Continuous Integration

Benefits, Challenges and Best Practices

Presented by

Slava Imeshev

Viewtier Systems, Inc.

vimeshev@viewtier.com

Introduction

Viewtier Systems, Inc.

- Software build management company.
- Develops Parabuild, a continuous integration and software build management server.
- Is a privately held, self-funded, and self-sustained company.

Notion Of Software Build

Software Build

- **Process of transforming project code base into a usable application**
- Traces its roots to 1952 when the first compiler was devised by Grace Murray Hopper
- Also is used as **a communication pattern** to define:
  1. Current state of the code base (“build is broken”)
  2. Actual results of build process (“build #650 is in production)
  3. Build tools and scripts (“run build at 2:00am”)
- **Broken build means no product**

Story of a Broken Build

- Real-life, unfortunately
- A software company with
  - 30 software engineers
  - 5 QA engineers
  - 1 release engineer
  - Source control system and issue tracking is in place
  - Is moving fast towards the release date

Story of a Broken Build

Monday evening

Richard:

- Finishes a critical chunk of work
- Runs local build and tests cleanly
- Submits changes to CVS
- Marks the issue closed
- Goes home, proud of his work and happy for meeting an important deadline

Story of a Broken Build

Tuesday morning, 9:15AM

Sandeep, Jason, Eric, and Chris:

- **cvs update ...** to integrate with the new Richard's code.
- Run **ant all.clean**
- Expecting to see "**BUILD SUCCESSFUL**" …
- … but instead see

```
[javac]    131.     return new Layout(0, 0, 1, 1);
[javac]                      ^----^
[javac] *** Semantic Error: A candidate for type "Layout" was found, but it

[javac] Found 5 semantic errors compiling "D:/bt/src/viewtier/au

[javac]      3. import viewtier.ui.*;
[javac]                 ^---------^
[javac] *** Semantic Error: You need to modify your classpath, sourcepath,

BUILD FAILED
file:D:/bt/build-prod.xml:45: Compile failed; see the compiler error
Total time: 21 seconds
```

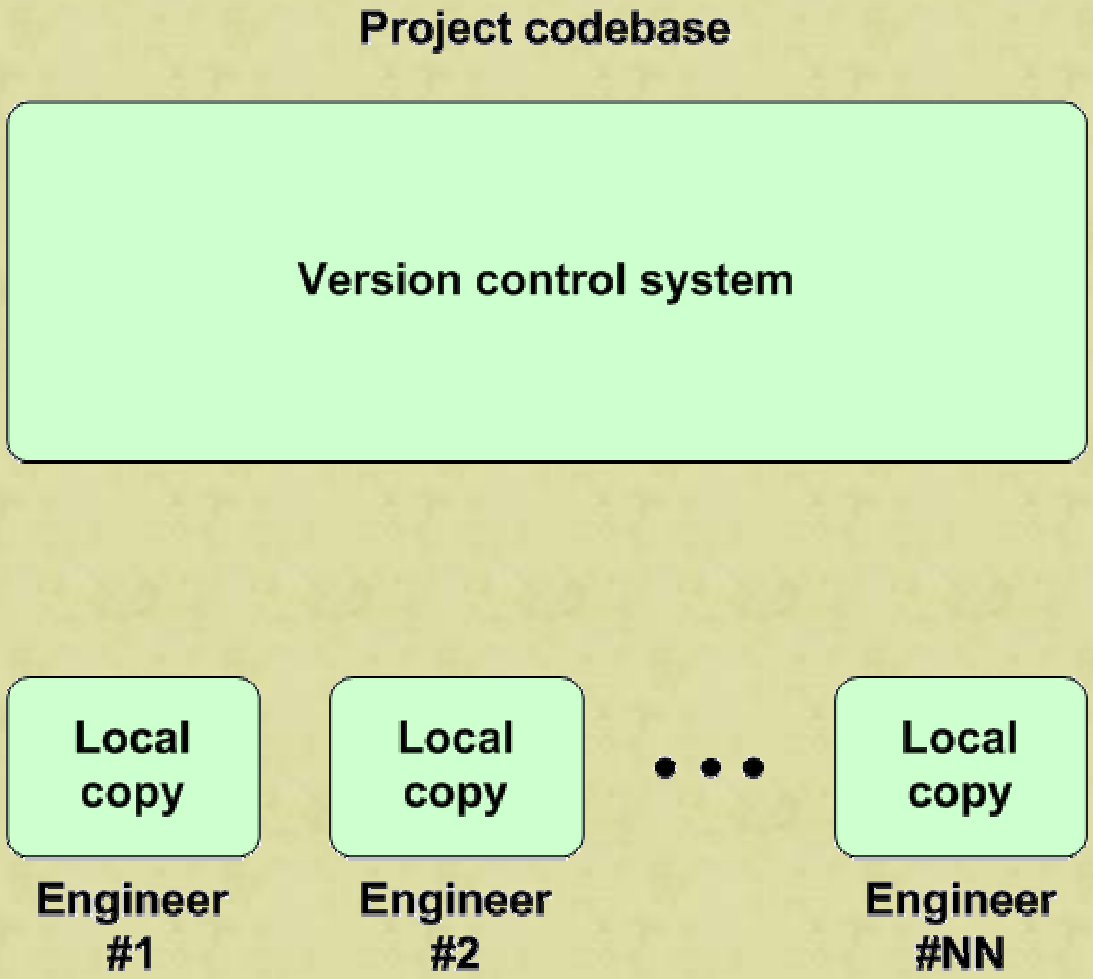
Story of a Broken Build

- Build **is broken**
- Richard forgot to add a new 3-rd party library
- Richard is out of the office and will be in at 1PM.
- Lucky ones have not updated to the latest and greatest

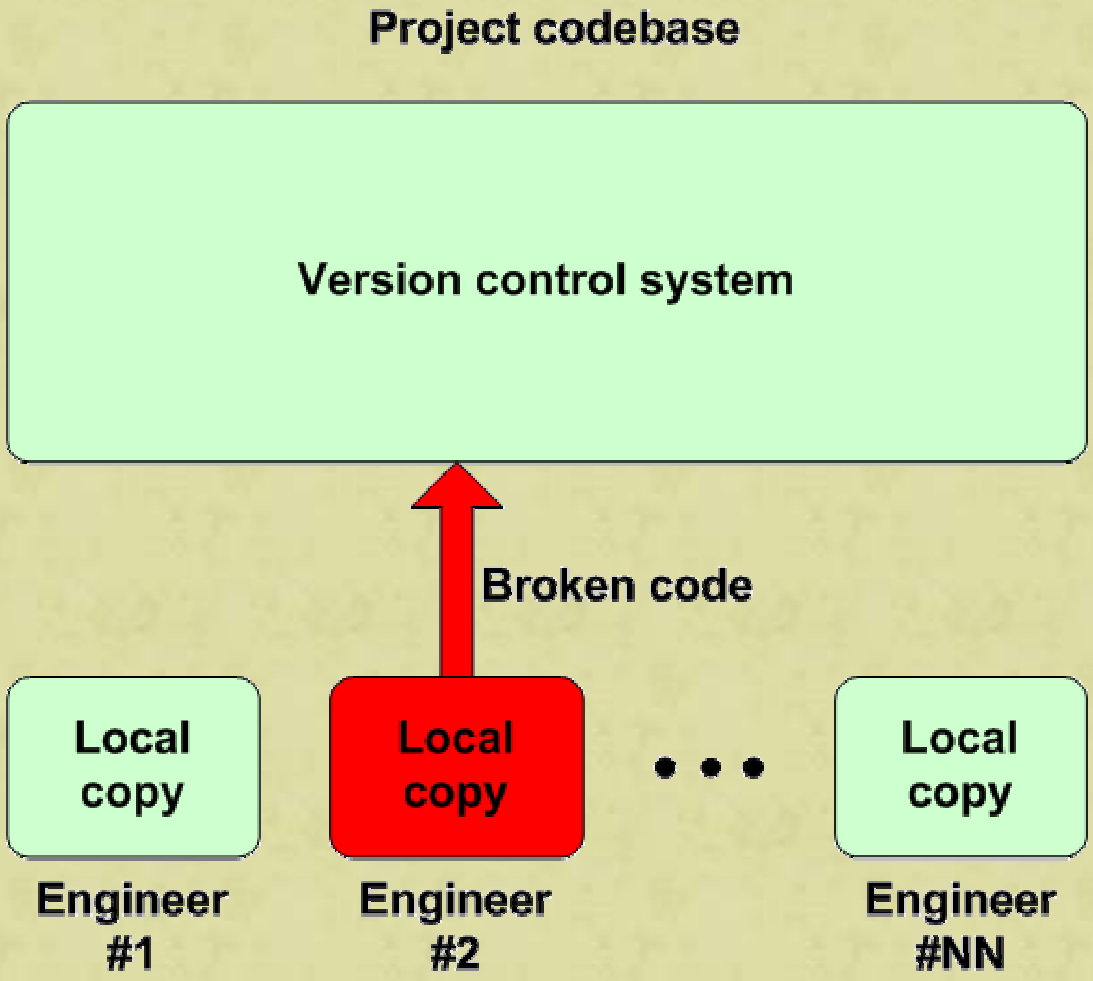
Story of a Broken Build (The End)

- Consequences
  - Sandeep, Jason, Eric and Chris couldn't work till 1PM when Richard arrives and submits the missing lib
  - The release engineer spent 4 hours trying to fix nightly build because it failed, too.
  - Five QA guys couldn't test new critical features till 2PM when new QA build is finally ready
- Result of the **single broken build**
  - Money: **45 man-hours → $4,500** went south at one shot
  - Time: Project time line slipped 5 hours → priceless ($100,000 fine per a day of delay)
  - Team spirit: you @#$%$%%!!!

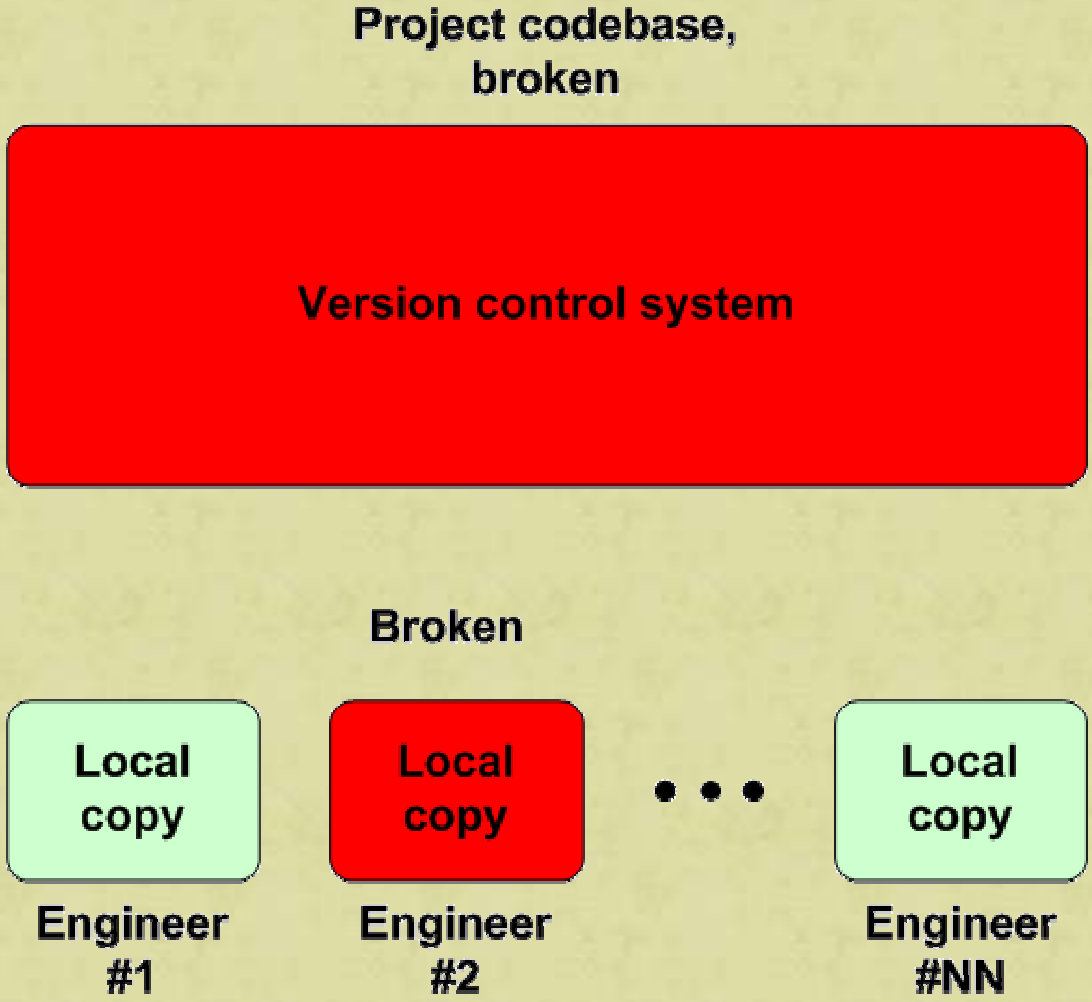
Spreading Build Breakage



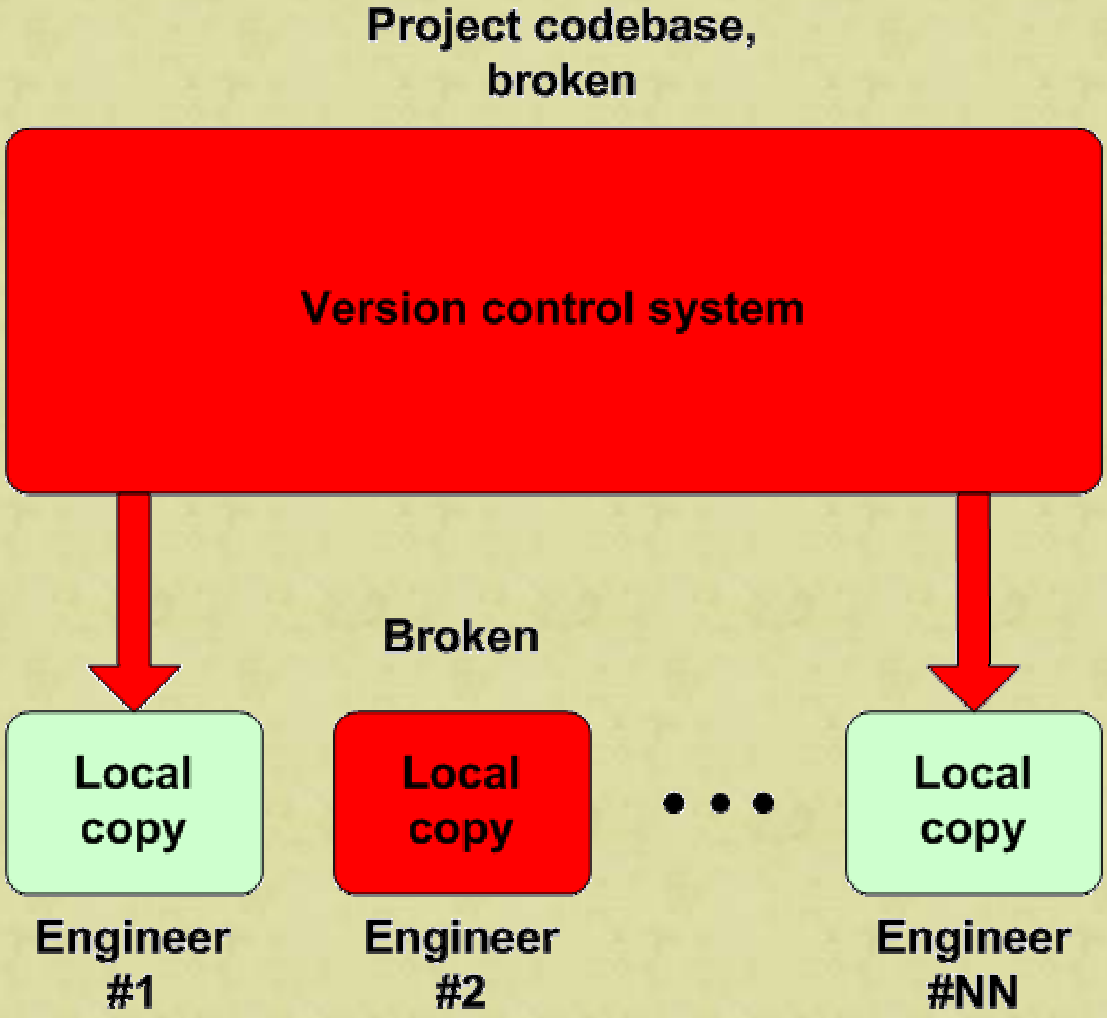
Spreading Build Breakage



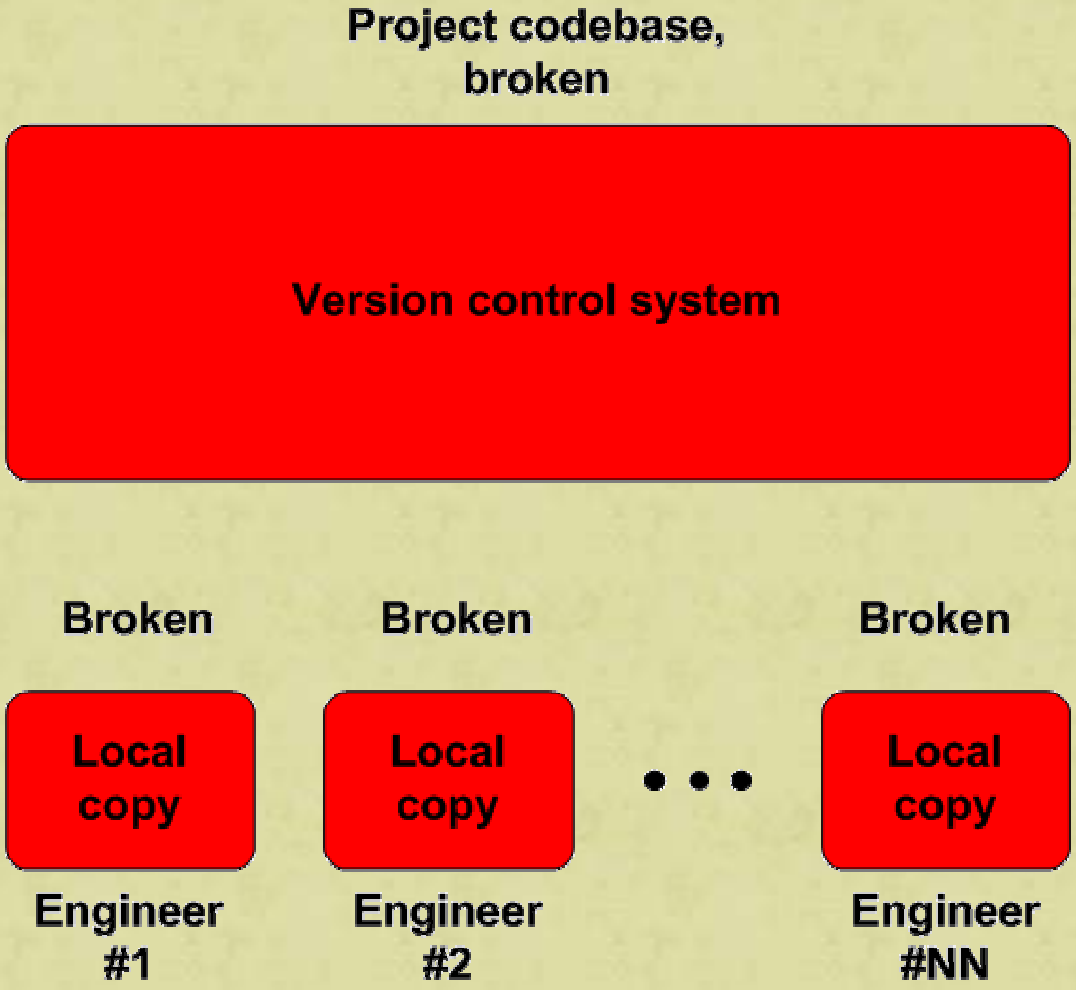
Spreading Build Breakage



Spreading Build Breakage



Spreading Build Breakage



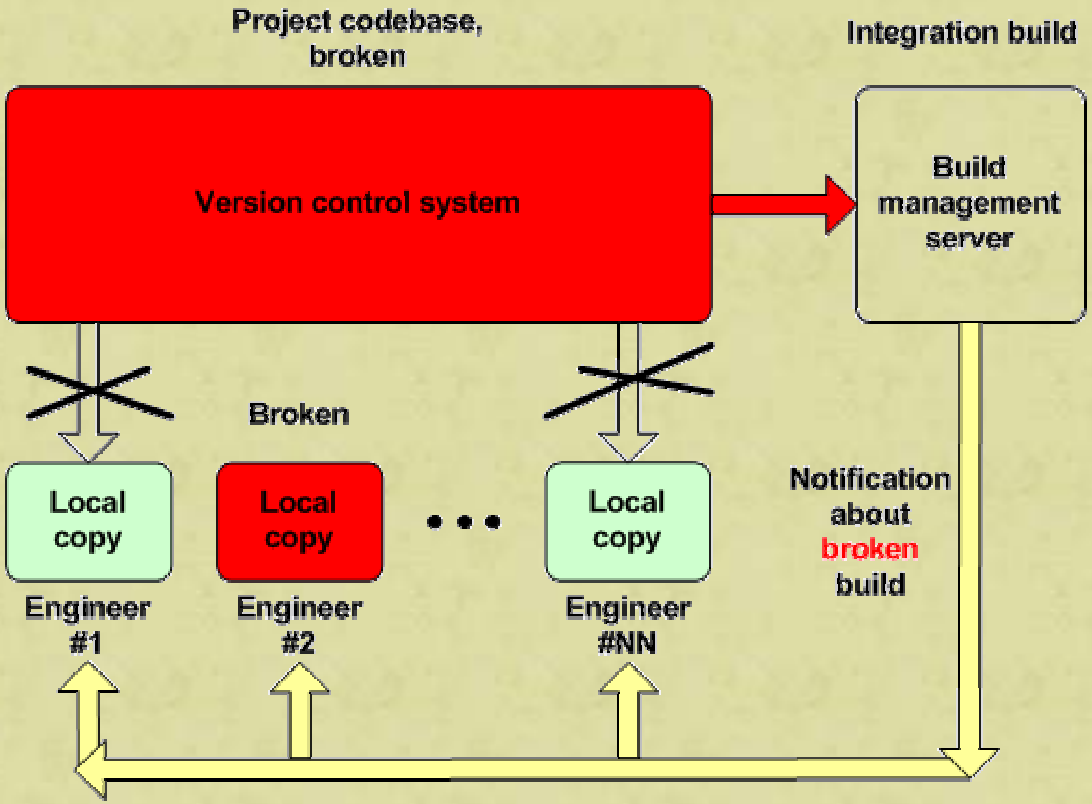
The Problem

- Broken builds increase risk of project failure
  - Cause **loss** of about **an hour per every member of engineering team.** A team of **10 engineers** loses about **80 hours** or **$8,000** per months **minimum** (two broken builds a week)
  - Losses grow linearly with the number of engineers, doubles for distributed teams
  - Schedule slips
- Broken builds mean no product
  - A build transforms project source code to a product
  - No clean build - no product
- Broken builds are dangerous
  - A team can be paralyzed in a matter of minutes if unwillingly checked out a broken code base

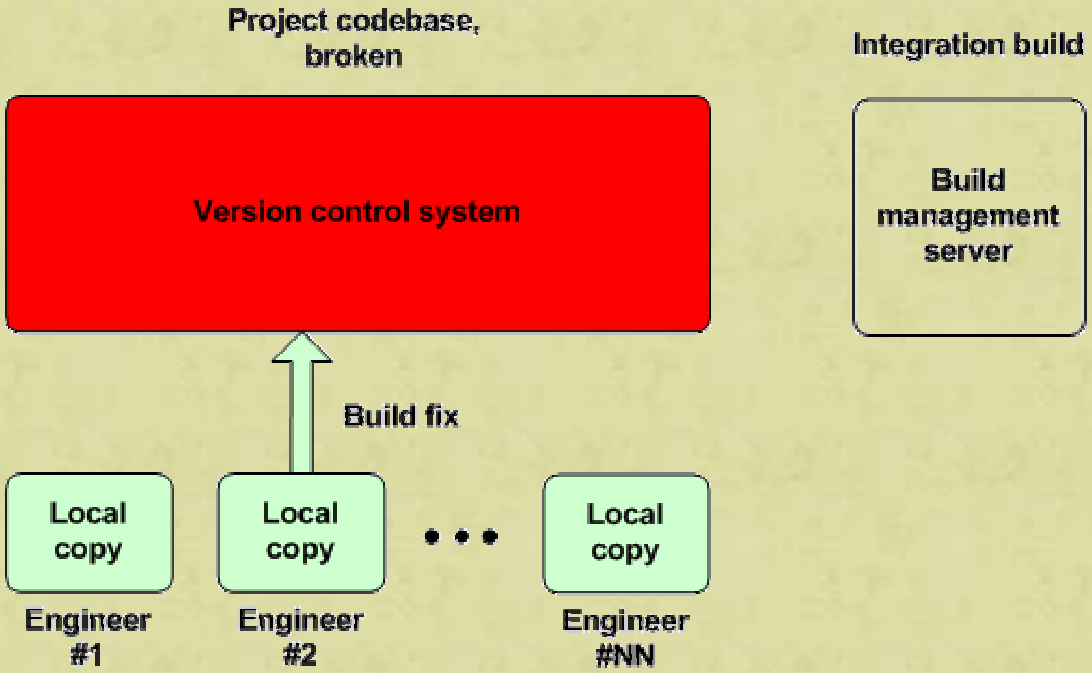
The Solution

- **Integration Build**
  - is a process of clean rebuilding of project code base to ensure that new changes integrate well into the existing code base
  - provides feedback on quality of new changes so that timely fixes can be delivered if the changes don't integrate and break the project code base
  - supposedly introduced by IBM software team in early 60-s when developing OS/360
  - run by a dedicated build management server.
- Running integration builds continuously is also known as **Continuous Integration**

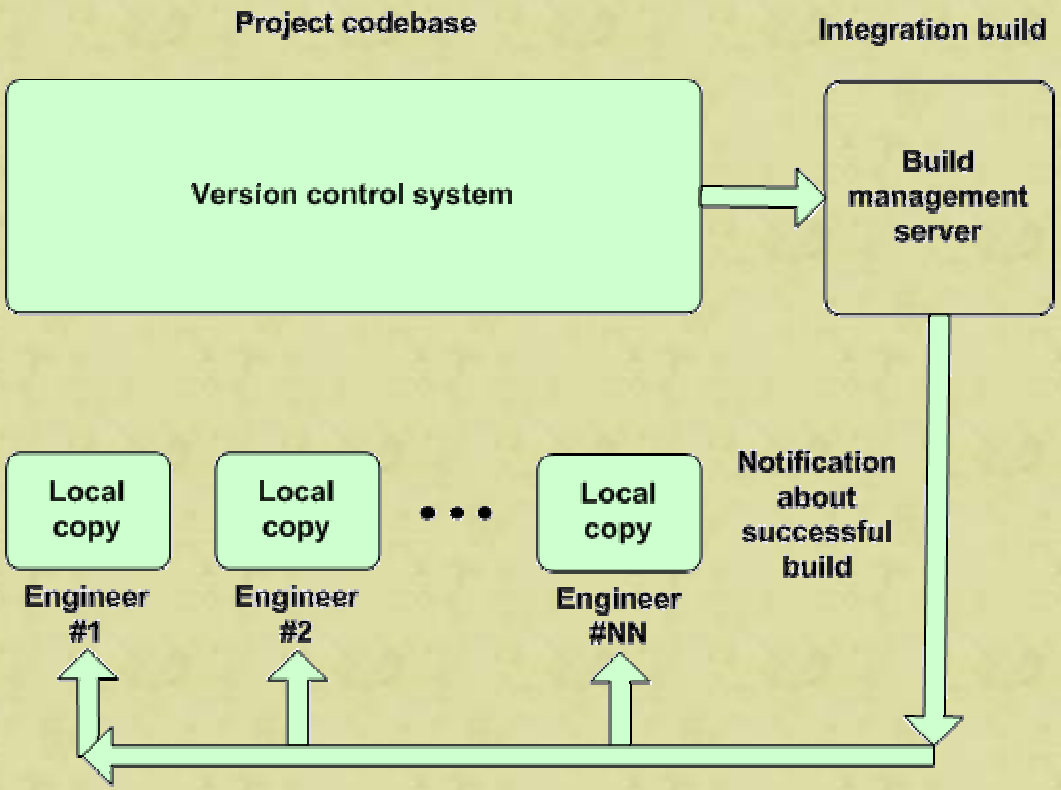
Preventing Spreading Of Build Breakage



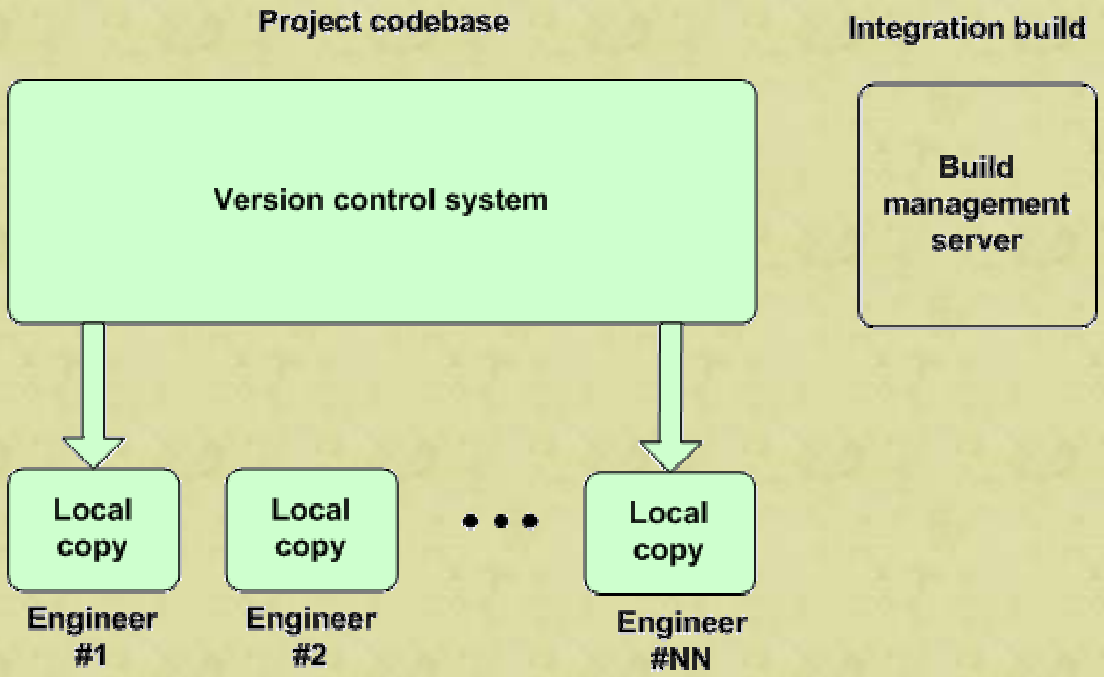
Preventing Spreading Of Build Breakage



Preventing Spreading Of Build Breakage



Preventing Spreading Of Build Breakage



Benefits Of Continuous Integration

Most of the software projects fail because they miss deadlines, cannot stay in budget, or are unable to meet the requirements. Continuous Integration allows to reduce risk of project failure by:

- Reducing slipping of schedule caused by significant time losses
- Reducing and oftentimes eliminating loss of productivity caused by code base breakage
- Increasing team morale and helping to maintain capable engineers on board
- Reducing development time and hitting the market before competitors

Reducing Significant Time Losses

- Single build breakage went into a codeline undetected costs about **one hour of time losses** per a member of engineering team; may double or triple for distributed or offshore team.
- Quick feedback on quality of new changes prevents spreading build breakage by giving build breakers a chance to deliver timely fixes.
- For a team of ten, Continuous Integrations saves 80 hours per months minimum that are lost otherwise

Reducing Loss Of Productivity

- Undetected build breakage is often a subject of domino effect – once codeline is broken, QA, PM, OPS and other organizations run idle.
- Continuous Integration helps stay in budget, saves $8,000 per months for a team of 10;
- This saving doubles when advanced servers like Parabuild are used.

Increasing Team Morale

- There is nothing worse than checking out latest and greatest code from VCS and finding that it doesn't compile. For most it just turns thought flow off.
- When (and if) found, build breakers are hated.
- Constantly broken code base creates a deserved sense of doom.
- Continuous Integration eliminates all of this because build breakage is getting fixed immediately after detected, and may even be unnoticed by the rest of the team.
- A happy team loves developing great code uninterrupted and enjoys sense of smooth process as compared to a doomed sweat-shop. Their resumes are collecting dust.

Hitting The Market Before Competitors

- High quality software gets developed faster than usual because major cause of slipping schedule, missed budget and loss of key people are eliminated.
- We have seen triple reduction of project delivery time.
- Many our customers hide the fact that they use our product because it gives them unfair advantage over competition.

Addressing Challenges

Introducing Continuous Integration, with all benefits it brings and however painless introduction is, may face some challenges. Here they are and ways to address them

1. **Nobody cares**
   - Buy commercial build management software. If nothing is invested, nothing is expected back.
2. **Release Engineering AKA "Mr. Nightly Build" resists**
   - RE has to become an owner of the infrastructure
   - Show that this is a relief, not a burden (lesser chances of daily build breakage or practically unbreakable ones with Parabuild)
   - Get them production-grade tools
3. **Builds are still constantly broken**
   - Educate your team to respond to breakage immediately
   - Follow best practices in the next section

Best Practices

Infrastructure

- Get a **production-grade software build management server**. A build server that need CPR every hour is no good.
- Dedicate the **fastest hardware money can buy**. Faster hardware produces quicker response about quality of changes.
- Make sure your **product can be build from command line**. It should be able to build in unattended, batch mode.
- Build scripts and dependencies are **under version control**. Everyone should be able to reproduce build breakage at given time.

People

- Remember - **you don't break the build if you don't do anything**. It OK to break a build from time to time. It is NOT OK to leave it broken.
- **Always claim build breakage**. People will appreciate knowing that the breakage is being taken care about, and will do same for you.
- **Avoid build breakage patterns**: Five O'clock Checkin, Spoiled Fruit and Small Change. This alone will reduce build breakage by 50%.
- Establish **fun, non-insulting ritual for hailing build breakers**. Buying donuts to the team works well!

Parabuild Server

Parabuild

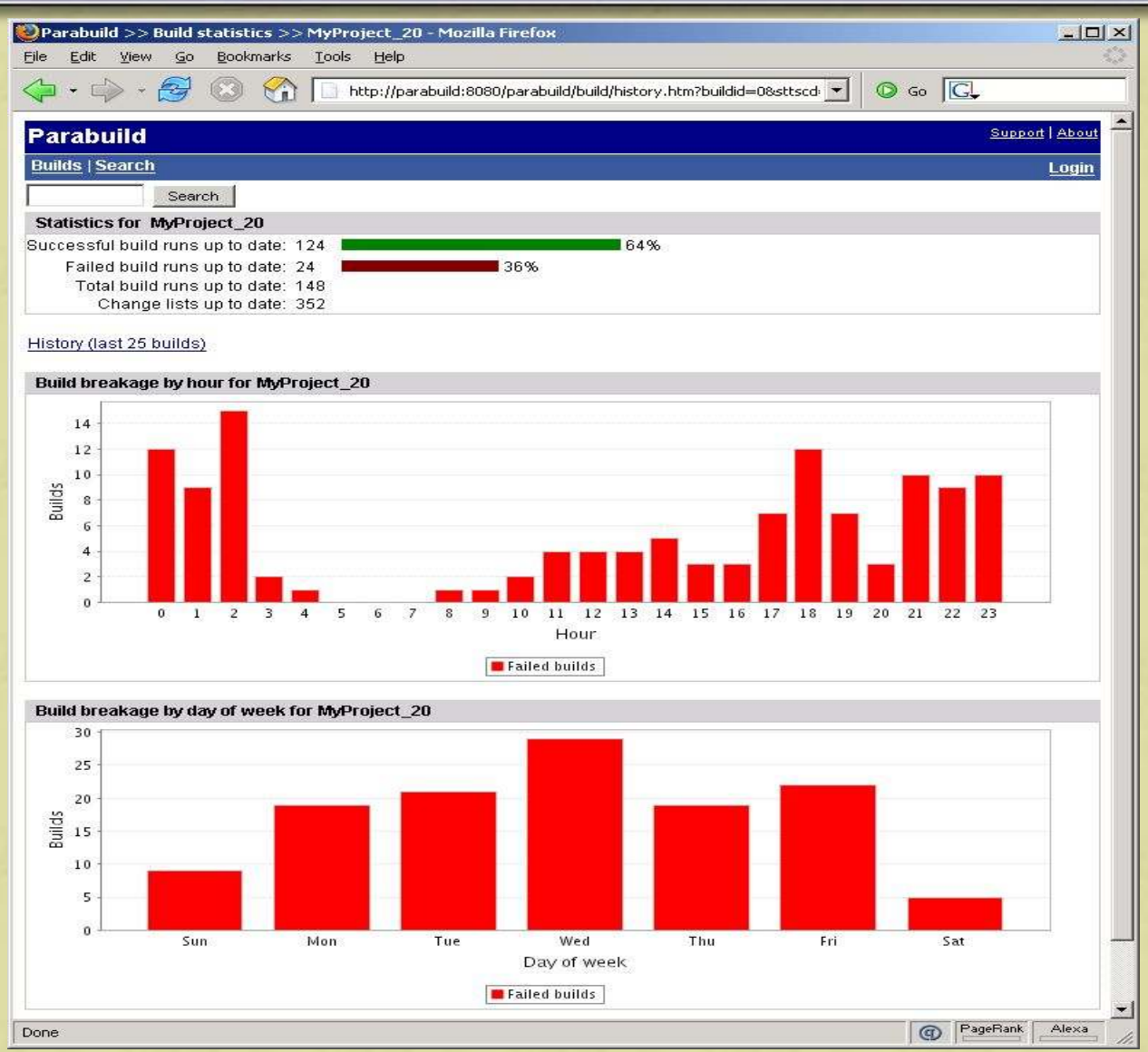
- **Continuous Integration and Software Build Management Server**
  - Developed By Viewtier Systems
  - For software companies who suffer from code base breakage
  - To help them reduce risk of failure of software projects
  - By offering them feedback on changes and providing practically unbreakable daily builds.
- **Key benefits**
  - Reduces risk of failure of software projects
- **Key features**
  - Practically unbreakable daily builds
  - Continuous Integration

Why Parabuild

Parabuild is better:

- **Product is superior**
  - Stable daily builds (no one else does)
  - Build statistics (no one else offers)
  - Release notes (no one else offers)
  - Remote builds (no one else does)
  - Tree minutes installation (others require hours and days)
  - Low to zero administration (others require constant babysitting)
  - Stable as in "Just Works" (others die on a daily basis)
  - ROI: Pays back after the first un-broken daily build (1-5 days)
- **Company is dedicated**
  - We are fully committed
  - We put our well-being on stake, not a couple of hours over weekend

Build Breakage Distribution



Q & A

Thank You

Visit Us At
www . viewtier . com

Slava Imeshev
Viewtier Systems, Inc.
vimeshev@viewtier.com

Parabuild Vs. CruiseControl

	CruiseControl	**Parabuild**
Definition	Toolkit	Build management server
Time to first build run	From a day to a month, depending on knowledge of Java	10 minutes
Web UI	Rudimentary	Extensive, Build management, detailed and, overview statuses, configuration, results, logs, search, statistics
Configuration	Barely documented XML file, knowledge of Java and application server required, multiple roundtrips to mailing list required.	Simple WEB UI
Build management	Rudimentary Continuous integration builds only	Extensive Continuous integration, unbreakable daily builds, build results and log archive, remote multiplatform builds
Security	None	Group-based
Administration	Constant babysitting is required	Zero to none, fire-and-forget stile
Cost	Free as in "free lunch"	Parabuild Std - $1,000, Pro - $2,400
Summary	Fun-to-play toy (if you know buttons)	Production-grade software build management server