LA-UR-19-21074
Approved for public release; distribution is unlimited.

Title: Continuous Integration! You Keep Saying Those Words. I Do Not Think They Mean What You Think They Mean.

Author(s): Adams, Terry R.

Intended for: Report

Issued: 2019-02-15 (rev.1)

Disclaimer:
Los Alamos National Laboratory, an affirmative action/equal opportunity employer, is operated by Triad National Security, LLC for the National Nuclear Security Administration of U.S. Department of Energy under contract 89233218CNA000001. By approving this article, the publisher recognizes that the U.S. Government retains nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or to allow others to do so, for U.S. Government purposes. Los Alamos National Laboratory requests that the publisher identify this article as work performed under the auspices of the U.S. Department of Energy. Los Alamos National Laboratory strongly supports academic freedom and a researcher's right to publish; as an institution, however, the Laboratory does not endorse the viewpoint of a publication or guarantee its technical correctness.

Continuous Integration!

You Keep Saying Those Words.
I Do Not Think They Mean What You Think They Mean.

Terry R. Adams

Monte Carlo Application ToolKit (MCATK)
Monte Carlo Methods, Codes, & Applications Group (XCP-3)
X-Computational Physics (XCP) Division
8 February 2019

Los Alamos
NATIONAL LABORATORY
EST. 1943

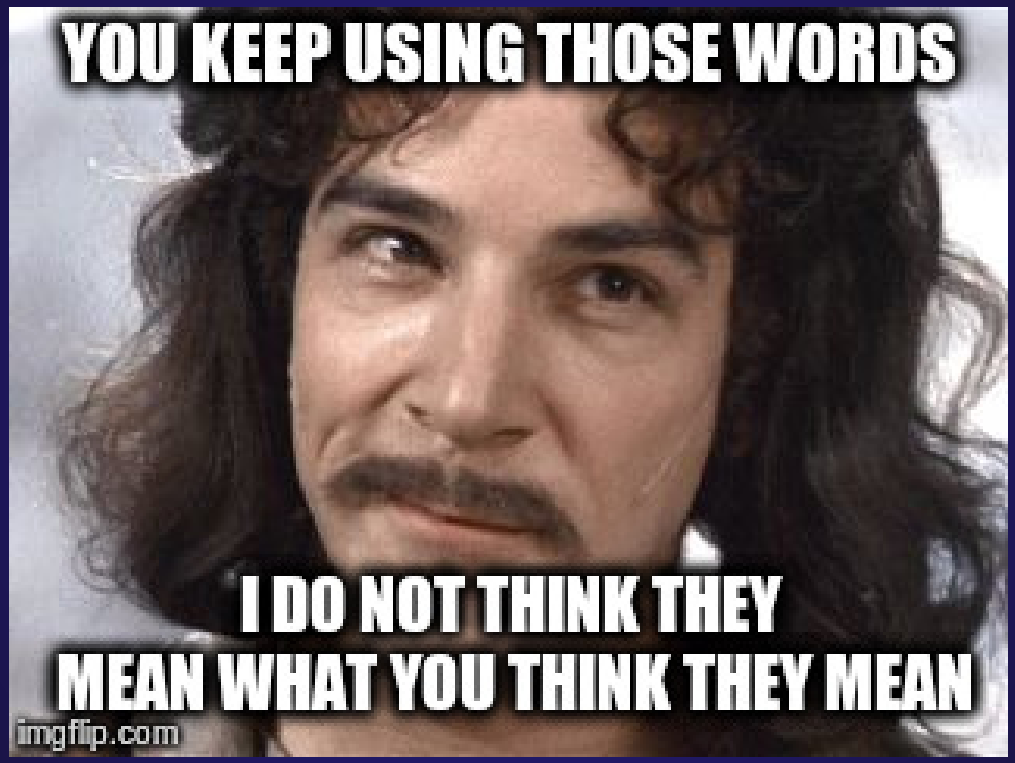
For more details, refer to the written companion document LA-UR-19-20745

Abstract

Continuous Integration (CI) is a best practice in the software community. Often though, there are several ideas about the meaning of continuous integration. With this uncertainty of definition often comes a loss of benefit to the practice. It is good to stop and reflect on how and why CI has been (and still is) practiced in software community. So, presented here is a historical perspective of CI along with associated practices, anti-patterns, and benefits. A subsequent presentation will discuss the workflows and controversy that had arised as CI has "changed" with time, especially in regards to feature-branch development.

Keywords:
Introduction, Continuous Integration, Software Development, MCATK

Continuous Integration !


YOU KEEP USING THOSE WORDS
I DO NOT THINK THEY
MEAN WHAT YOU THINK THEY MEAN
imgflip.com

A Common refrain Often Heard

“We do continuous integration!
We use [*insert your favorite CI software name*] …. ”

And …….

A Common refrain Often Heard

“We do continuous integration!
We use [*insert your favorite CI software name*] …. ”

“….. and it automatically runs ‘builds and tests’ when we make a commit (or a pull request).”

“Continuous Integration, *in its simplest form*, involves a tool that monitors your version control system for changes.”

- <u>Jenkins: The Definitive Guide: *Continuous Integration for the Masses*</u>

“The concept of continuous integration *means automating* the overall deployment process for an application *after a code has been committed*. It helps in reducing bugs in the deployment process.”

- ‘Continuous Integration: Its History and Benefits’ by Soumyajit Basu

"Continuous Integration, *in its simplest form*, involves a tool that monitors your version control system for changes."

- Jenkins: The Definitive Guide: *Continuous Integration for the Masses*

"The concept of continuous integration *means automating* the overall deployment process for an application *after a code has been committed*. It helps in reducing bugs in the deployment process."

- 'Continuous Integration: Its History and Benefits' by Soumyajit Basu

Perplexed

A Continuous Integration History: Why Should We Care?

"Something else in software that we are particularly bad at, we have a weak sense of HISTORY."

- Kevlin Henney 'Old Is The New New ', GOTO Conference 2018 video on YouTube

A Continuous Integration History: Why Should We Care?

"Something else in software that we are particularly bad at, we have a weak sense of HISTORY."

- Kevlin Henney 'Old Is The New New', GOTO Conference 2018 video on YouTube

"SEMANTIC DIFFUSION occurs when you have a word that is coined by a person or group, often with a pretty good definition, but then gets spread through the wider community in a way that weakens that definition. This weakening risks losing the definition entirely - and with it any usefulness to the term."

- Martin Fowler 'Semantic Diffusion'
https://martinfowler.com/bliki/SemanticDiffusion.html

And caution p.s.

A Continuous Integration History: Why Should We Care?

"Something else in software that we are particularly bad at, we have a weak sense of HISTORY."

- Kevlin Henney 'Old Is The New New', GOTO Conference 2018 video on YouTube

"SEMANTIC DIFFUSION occurs when you have a word that is coined by a person or group, often with a pretty good definition, but then gets spread through the wider community in a way that weakens that definition. This weakening risks losing the definition entirely - and with it any usefulness to the term."

- Martin Fowler 'Semantic Diffusion'
https://martinfowler.com/bliki/SemanticDiffusion.html

P.S.
"Let's not forget there's a tricky balance between holding to a clear definition and dogmatism."
- Martin Fowler 'Semantic Diffusion'

Continuous Integration! – *You Keep Using Those Words …*

CONTINUOUS INTEGRATION (CI) is

- **A *PRACTICE*, not a software tool;**
- **An *ATTITUDE* as much as a practice;**
- **As much about what happens *BEFORE* a commit, as during and after a commit.**

Not the focus

Continuous Integration: Not The Focus Here

Please Note, **that this presentation will *Not* address the last 10 years worth of new software development tools and accompanying integration practices.**

Such As:

- **Tools ("CI" Software Automation Servers, Distributed Version Control Systems (DVCS))**
- **Git, GitHub, Feature-Branches, GitFlow, and Pull-Requests**

Because of Its Importance, and I Want To Do It Justice:

This topic will be reported on subsequently at a later date.

Focus will be

Continuous Integration: The Focus (For The Most Part)

WILL BE ON:

- Advances in Software Integration Leading Up to CI
- One *Lineage* of "Historical" CI
- Some Controversy

Historical Background: Major Steps Forward on Integration

(1980s) Microsoft famed practice of "daily builds" (*Cusumano and Selby 1995*).

(1990s) First published use of the term "Continuous Integration" (*Booch 1991*).

(1990s) "Integrate and test several times a day". First publication of XP. (*Beck 1998*).

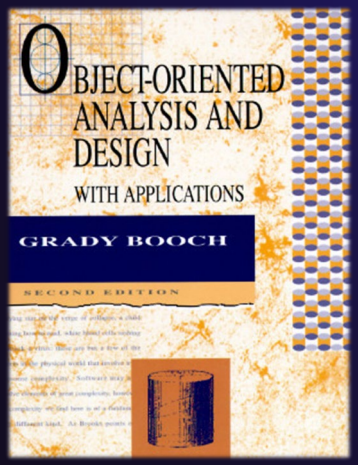
(1990s) Embracing Change with XP paper. CI included as one of the practices for XP. (*Beck 1999*).

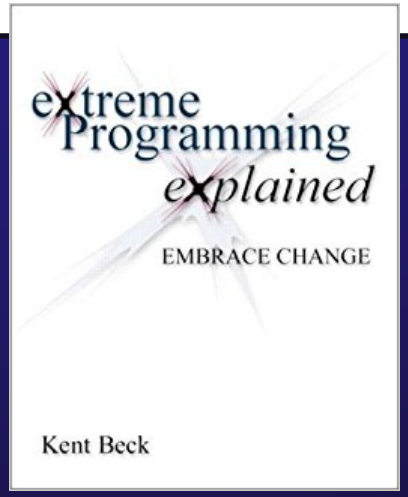
(1990s) Extreme Programming Explained book (seminal)**. *Describes CI* (*Beck 2000*).**

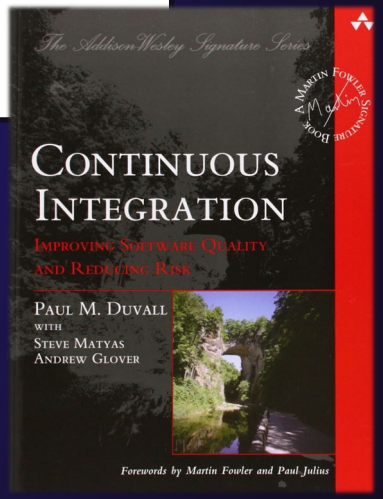
(2000) Continuous Integration paper (classic)**. *Outlines practices of CI* (*Fowler 2000 & 2006*).**

(2007) Continuous Integration book (definitive)**. (*Duvall 2007*).**

(2010) Continuous Delivery book (definitive)**. *(Humble & Farley 2010)*.**

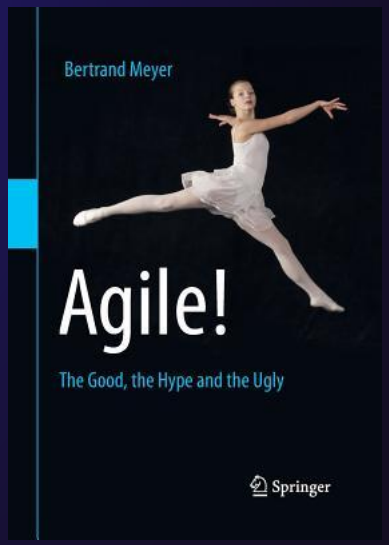






Software Integration

"*INTEGRATING* a software project means taking the components of the software as written so far, compiling them together and running the tests (the regression suite)."
– Bertrand Meyer (2014)



Meyer is a software engineer who invented Design by Contract and was one of the earliest proponents of object-oriented programming. See https://en.wikipedia.org/wiki/Bertrand_Meyer.

Phased Integration (McConnell, 2004)

Several Defined Phases:

1. **To Design, Code, and Test Each Component**
2. **To Take the Components and Combine Them Into One Large System**
3. **To Test (and Debug) the Whole System**

Start to integrate and test the *entire* system,
***end to end*,**
late in the scheduled release date.
(or just before the scheduled release date)

Sometimes called, "Big Bang" Integration

Integration Hell

Often the Resultant Of A 'Big Bang' Integration

Phone line Photo by G. Crescoli on Unsplash

Incremental Integration (McConnell, 2004)

Spreading the Pain of Integration Across the Project

- ***The increments can be small pieces of functionality in either classes or components that are developed and tested alone.***
- ***Then incrementally, integrated and tested back in pieces to the full system, sooner than later.***

Some of the Benefits:

- ***Errors are easier to locate***
- ***The system succeeds early in the project***
- ***You get improved progress monitoring***
- ***Improve customer relations***
- ***The units of the system are tested more fully***
- ***And you can build the system on a shorter development schedule***

INCREMENTAL INTEGRATION PROVIDES FASTER FEEDBACK

"Daily Builds" Microsoft 1980s (*Cusamano and Selby, 1995*)

Spreading the Pain of Integration Across the Project

- Stop all source commits to main at the end of day
- Run a build of integrated source
- Execute "Quick-Tests"

Quick-Tests cover a judicious, yet significant, set of end-to-end tests. Sometimes called, "Smoke-Tests".

Things Of Note:

- Daily synchronization cadence
- Suite of Quick-Tests
- Quick-Tests could be used by developers all during the development phase.
- Any issues fixed immediately
- faster feedback
- Commit to mainline code repository

Some of the Benefits:

- It minimizes integration risk
- It reduces the risk of low quality
- It supports easier defect diagnosis
- It improves morale

"Daily Build and Smoke-Tests" (*McConnell, 2004*) Code Complete

DAILY BUILDS REDUCE RISKS FOR THE FULL SYSTEM.

Continuous Integration (CI) : A Definition

"CONTINUOUS INTEGRATION is a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily – leading to multiple integrations per day. Each Integration is verified by a 'build' (including tests) to detect integration errors as quickly as possible."

– Fowler and Foemmel (2000, 2006)

Note: From now on in this presentation the term "Build" will mean to 'build and test'.

- INTEGRATE FREQUENTLY
- BY EACH MEMBER (AT LEAST DAILY)
- EACH INTEGRATION VERIFIED BY A "BUILD"

Side Note: Perhaps A More Apt Name, "*Continual Integration*"

Starting to Quantify CI

With These Publications:

- **Extreme Programming (XP) – Beck (1998,1999,2000)**
- **Continuous Integration Paper – Fowler and Foemmel (2000, 2006)**

Beck and Fowler started to Quantify CI.

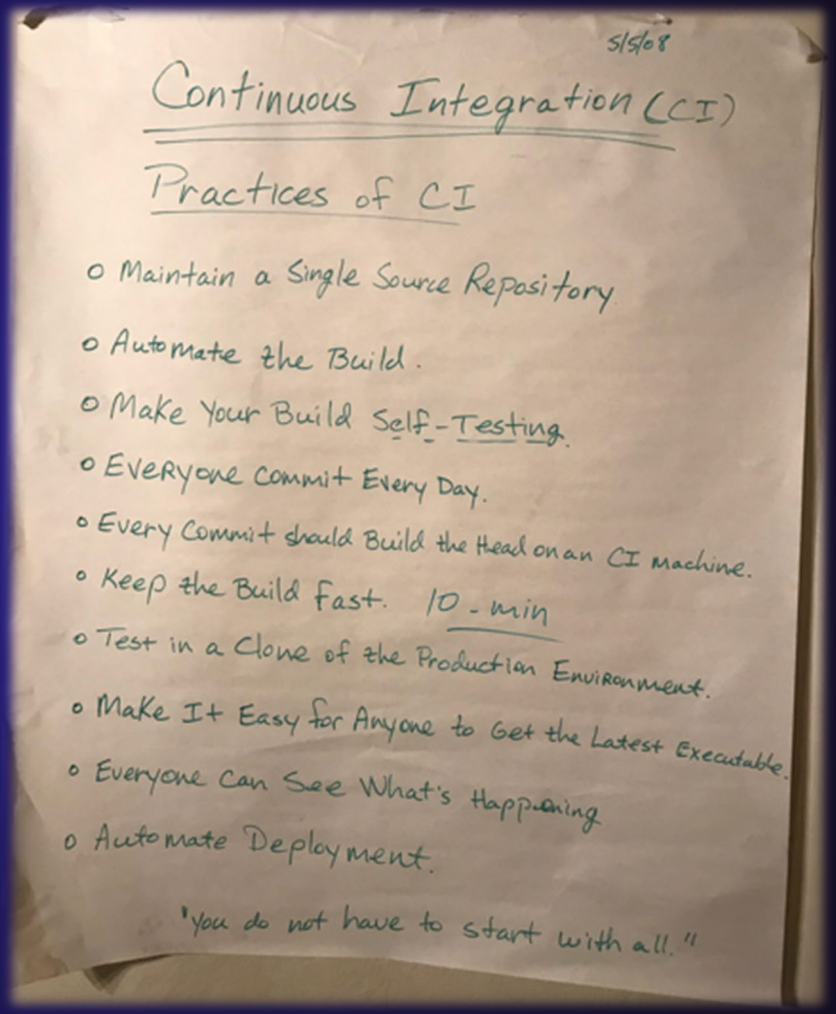
> ***"We are using the term Continuous Integration, a term used as one of the practices of XP (Extreme Programming). However we recognize that the practice has been around for a long time and is used by plenty of folks that would never consider XP for their work."***
>
> **– Fowler and Foemmel (2000)**

Practices of Continuous Integration
(*Fowler and Foemmel, 2006*)

(*MCATK 2008*)

- Maintain a Single Source Repository
- Automate The Build
- Make Your Build Self-Testing
- Everyone Commits to the Head Every Day
- Every Commit Should Build the Head on a CI Machine
- Keep The Build Fast (Ten-Minute Builds)
- Test In A Clone Of The Production Environment
- Make It Easy For Anyone To Get The Latest Executable
- Everyone Can See What's Happening
- Automate Deployment

You DO NOT Have To Start With All Of These



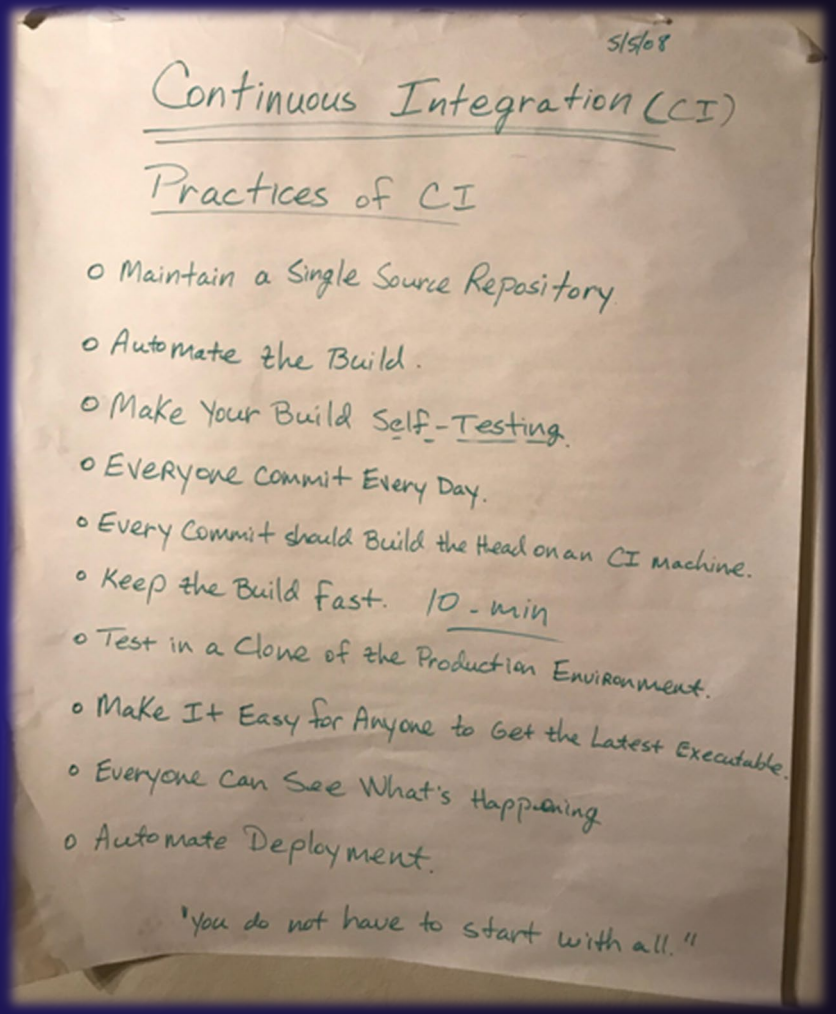
Practice to talk on more later.

Practices of Continuous Integration
(*Fowler and Foemmel, 2006*)

(*MCATK 2008*)

- Maintain a Single Source Repository
- Automate The Build
- Make Your Build Self-Testing
- Everyone Commits to the Head Every Day
- Every Commit Should Build the Head on a CI Machine
- Keep The Build Fast (Ten-Minute Builds)
- Test In A Clone Of The Production Environment
- Make It Easy For Anyone To Get The Latest Executable
- Everyone Can See What's Happening
- Automate Deployment

You DO NOT Have To Start With All Of These



Anti-Patterns (*Duvall, 2007b and 2008*)

1. Infrequent Check-Ins
2. Broken Builds
3. Minimal Feedback
4. Receiving Spam Feedback
5. Slow Machine
6. Bloated Build
7. Bottleneck Commits
8. Continuous Ignorance
9. Scheduled Builds
10. Works On My Machine
11. IDE-Only Build
12. Myopic Environment
13. Polluted Environment

Also see “Forces Affecting Continuous Integration” (*Shore, 2008*)

Continuous Integration Benefits

Again: Spreading the Pain of Integration Across the Project

- Reduces Risks to Integration and the Overall Project
- Increases Visibility and Communication of Code Changes Between Developers
- Establishes Greater Confidence in the Software Product for Developers, Managers, and Users
- Increases Product Visibility for Decisions and Trends
- Generates Deployable Software at any Time, any Place, and Frequently
- Reduces Repetitive Manual Processes
- Helps the Human Factor with Natural Breaks, Increased Confidence and Morale

INTEGRATE OFTEN
KEEP BATCH SIZE SMALL
BE PREPARED TO DEPLOY

CI practice two contentous ideas.

“Everyone Commits to the Head Every Day”

TWO CONTENTIOUS IDEAS:

1. **Daily Commits (at Least)**
2. **Commit Directly to the Head (Trunk-Based Development)**

Future Presentations On:
Trunk-Based Development (TBD) and Feature-Branch Development (FBD)

Continuous Integration : Daily Integration

Something to Remember

The Question: What is a Good Frequency of Integration?

"It is not always practical to integrate all of the code all the time. How often you integrate and test depends on what it takes to find defects… The proof that you integrating frequently enough lies in your ability to integrate rapidly at any time without finding defects."

- Poppendieck (2010)

Integration Cadences:

- ***Every Few Minutes***
- ***Every Few Hours***
- ***Every Day***
- ***Every Iteration***
- ***Every Deployment***
- ***After Deployment***

Continuous Integration: Trunk-Based Development (TBD)

Yes ….

Continuous Integration: Trunk-Based Development (TBD)

YES, IT IS STILL A THING

No it is not …

Continuous Integration: Trunk-Based Development (TBD)

YES, IT IS STILL A THING

AND NO, THEY ARE NOT

TrunkBasedDevelopment.com (Paul Hammant)

Trunk Based Development: Introduction

Trunk Based Development

Introduction
- One line summary
- Claims
- Caveats
- History
- This site

Context

Five-minute overview

Deciding factors

Version control system features

Version control system choices

Feature flags

Branch by Abstraction

Branch for release

Release from trunk

Continuous Integration (CI)

Committing straight to the trunk

Short Lived Feature Branches

Continuous Code Review

Continuous Delivery (CD)

Concurrent development of co...

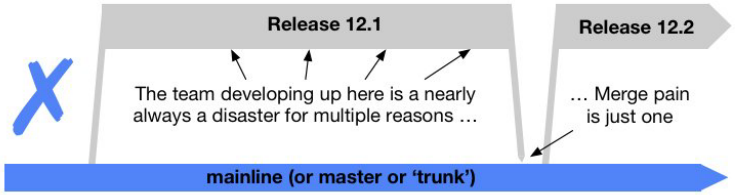
Introduction

One line summary

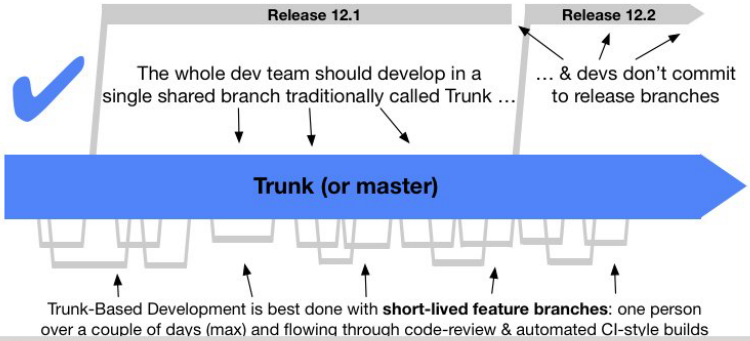
A source-control branching model, where developers collaborate on code in a single branch called 'trunk' *, resist any pressure to create other long-lived development branches by employing documented techniques. They therefore avoid merge hell, do not break the build, and live happily ever after.

* *master*, in Git nomenclature

Shared branches *off* mainline/master/trunk are bad at any release cadence

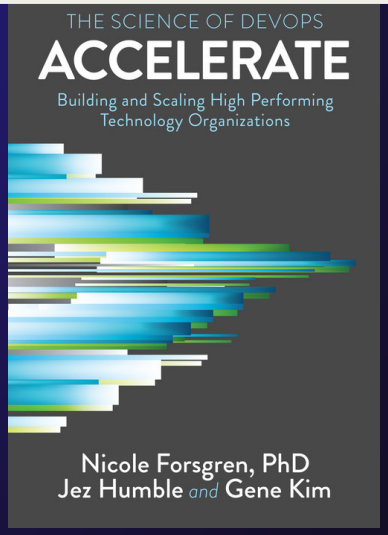


Release branches are good practice - but only up to a certain release cadence

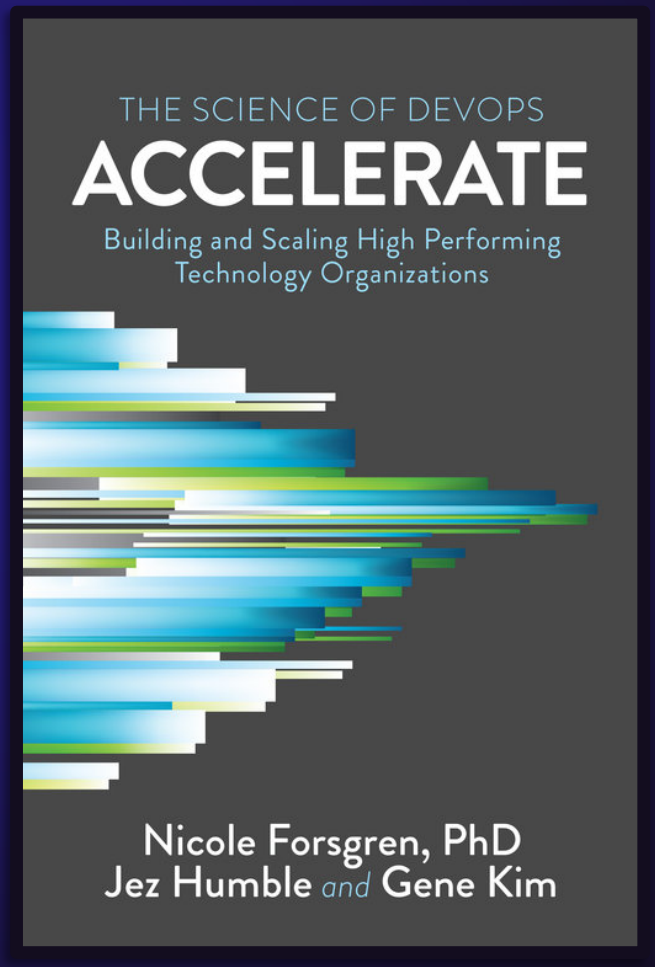


6:31 pm

DORAs State DevOps Reports (2014-2018) and Accelerate



DORAs State DevOps Reports (2014-2018) and Accelerate



Continuous Integration: Trunk-Based Development (TBD)

"….. uncovered 24 key capabilities that drive improvements in software delivery performance" (*Forsgren et al., Accelerate, p. xix*)

Two were CI and TBD

Continuous Integration: Trunk-Based Development (TBD)

"….. uncovered 24 key capabilities that drive improvements in software delivery performance" (*Forsgren et al., Accelerate, p. xix*)

Two were CI and TBD

(2016) THESE PRACTICES CONTRIBUTE TO HIGHER SOFTWARE DELIVERY PERFORMANCE:

- **Merging Code into Trunk on a Daily Basis**
- **Having Branches or Forks that are Short Lived (Less than a Day)**
- **Having Fewer than Three Active Branches**

Continuous Integration: Trunk-Based Development (TBD)

"….. uncovered 24 key capabilities that drive improvements in software delivery performance" (*Forsgren et al., Accelerate, p. xix*)

Two were CI and TBD

(2016) THESE PRACTICES CONTRIBUTE TO HIGHER SOFTWARE DELIVERY PERFORMANCE:

- **Merging Code into Trunk on a Daily Basis**
- **Having Branches or Forks that are Short Lived (Less than a Day)**
- **Having Fewer than Three Active Branches**

(2017) THEY INVESTIGATED BRANCH LIFETIMES AND FOUND:

- **High Performers had the Shortest Integration Times and Branch Lifetimes, with Branch Life and Integration Typically Lasting Hours.**
- **Low Performers have the Longest Integration Times and Branch Lifetimes, with Branch Life and Integration Typically Lasting Days.**
- **The Differences were Statistically Significant.**

A Few Users of Continuous Integration (or Delivery)

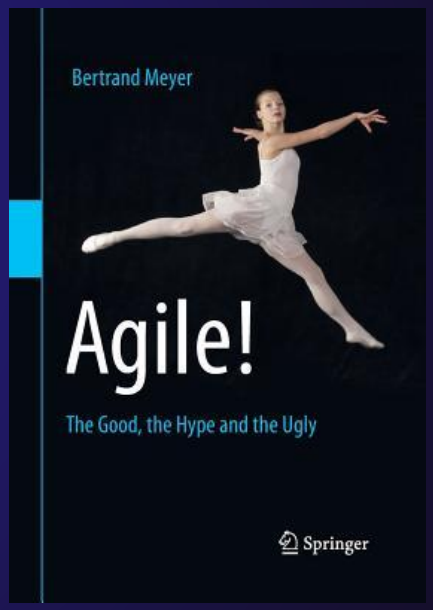
- IBM
- Microsoft
- Google
- Facebook
- Amazon
- Etsy
- Target
- Walmart
- Norstorms
- Fidelity Investments
- Intel
- Kitware
- Flickr
- LinkedIn
- Netflix
- Tesla
- Adobe
- Airbnb
- Capital One

For several, if not most, of these companies like Google and FaceBook, to name a few, Trunk-Based Development is the norm.

Bertrand Meyer
Agile! The Good, the Hype, and the Ugly

Out of all the XP practices, he considers *Test-First* and *CI* as the most lasting technical contributions to the software development industry and community.

Meyer, also, includes *CI* under the *Brilliant* category of his Ugly, Hyped, and Good categories of Agile.



Challenge To You

Analyze Your Process;
Find Ways to Improve It;
And in Doing so,
Strongly Consider Adding
Continuous Integration Ideas and Practices

princess bride quote

I have told you the truth,
and it's up to you to live with it.

– inspired by Yeste of The Princess Bride

Questions or Comments?

References

- Beck, Kent (1998), "Extreme Programming: A Humanistic Discipline of Software Development", Proceedings of International Conference on Fundamental Approaches to Software Engineering, Springer Berlin Heidelberg, Berlin, Germany.
- Beck, Kent (2000), Extreme Programming Explained: Embrace Change, Addison-Wesley, Boston MA.
- Booch, Grady (1991), Object-Oriented Design: With Applications, The Benjamin/Cumming Publishing Company Inc., Redwood City, CA.
- Cusumano, Michael A. and Richard W. Selby (1995), Microsoft Secrets, The Free Press, New York, N.Y. 10020.
- Duvall, Paul M., Steve Matyas, and Andrew Glover (2007a), Continuous Integration: Improving Software Quality and Reducing Risk, Addison-Wesley, Upper Saddle River, NJ.
- Duvall, Paul (2007b), "Automation for the People: Continuous Integration Anti-Patterns: Make Your Life with CI Easier by Learning What Not To Do", Url: https://www.ibm.com/developerworks/library/j-ap11297/index.html, Retrieved, 1 December 2017.
- Duvall, Paul (2008), "Automation for the People: Continuous Integration Anti-Patterns, Part 2: Make Your Life with CI Easier by Learning What Not To Do", Url: https://www.ibm.com/developerworks/java/library/j-ap03048/j-ap03048-pdf.pdf, Retrieved, 1 December 2017.
- Forsgren, N., Humble, J., Kim, G., 2018. Accelerate: The Science of Lean Software and DevOps Building and Scaling High Performing Technology Organizations, 1st Edition. IT Revolution Press.
- Fowler, Martin and Foemmel, Mat (2000), "Continuous Integration (Original Version)", *Url*: https://www.martinfowler.com/articles/originalContinuousIntegration.html, Retrieved, 01 January 2018.
- Fowler, Martin and Foemmel, Mat (2006), "Continuous Integration", *Url*: https://www.martinfowler.com/articles/continuousIntegration.html, Retrieved, 01 January 2018.
- Humble, Jez and David Farley (2010), Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation, Addison-Wesley, Upper Saddle River, NJ.
- McConnell, Steve (2004), Code Complete (2nd Edition), Microsoft Press, Redmond, WA.
- Meyer, Bertrand (2014), Agile! The Good, the Hype and the Ugly, Springer International Publishing Switzerland.
- Poppenendieck, Mary and Tom Poppendieck (2010), Leading Lean Software Development, Addison-Wesley, Boston, MA.
- Shore, James (2005), "Continuous Integration is an Attitude, Not a Tool", *Url*: http://www.jamesshore.com/Blog/Forces-Affecting-Continuous-Integration.html, Retrieved, 6 November 2017.
- Shore, James (2006), "Continuous Integration on a Dollar a Day", *Url*: http://www.jamesshore.com/Blog/Continuous-Integration-on-a-Dollar-a-Day.html, Retrieved, 6 November 2017.
- Shore, James (2008), "Forces Affecting Continuous Integration", *Url*: http://www.jamesshore.com/Blog/Forces-Affecting Continuous-Integration.html, Retrieved, 6 November 2017.

Appendix A: MCATK Continuous Integration Process

This Appendix is included to document the Monte Carlo Application ToolKit's (MCATK) CI Practices and Process.

Continuous Integration: Simplest Form (*Shore 2006 & 2009*)

Basis of the MCATK CI process.

Checklist

- ❑ Integration Machine. (Find an Old Development Computer)
- ❑ Get a Token (Sock Monkey, Rubber Chicken, or pick your own)
- ❑ Procure a Desk Bell
- ❑ Automate the Build (at the least, a ONE button builds)
- ❑ Drink the “Kool-Aid”
- ❑ Launch

Continuous Integration: Simplest Form (*Shore 2006 & 2009*)

Checklist

- ✓ Integration Machine. (Find an Old Development Computer)
- ✓ Get a Token (Sock Monkey, Rubber Chicken, or pick your own)
- ✓ Procure a Desk Bell
- ✓ Automate the Build (at the least, a ONE button builds)
- ✓ Drink the "Kool-Aid"
- ✓ Launch

Steps

1. Build from clean local sandbox with 100% pass
2. Update sandbox from latest from Repo
3. Repeat Step 1.
4. Take the Token and/or Announce your intent to commit code.
5. Check-in code
6. Use integration machine to do a clean checkout and build with 100% pass.
7. Ring the bell; *you are done!*

Please Keep In Mind, As Part of the MCATK CI, …..

As your read the upcoming process outline, that it does not reflect that the MCATK Team also practiced:

- ❑ **Self-Testing "Build"**
- ❑ **Run A Private "Build"**
- ❑ **Ten-Minute "Build"**
- ❑ **Working in Small Chunks & Durations**
- ❑ **Commits Every Day (if not *several times* a day)**
- ❑ **Used a Clone Production Environment Upon Commit; With a Clean, New Checkout and "Build".**

Being a small team in a bullpen for co-location, and using "The One Button" and "Ten-Minute Builds" CI practices help make the following process doable.

Initial MCATK CI (*from MCATK's TeamForge Wiki July 2008*)

Process For Continuous Integration

1) Build & Test Local "Sandbox"
- make clean
- make all/run tests (multi-compilers)
- 100% NO errors (or warnings)

2) Update Local "Sandbox" With Head
- svn update
- Repeat step 1).

3) Communicate Intention Of Integration
- Verbally and/or with a CI Token ("Chucklehead")
- Brief code "freeze"

4) Commit Your Code
- svn ci -m "Really well tested and communicative code"

5) **Go To CI Machine & Build, from SCRATCH, the Head**
- svn co repository head
- make all/run tests
- If build fails
  - Evaluate Problem:
    - If a quick-fix then FIX-IT in your own local sandbox and Start Over at 1)
    - Else If NOT a quick-fix then UNDO Commit to SVN
    - Communicate your DONE
    - Start Over at step 1)
- 100% NO Errors or Warnings!

6) COMMUNICATE YOUR DONE BY PUTTING Sockmonkey Back in His Place!!

"Chucklehead" The Sockmonkey