Software Production Management
Agile/Continuous Integration

Backend infrastructure to realize Agile software development methodologies

November 18, 2008

Jeff Price

Account Manager

Dave Finnegan

Systems Engineer

www.electric-cloud.com

What am I going to tell you today?

- **Four practical steps to support Agile / Continuous Integration**
- **Impacts on the Build/Integration team**
- **Best practices you can adopt *right now***
- **Commercially-available tools to overcome these impacts**

Practical steps for Agile / CI

- One : Enable developers to build small pieces of functionality
- Two: Automate the full integration build
- Three: Perform an integration build for each SCM commit
- Four: Minimize broken builds and maximize integrated testing of new features

Impacts on Build / Integration

- **One : Enable developers to build small pieces of functionality**
  - Problem : Developers work on isolated components; Integration becomes a trouble spot
- **Two: Automate the full integration build**
- **Three: Perform an integration build for each SCM commit**
- **Four: Minimize broken builds and maximize integrated testing of new features**

Impacts on Build / Integration

- **One : Enable developers to build small pieces of functionality**
  - Problem : Developers work on isolated components; Integration becomes a trouble spot
- **Two: Automate the full integration build**
  - Problem : DIY build scripts are fragile and expensive
- **Three: Perform an integration build for each SCM commit**
- **Four: Minimize broken builds and maximize integrated testing of new features**

Impacts on Build / Integration

- **One : Enable developers to build small pieces of functionality**
  - Problem : Developers work on isolated components; Integration becomes a trouble spot
- **Two: Automate the full integration build**
  - Problem : DIY build scripts are fragile and expensive
- **Three: Perform an integration build for each SCM commit**
  - Problem : Number of builds and total build time explodes
- **Four: Minimize broken builds and maximize integrated testing of new features**

Impacts on Build / Integration

- **One : Enable developers to build small pieces of functionality**
  - Problem : Developers work on isolated components; Integration becomes a trouble spot
- **Two: Automate the full integration build**
  - Problem : DIY build scripts are fragile and expensive
- **Three: Perform an integration build for each SCM commit**
  - Problem : Number of builds and total build time explodes
- **Four: Minimize broken builds and maximize integrated testing of new features**
  - Problem : keeping the main line buildable and clean

Overcoming Impacts

- **Problem : Developers work on isolated components; Integration becomes a trouble spot**
- **Problem : DIY build scripts are fragile and expensive**
- **Problem : Number of builds and total build time explodes**
- **Problem : keeping the main line buildable and clean**

Give the developers access to the build!

- **Put full builds in the hands of developers**
- **Set up an internal web page that allows a developer to fire off a build**
- **The developer should be able to select:**
  - Which branch to build…
  - … or the location of his personal source tree
  - Whether to run unit tests or not
  - Select platforms
- **The build runs and the engineer gets an email with the result**

The Problems for the Build team…

- Problem : Developers work on isolated components; Integration becomes a trouble spot
- **Problem : DIY build scripts are fragile and expensive**
- Problem : Number of builds and total build time explodes
- Problem : keeping the main line buildable and clean

Script your build the way you write code!

- **Design generic, modular build elements which can be reused**
- **Make use of parameterization**
- **Develop or apply Continuous Integration systems (start with `cron`)**
- **Standardize on a single set of build tools**
  - Pick a scripting language and stick with it
  - Enforce SCM for build scripts
  - Code review for build scripting
- **Know when to stop fiddling**

The Problems for the Build team…

- Problem : Developers work on isolated components; Integration becomes a trouble spot
- Problem : DIY build scripts are fragile and expensive
- **Problem : Number of builds and total build time explodes**
- Problem : keeping the main line buildable and clean

Make the build fast, automatic, and visible

- **Achieve fast builds**
  - That means better build structure
  - Fast build hardware, special software for parallel builds
  - `gmake -j` may get you 2-3x improvements
- **Schedule builds through automation**
  - `cron` for CI builds
  - Web front-ends and request systems for developer builds
- **Investigate better build reporting**
  - Collect and publish build result information
    - Build status
    - Collect metrics: errors, warnings, tests, etc
    - Include changelogs
  - Build tracking repository

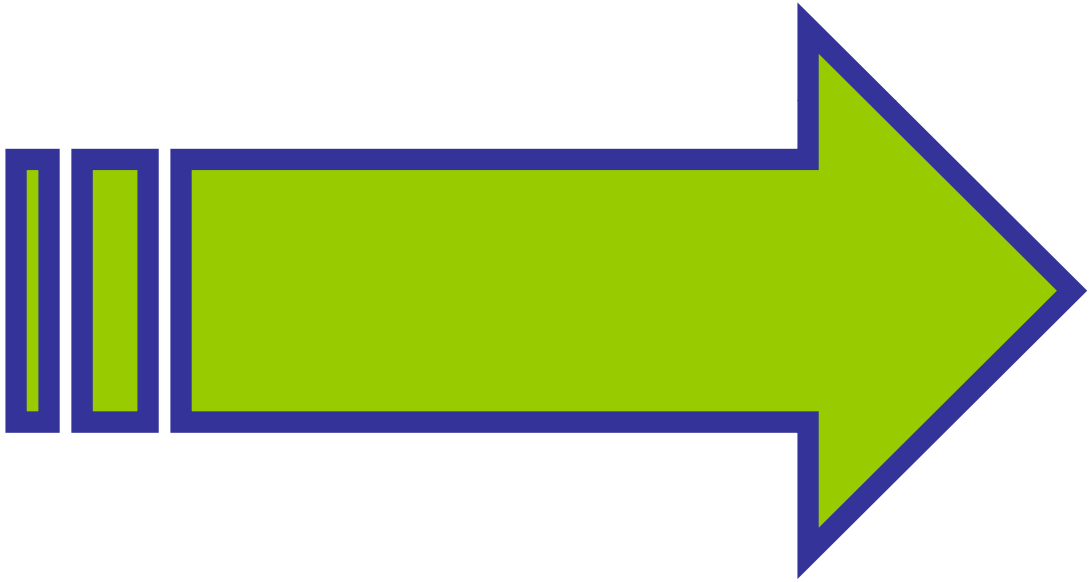
The Problems for the Build team…

- Problem : Developers work on isolated components; Integration becomes a trouble spot
- Problem : DIY build scripts are fragile and expensive
- Problem : Number of builds and total build time explodes
- **Problem : keeping the main line buildable and clean**

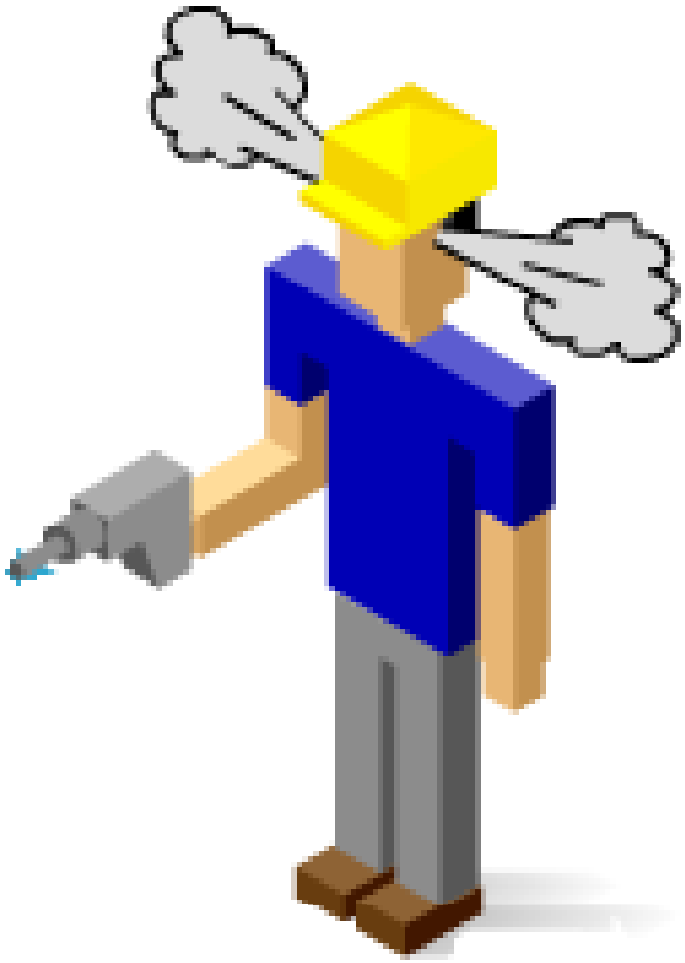
Guard the mainline and create policies

- **Private Branches**
  - Give developers private branches, or multi-stage CI streams
- **Integration Builds**
  - Require developers to run integration builds before merging
- **SCM Triggers**
  - Use SCM hooks to fire an integration build for every checkin
- **Automate Checkin Reverts**
  - Integrate SCM and CI to revert checkins which break the build
- **Communicate Status**
  - Use notification, dashboards, or lava lamps to communicate mainline status

Going Beyond DIY

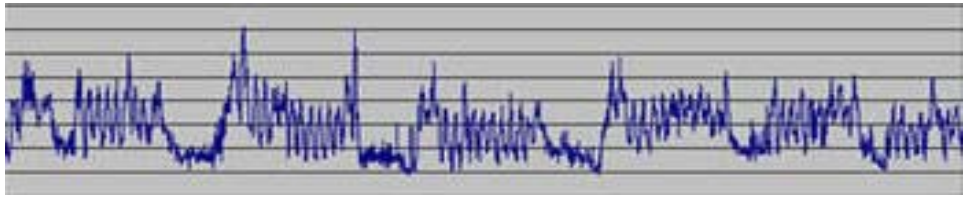


The Build Manager

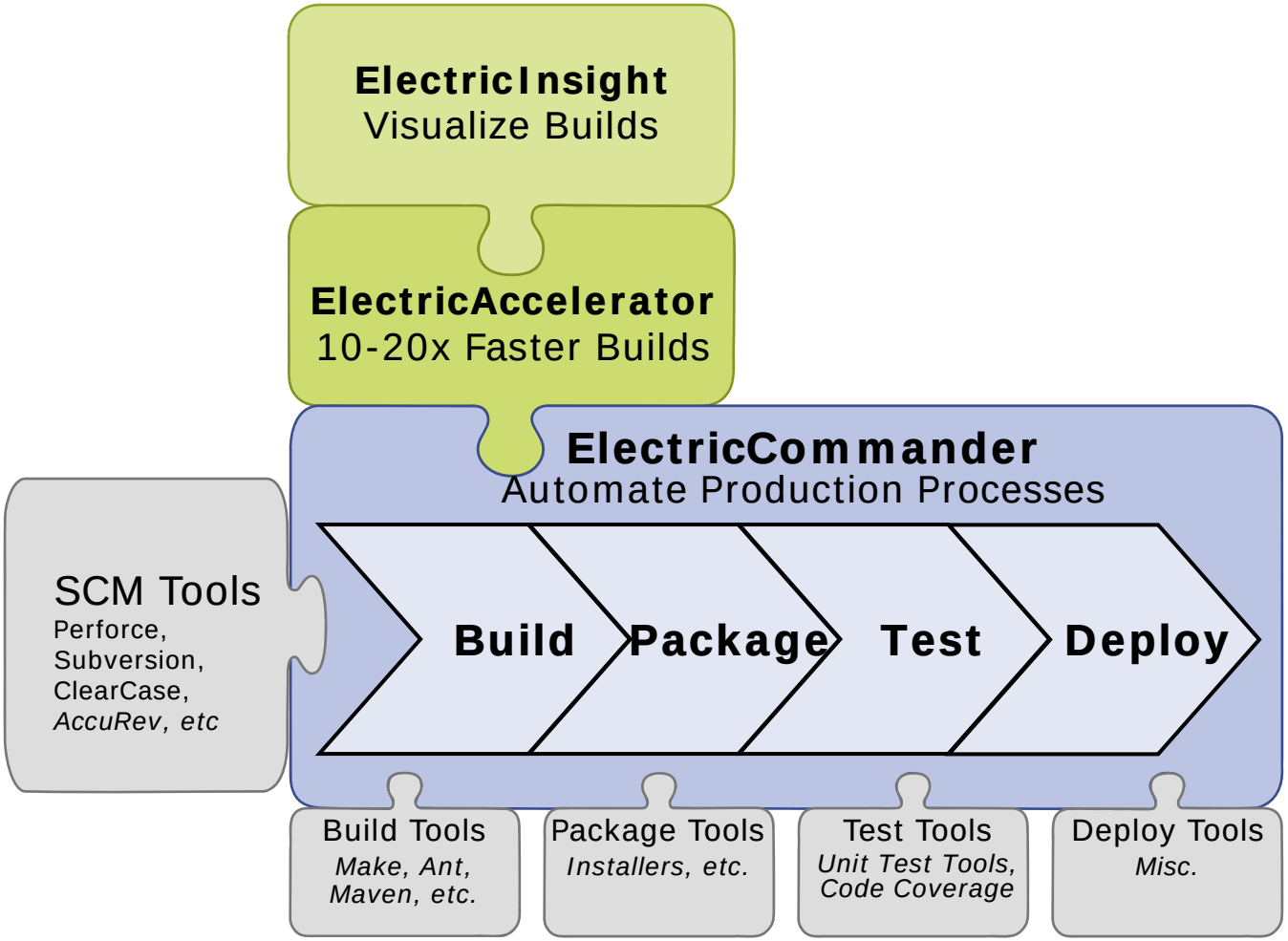


Improving on in-house

- **I need fast builds**
  - That means better build structure
  - Fast build machines, special software for parallel builds
  - `gmake -j` doesn't give the speedup up I need
- **I'm doing 200 builds a day, I need better tools**
  - `cron` just won't cut it
  - I need resource management tools
  - I need developers to build in the production environment
- **I need reporting tools**
  - Everyone wants live build feedback
  - Perl + Excel isn't enough
- **I need this because builds are now the heartbeat of software engineering**



Electric Cloud SPM Solutions



Improving on in-house

- **I need fast builds**
  - That means better build structure
  - Fast build machines, special software for parallel builds
  - `gmake -j` doesn't give the speedup up I need
- **I'm doing 200 builds a day, I need better tools**
  - `cron` just won't cut it
  - I need real management tools
  - I need developers to build in the production environment
- **I need reporting tools**
  - Everyone wants live build feedback
  - Perl + Excel isn't enough
- **I need this because builds are now the heartbeat of software engineering**

ElectricAccelerator

- **Problem it solves: Build cycle is too long**
  - Very long builds
  - Complex matrix of builds and targets
- **Benefits:**
  - 10-20x faster builds
- **How it works:**
  - Run builds in parallel using a cluster of standard hardware
  - Automatically discover and track dependencies
  - Plug-compatible replacement for GNU make, Microsoft nmake, Visual Studio, Ant
  - No process changes required
- **Works with ElectricCommander or stand-alone**

Based on proven, patented dependency management technology

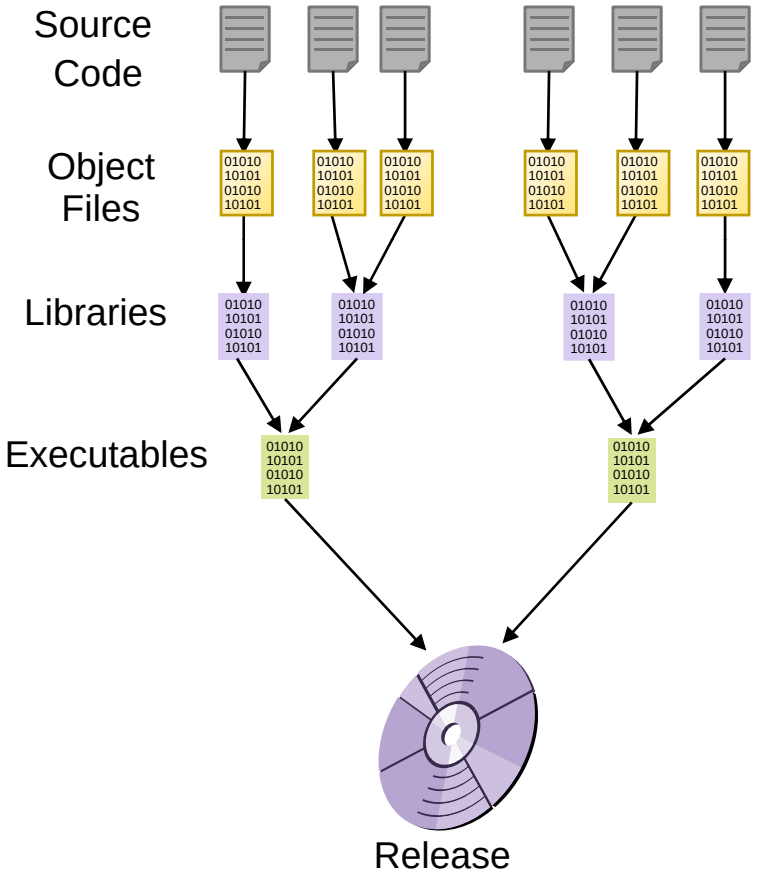
How Do You Get There?

Requirements for real build acceleration:

- ✓ **Fast and accurate builds via parallelism**
  - Requires understanding of dependencies
- ✓ **Robust and scalable**
  - Support high build volume and avoid resource failure
- ✓ **Work within existing environment**
  - Must not require costly re-write
- ✓ **Visibility and reporting**
  - Provide status reports and tools for diagnostics

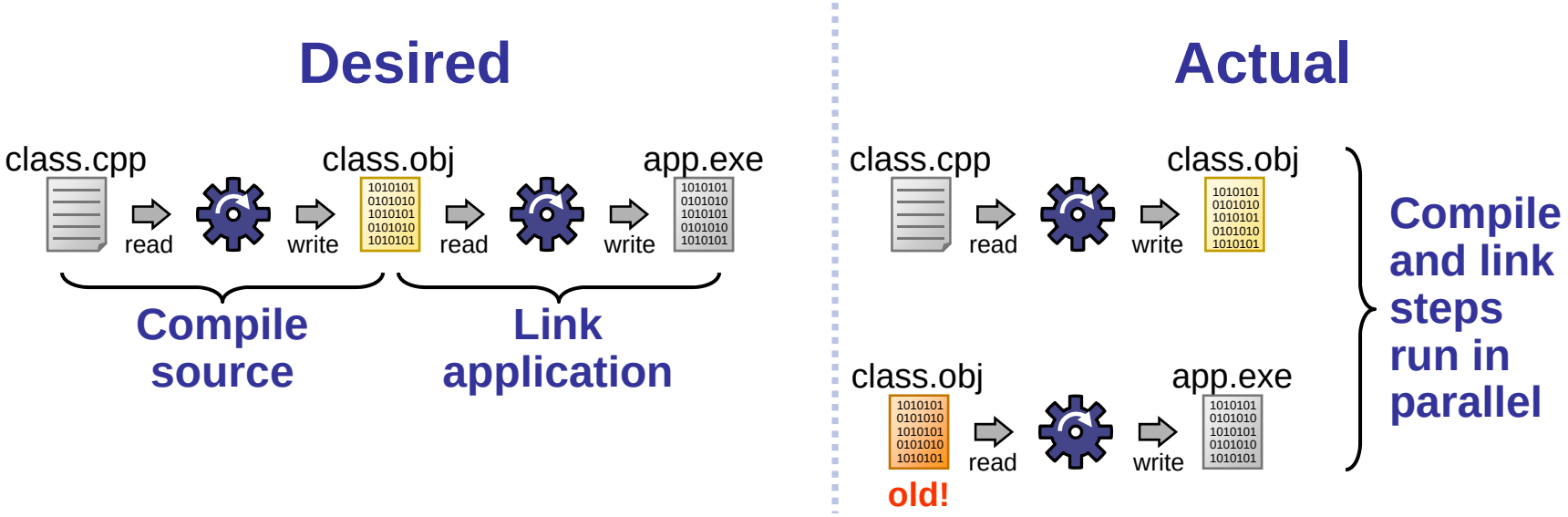
Managing Dependencies for Parallel Builds

- **Builds have inherent parallelism**
- **Should be able run pieces concurrently**
  - Large SMP Machines (gmake –j)
  - Distributed builds (distcc)
- **Other attempts yield small results due to dependencies:**
  - Incomplete or unknown
  - Can't be expressed between Makefiles
- **Result: broken builds**



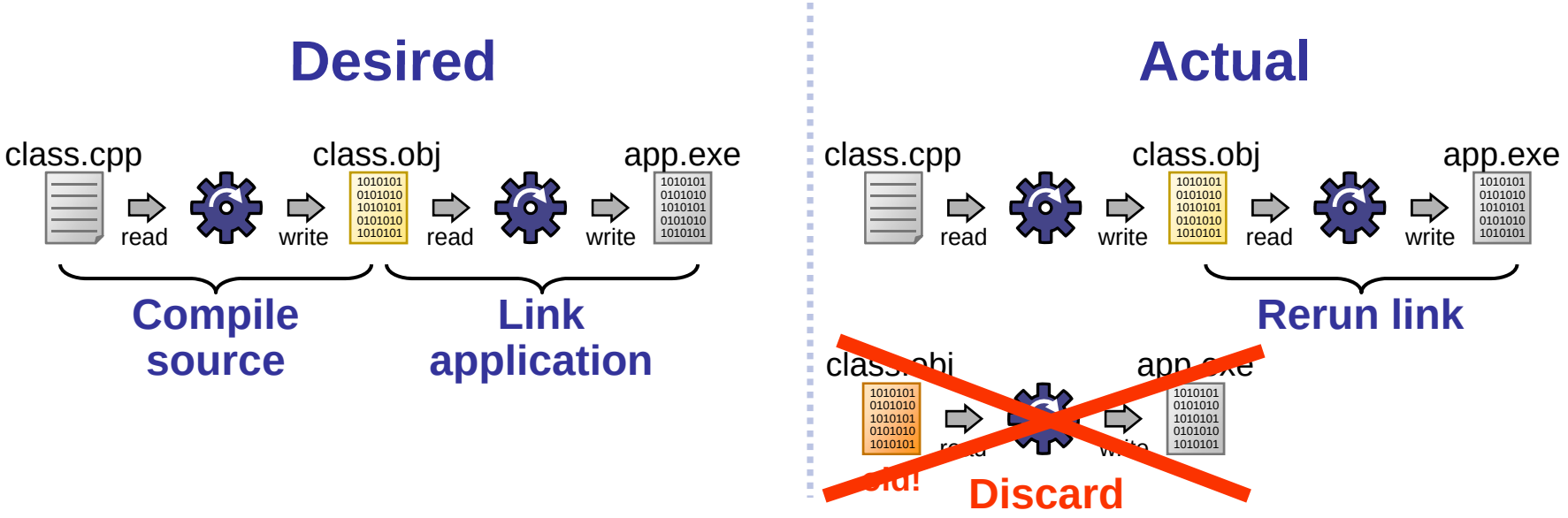
Solution for Fast, Accurate Builds

- **Deduce dependencies on-the-fly:**
  - Watch all file accesses: these indicate dependencies
  - Automatically detect out-of-order steps

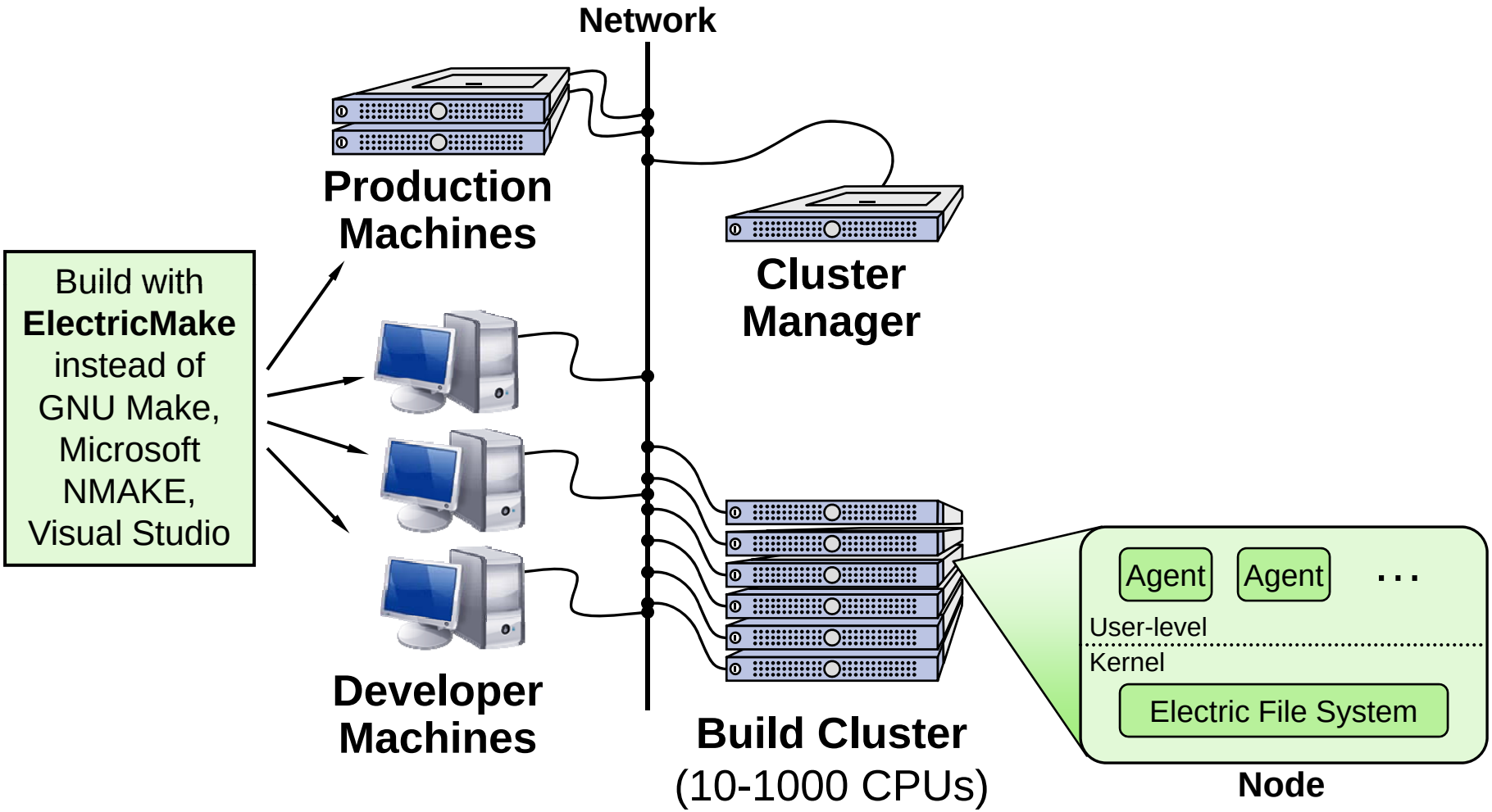


Solution for Fast, Accurate Builds

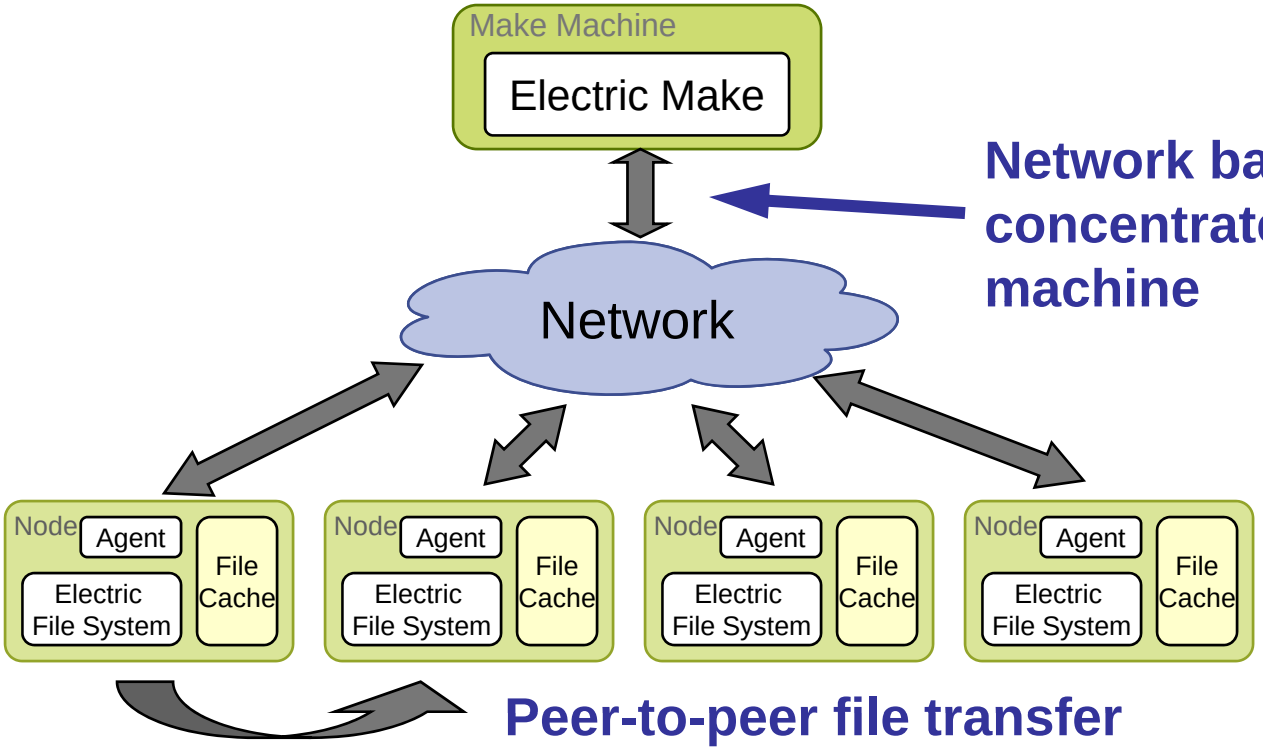
- **Deduce dependencies on-the-fly:**
  - Watch all file accesses: these indicate dependencies
  - Automatically detect and correct out-of-order steps
  - Save discovered dependencies for future builds
  - Result: high concurrency is safe



ElectricAccelerator Architecture

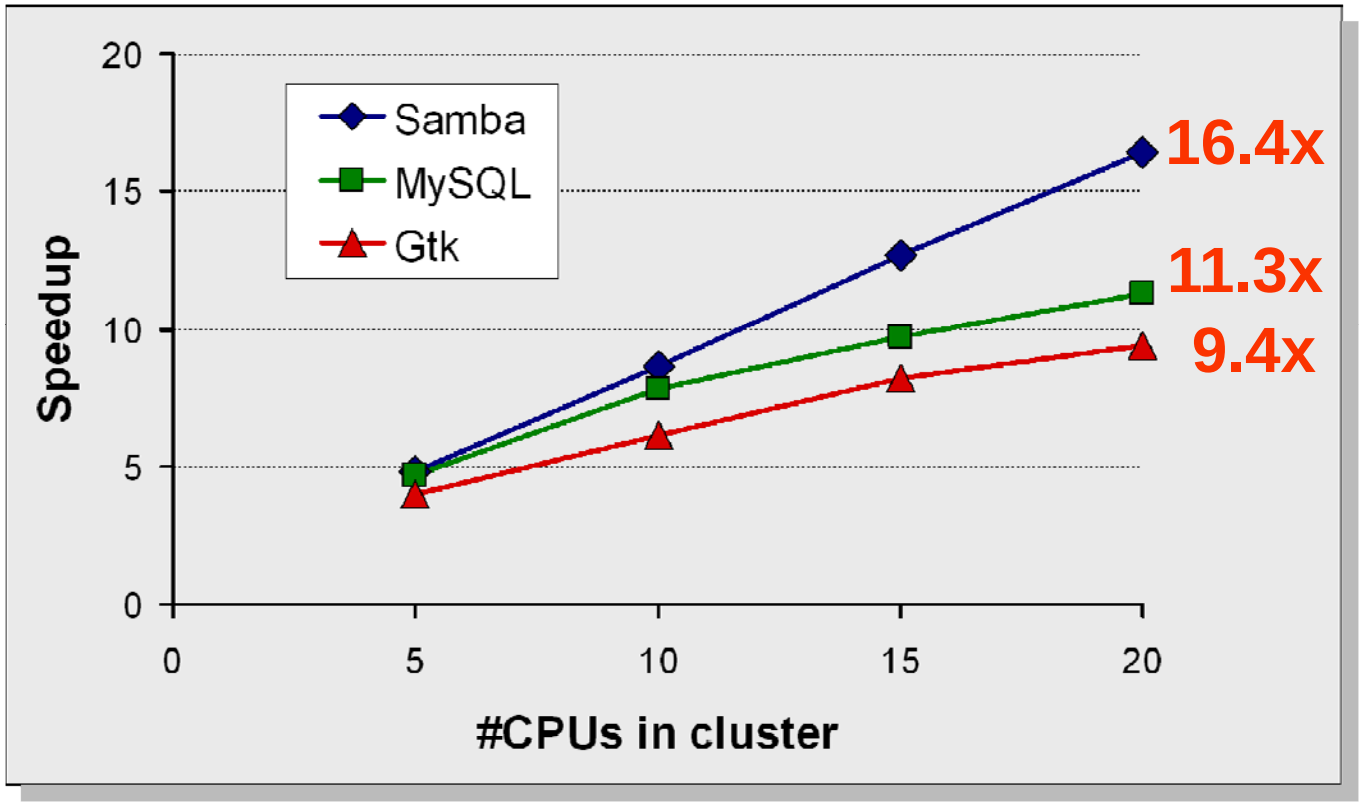


File Caching for Network Optimization



- File caching on nodes **cuts 90-95% of network traffic**
- Just-in-time compression **cuts traffic 2.5-3x**
- P2P file transfers **offload 20-25% of outbound traffic**

Linear Scalability



	Local	20 CPUs	Speedup
Samba	16m	58s	16.4x
MySQL	23m	124s	11.3x
Gtk	15m	95s	9.4x

Works Seamlessly in Environment

- **Works seamlessly with existing infrastructure**
  - GNU make, Nmake, Visual Studio
  - No learning a new interface/tool
  - Use existing Makefiles and scripts
  - Identical log files
- **Works with grid system**
  - Use dedicated cluster of servers or central data center
  - Fully integrated with the Platform LSF grid computing solution
  - Provide build resources just when needed

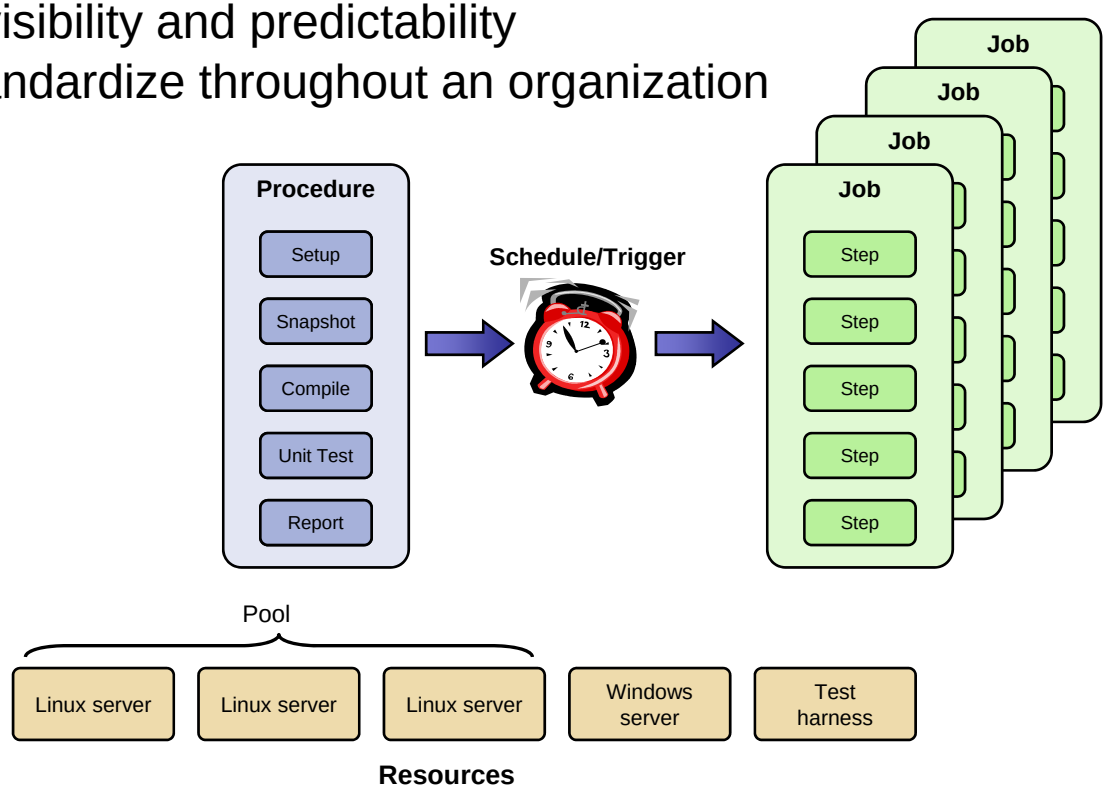
Minimal changes required for end users

Improving on in-house

- I need fast builds
  - That means better build structure
  - Fast build machines, special software for parallel builds
  - `gmake -j` doesn't give the speedup up I need
- **I'm doing 200 builds a day, I need better tools**
  - `cron` just won't cut it
  - I need real management tools
  - I need developers to build in the production environment
- **I need reporting tools**
  - Everyone wants live build feedback
  - Perl + Excel isn't enough
- **I need this because builds are now the heartbeat of software engineering**

ElectricCommander

- **Problem it solves: Manual, inflexible build and test infrastructure**
- **Web-based platform for managing distributed processes**
  - Faster, more efficient build and release processes
  - More efficient resource utilization
  - Better visibility and predictability
  - Can standardize throughout an organization

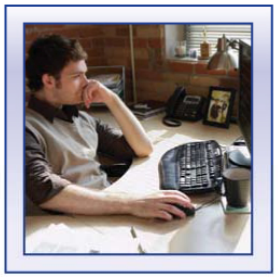


Managing Global Teams

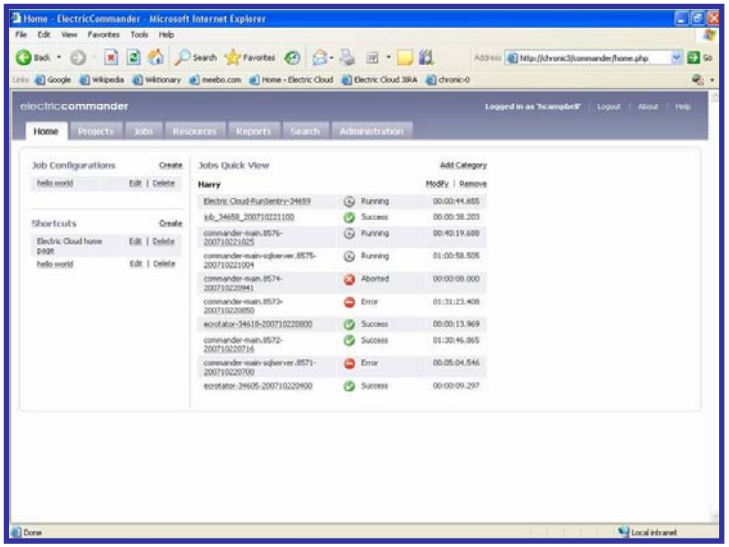
SW DEVELOPERS
Munich

ENGINEERING MGR
Boston

BUILD TEAM
San Francisco



SW DEVELOPERS
San Francisco



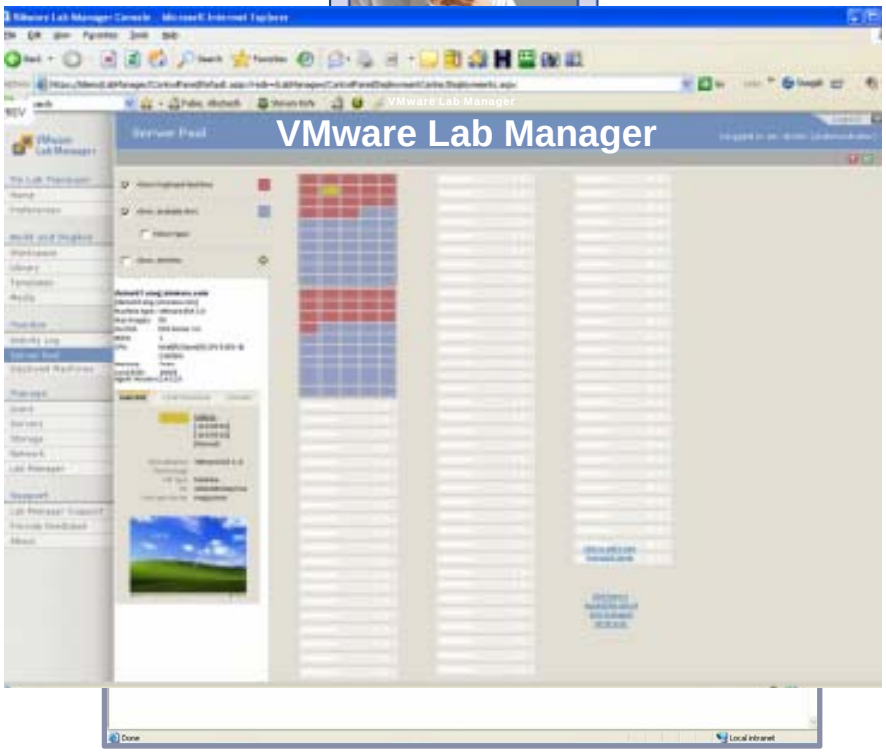
OUTSOURCED QA
Bangalore

Specific Access and Permissions Based on Role Anywhere in the World

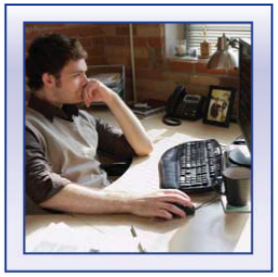
Slide 32

Managing Resources

SW DEVELOPERS

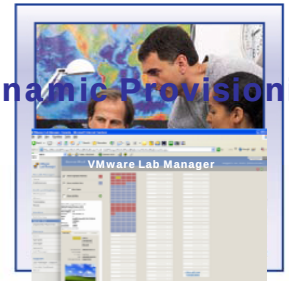


BUILD TEAM



SW DEVELOPERS

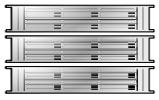
Dynamic Provisioning



OUTSOURCED QA

Build Servers

Test Servers

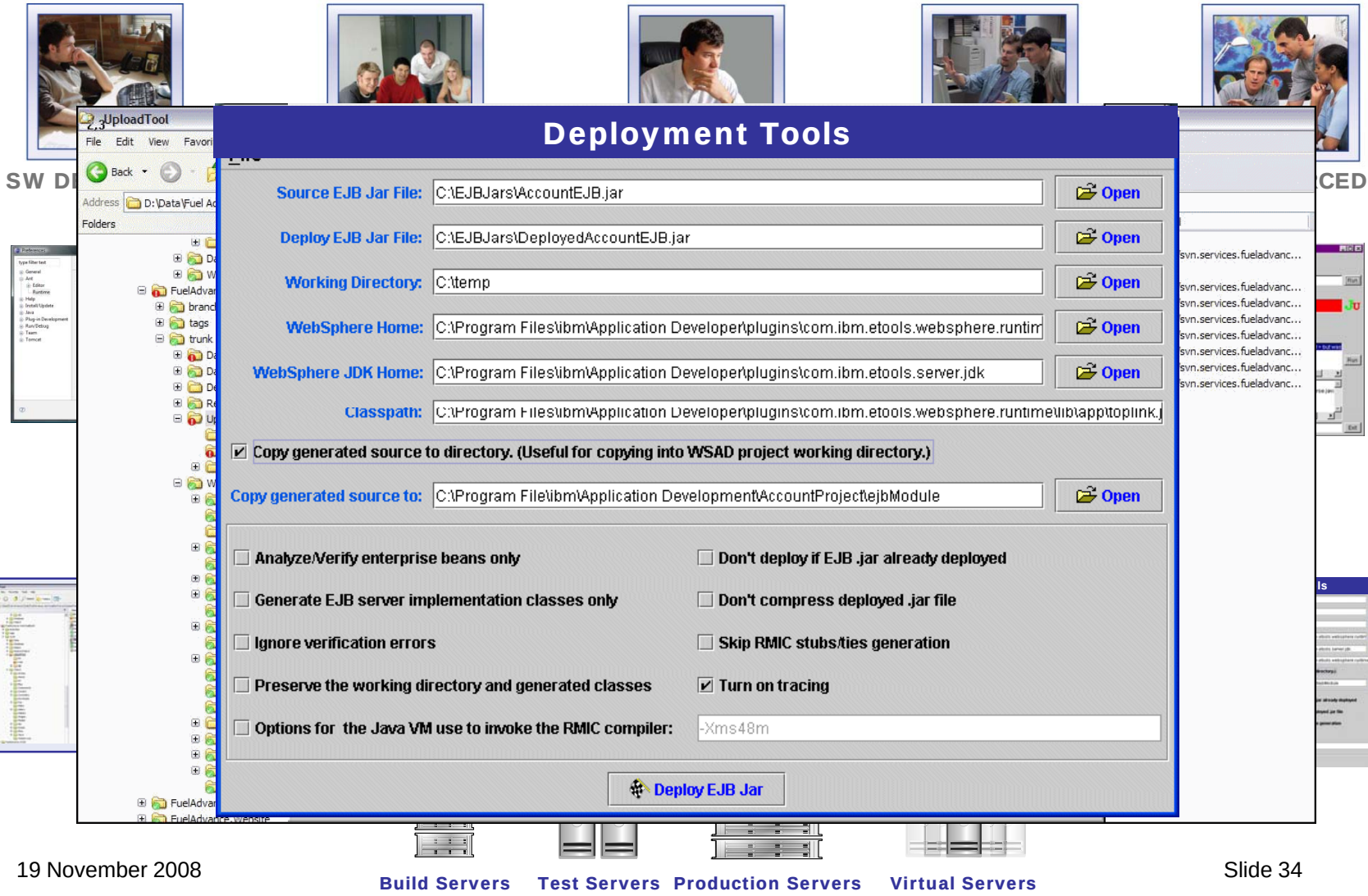


Production Servers

Virtual Servers

Slide 33

Integrate Tools and Processes



Tie it all Together

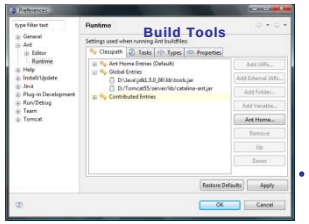
SW DEVELOPERS

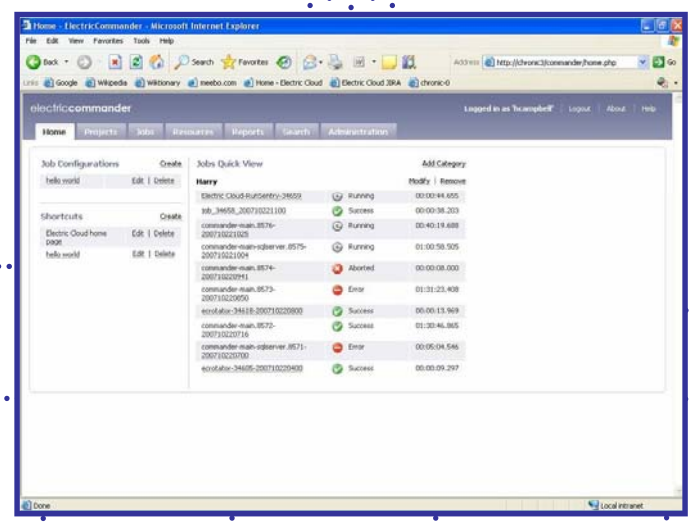
SW DEVELOPERS

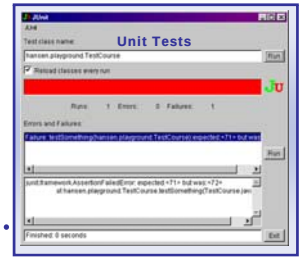
ENGINEERING MGR

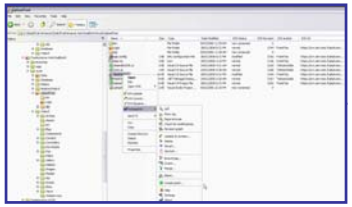
BUILD TEAM

OUTSOURCED QA



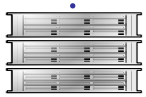






Build Servers

Test Servers

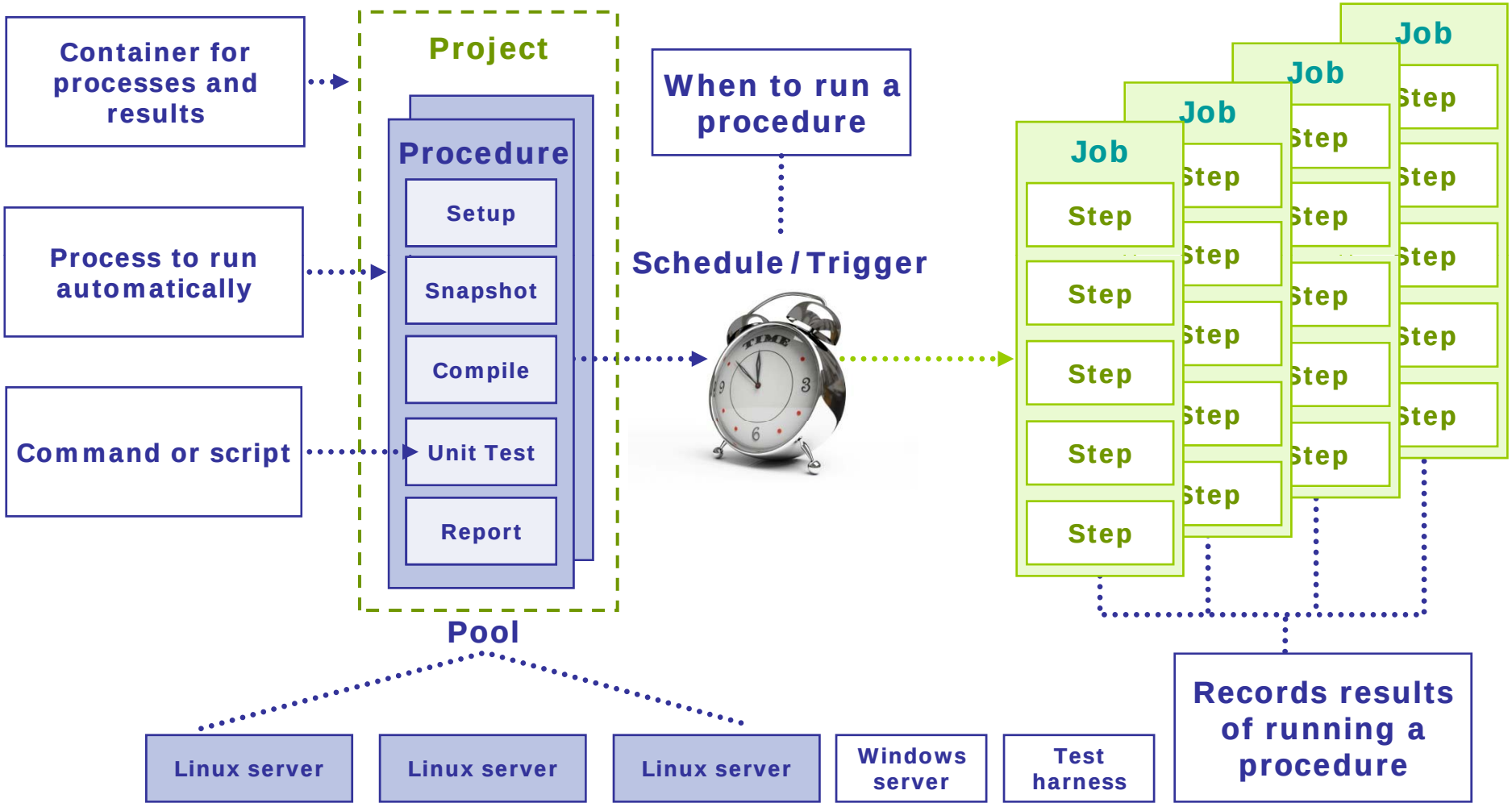


Production Servers

Virtual Servers

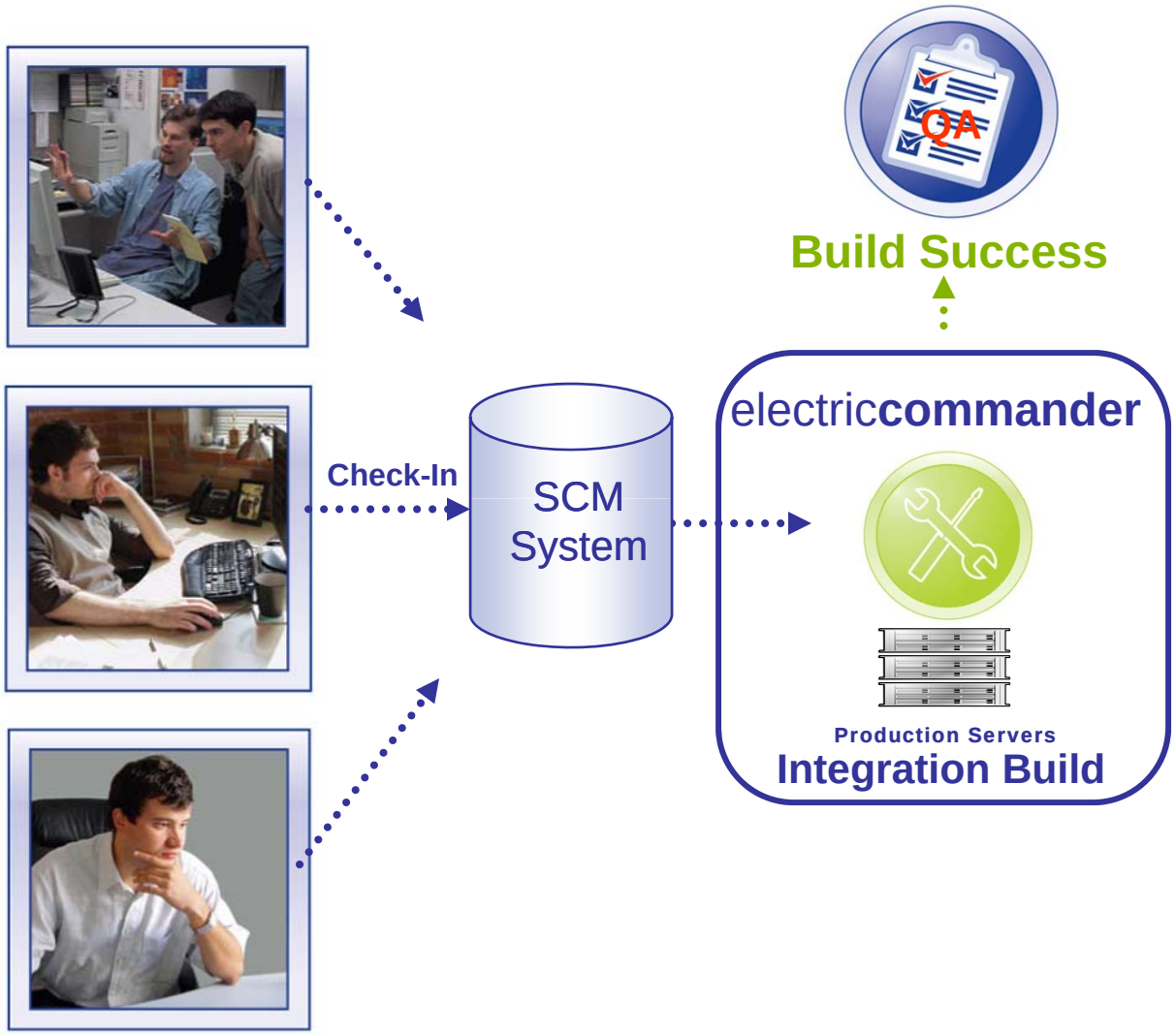
Slide 35

ElectricCommander Basics



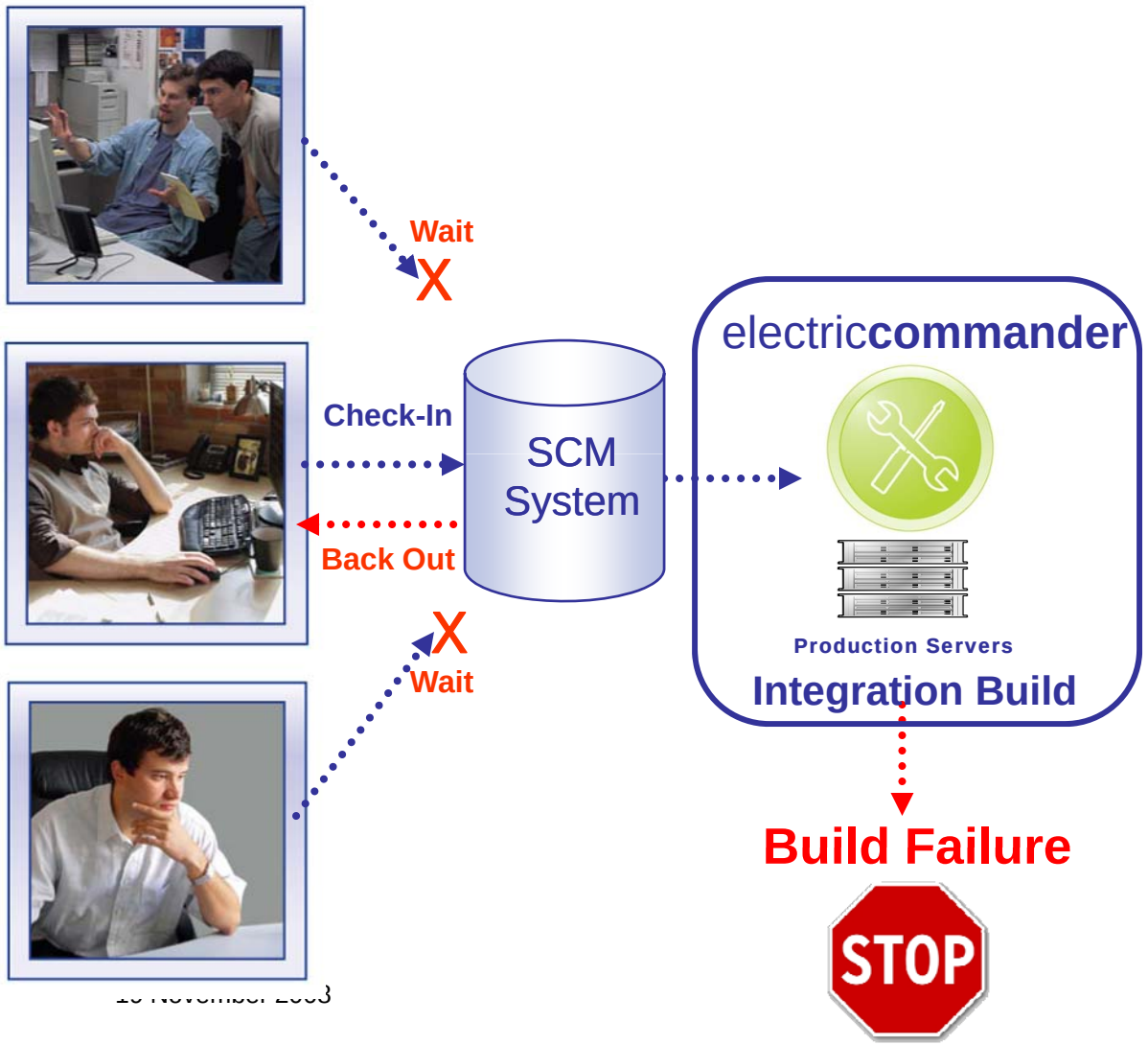
Slide 36

Continuous Integration



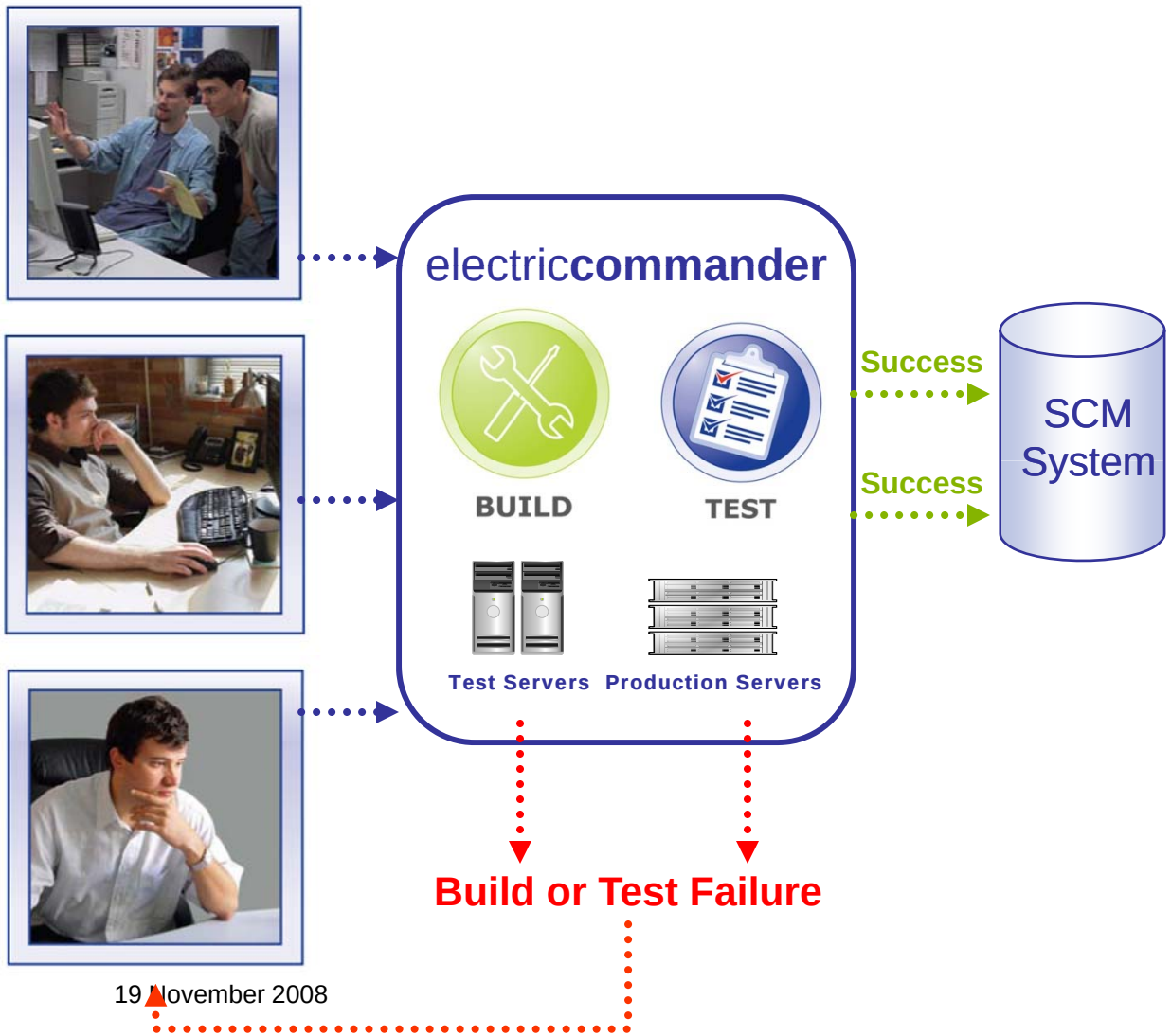
- Developer runs local build and smoke tests
- Developer checks tested code into SCM system
- Integration build at frequent intervals or upon check-in

Frequent Problem: Continuously Broken Builds



- 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

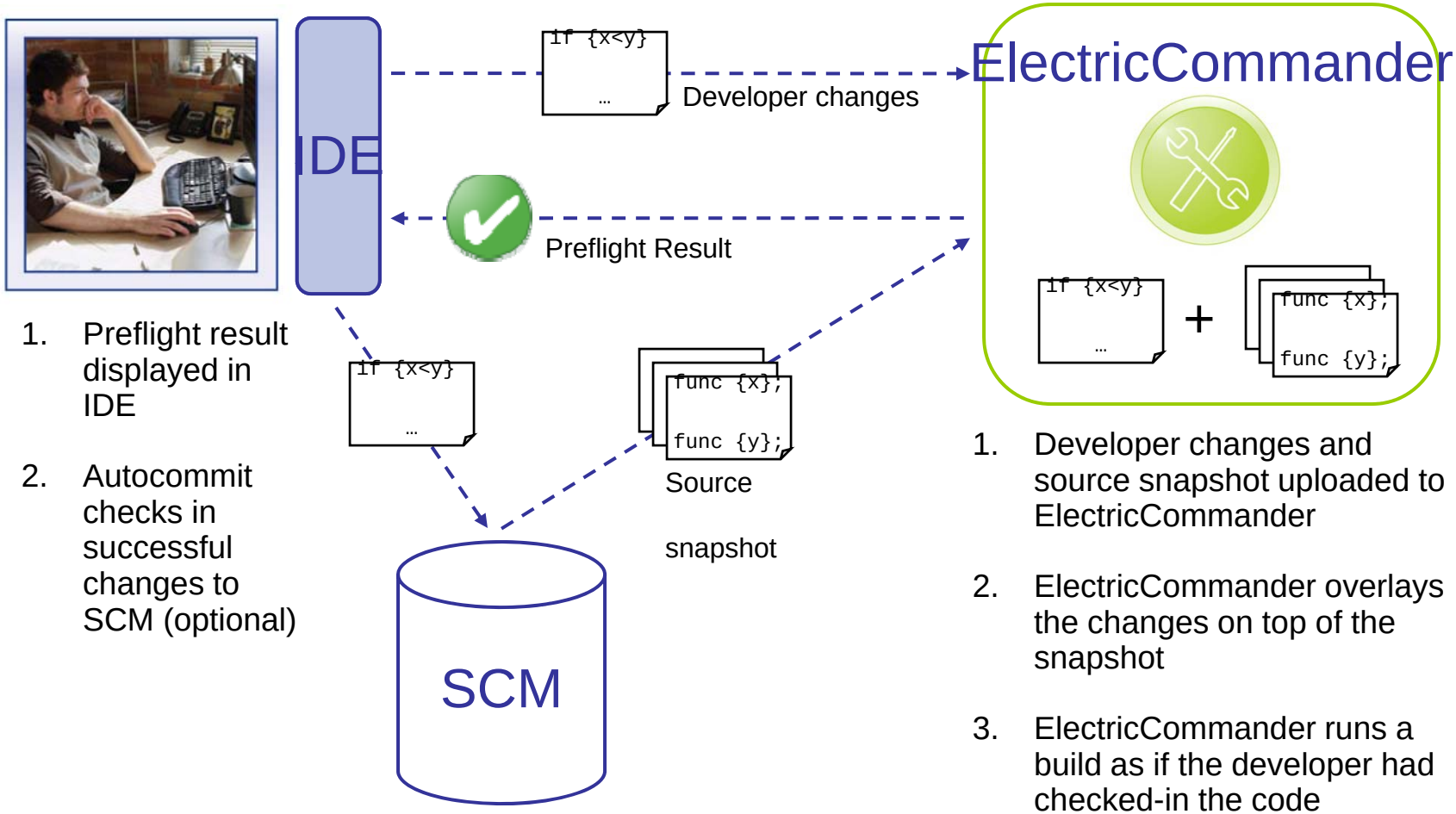
Solution: Pre-Flight Builds and Tests



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

Preflight Workflow

1. Developer invokes preflight through Visual Studio, Eclipse or command line



1. Preflight result displayed in IDE
2. Autocommit checks in successful changes to SCM (optional)

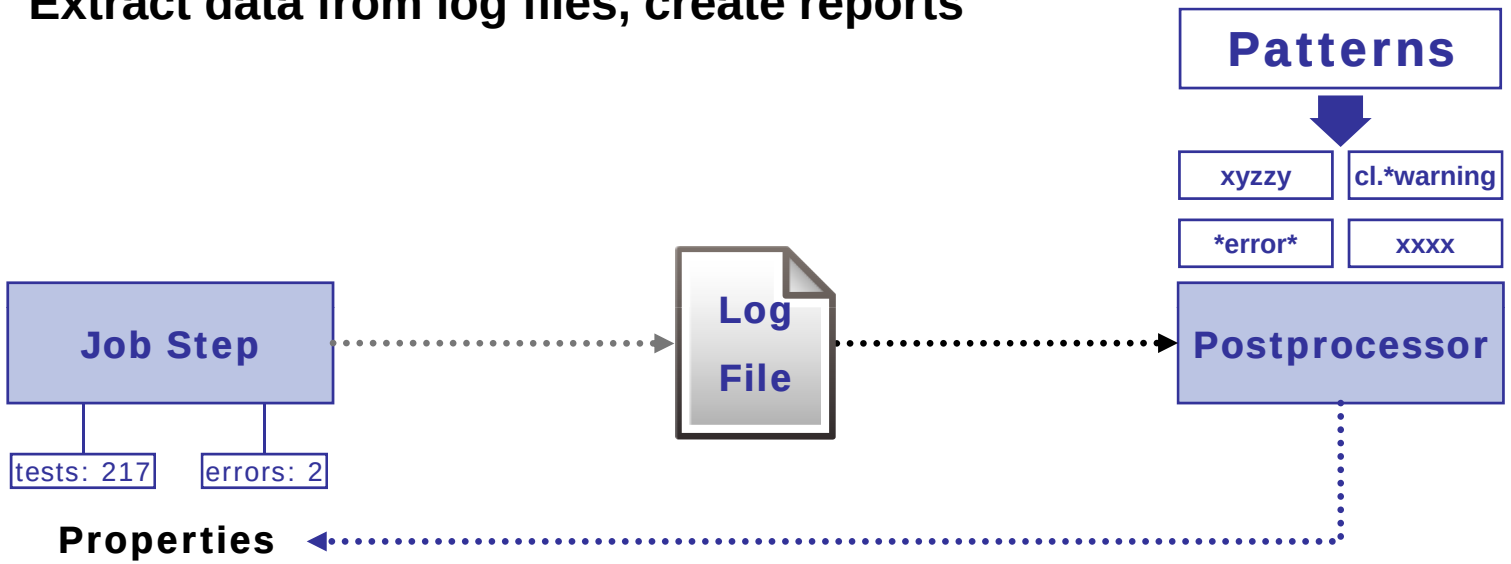
1. Developer changes and source snapshot uploaded to ElectricCommander
2. ElectricCommander overlays the changes on top of the snapshot
3. ElectricCommander runs a build as if the developer had checked-in the code

Improving on in-house

- I need fast builds
  - That means better build structure
  - Fast build machines, special software for parallel builds
  - `gmake -j` doesn't give the speedup up I need
- I'm doing 200 builds a day, I need better tools
  - `cron` just won't cut it
  - I need real management tools
  - I need developers to build in the production environment
- **I need reporting tools**
  - Everyone wants live build feedback
  - Perl + Excel isn't enough
- I need this because builds are now the heartbeat of software engineering

Pinpoint Data for Reporting

- Extract data from log files, create reports



Problem:

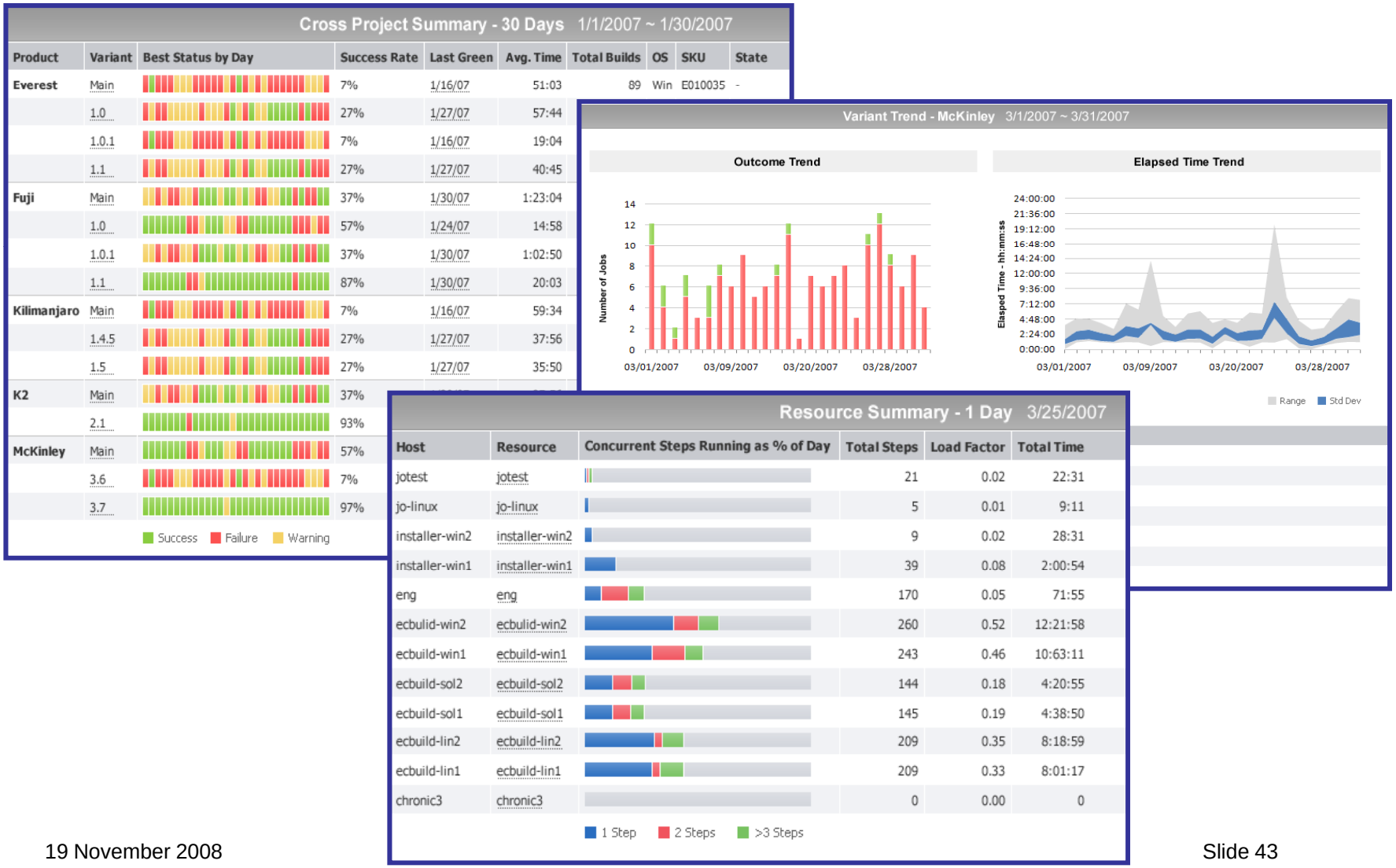
- Large, complex log files
- Only a small amount of data matters
- Log files get deleted

ElectricCommander solution:

- Extract key data from log:
  - Tests executed
  - Errors
  - Diagnostic messages
- Save in database

Slide 42

Visibility: Management Reports



Improving on in-house

- I need fast builds
  - That means better build structure
  - Fast build machines, special software for parallel builds
  - `gmake -j` doesn't give the speedup up I need
- I'm doing 200 builds a day, I need better tools
  - `cron` just won't cut it
  - I need real management tools
  - I need developers to build in the production environment
- I need reporting tools
  - Everyone wants live build feedback
  - Perl + Excel isn't enough
- **I need this because builds are now the heartbeat of software engineering**

Builds are too expensive to ignore!

Multi-Platform Gaming Company

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

Solution: ElectricAccelerator + ElectricCommander

- Build acceleration led to more frequent builds and tests
  - Build time from 2.5 hours to 12 minutes
- Developer “preflight” builds reduced broken builds by 90%
  - Preflight build and test run on 3 platforms prior to check-in
  - 10% due to developers not doing preflight

Customer Examples

Networking	Semiconductor	Cellular	ISV	Other
Cisco	Broadcom	Motorola	Intuit	VMware
tellabs	Texas Instruments	LG	Wind River	Expedia.com
Ericsson	Conexant	UIQ Technology	Fanfare	Thales
Force10	Magnum Semiconductor	Kyocera	GoldenGate	BioWare Corp
Redback Networks	freescale semiconductor	Samsung	PTC	Siemens
Agilent Technologies	Xilinx	Sony Ericsson	cadence	Computer Systems, Inc. Mercury
Brocade	Qualcomm		PayPal	Delphi
				CAT

Slide 46

What we have covered today

- Four practical steps which are required to support Agile
- Impacts of these steps on the Build/Integration team
- Best practices you can adopt *right now* to handle these impacts
- Commercially-available tools which overcome these impacts

Thanks for attending

For more information:

- Visit our website: www.electric-cloud.com
- E-mail: info@electric-cloud.com
- Phone: 408.419.4300

Questions and Answers

Backend infrastructure to realize Agile software development methodologies

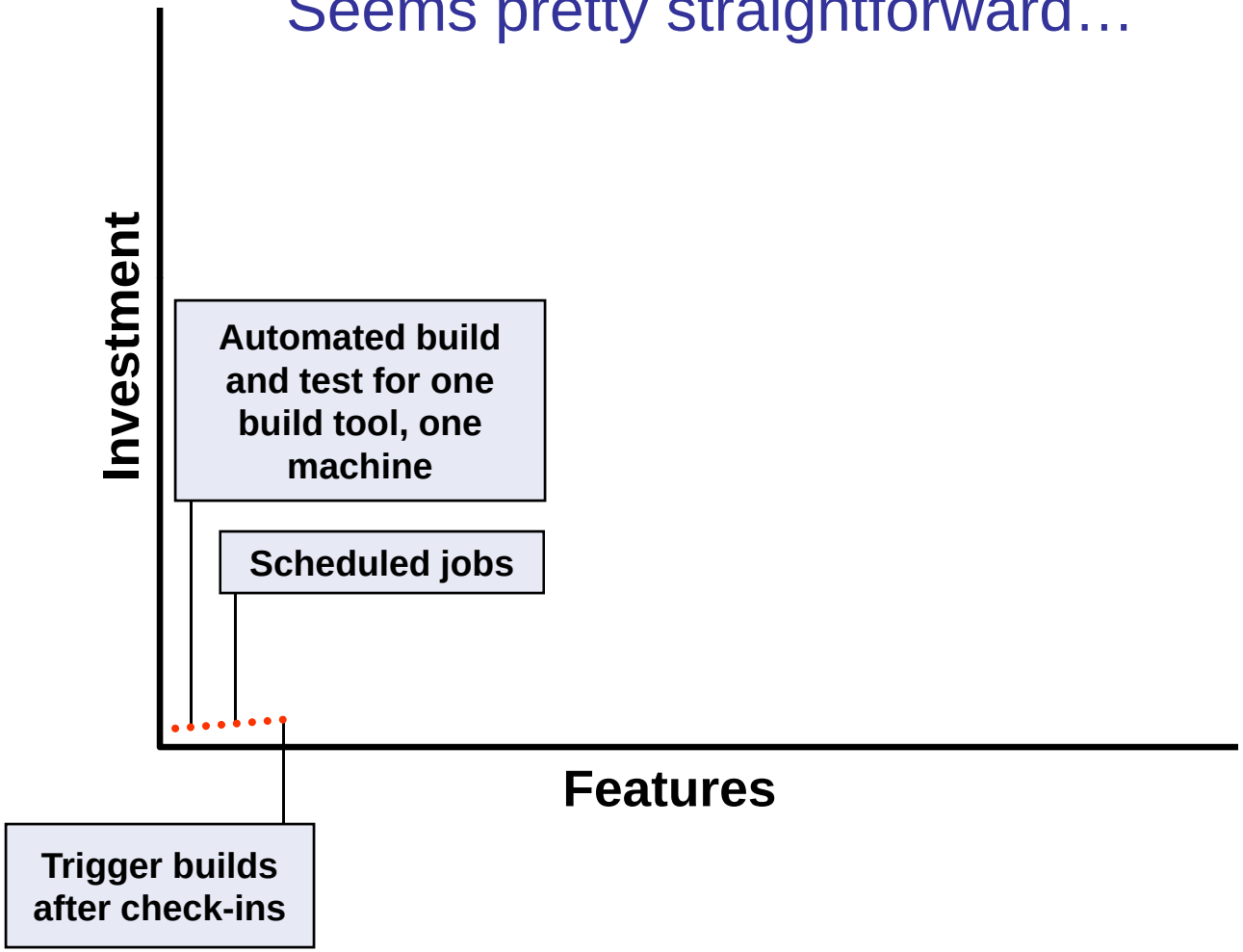
www.electric-cloud.com

Supplemental Slides
Build Your Own

www.electric-cloud.com

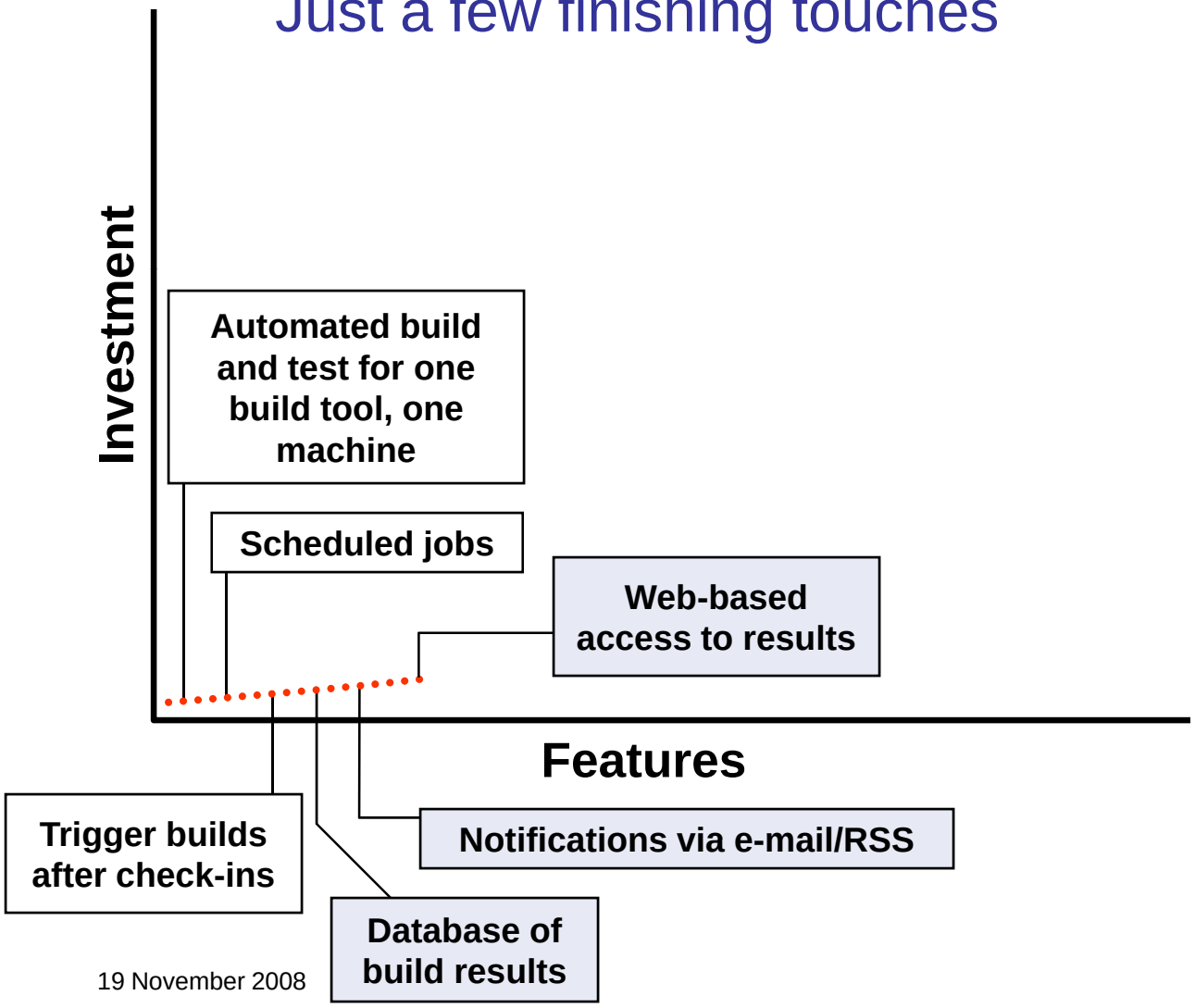
Build Your Own?

Seems pretty straightforward…



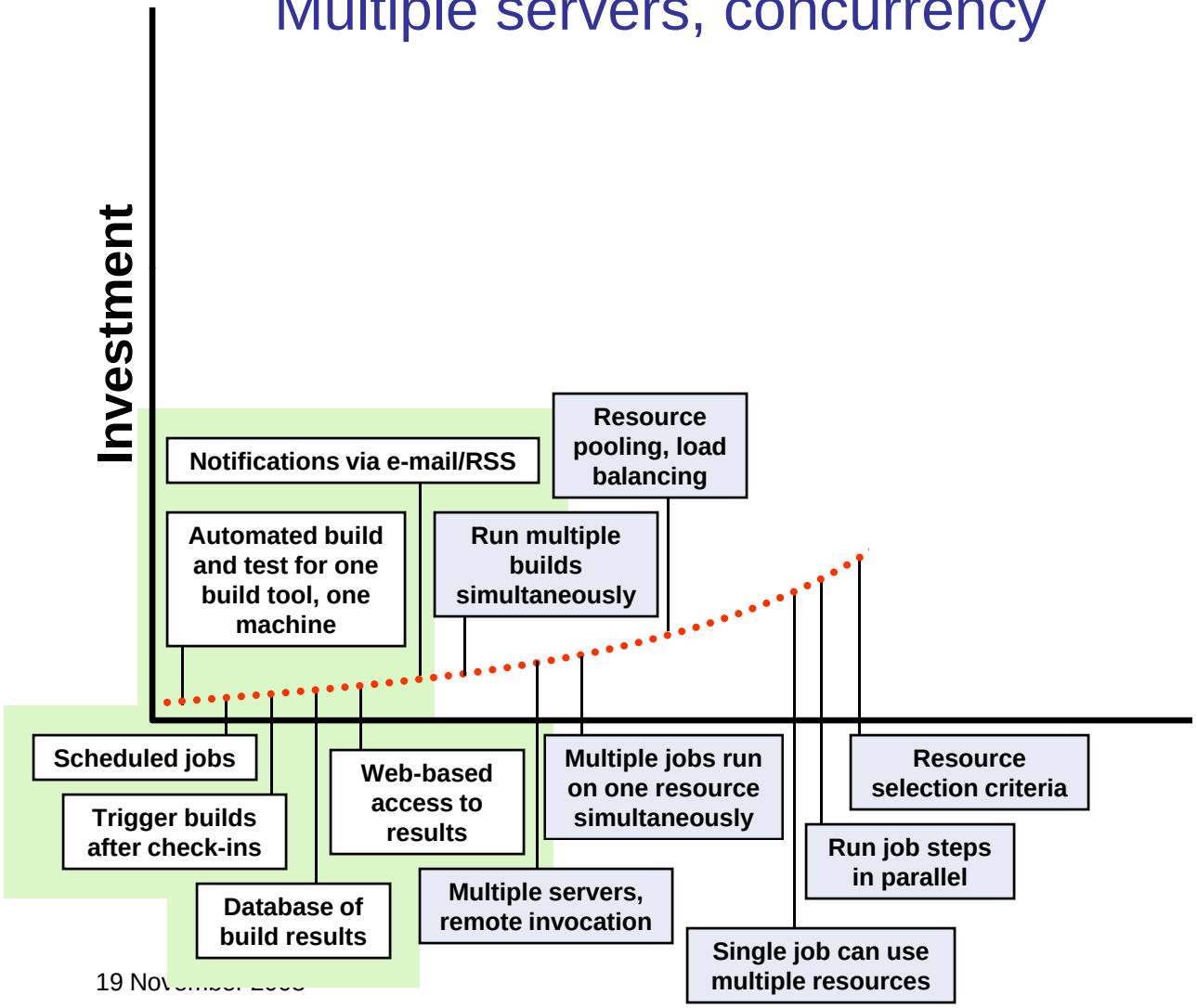
Build Your Own?

Just a few finishing touches



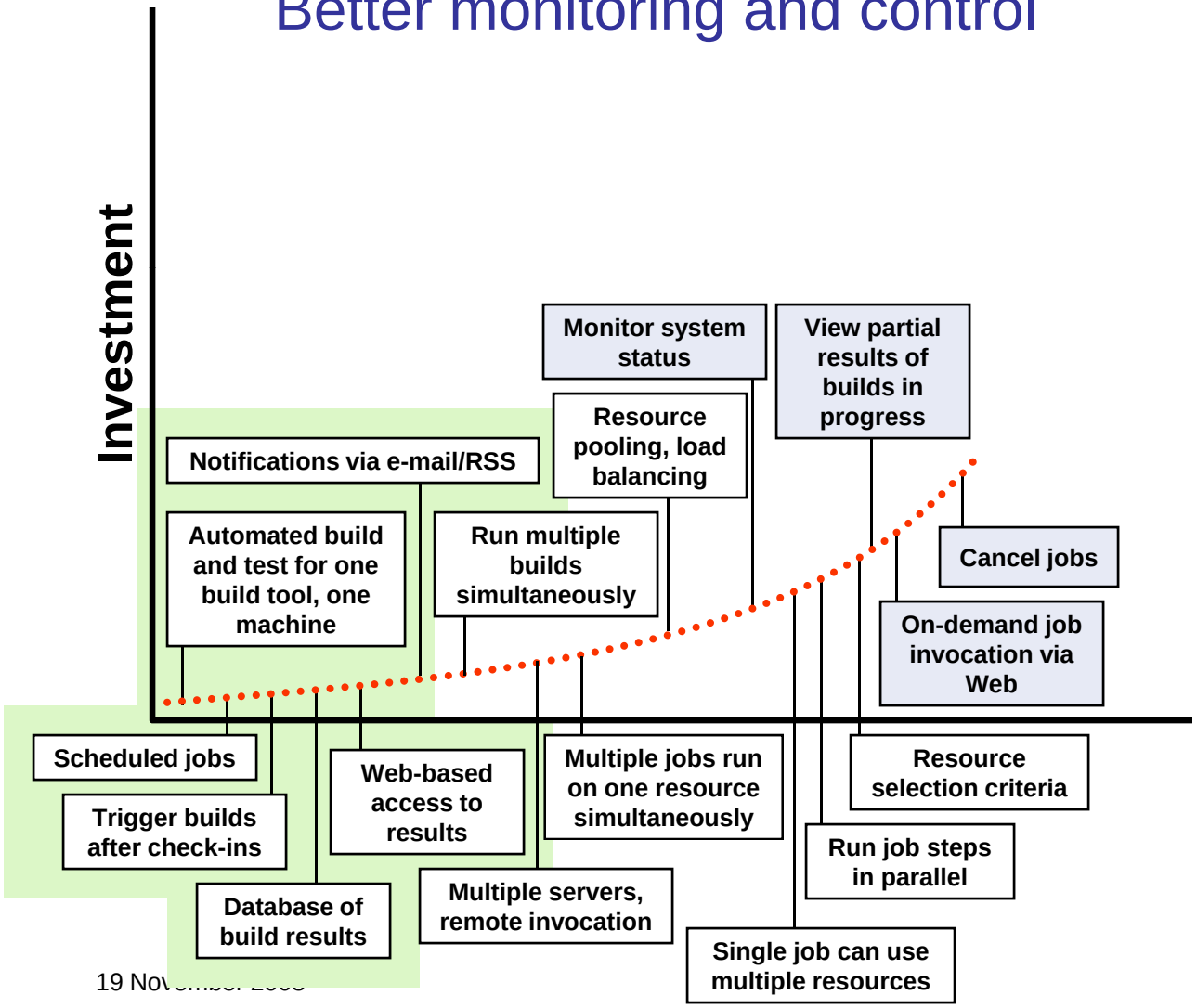
Build Your Own?

Multiple servers, concurrency



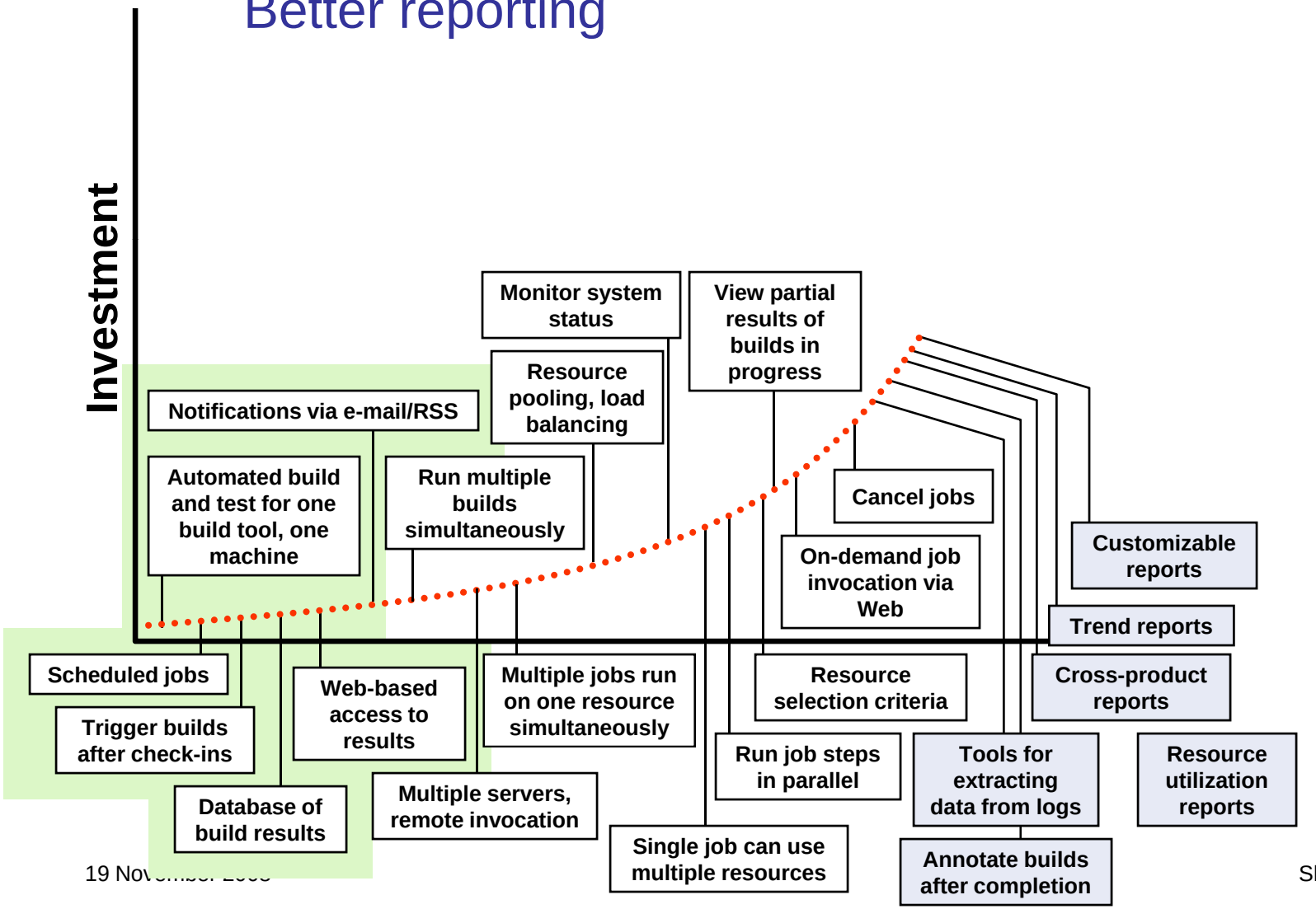
Build Your Own?

Better monitoring and control



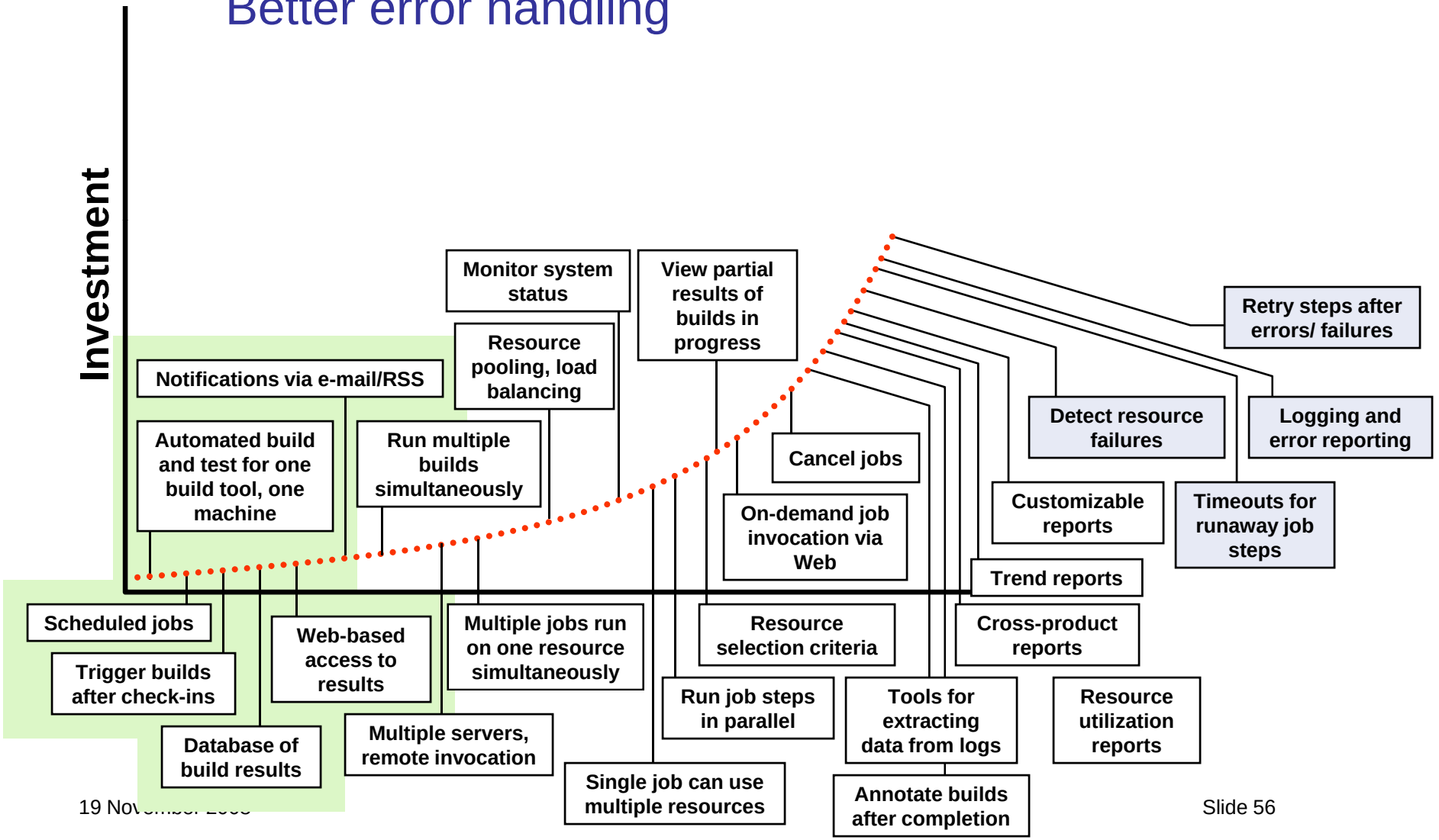
Build Your Own?

Better reporting

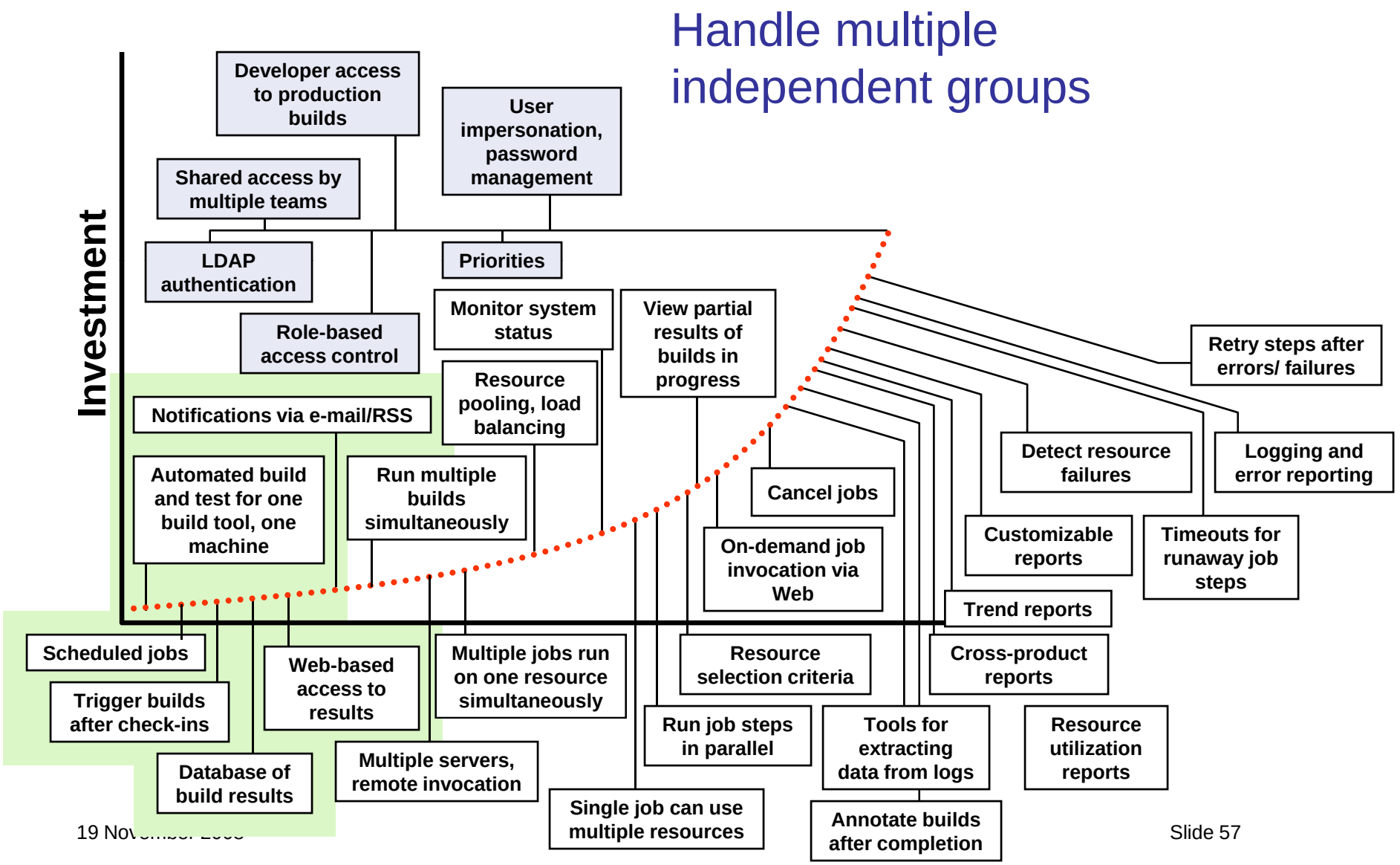


Build Your Own?

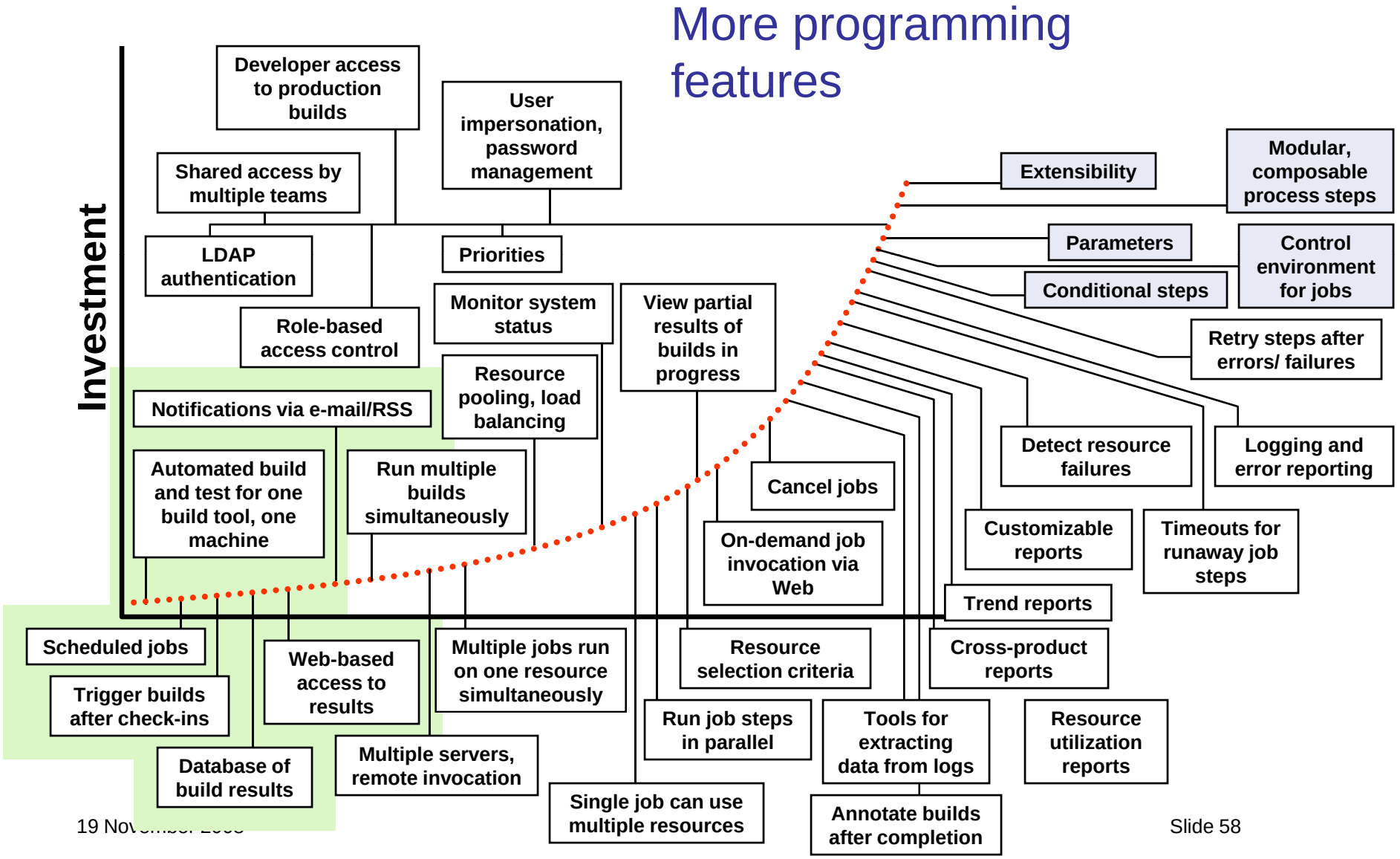
Better error handling



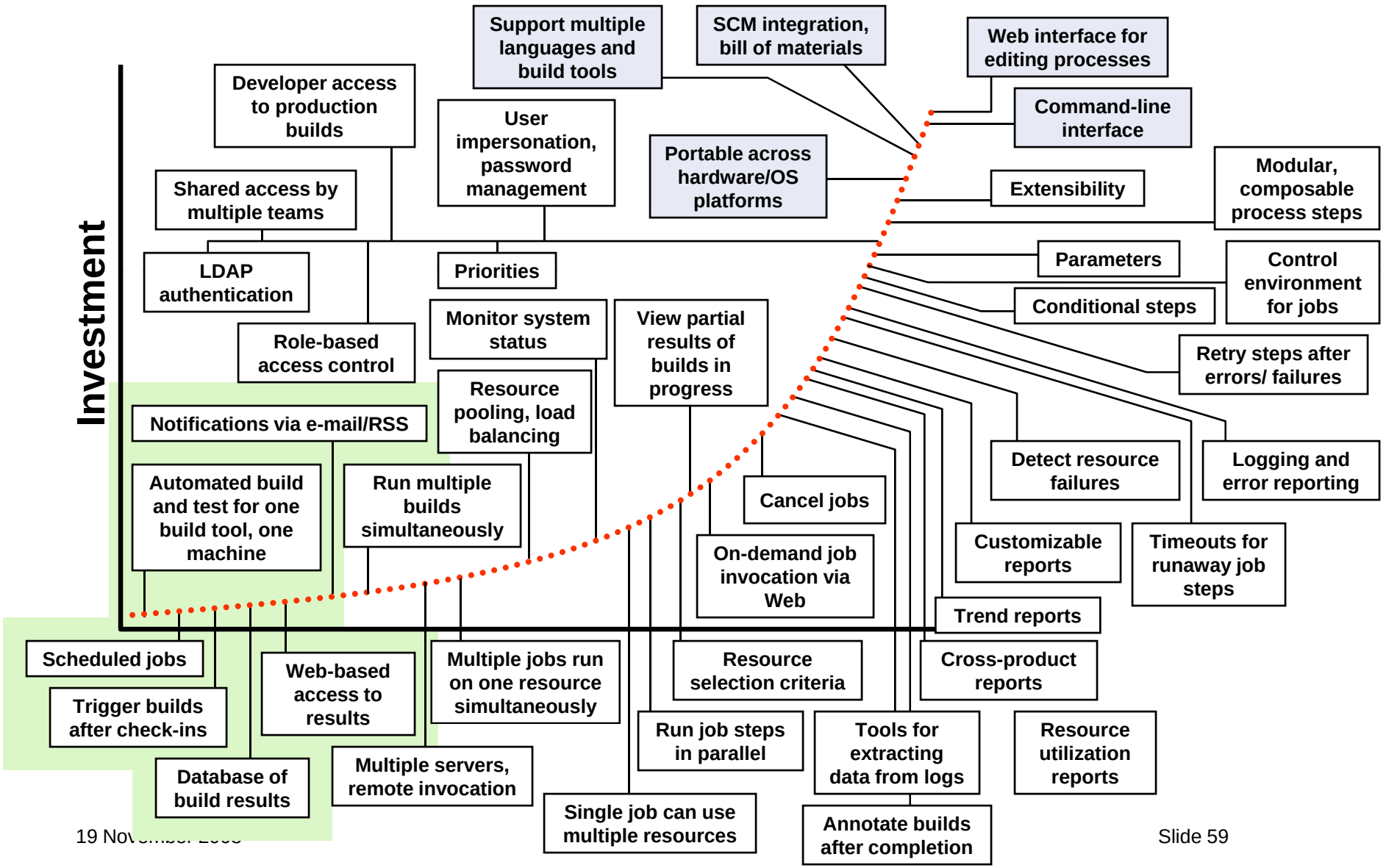
Build Your Own?



Build Your Own?



Build Your Own? It Never Ends



Questions and Answers

www.electric-cloud.com