

# Making Agile Work

---

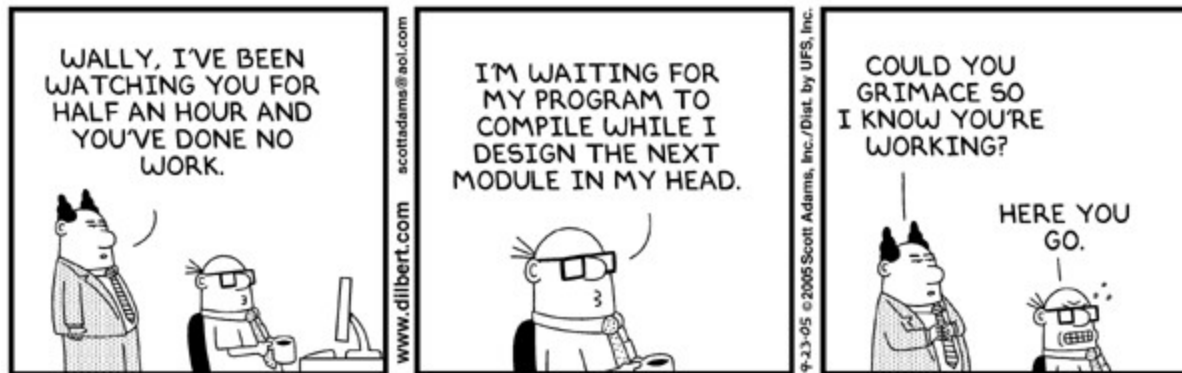
Brian Handley



[www.electric-cloud.com](http://www.electric-cloud.com)

# Funny, but too often true

---



© Scott Adams, Inc./Dist. by UFS, Inc.

## **Survey: Are You Adopting Agile?**

---

- a) No Agile, no plans**
- b) Not Agile now, but planning for it**
- c) Starting Agile (pilots, small groups)**
- d) Scaling Agile (broadening deployment)**
- e) Standardized on Agile**

## Once Upon a Time...

---

- **The Dark Ages**
- **Developers worked in small groups**
- **They integrated their software with each other in an ad hoc manner**
- **QA was ad hoc using builds thrown over the wall by developers**
- **When the software had to be shipped someone typed 'make' on The Build Machine**
- **Then they copied the installer to a floppy disk**
- **There was no such thing as a build manager**

## ...Then Assumptions Changed

---

- **Nightly build doesn't cut it**
  - The build takes longer than a night to build: 10, 11, 12 hours
  - There is no night anymore: Waltham hands off to Bangalore, which hands off to Krakow
- **Ad hoc process doesn't cut it**
  - "I don't know why it broke, it worked on my machine"
- **"Test later" doesn't cut it**
  - Too expensive to let QA find all bugs
- **The Build Team is under enormous pressure to DO SOMETHING**
  - They still use rough-edged tools, many DIY

# The Agile Chant

---

- **Build early, build often**
- **Test early, test often**
- **Developers adopt Continuous Integration**
  - Check-in and get instant feedback on integration problems
  - That means really fast build and test on demand
- **QA needs fully-tested builds for their work**
  - They need regular builds (perhaps at most once daily) that have undergone more testing
- **Management needs to measure progress**

**The build delivers one of the best indicators of working code: working code**



# Making Agile Work

---

Step 1: Getting There



[www.electric-cloud.com](http://www.electric-cloud.com)

# Making Agile Work: What's Needed?

---

- **Fast builds** because 1/day isn't enough!
  - "Overnight" is too slow, and so is "over lunch"
  - Need "coffee-break" builds
- **Automated builds** for consistent results
  - Same result no matter who runs the build, where, or when
  - Eliminate "But it worked on my machine!"
- **Access for all developers, build/release, and QA**
  - Scheduled builds, on-demand builds, and stimulus builds need to use the same process
- **Visibility** into process and results
  - Developers need to detect and fix problems quickly
  - Management needs trend info for planning

# Getting There: DIY

---

## ● **Fast builds**

- Buy lots of SMP hardware and try out GNU Make parallelism or manually parallelize the build.

## ● **Automated builds**

- Lots of scripting around builds & tests

## ● **Scheduled vs. On-demand vs. Stimulus builds**

- Call scripts from `cron`
- Write your own scm integration

## ● **Visibility**

- Build a web front end
- Lava lamps
- Skunk

# DIY

---

- **“We’re SW developers, we can build this”**
- **Before you do, think about...**
  - How much time do you have to write all that code?
  - How you are going to manage 200+ builds per day?
  - How are you going to get the build down from 4 hours to 15 minutes?
  - How are you going to manage hardware failure in your build system?
  - How are you going to store your process results?
- **Build teams now control the heartbeat of software engineering**
  - They need something more than a creaky Perl script

## Better Tools

---

- **Build speed tools**
  - ElectricAccelerator® speeds up builds 5-20x
- **Build visualization and troubleshooting tools**
  - ElectricInsight® shows you how your build works
- **Build-test-deploy management tools**
  - ElectricCommander® orchestrates, automates and reports on complex processes
- **Together these tools enable the release team to automate, accelerate, and analyze the build-test-deploy process**
- **And they enable effective CI and Agile development**

# Making Agile Work

---

Step 2: Making it Reliable and Effective



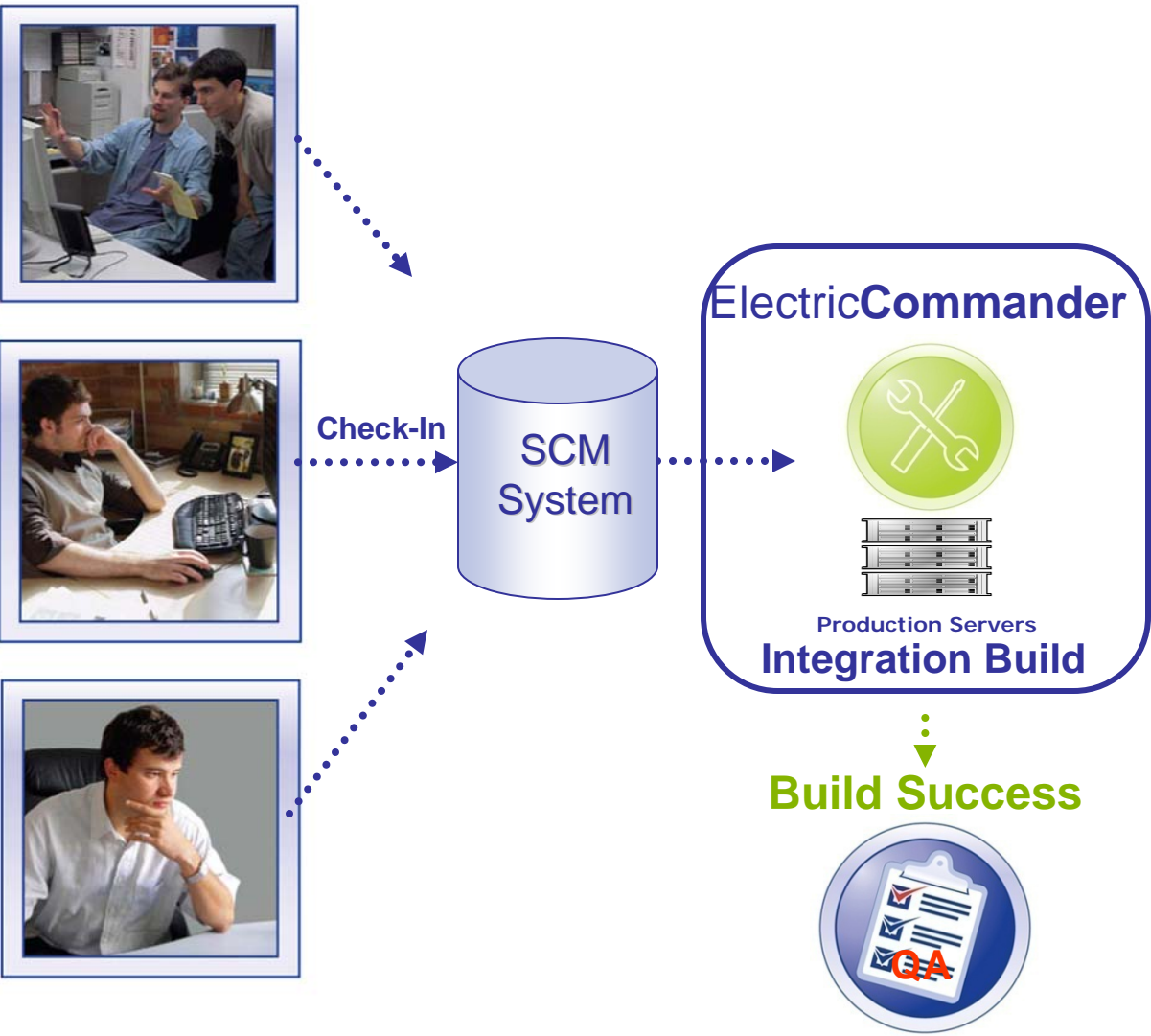
[www.electric-cloud.com](http://www.electric-cloud.com)

## OK – So You're Agile...

---

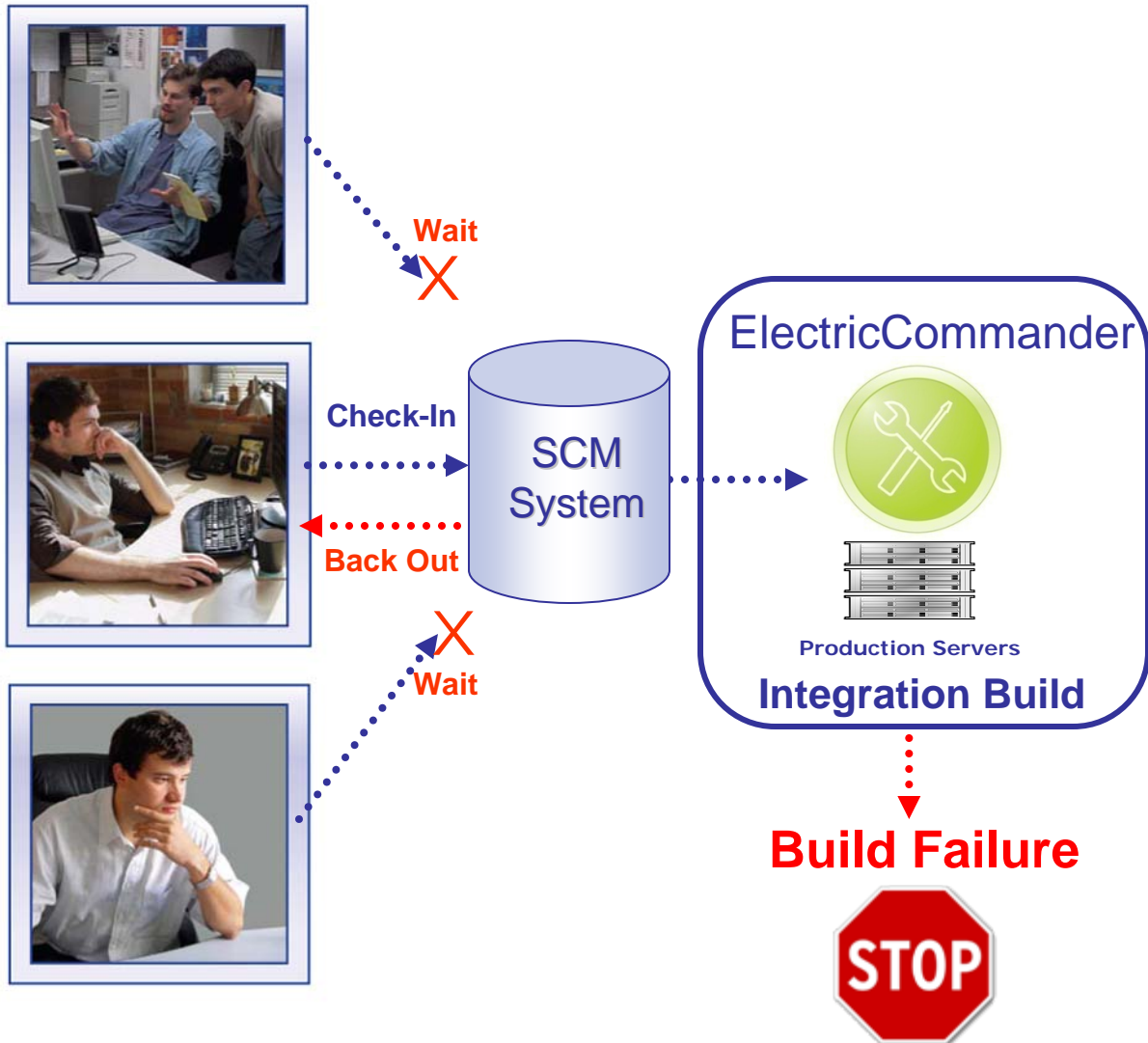
- **Continuous Integration delivers quality and productivity benefits**
- **But CI can break down**
  - The “it worked on my machine” phenomenon in a multi-platform environment
  - Continuously broken builds
- **The answer: “preflight” builds and tests**
  - Just as a pilot goes through a rigorous preflight check routine to ensure the plane is in top condition prior to takeoff
  - Preflight builds and tests give developers the power to compile and test each change across all targets, in the production environment without impacting the rest of the team

# Continuous Integration Theory



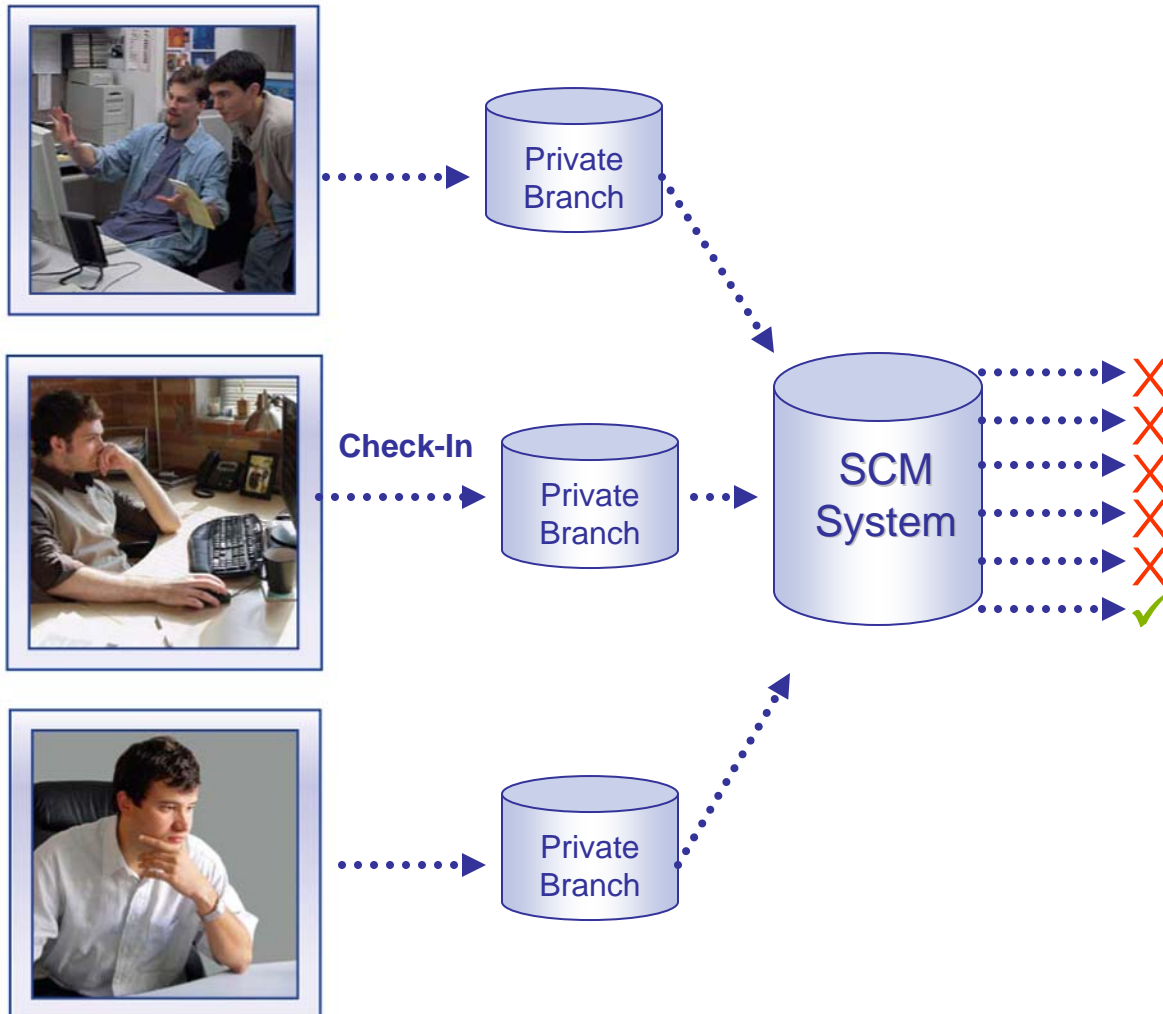
- Developer runs local build and automated tests
- Developer checks tested code into SCM system
- Integration build run at frequent intervals or upon check-in

# Continuous Integration Reality



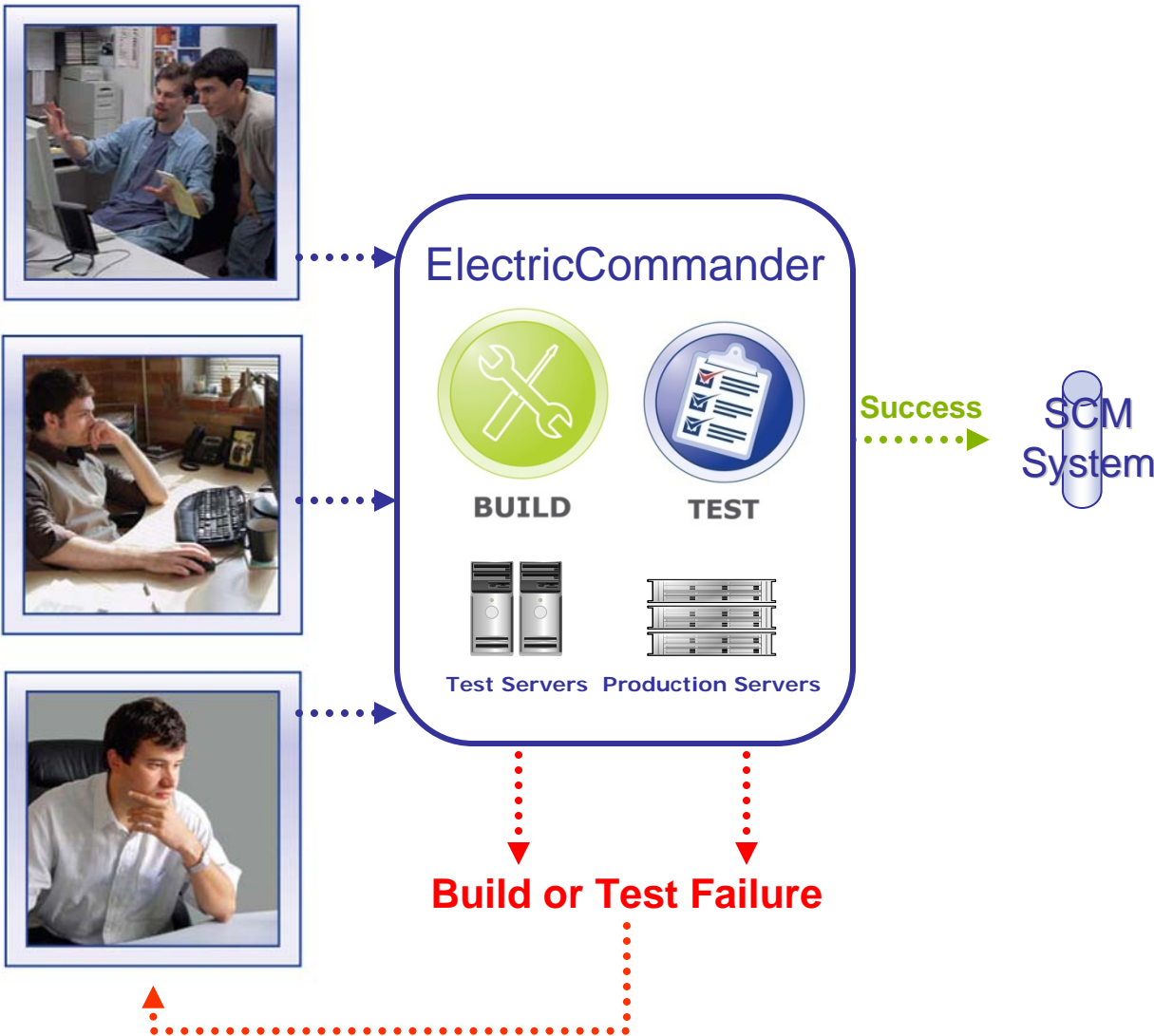
- Developer builds/tests on local system, checks in code
- Integration build started, breaks (“it worked on my machine”)
- Team impacted while check-in is backed out or build fixed

# Private Branches Can Solve Part of the Problem



- Developers don't step on each other
- Wind up with integration storms where many branches are merged
- Because so many changes have been introduced, the build is more likely to break repeatedly
- This just pushes the problem later in the cycle

# Preflight: Continuous Integration Evolved



- Developers build and test *across all targets/platforms*
- Ensures successful integration build
- Developers can check in changes only upon successful preflight
- Broken builds less likely to affect the entire team

## Put Builds and Tests in Developers' Hands

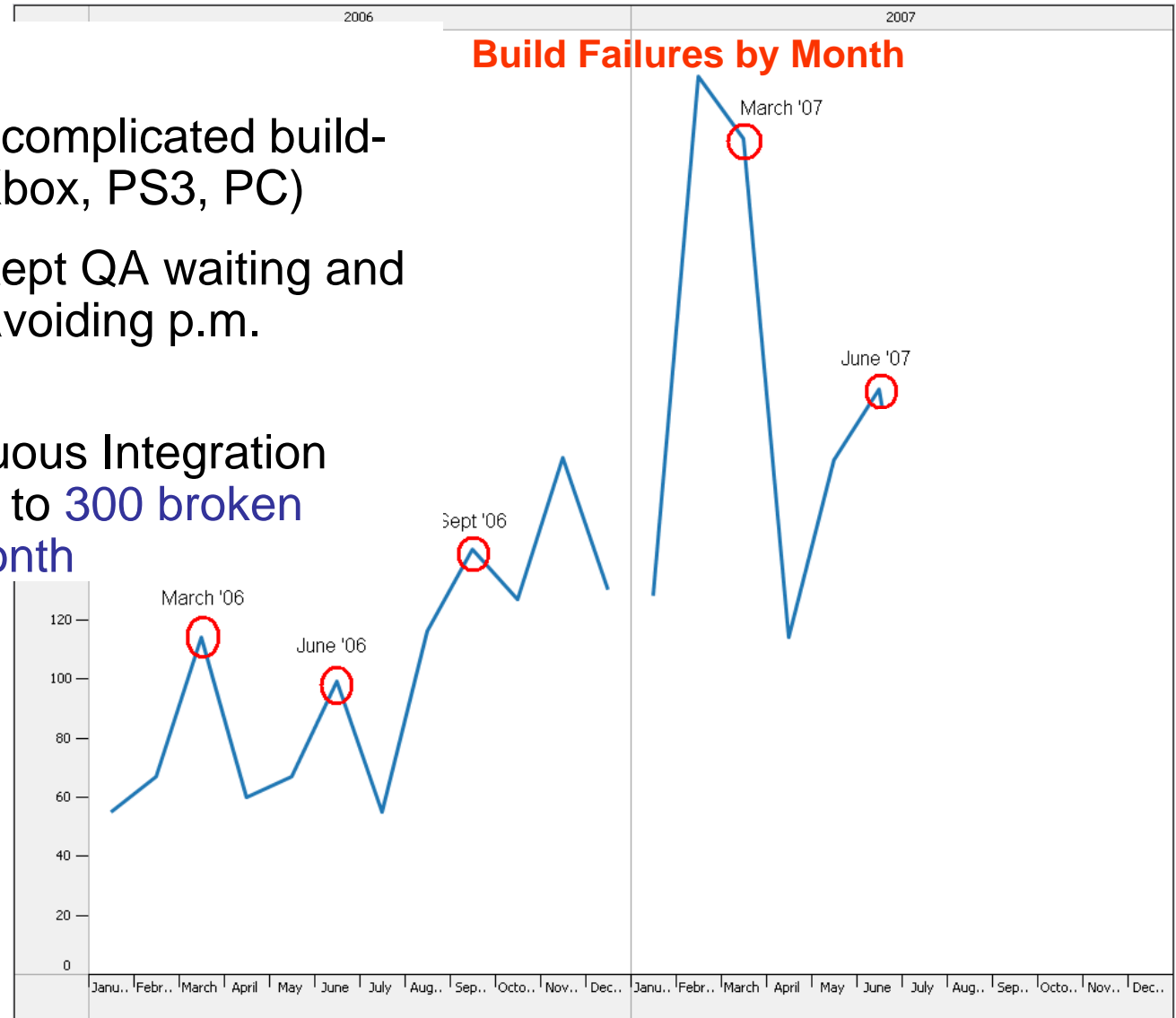
---

- **Allow developers to run preflight builds**
  - Developers submit change packages
  - Trial-merge the change and build it
  - Execute unit and other (“smoke”) tests
  - Invoke these procedures from within the IDE
- **If the build fails or a test fails**
  - Commit is rejected
  - Developer receives notification
- **If the build succeeds**
  - Commit (or promotion) is accepted
  - Developer gets the green light to work on other things

# Customer Example: BioWare

## Problem:

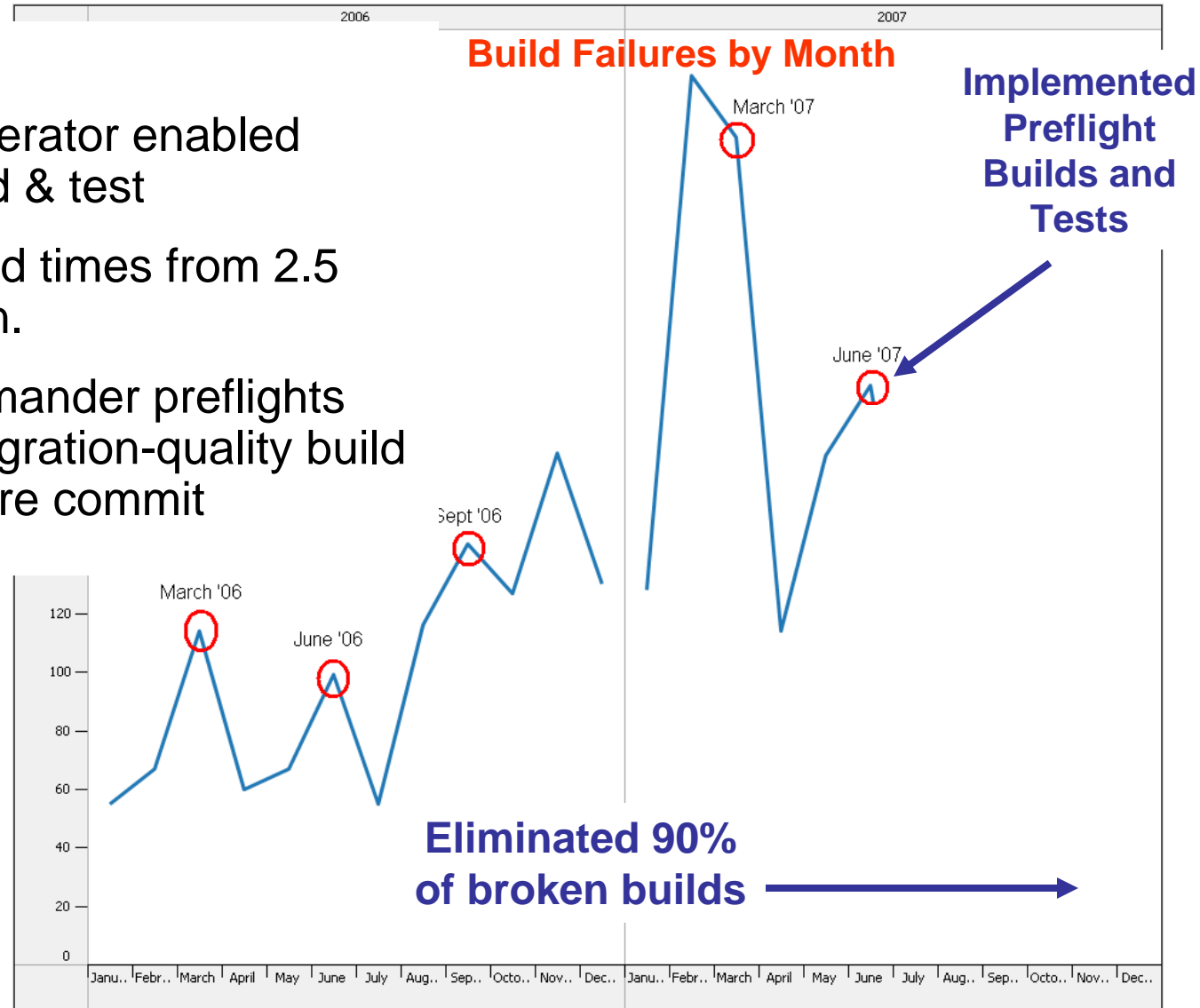
- Challenging, complicated build-test matrix (Xbox, PS3, PC)
- Slow builds kept QA waiting and Developers avoiding p.m. checkins
- Initial Continuous Integration approach led to **300 broken builds per month**



# BioWare: After Preflight Implemented

## Solution:

- ElectricAccelerator enabled frequent build & test
- Reduced build times from 2.5 hrs. to 15 min.
- ElectricCommander preflights provided integration-quality build and test before commit



## Conclusion

---

- **The build-and-test process is more important than ever – it's the heartbeat of development**
- **Making this process fast and automated is essential to agile development**
- **CI is a great place to start...**
- **...but preflight builds and tests are key**
  - makes CI reliable and effective in an enterprise development environment

# Questions

---



[www.electric-cloud.com](http://www.electric-cloud.com)