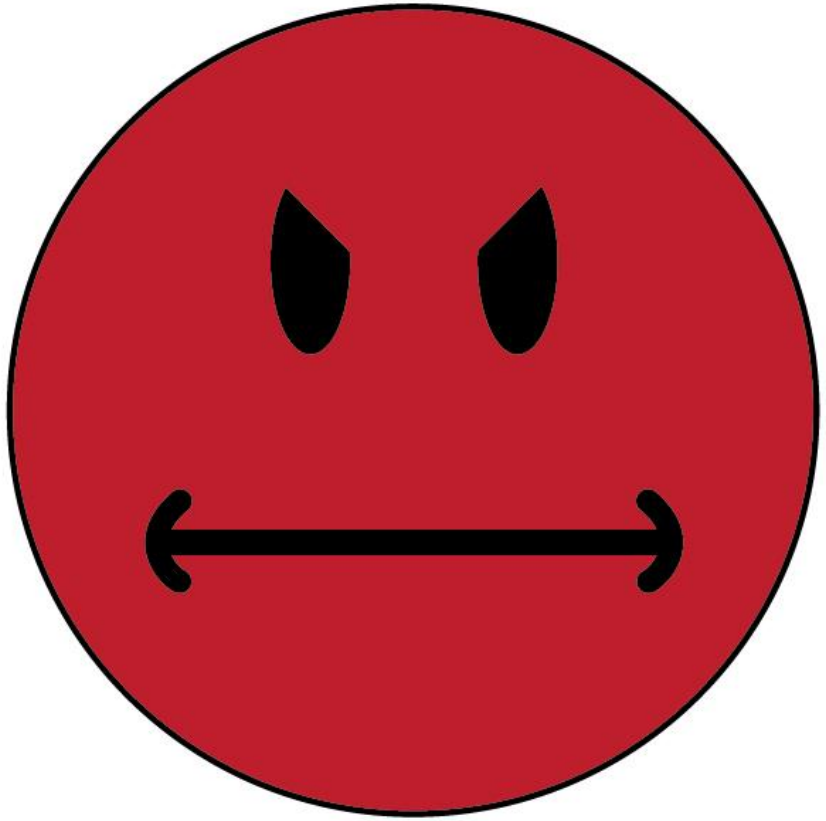
What we enjoy



What we loathe

Reality for most of us…

What we enjoy



What we loathe

How can we turn this around?

We have put a lot of emphasis on the ability of our software teams and development culture to achieve fun and personal fulfillment in our work…

…helping to realize the things that got us into software while minimizing the pain…

… we call this Funability

… the measure of how well our culture and process enable us to realize the motivations that got us into this business in the first place

What contributes to Funability?

- Frequent Delivery of Value to Customers
- Being Part of a Team
- Maintainability of the System
- Effective Management of Technical Debt
- Sound Software Design
- Consistent Quality of Product Releases
- Productivity and Efficiency of the Developers

What contributes to Funability?

- Frequent Delivery of Value to Customers
  - Visibility of progress by internal stakeholders
  - Regular and frequent releases to external customers
  - Minimizing long, drawn-out development efforts

What contributes to Funability?

- Being Part of a Team
  - Frequent interactions with teammates on project items
  - Ability to leverage pair programming when necessary
  - Esprit de Corps - a feeling of pride, fellowship, and common loyalty shared by the members of a particular group.
  - Mutual accountability amongst the team

What contributes to Funability?

- Maintainability of the System
  - Ability to efficiently read and understand the code throughout the system
  - Ability to effectively debug the system
  - Ability to understand the impact of a change on the entire system
  - Ability to avoid unintended behavior changes
  - Maximizing the useful life of a software system
  - Avoidance of silos in the System

What contributes to Funability?

- Effective Management of Technical Debt
  - Recognizing when choices will lead to technical debt
  - Ability to efficiently reduce technical debt as part of normal feature development
  - Leveraging tools to identify technical debt

What contributes to Funability?

- Consistent Quality of Product Releases
  - Stress-free release days
  - Automation of processes
  - No stabilization phases
  - Hot fixes as the exception, not the rule

What contributes to Funability?

- Productivity and Efficiency of the Developers
  - Creating a project management discipline that reduces the mental burden on developers and leads
  - Enabling designers and developers enough time to actually do some work

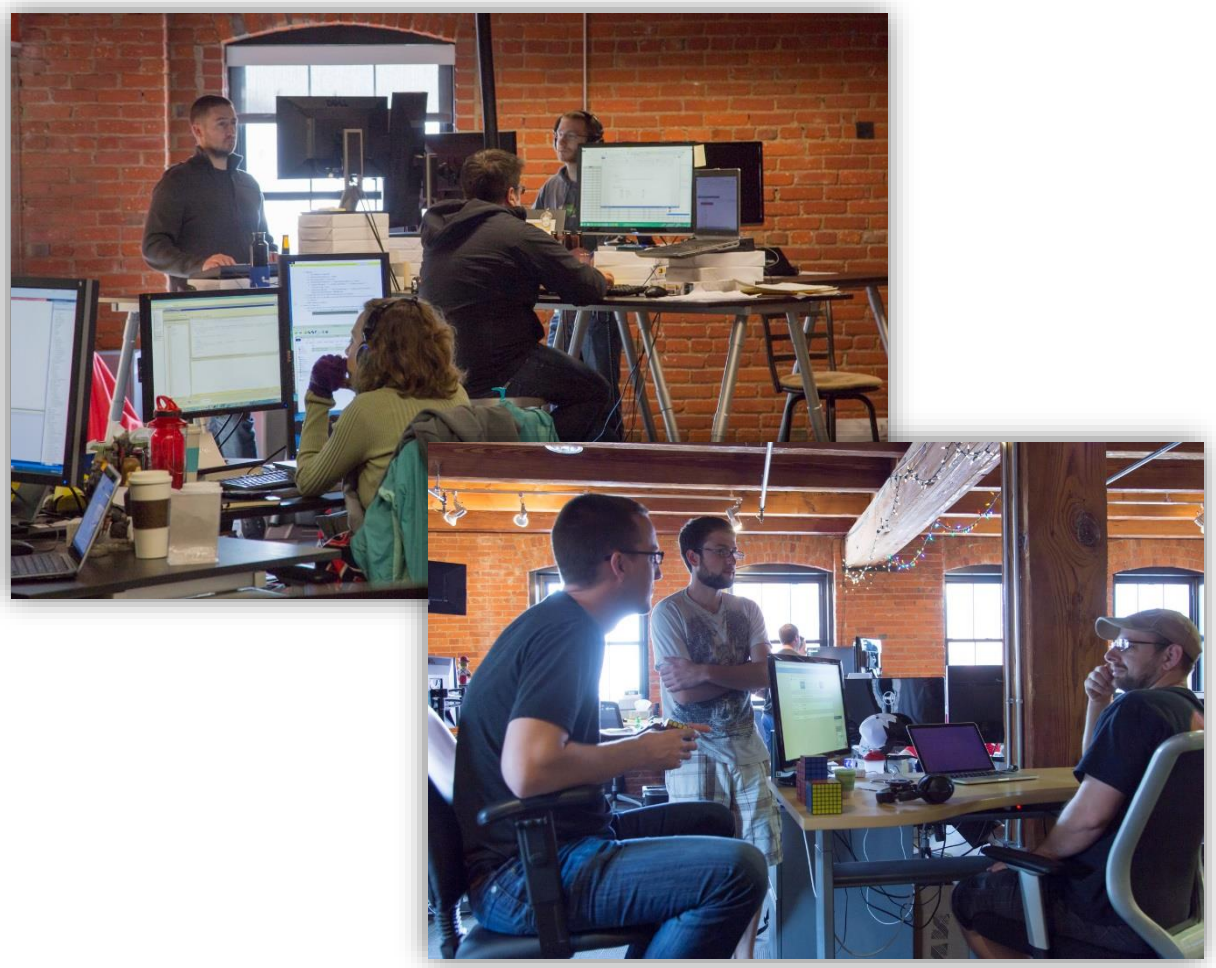
What contributes to Funability?

- Sound Software Design
  - Consistency of the conceptual design of the software
  - Consistent adherence to common design principles and criteria
  - Simplicity over complexity and cleverness
  - Disciplined and consistent approach to decomposition and estimation
  - This is the “bedrock” of our culture

Can’t a lot of this be managed through office layout and agile processes?

Cool spaces are important

- Creates a relaxed and collegial environment
- Enables collaboration
- Helps with recruiting
- Enables play
- Increases socialization



… and Agile methods are essential, but…

… they are not enough to effectively address and manage the ever-increasing essential complexity of the problems we are trying to solve with software

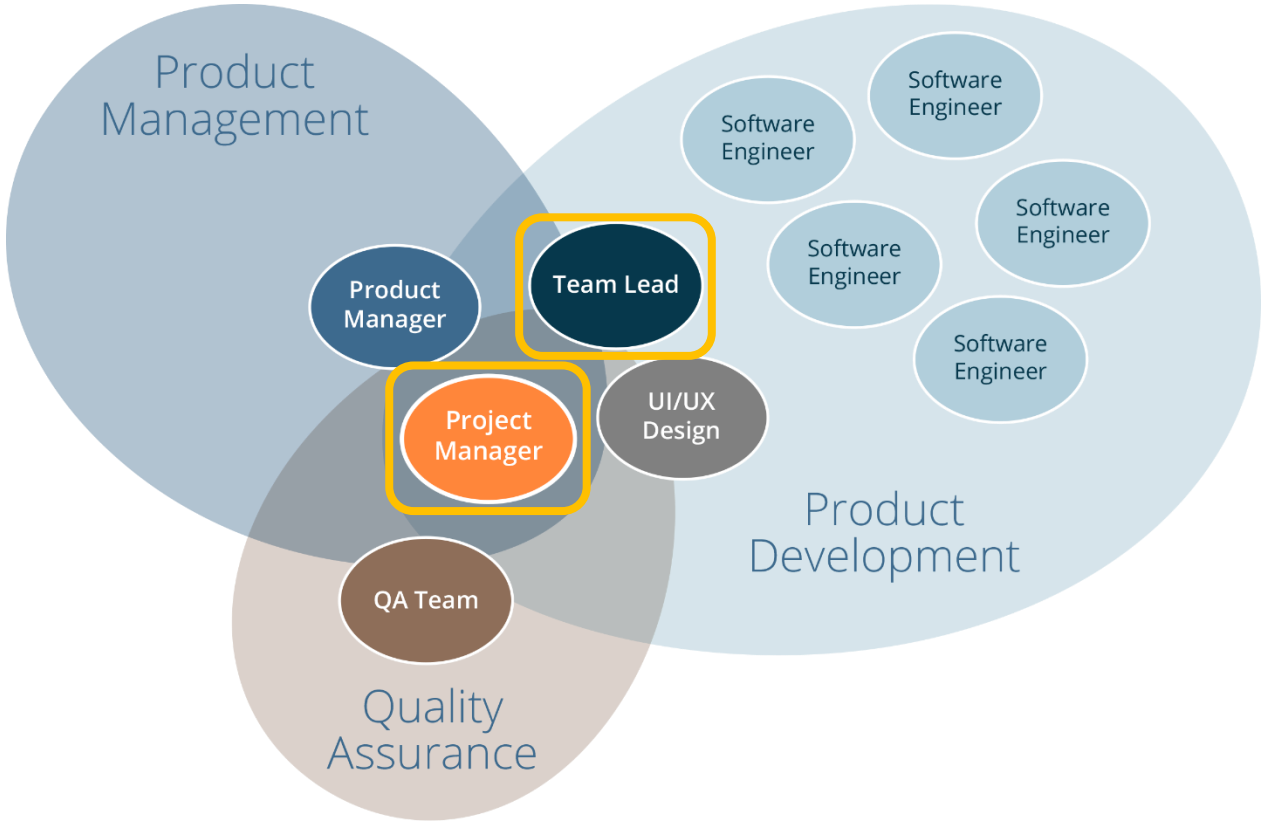
Bottom line: Agile is not a silver bullet

Strategies and techniques for increasing Funability

- Leverage Your Leadership Roles
- Have a Design
- Productivity Awareness
- Use a Layered Process Towards Quality

Leverage Your Leadership Roles

Key Team Roles



Establish a strong development lead role

- Lead Developer
  - Qualities of good programmer +…
  - Coaches Jr programmers
  - Works with programmers to design new features

- Lead Engineer
  - Qualities of good engineer +…
  - Coaches and mentors team on design principles and standards
  - Responsible for maintaining the conceptual design
  - Maintains big picture of product
  - Proactive communicator
  - Responsible for performance of product
  - Ensures engineers are testing their code
  - Performs code reviews

Establish a strong project management role

- Primary responsibility: Process facilitator
  - Ensure steps are followed
  - Maintain consistency
  - Keep a productive rhythm
  - Schedule/facilitate meetings
  - Keep meetings productive
  - Ensure proper task prioritization
- Central communication for project
- Tight coordination with lead engineer, UI/UX, QA and product manager
- Decision tracking and documentation
- Release plan development
- Task/action item tracking
- Project status monitoring / reporting
- Project health monitoring / reporting
- Information/decision coordination
- Retrospectives
- Management of external communications
- Lead daily standups
- Keeps sprint planning < 1 hour

Have a Design

Establish a consistent design identity

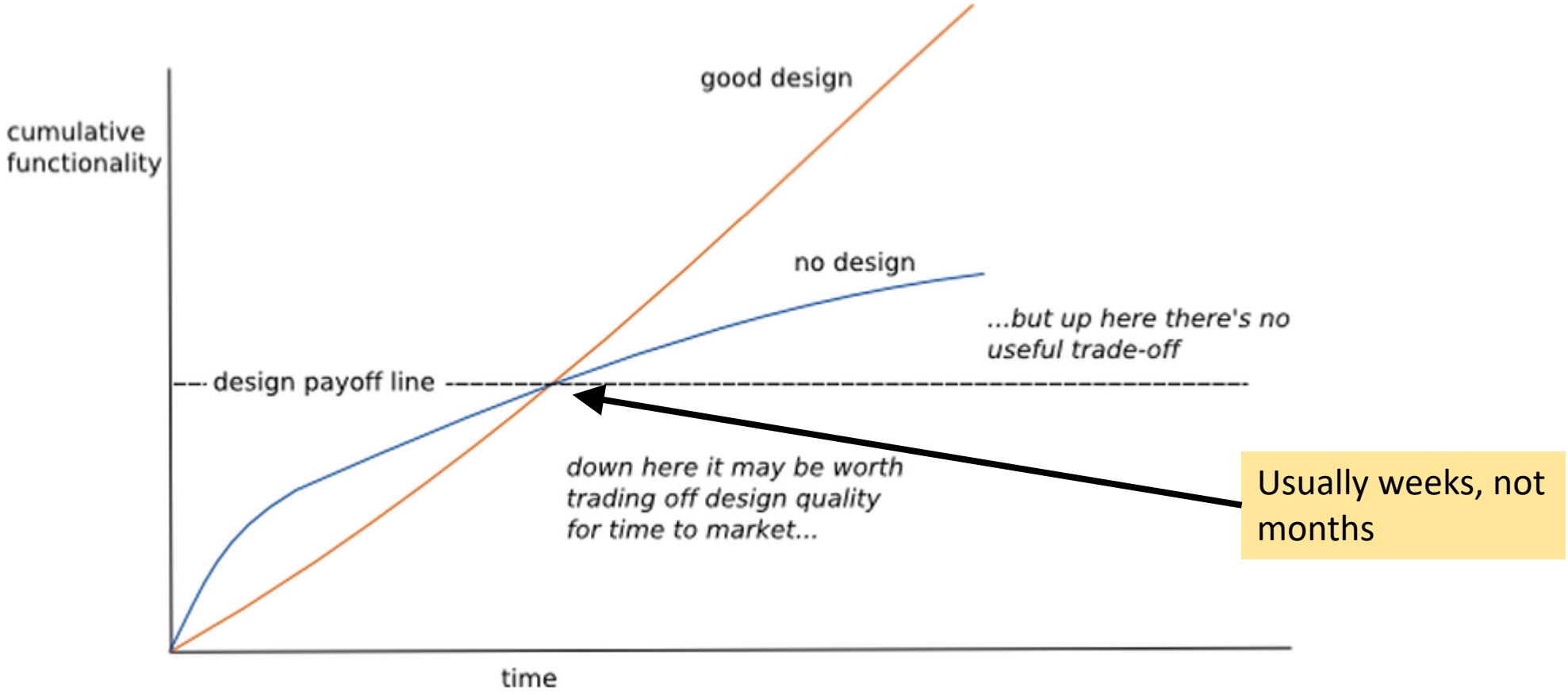
> "I will contend that Conceptual Integrity is the most important consideration in system design. It is better to have a system omit certain anomalous features and improvements, but to reflect one set of design ideas, than to have one that contains many good but independent and uncoordinated ideas."
>
> Fred Brooks (1975)

Establish a consistent design identity

- Avoid treating every new project as a unique design effort
  - When done right, the methodology for decomposing a system can (and should) be the same for every project
- System feels created by a single mind
- Ensures things such as testability remain high in all areas of system
- Enables movement of developers from one area of the system to another, and from project to project

- Examples: object-orientation, services, micro-services, IDesign

Design Stamina Hypothesis



Usually weeks, not months

http://www.martinfowler.com/bliki/DesignStaminaHypothesis.html

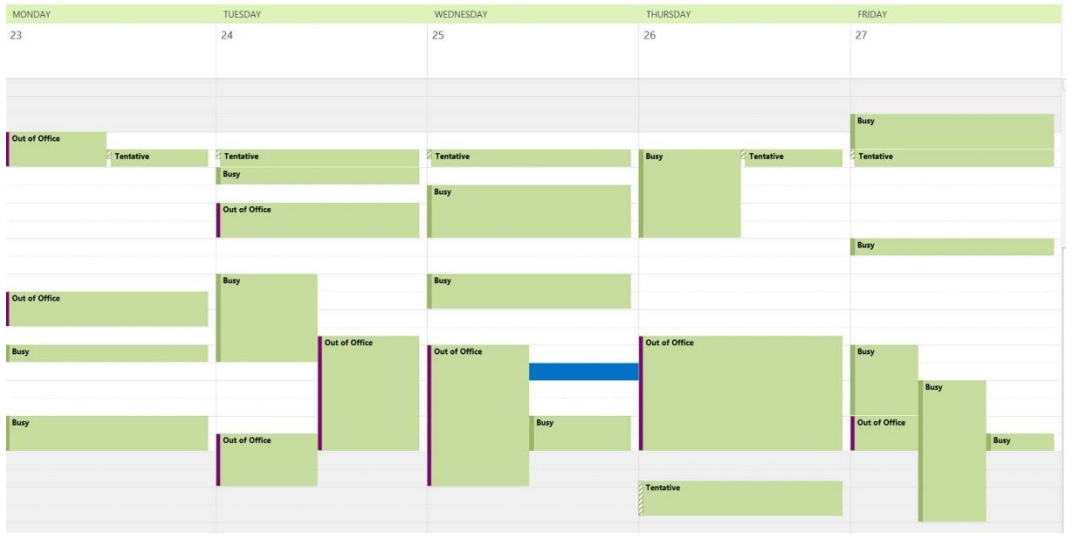
Practice test-driven design

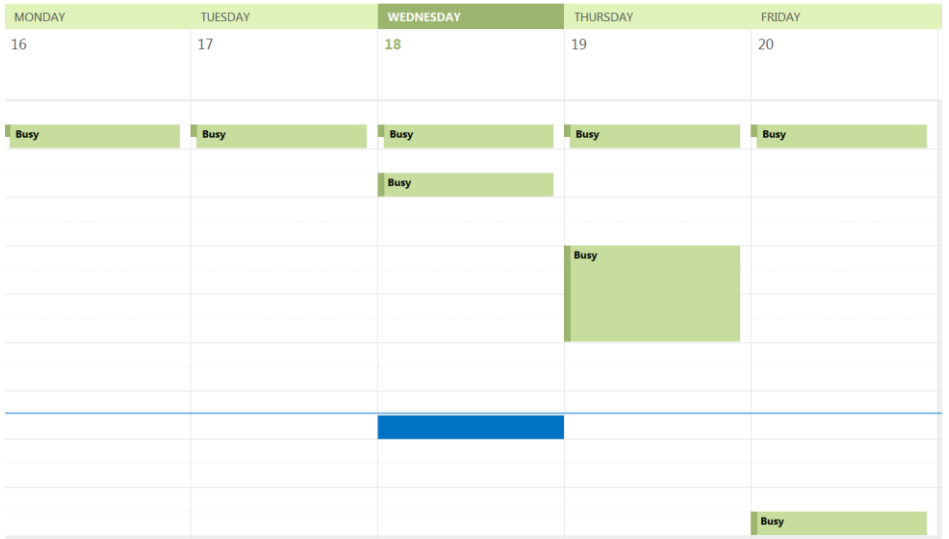
- Forces consumption awareness in your code because you create the first consumer
  - Gets you focused on the interface rather than the implementation
  - Tends to create interfaces that are conveniently callable
- Forces the software to be more testable which usually requires more decoupling from its surroundings
- Things that are difficult to test tend to be simplified in order to achieve testability
- Tests allow you to play “what if” games with broad changes to assess the impact to the design
- Makes the code more understandable/readable
  - Unit tests that describe how the developer intended the code to be consumed
  - Built in example code!

Productivity Awareness

Protect the schedules of your “makers”

- Manager vs Maker Schedule
  - Paul Graham (Y Combinator): http://www.paulgraham.com/makersschedule.html





Iterations

- Keep changes outside of iterations
- Allow devs to work to completion during an iteration

Require ability to do integration tests on the desktop

- Enables a developer to build and run end-to-end tests and validation on their own machines
  - Includes hosting of databases locally
  - Avoids collisions with other developers
  - Enables development in isolation
- Ability to do this must be a conscious design all along the way
  - Your system design choices can enable or prevent this

Use a Layered Process Towards Quality

Require code review of every pull request

- Code reviews single best way to improve quality
- Reduces the stabilization cycle
- Enables us to develop confidently
- Provides mechanism for coaching and mentoring
- Ensures the code is consistent with the software architecture/design

Use continuous integration with tests

- We want confidence that our systems work
- We don’t want that dreaded support call
- Enables us to sleep at night
- Avoids “broken window” syndrome

People

People

- People make it work

Summary

- No silver bullet to creating a dev culture with high Funability
- To truly change your development culture you need to go beyond cool spaces and agile/scrum and change the way the software is designed and constructed
  - Only an integrated view of these processes and best practices will get you where you want to be
- Constantly review practices and push for higher Funability
- Challenges remain
  - Still feeling a lot of pain in the web client tier and some mobile app development
  - How to actually measure funability in the workplace

Thanks!

"If builders built buildings the way programmers wrote programs, then the first woodpecker that came along would destroy civilization."

Gerald Weinberg

- cmichel@dontpaniclabs.com / @chadmichel
- http://blog.dontpaniclabs.com / @dontpaniclabs

nebraskaGlobal
/* Develop here */

DON'T PANIC LABS