Visual Studio LIVE! | Redmond
EXPERT SOLUTIONS FOR .NET DEVELOPERS

~~Develop~~ Deliver on Cadence, Release on Demand

Richard Hundhausen
Scrum / DevOps Trainer
Accentient, Inc.

Yesterday's Knowledge; Tomorrow's Code!

Visual Studio LIVE! 25 YEARS OF CODING INNOVATION

About Me

- ✓ From Boise, Idaho, USA
- ✓ President of Accentient
- ✓ Microsoft MVP (Dev Technologies)
- ✓ Professional Scrum Developer
- ✓ Professional Scrum Trainer
- ✓ Co-creator of the Nexus (scaled Scrum Fx)
- ✓ richard@accentient.com

@rhundhausen

Microsoft
Professional Scrum Development with Microsoft Visual Studio 2012
Richard Hundhausen

Cadence \ ˈkā-dᵊn(t)s \

Cadence

A rhythmic sequence or
flow of sounds in language

Cadence

A rhythmic sequence or

flow of ~~sounds in language~~

value in the form of working software

More specifically …

Getting the team into a regular, comfortable, and sustainable development rhythm

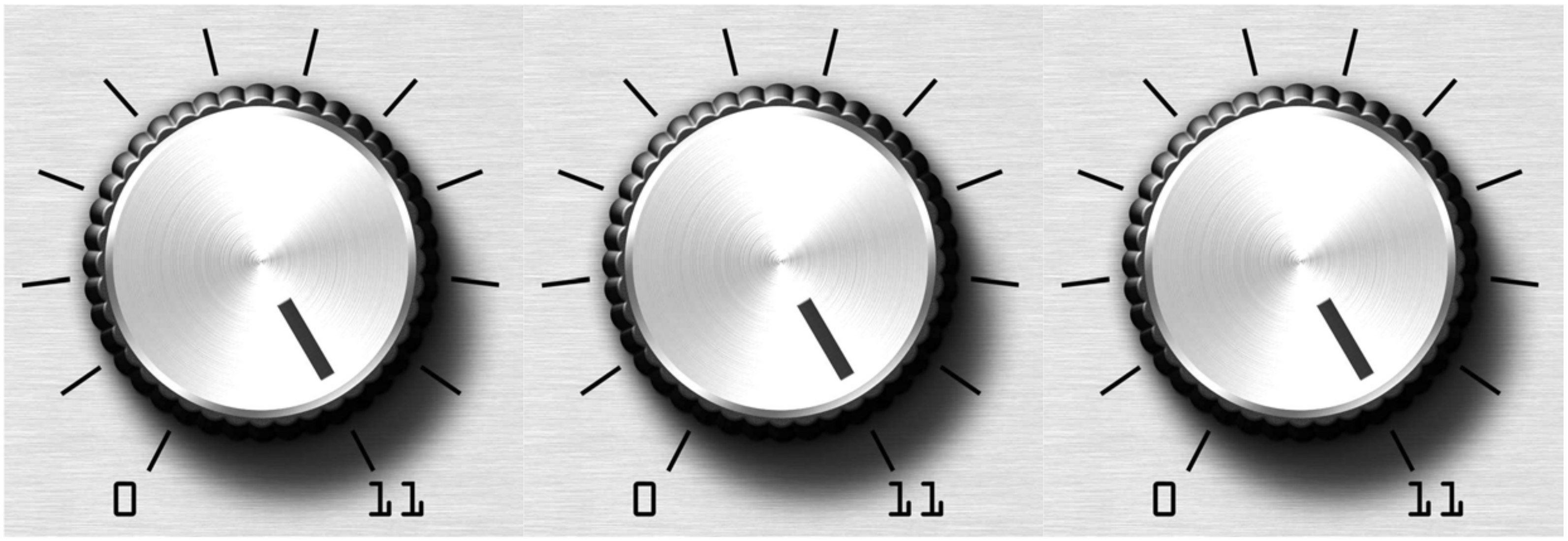
Why would we want this?

- ✓ Maximize value (in the form of working software)
- ✓ Manage variability
- ✓ Reduce risk
- ✓ Increase predictability
- ✓ Increase collaboration, creativity, morale

Unfortunately, many forces prevent cadence

I refer to them as *friction*

Friction falls into three categories …



PEOPLE

PROCESS

PRODUCTS

Let’s discuss a few of them …

People Frictions

Friction: Command and Control

Self-
Organization
Command
and Control

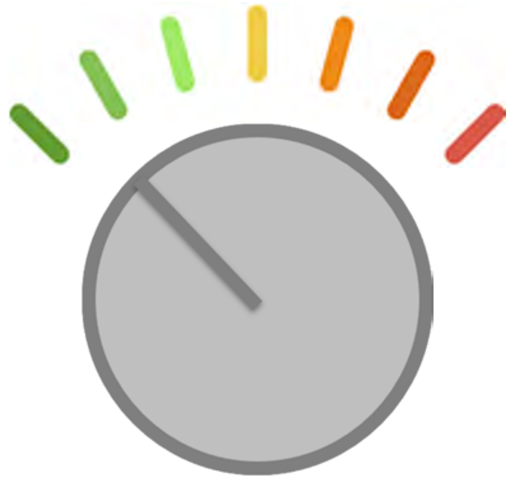
self-organizing == separating what/how == enabling creativity;
Management's job should be to optimize the system

Friction: Component Teams

Feature Teams
Component Teams

Cross-functional, cross-component teams that can complete an end-to-end customer feature

Friction: QA or DevOps Teams



Cross-functional Development Team

QA or DevOps Teams

Only the Dev Team does the work, regardless of domains that need to be addressed: testing, operations, architecture, business analysis

Friction: Working as Individuals

Working as a team

Working as individuals

Mobbing > Swarming > Pairing > 1 person/task > 1 person/story

Friction: Specialists

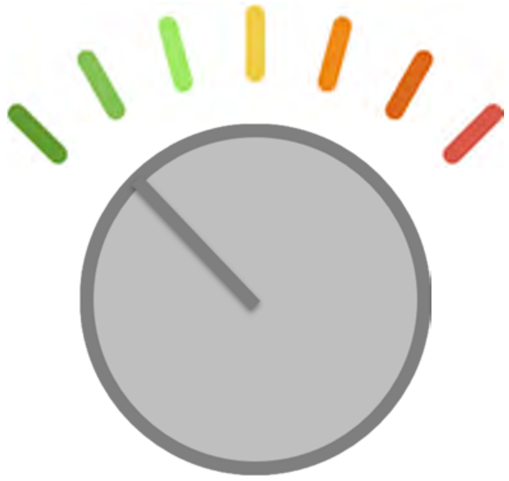
T-Shaped

Specialists

Generalists with expertise in certain skillsets
(or at least no allergies to doing other kinds of work)

Friction: Adhoc Team Formation

Long-lived Teams



Adhoc Team Formation

Team members are 100% dedicated to one (and only one) team and the team remains unchanged for the sake of stability

Visual Studio LIVE 25 YEARS OF CODING INNOVATION

Friction: Dislocated Teams

Collocated Teams

Dislocated Teams

Each team is collocated in the same room to maximize learning, focus, and the shared responsibility for team outcome

Visual Studio LIVE 25 YEARS OF CODING INNOVATION

Process Frictions

Friction: Obfuscated Work

Visualize Work and Progress

Obfuscated Work (and Progress)

Kanban board, Taskboard, Burndown, Burnup, SpecMap

Friction: Multitasking

Focus

Multitasking

Multitasking damages the brain (2009 Stanford study)
Slow down and get more "Done"

Friction: Multiple Projects

Sprints

Multiple Projects

Shift from *project* to *product* thinking and then projects simply become batches of related work delivered over sprints

Friction: Develop By Layers

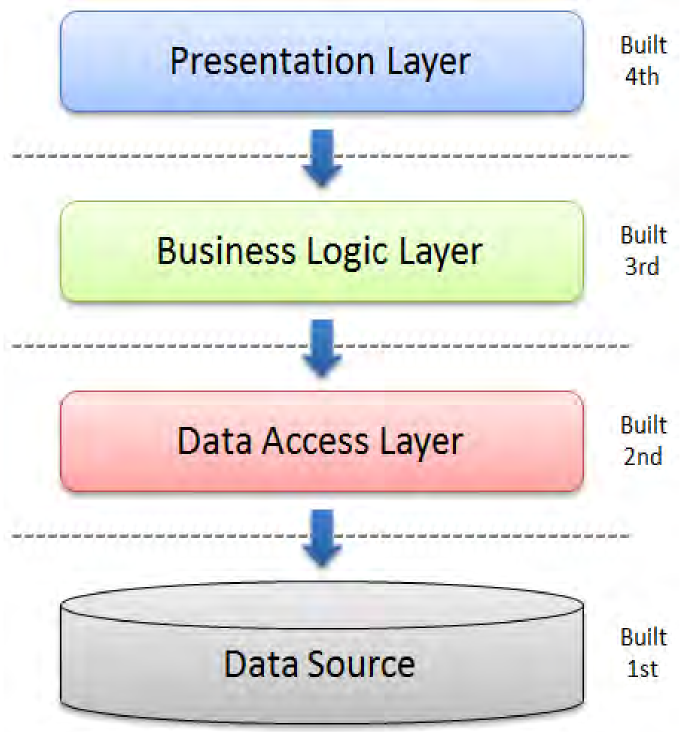
Thin-Slicing

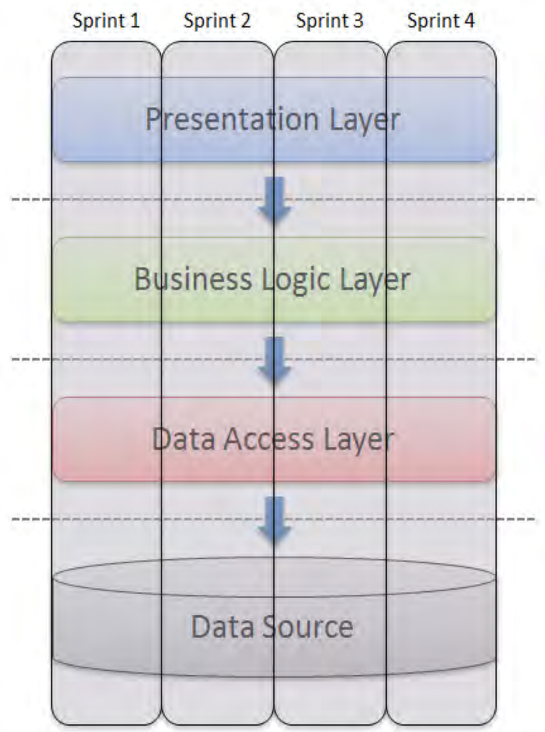
Develop by Layers

Develop the simplest possible functional, usable, end-to-end slice of functionality in order to obtain feedback

Thin-Slicing Example

- ✓ Layers = delayed value
- ✓ Slices = value every Sprint





Friction: Done is Subjective

Definition of "Done"

Done is Subjective

Defined by the Dev Team as a shared understanding of what "Done" is for all work in the product – for transparency

Friction: Dependencies

Make them transparent, remove them

Dependencies

Dependencies can be people, domain, technology, or software related; *external* dependencies are especially risky

Friction: Team Decides *What*

Stakeholder feedback

Team decides *what* to work on

Software development is complex, invisible work; we must work in small batches with regular feedback to be successful

Friction: Discontinuous Integration

Continuous Integration

Discontinuous Integration

Integrate your code with others several times a day; each push verified by automated build and tests == immediate feedback

Friction: Code Reviews

**Pairing,
Swarming,
Mobbing**

Code Reviews

Pairing, swarming, and mobbing increases learning and quality while increasing flow and reducing risk (by limiting WIP)

Product Frictions

Friction: Private Repositories

Collective Code Ownership

Private Repositories

The Development Team collectively owns everything, including the code;
All team members should have equal access to all repositories

Friction: Pull Requests

Pairing, Swarming, Mobbing + CI

Pull Requests

Pull Requests and other quality gates impede flow; Working as a team with immediate/automated feedback reduces friction

Friction: Working in Branches

Trunk-Based Development

Working in Branches

Shared branches off main/master/trunk are bad at any cadence;
www.trunkbaseddevelopment.com

Friction: Manual Deployments

Automated Deployments (RM)

Manual Deployments

A service that fully automates the testing and delivery of software in multiple environments all the way to production

Friction: Manual Testing

Automated Testing

Manual Testing

Manual testing doesn’t scale, especially during regression; Automated API testing > automated UI testing

Friction: User Acceptance Testing

Acceptance Testing

User Acceptance Testing

External dependencies, such as users/customers running tests, are a risk; Only the Dev Team does the work and can say if it's "Done"

Friction: Code Deployment

LaunchDarkly

Feature Release

Code Deployment

Use feature flags to deploy releases on demand and enable canary releases, A/B testing, and even Hypothesis-Driven Development

And now for a bonus friction …

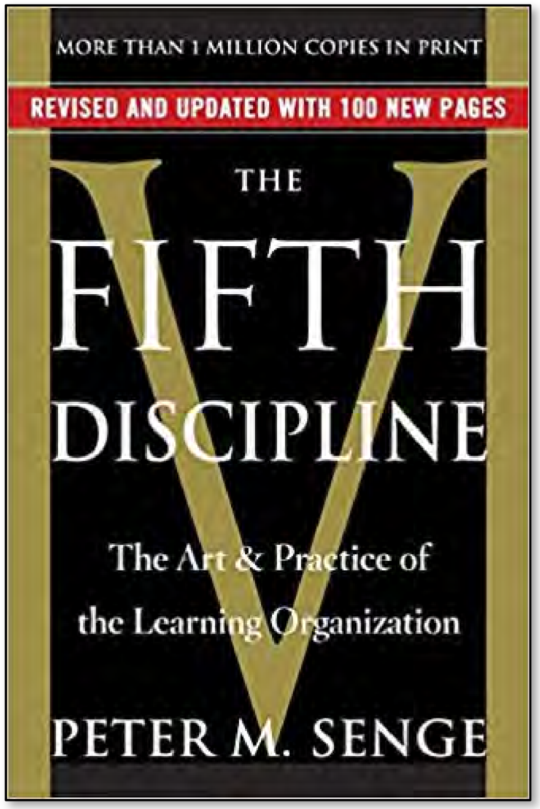
Friction: Local Optimization

Global Optimization (Systems Thinking)

Local Optimization

Local Optimization is making the "best" decision from the viewpoint of a person, team, or department, rather than a global optimization for the systems-level goal (e.g. *deliver value fast with high quality and high morale*)

Good books to read …

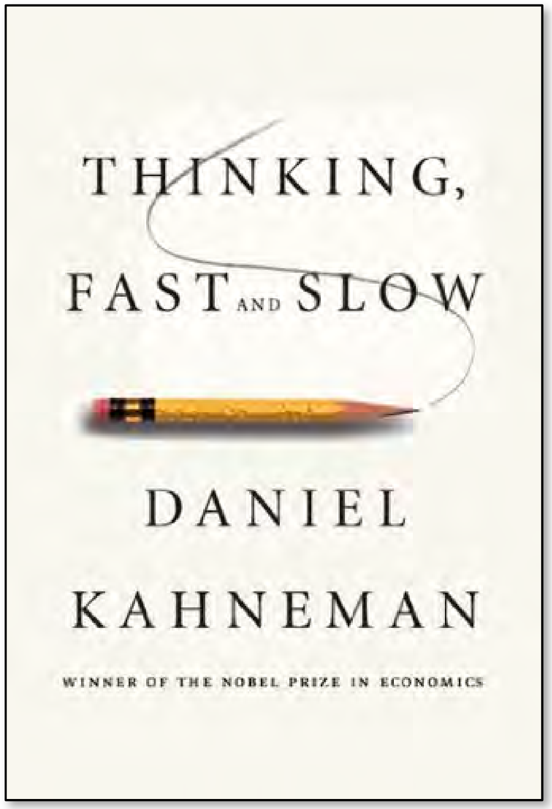

MORE THAN 1 MILLION COPIES IN PRINT

REVISED AND UPDATED WITH 100 NEW PAGES

THE FIFTH DISCIPLINE

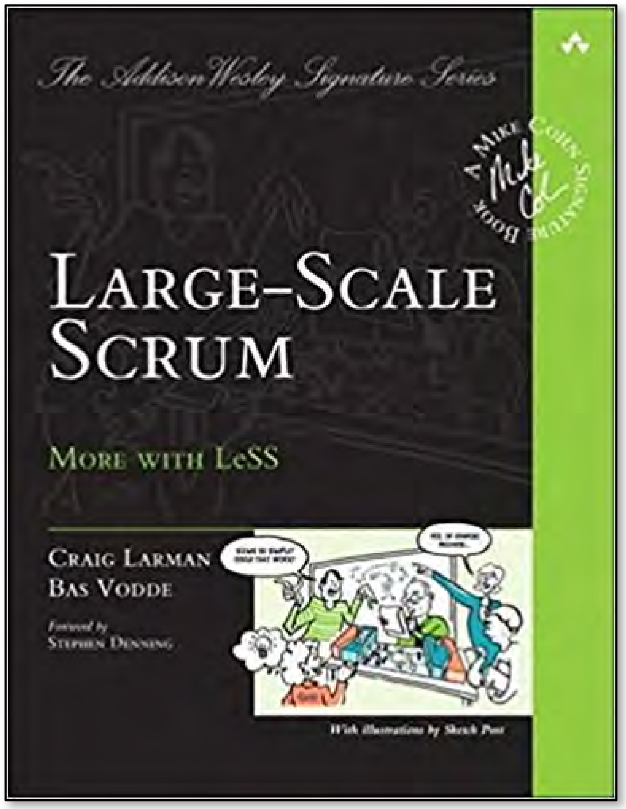
The Art & Practice of the Learning Organization

PETER M. SENGE


THINKING, FAST AND SLOW

DANIEL KAHNEMAN

WINNER OF THE NOBEL PRIZE IN ECONOMICS


The Addison Wesley Signature Series

A Mike Cohn Signature Book

LARGE-SCALE SCRUM

MORE WITH LeSS

CRAIG LARMAN
BAS VODDE

Foreword by
STEPHEN DENNING

With illustrations by Sketch Post

Visual Studio Live 25 YEARS OF CODING INNOVATION

Done();

(thank you)

richard@accentient.com | @rhundhausen

https://github.com/rhundhausen