escuela técnica superior
de ingeniería informática

Gestión de la construcción e integración continua
Build engineering, continuous integration

Departamento de
Lenguajes y Sistemas Informáticos

EGC

Varias páginas de integración continua están tomadas de Marty Stepp
http://www.cs.washington.edu/403/ y Bruce Altner y Brett Lewinski (NASA)

Ejemplo de otro dominio

ENSAMBLAR BICIS

¿Qué hay que hacer para “ensamblar” /”construir” software?

Escenarios habituales

¿Cómo hacemos las construcciones en estos casos?

1. Tenemos un equipo de trabajo distribuido
2. Tenemos varias construcciones y entregas
3. Nuestro proyecto usa librerías externas al proyecto (e.g. *.jar)

- Introducción
- Escenarios
- Integración continua
- Resumen
- Bibliografía

Compilaciones locales o remotas

Equipo distribuido

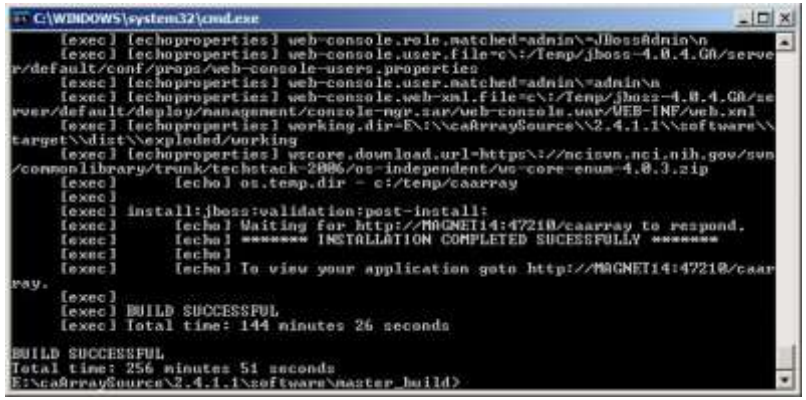
Puesto 1

Puesto 2

Puesto n

Repositorio de código

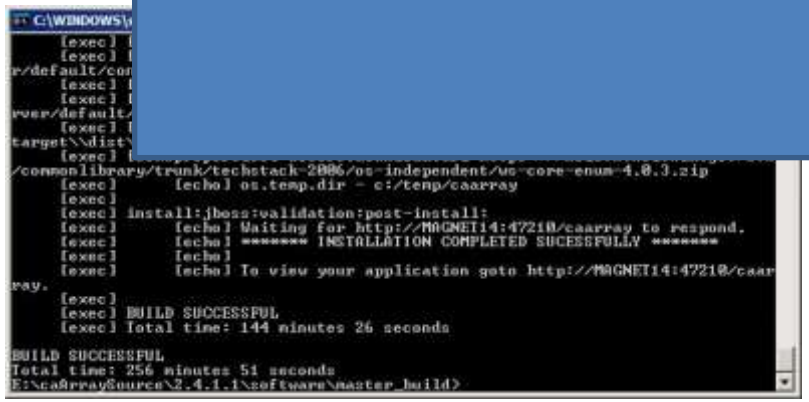
Servidor de compilación



Compilaciones locales o remotas

Puesto 1

INTEGRACIÓN

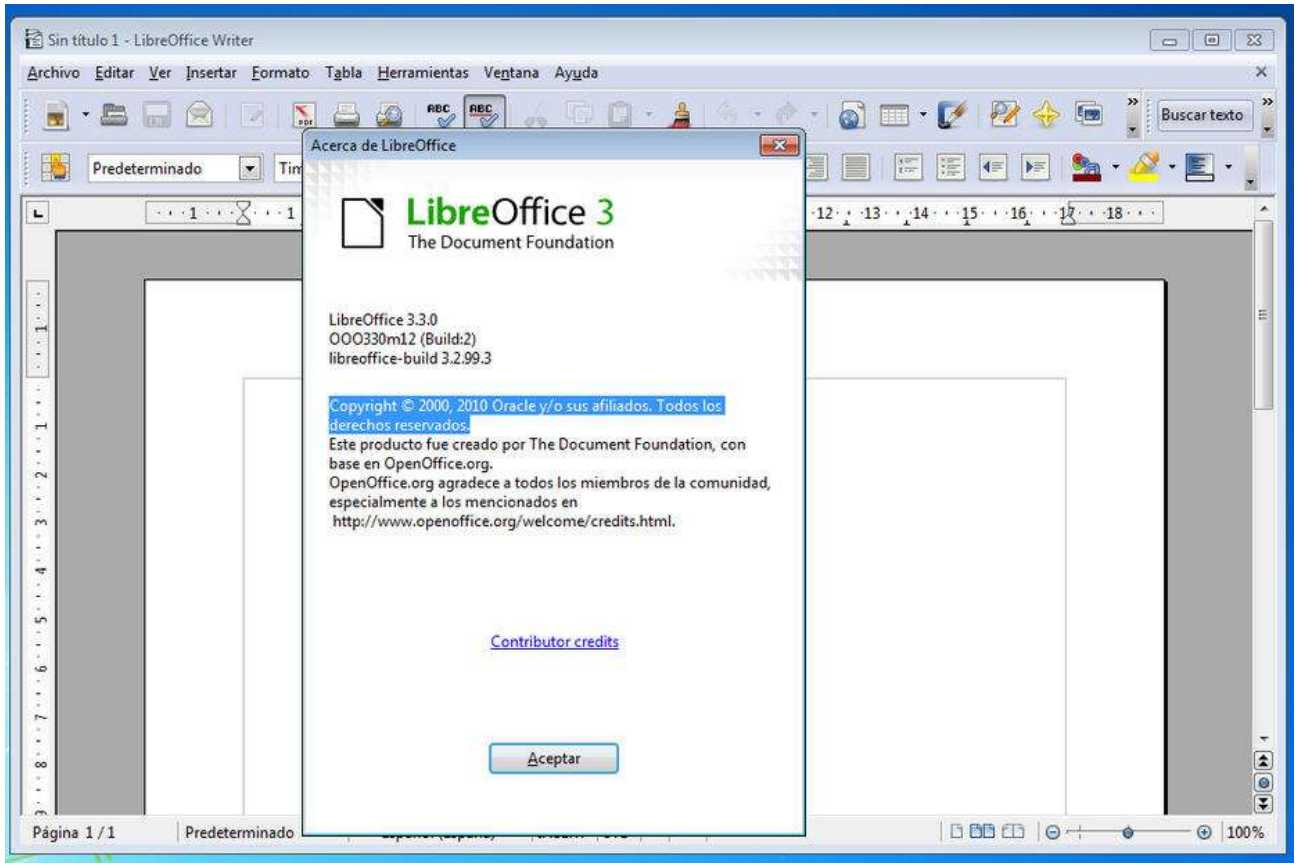


Servidor de compilación

Versionado de los resultados

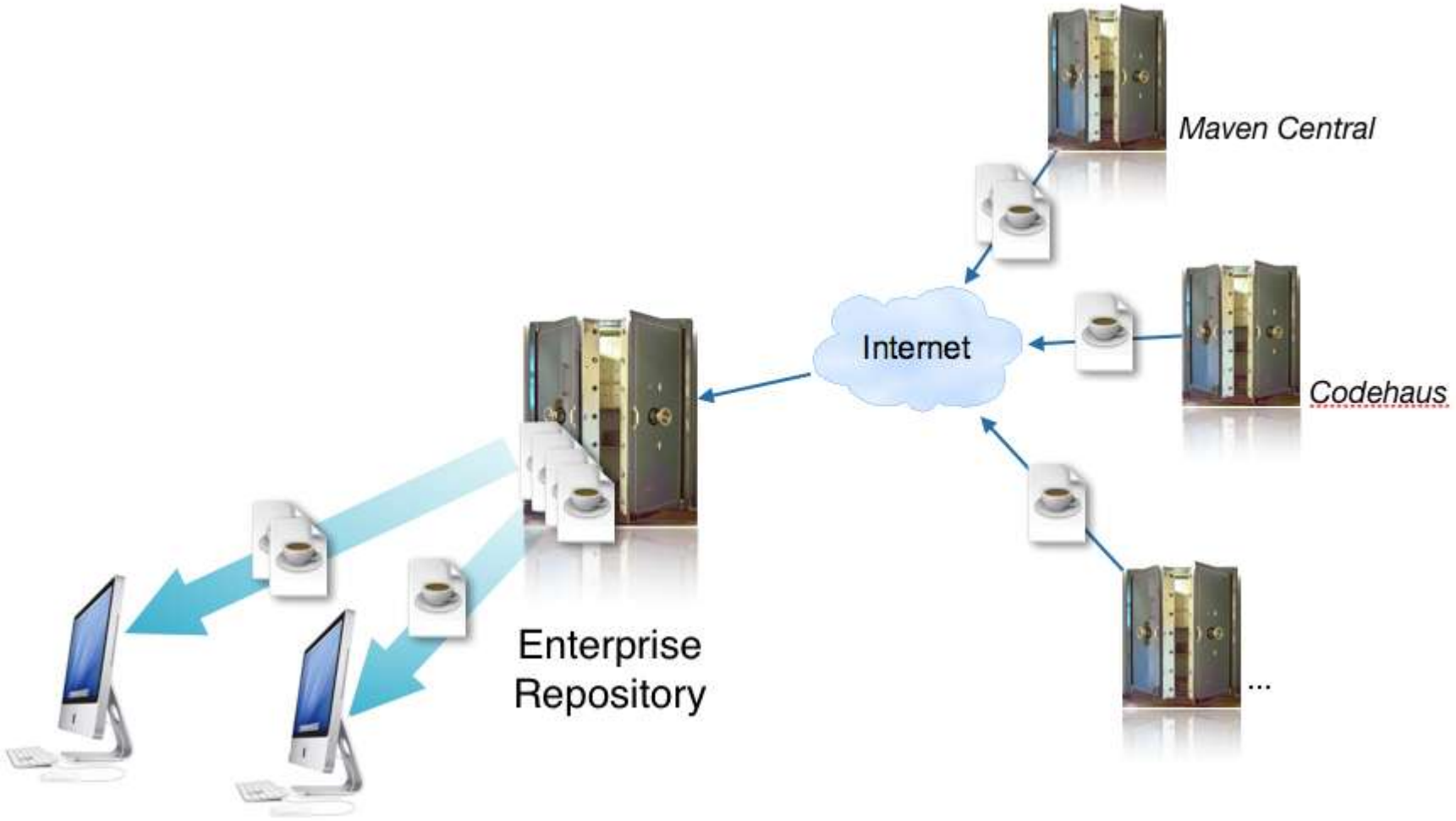
Varias construcciones y entregas

```
$>java -version
java version "1.6.0_20"
Java(TM) SE Runtime Environment (build 1.6.0_20-b02)
Java HotSpot(TM) 64-Bit Server VM (build 16.3-b01, mixed mode)
```



Repositorio de artefactos

Gestión de dependencias



Integración

- **Integración**: Combinar 2 o más unidades de software.
  - A menudo un subconjunto del total del proyecto (!= system testing)

¿Por qué hay que preocuparse de la integración?

Integración

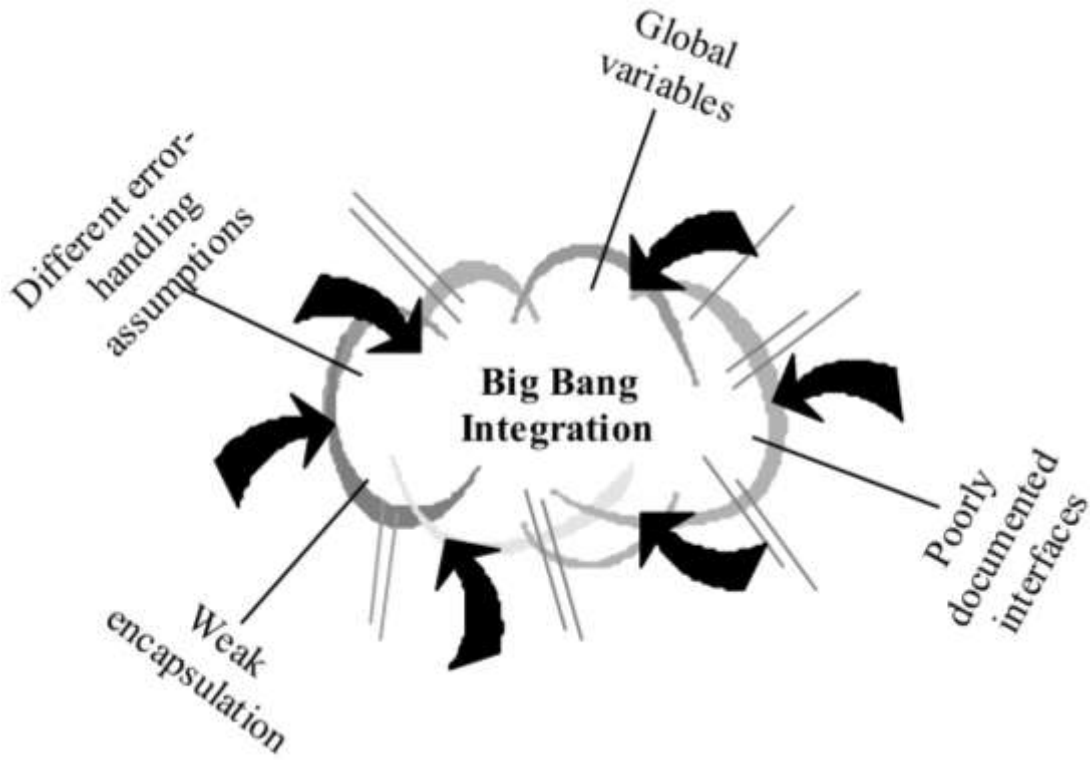
- **Integración**: Combinar 2 o más unidades de software.
  - A menudo un subconjunto del total del proyecto (!= system testing)

¿Por qué hay que preocuparse de la integración?

"Siempre que haya algo difícil ("painful"), hay que hacerlo cuánto antes mejor" [Humble]

Integración por fases

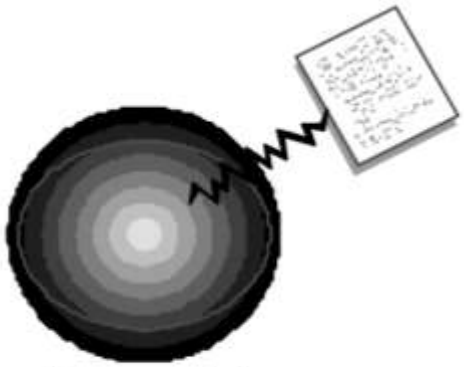
- **Integración por fases ("big-bang")**:
  - Diseño, códificación, pruebas, depuración, cada clase/unidad/subsistema de manera separada.
  - Lo combinamos todo
  - "rezamos"



Integración incremental

- **Integración incremental**:
  - Se desarrolla un "esqueleto" funcional de la aplicación/sitema
  - Diseñar, codificar, probar y depurar cada nueva pieza pequeña
  - Integrar esta pieza con el equeleto.
    - Probar/depurar antes de añadir nada nuevo

Phased Integration

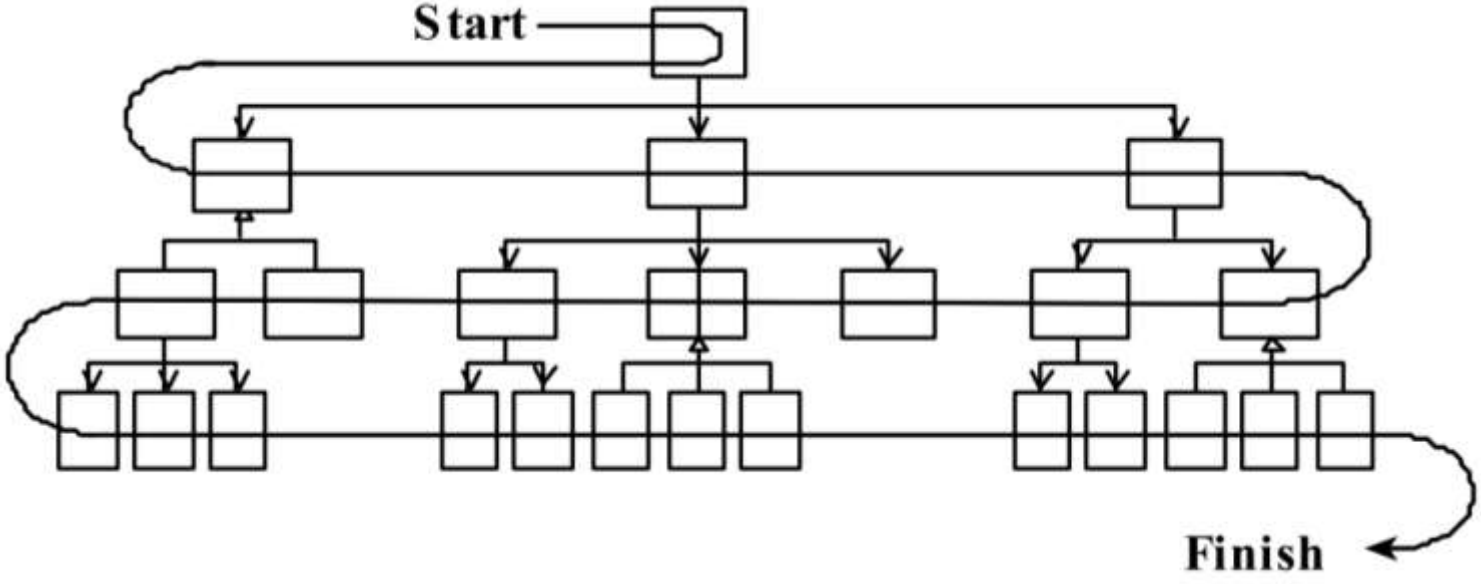


Incremental Integration

¿Ventajas e inconvenientes de integración por fases e integración incremental?

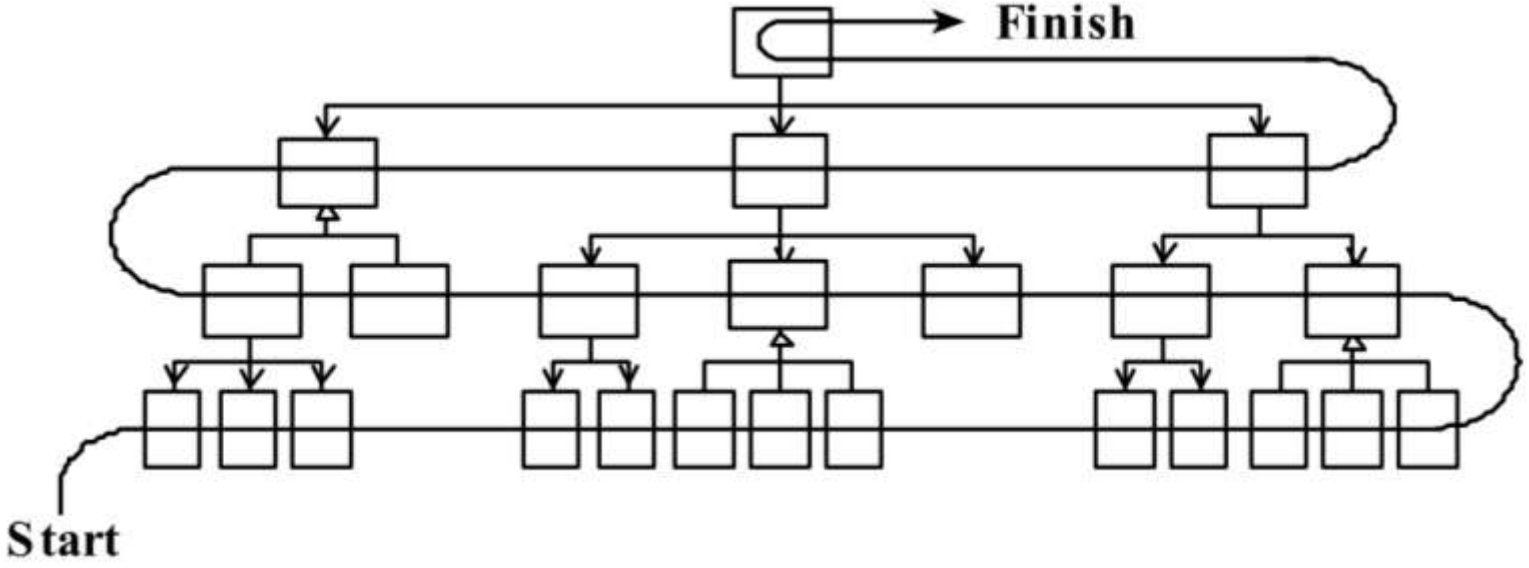
Top-down integration

- **Top-down integration**:
  Start with outer UI layers and work inward
  - must write (lots of) stub lower layers for UI to interact with
  - allows postponing tough design/debugging decisions (bad?)



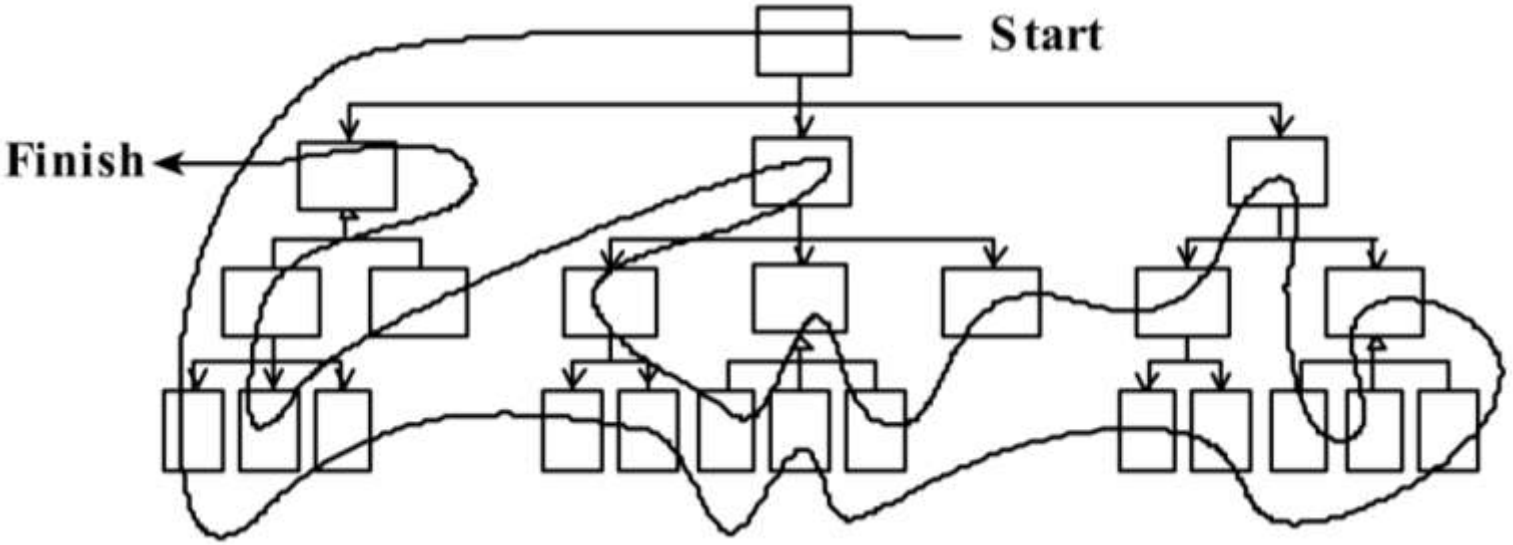
Bottom-up integration

- **bottom-up integration**:
  Start with low-level data/logic layers and work outward
  - must write test drivers to run these layers
  - won't discover high-level / UI design flaws until late



"Sandwich" integration

- **"sandwich" integration**:
  Connect top-level UI with crucial bottom-level classes
  - add middle layers later as needed
  - more practical than top-down or bottom-up?



Introducción

Escenarios

Integración continua

- Definición
- Arquitectura
- Builds
- Servidor de CI
- Pruebas y CI
- Inspección
- Feedback
- Pros/cons
- Herramientas
- Lo esencial

Resumen

Bibliografía

Continuous Integration

- Pioneered by Martin Fowler; part of Extreme Programming – Agile methods
- Ten principles:

1. maintain a single source repository
2. automate the build
3. make your build self-testing
4. everyone commits to mainline every day
5. every commit should build mainline on an integration machine
6. keep the build fast and short
7. test in a clone of the production environment
8. make it easy for anyone to get the latest executable
9. everyone can see what's happening
10. automate deployment

What is CI?, The Build Cycle

There are many variations to the process, but the basic CI *build cycle* consists of these key steps:

- Developer commits changes to the source code repository
- Build server executes the master build script, or delegates execution to another server
  - » Checks out source code
  - » Builds executable version of the application
  - » Runs other jobs, such as testing and code inspection
- Team is notified of build results through a feedback mechanism
  - » If alerts are generated, the team takes immediate action to correct problems
  - » If a code fix is needed, developer commits the corrected code back to the repository; this action kicks off a new build cycle.

Introducción

Escenarios

Integración continua

- Definición
- Arquitectura
- Builds
- Servidor de CI
- Pruebas y CI
- Inspección
- Feedback
- Pros/cons
- Herramientas
- Lo esencial

Resumen

Bibliografía

Aquitectura del sistema de CI

The key to fixing problems quickly is finding them quickly.
– (Fowler, 2006)

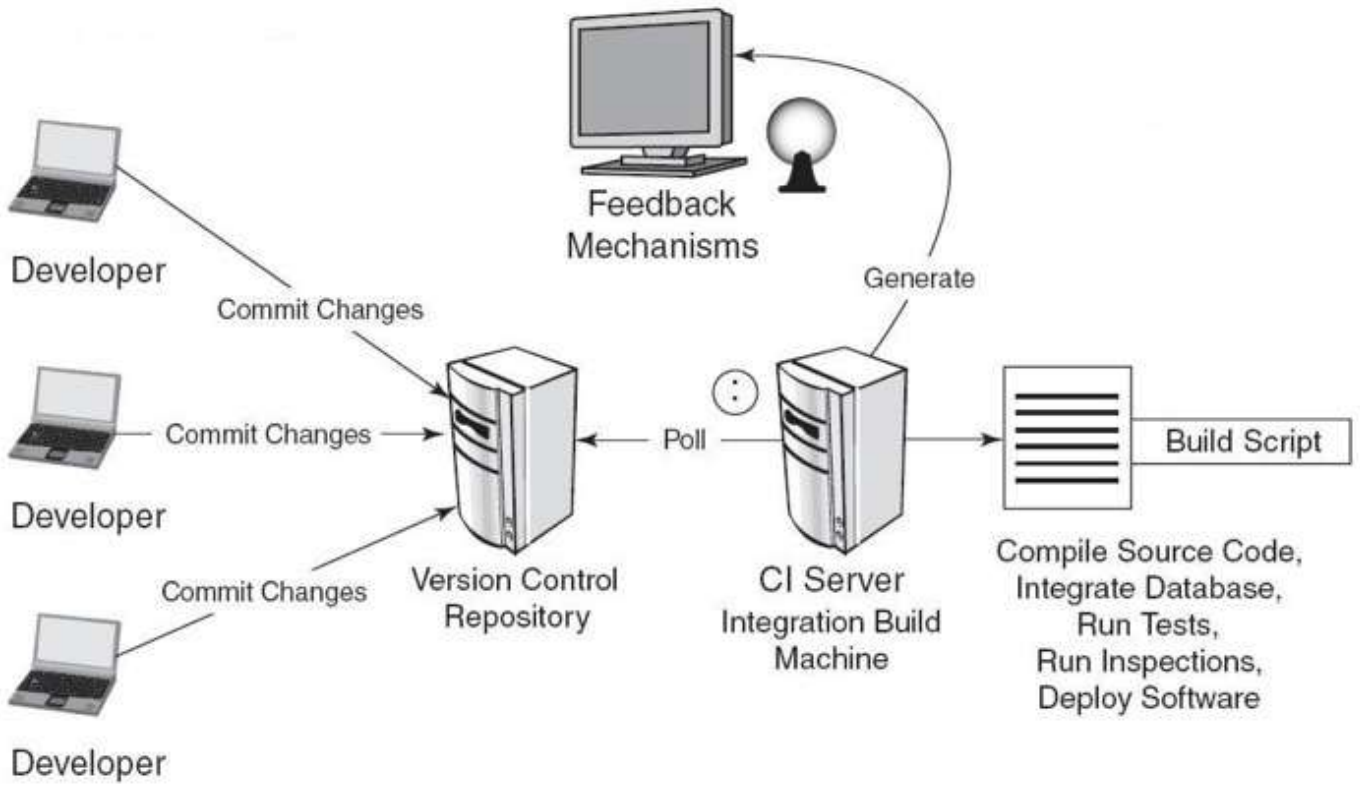


Figure 1: System and Software Architecture Supporting a CI Build

Definition: CI is the practice of regular, comprehensive, and automatic building and testing of applications in software development.

Artifacts Needed

- A build usually involves more than source code files. Other items may include:
  - Project and third-party components and libraries
  - Configuration files (e.g. *.properties)
  - Data files and database scripts
  - Test scripts
  - Build scripts
  - Properties files
- All items except project and third-party libraries should be stored in the source code repository…. What about binaries?

Introducción

Escenarios

Integración continua

- Definición
- Arquitectura
- Builds
- Servidor de CI
- Pruebas y CI
- Inspección
- Feedback
- Pros/cons
- Herramientas
- Lo esencial

Resumen

Bibliografía

Daily commits

"Everyone commits to the mainline every day."

- **daily commit**: Submit work to main repo at end of each day.
  - Idea: Reduce merge conflicts; avoid later integration issues.
  - This is the key to "continuous integration" of new code.

  - *Caution:* Don't check in faulty code (does not compile, does not pass tests) just to maintain the daily commit practice.
  - If your code is not ready to submit at end of day, either submit a coherent subset or be flexible about commit schedule.

Essential Characteristics of a CI Build

- The build is completely automated, usually through the execution of a single script.
- No matter how triggered or how often run, *a build always begins by retrieving code from the source code repository*.
- Unless terminated prematurely, the end product of a build is always executable code (define semantics of "executable").
- Notification of build status always occurs through a feedback mechanism.

The "Integrate Button" Metaphor

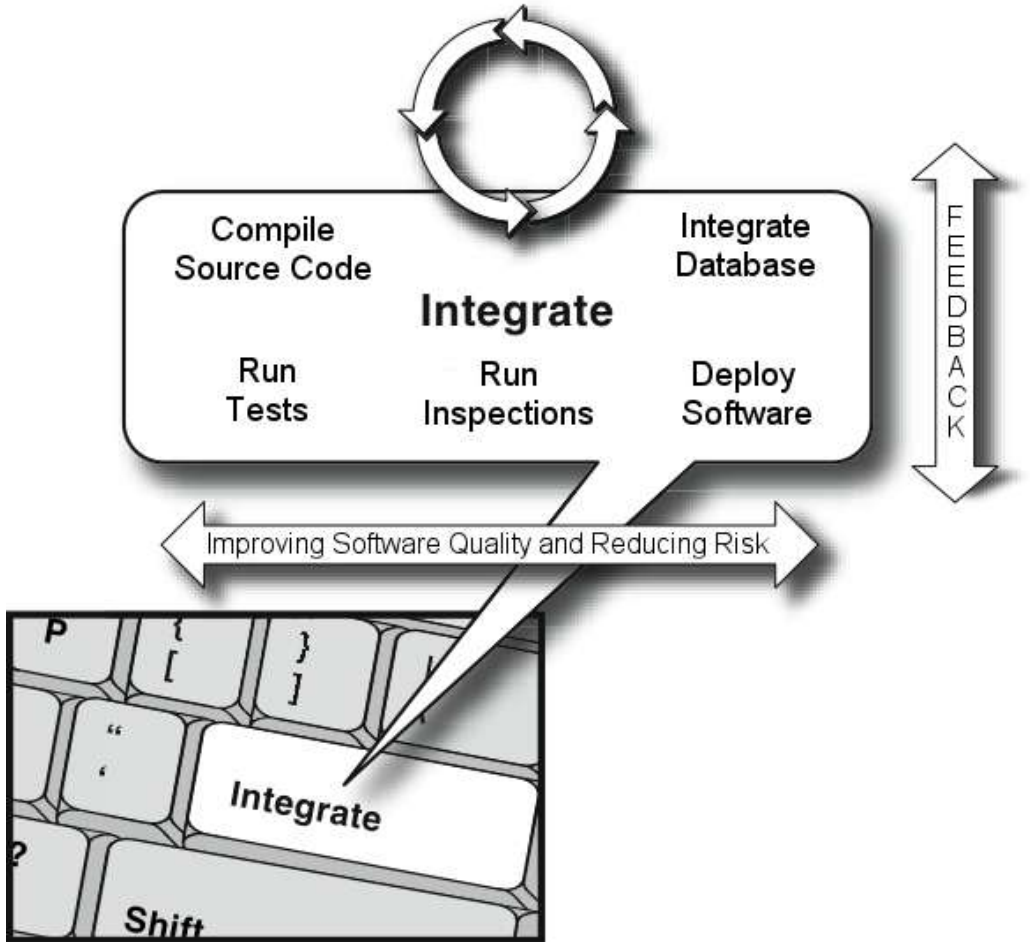


Figure 2: The "Integrate Button" Metaphor

Source: Duvall et al. (2007)

Build Types

There are three basic types of builds: private, integration, and release

- All builds are automated, script-driven processes that pull source code from the repository and create deployable artifacts
- These build types differ in several ways
  - Where, when, by whom, and for what reason they are run
  - The code that feeds the process
  - What is done with the output

Build Types *(continued…)*

- **Private Builds**
  - Run by developers in their local environments (IDE)
  - Ensure that code compiles before committing back to source code repository
  - Triggered manually
  - Very stripped-down and designed to finish quickly
- **Integration Builds**
  - Compile and package the application for testing and quality inspection in response to recent changes
  - Performed on a dedicated build server either on a scheduled basis (nightly) or continuously
  - Usually drawn from the trunk of the repository tree
  - Fast or slow depending on the number and type of processes included in the build script
- **Release Builds**
  - Performed when code is ready to be deployed to other environments
  - Performed by the team's release manager
  - Use tagged code from the repository
  - Run the full suite of automated and manual tests and inspections
  - Tend to be slow

CI vs. Build Management

Property	Continuous Integration	Build Management
Build Type	Integration	Release
Purpose	To determine whether the latest integrated changes degraded the code quality	To produce an unambiguous set of artifacts to be released to third parties outside of development
Audience	Development team	QA, CM, Operations teams, etc.
Lifecycle	Development	Boundary between development and next application lifecycle stage
Source	Most current version in repository (trunk); always changing	Specific snapshot (tag); unchanging
Traceability	To newly integrated changes	To full source snapshot
Degree of Automation	Completely automated	Combination of automated scripts and manual processes (¿?)
Artifacts	Artifacts are a mere by-product; quality determination is the primary output	Production of artifacts is instrumental to the purpose

Daily builds

"Automate the build."

- **daily build**: Compile working executable on a daily basis
  - allows you to test the quality of your integration so far
  - helps morale; product "works every day"; visible progress
  - best done *automated* or through an easy script
  - quickly catches/exposes any bug that breaks the build

- **Continuous Integration (CI) server**: An external machine that automatically pulls down your latest repo code and fully builds all resources.
  - If anything fails, contacts your team (e.g. by email).
  - Ensures that the build is never broken for long.

Build from command line

- An Android project needs a `build.xml` to be used by Ant.
  - This file allows your project to be compiled from the command line, making automated builds possible.

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="MainActivity" default="help">
    <property file="ant.properties" />
    <import file="${sdk.dir}/tools/ant/build.xml" />
    <taskdef name="findbugs" classname="edu.umd.cs.findbugs.anttask.FindBugsTask"/>
    <target name="findbugs">
        <findbugs home="${findbugs.home}" output="xml" outputFile="findbugs.xml" excludeFilter="findbugs-exclude.xml">
            <auxClasspath path="${android.jar}" />
            <auxClasspath path="${rt.jar}" />
            <auxClasspath path="libs\android-support-v4.jar" />
            <class location="${out.dir}" />
        </findbugs>
    </target>

    <taskdef resource="checkstyletask.properties" classpath="${basedir}/libs/checkstyle-5.6-all.jar"/>
    <checkstyle config="sun_checks.xml" failonviolation="false">
        <fileset dir="src" includes="**/*.java"/>
        <formatter type="plain"/>
        <formatter type="xml" toFile="checkstyle-result.xml"/>
    </checkstyle>
</project>
```

The Importance of Fast Builds

- In general, the more processes included in the build, the better the quality of the final product; however, there are tradeoffs.
- Each subprocess takes time to execute; extra steps result in loss of productivity, as developers wait for long build cycles to complete before they can continue working.
- Frustration with overly long builds* can be a significant roadblock to team acceptance of the CI process.

* Kent Beck, in his book Extreme Programming Explained (2004), suggests a 10-minute limit for basic integration builds. Some developers would consider even 10 minutes to be too long a time.

Improving Build Speed

- Three approaches to dealing with build speed:
  - Speed up the build by eliminating bottlenecks or by running the build on a faster machine with plenty of memory.
  - Run complete builds less often or at a time when developers are less likely to be working, and run only basic builds on a continuous basis.
  - Employ a “staged build” approach in which the basic build executes to validate code that developers have just checked in, after which a second process completes the remaining steps.
- A combination of the three approaches is often used.

Introducción

Escenarios

Integración continua

- Definición
- Arquitectura
- Builds
- Servidor de CI
- Pruebas y CI
- Inspección
- Feedback
- Pros/cons
- Herramientas
- Lo esencial

Resumen

Bibliografía

The CI Build Server

- CI build servers provide many valuable services:
  - Polling for changes in the source code repository, which trigger a new build
  - Avoiding overlapping builds (due to repository commits very close together in time) by specifying a configurable quiet-time interval
  - Support for different scripting tools and source code repositories
  - Support for a variety of notification mechanisms
  - Logging of build results to provide a history of previous builds
  - Tagging of build numbers
  - Web-based configuration and reporting of build results
  - Support for distributed builds via a master/slave architecture

Introducción

Escenarios

Integración continua

- Definición
- Arquitectura
- Builds
- Servidor de CI
- Pruebas y CI
- Inspección
- Feedback
- Pros/cons
- Herramientas
- Lo esencial

Resumen

Bibliografía

Automated tests

"Make your build self-testing."

- **automated tests**: e.g. Tests that can be run from the command line on your project code at any time.
  - can be unit tests, coverage, static analysis / style checking, ...

- **smoke test**: A quick set of tests run on the daily build.
  - NOT exhaustive; just sees whether code "smokes" (breaks)
  - used (along with compilation) to make sure daily build runs

Automated Processes:Continuous Testing

Automated testing is an important component of the CI build cycle.

Test Type	Description	Impact on CI Build
Unit	Tests discrete units of code to verify correct behavior; written and performed by developers as part of the development process.	Easily included in the build cycle; very low impact on build speed.
Integration/ Component	Tests to verify specific components, including their interaction with other internal and external components; may exercise code not exposed to clients or end users.	May be included in the build cycle; tends to run longer than unit tests so there is some impact on build speed.

Continuous Testing *(continued…)*

Test Type	Description	Impact on CI Build
Functional (Regression)	Tests all aspects of the fully deployed system, exercising all actions that may be taken by end users; may use a wide range of input data to force errors; fully aligns with use cases; specialized tools allow feature-by-feature capture of user actions as executable scripts that can be run on an automated basis.	The impact on build speed may be significant; should not be a part of builds that run every time code changes; appropriate for nightly runs or a staged build process.
Functional (Targeted)	Tests only specific features that were changed for a particular build.	Minimal impact on build speed since only a small part of the full application is tested.

Continuous Testing *(continued…)*

non-functional

Test Type	Description	Impact on CI Build
Load	Subjects the application to levels of use approaching and beyond the limits of its specification to ascertain the maximum amount of work a system can handle without significant performance degradation.	The impact on build speed may be significant; should not be a part of builds that run every time code changes; appropriate for nightly runs or a staged build process.
Stress	Evaluates the extent to which a system keeps working when subjected to extreme work loads or when some of its hardware or software has been compromised; includes availability or resistance to denial of service (DoS) attacks.	The impact on build speed may be significant; should not be a part of builds that run every time code changes; appropriate for nightly runs or a staged build process.

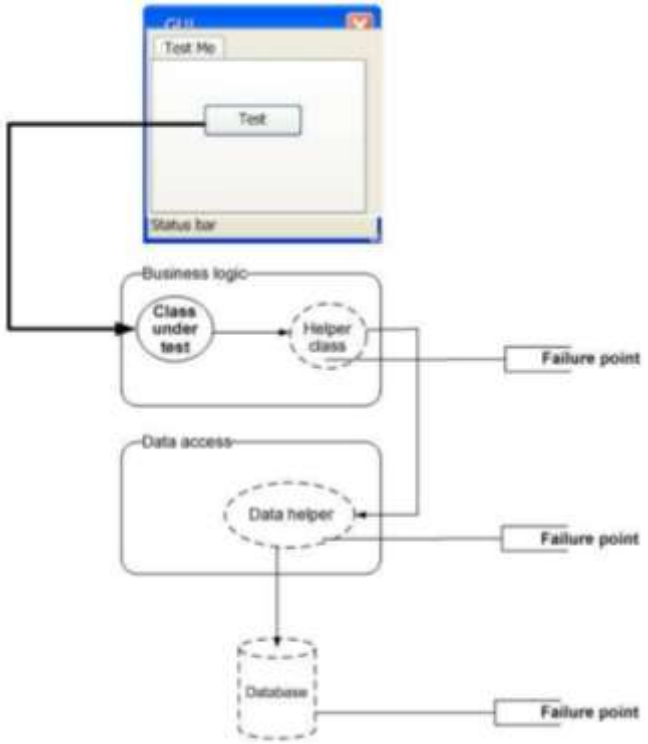
Continuous Testing *(continued…)*
non-functional

Test Type	Description	Impact on CI Build
Security	Used in conjunction with manual or automated functional testing. Many types of tools for web application security assessment exist, such as source-code analyzers, web application (black-box) scanners, database scanners, binary analysis tools, runtime analysis tools, configuration analysis tools, and proxies. Proxies, for example, watch what the user or test script does and report potential security holes.	The impact on build speed could be significant; should not be a part of builds that run every time code changes; appropriate for nightly runs or a staged build process.

Integration testing

- **integration testing**: Verifying software quality by testing two or more dependent software modules as a group.

- challenges:
  - Combined units can fail in more places and in more complicated ways.
  - How to test a partial system where not all parts exist?
  - How to "rig" the behavior of unit A so as to produce a given behavior from unit B?



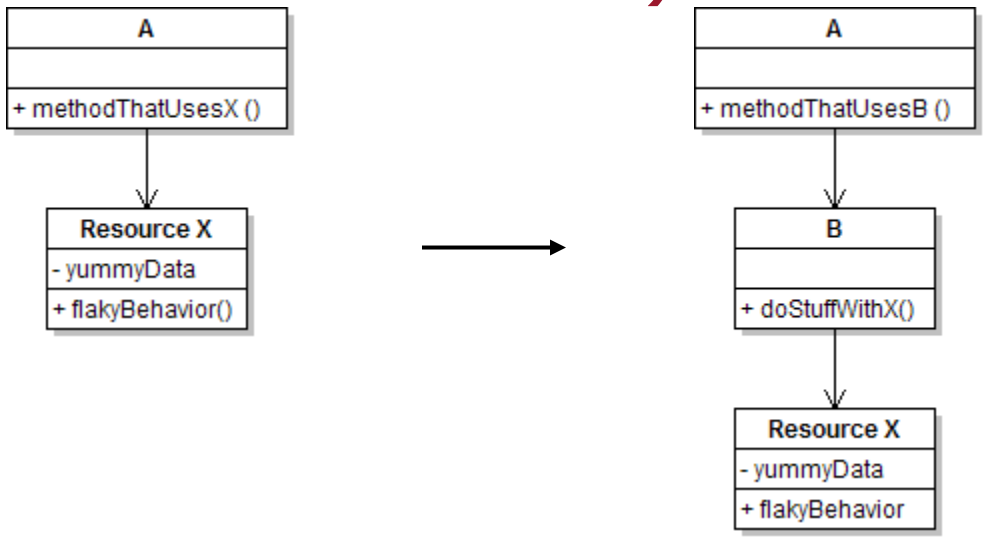
Stubs

- **stub**: A controllable replacement for an existing software unit to which your code under test has a dependency.

  - useful for simulating difficult-to-control elements:
    - network / internet
    - database
    - time/date-sensitive code
    - files
    - threads
    - memory

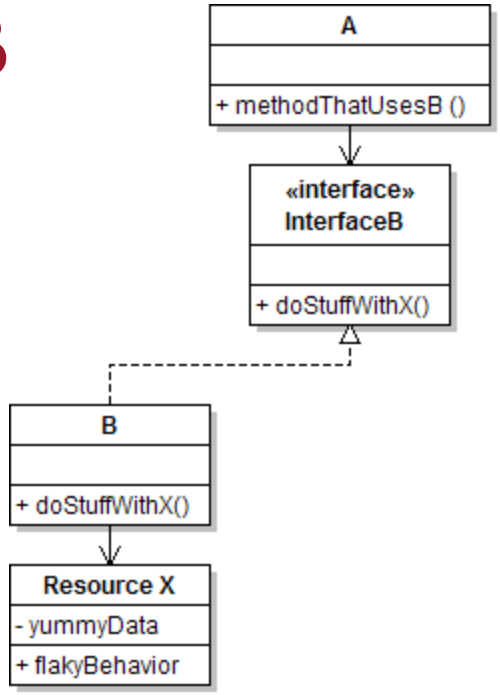
Create a stub, step 1

- Identify the external dependency.
  - This is either a resource or a class/object.
  - If it isn't an object, wrap it up into one.
    - (Suppose that Class A depends on troublesome Class B.)



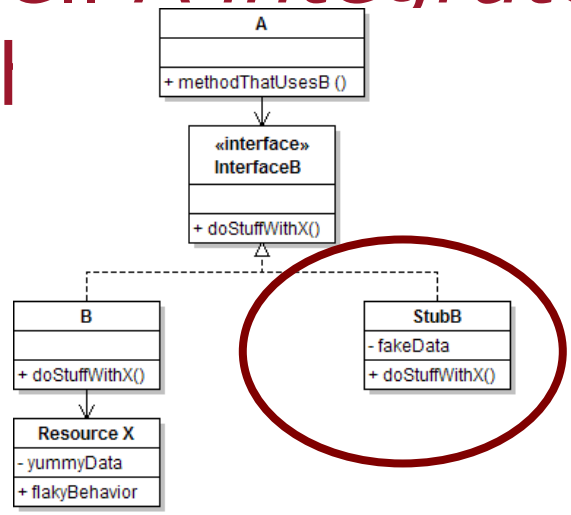
Create a stub, step 2

- Extract the core functionality of the object into an interface.
  - Create an InterfaceB based on B
  - Change all of A's code to work with type InterfaceB



Create a stub, step 3

- Write a second "stub" class that also implements the interface, but returns pre-determined fake data.
  - Now A's dependency on B is dodged and can be tested easily.
  - Can focus on how well A *integrates* with B's external bel



Injecting a stub

- **seams**: Places to inject the stub so Class A will talk to it.
  - at construction (not ideal)

```
A aardvark = new A(new StubB());
```

  - through a getter/setter method (better)

```
A apple = new A(...);
aardvark.setResource(new StubB());
```

  - just before usage, as a parameter (also better)

```
aardvark.methodThatUsesB(new StubB());
```

  - You should not have to change A's code everywhere (beyond using your interface) in order to use your Stub B. (a "testable design")

Introducción

Escenarios

Integración continua

- Definición
- Arquitectura
- Builds
- Servidor de CI
- Pruebas y CI
- Inspección
- Feedback
- Pros/cons
- Herramientas
- Lo esencial

Resumen

Bibliografía

Continuous Code Inspection

A stated objective of CI is to integrate frequently in order to detect and remedy quality issues as they are introduced.

- Code quality issues are not just errors that cause compile-time or runtime problems.
- Tangled, hard-to-read, overly complex, undocumented, and inconsistently formatted code may compile, but committing this kind of code to the repository is not a good practice.
- Many teams apply stylistic standards for “clean” code in their particular language, a process that can be automated.

Continuous Code Inspection *(continued…)*

- Automated processes are very good at code inspection, once a set of standards has been defined.
  - Noncompliance with team standards can cause a build to fail or simply to be reported.
  - Tools are available for most programming languages.
  - Rules can be based on a project's coding standards.
  - Can be run inside an IDE or in a build tool such as Ant.
  - Tools are available to not only enforce style and format compliance but also reduce code complexity, eliminate duplicated code, and ensure complete unit test coverage.
- Continuous code inspection allows the team to concentrate on the important issues during peer reviews, such as alignment with requirements and sound architecture and design.

Introducción

Escenarios

Integración continua

- Definición
- Arquitectura
- Builds
- Servidor de CI
- Pruebas y CI
- Inspección
- Feedback
- Pros/cons
- Herramientas
- Lo esencial

Resumen

Bibliografía

Feedback

- Is essential to the purpose of CI
  - Without notification the team has no way of knowing whether recent changes introduced new problems.
- Can take many forms and be directed at many stakeholders
- Must be managed carefully
  - “Spam Feedback” is a CI anti-pattern in which team members become inundated with build status e-mails, to the point where they start to ignore messages.
- Is most effective when it directs the right information to the right people in the right situation and uses the right methods

Feedback *(continued…)*

There are many ways to send notification of build status:

- E-mail: Provides build status at discrete points in time
- RSS: Pushes alerts regarding build status to an RSS reader
- SMS: Provides build status with text messages sent to cell phone
- Visual devices: Ambient Orb, X10 Lava lamps
- Twitter: Provides build status with “tweets” sent to Twitter account
- Sounds: Provides build status through sound
- Displays: Provides feedback through an LCD monitor
- Instant Message: Allows for instantaneous notification via IM
- Browser plug-ins: Provides build status in the browser status bar
- Widgets: Display the build status on the user's desktop

Introducción

Escenarios

Integración continua

- Definición
- Arquitectura
- Builds
- Servidor de CI
- Pruebas y CI
- Inspección
- Feedback
- Pros/cons
- Herramientas
- Lo esencial

Resumen

Bibliografía

Benefits of CI

- Increased productivity
  - Enables shorter feedback cycle when changes are made
  - Code is kept in a “releasable” state
  - Code gets back into the hands of testers quickly
  - Frees the team to do more interesting and valuable work
  - Improves morale, making it easier to retain good developers
  - Enables more frequent releases with new features
- Improved quality
  - Makes it easier to find and remove defects because frequent integration and testing identifies bugs *as they are introduced.*
  - Multi-platform builds help in finding problems that may arise on some, but not all, versions of the supported platform.
- Reduced Risk
  - Reduces uncertainty greatly because at all times the team knows what works, what does not, and what the major issues are.

Barriers to CI

- Why doesn't every team already practice CI?
  - It is not easy.
  - Establishing a *solid* CI practice takes a lot of work and technical knowledge.
  - A number of new tools and processes must be mastered.
  - Setting up a CI server requires that the build, unit test, and executable packaging processes all be automated.
  - Requires mastering a build scripting language, a unit testing platform, and potentially a setup/install platform as well.
- Duvall (2007) says, "I often hear statements like 'CI doesn't work for large projects' or 'our project is so unique that CI won't work' when, in fact, CI isn't the issue at all — *it's the ineffective application, or nonexistence, of certain practices that have led to frustration."*

Introducción

Escenarios

Integración continua

- Definición
- Arquitectura
- Builds
- Servidor de CI
- Pruebas y CI
- Inspección
- Feedback
- Pros/cons
- Herramientas
- Lo esencial

Resumen

Bibliografía

Build Servers

The most important criterion in choosing a tool is whether it does what you need it to do.
-- Duvall et al. (2007)

A comprehensive and up-to-date listing of the many open source and commercial CI build servers available today is maintained on the ThoughtWorks CI Feature Matrix.*

- At this time, 25 products are compared on dozens of attributes.
- Adds products and attributes as they are identified.
- Attempts an unbiased comparison of servers to assist teams considering CI adoption in selecting an appropriate product.
- Identifies the most important features for companies and projects involved in the developing CI build servers.
- Does not provide market share information.

* See http://confluence.public.thoughtworks.org/display/CC/CI+Feature+Matrix

Build Servers: What are People Saying?

- Using an online survey of software professionals with an interest in CI, Fleisher (2009) attempted to learn which CI servers people are using and why.
- He identified 12 products and reported their support for:
  - automated build tools
  - version control systems
  - build triggers
  - test frameworks
  - notification mechanisms
  - integration with IDEs
  - access control
  - multi-platform builds
  - history builds
  - virtualized environments
  - extensibility

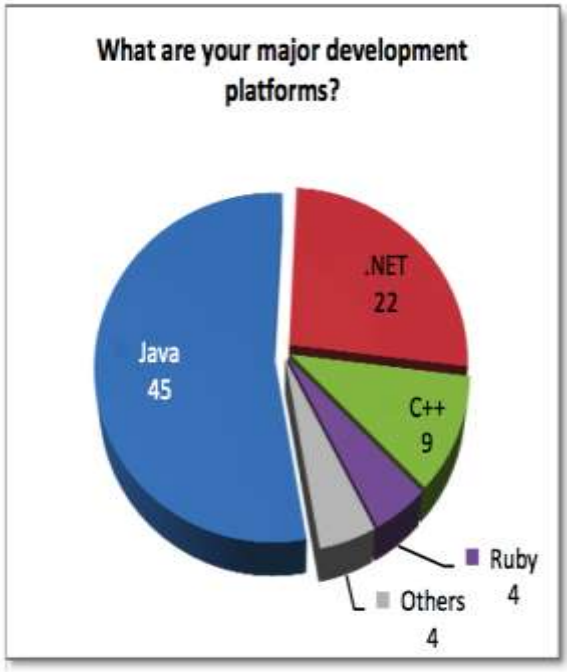
Build Servers: Survey Results

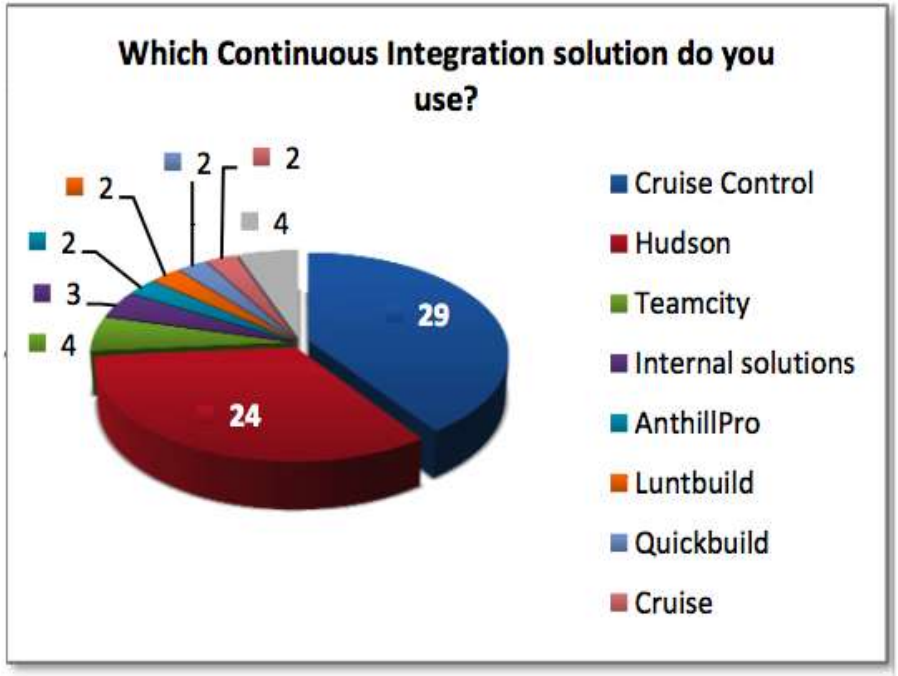
CI Servers from Fleisher (2009)

Solution	Company/Project	Open Source	First Release	URL
AnthillPro	Urbancode	-	2001	http://www.anthillpro.com
Bamboo	Atlassian	-	2007	http://www.atlassian.com
Continuum	Apache project	yes	2005	http://continuum.apache.org
Cruise *	ThoughtWorks	-	2008	http://studios.thoughtworks.com/cruise
CruiseControl	Sourceforge project	yes	2001	http://cruisecontrol.sourceforge.net
FinalBuilder	VSoft Technologies	-	2001	http://www.FinalBuilder.com
Hudson	java.net project	yes	2007	http://hudson.dev.java.net
Lunt build	Javaforge project	yes	2004	http://luntbuild.javaforge.com
Parabuild	Viewtier	-	2005	http://www.viewtier.com
Pulse	Zutubi	yes	2006	http://www.zutubi.com
Quick build **	PMEase	yes	2004	http://www.pmease.com
TeamCity	JetBrains	yes	2006	http://www.jetbrains.com/teamcity

* Cruise is a successor to CruiseControl
** Quick build is the commercial version of Lunt build

Build Servers:Survey Results





It is not surprising that CruiseControl fared so well in this survey, since it has been around the longest of any of the open source solutions (since 2001). What is worth noting is the strong showing by Hudson, a full-featured but rather new entry in the field (since 2007).

Automated Build Tools

Build automation is the scripting of tasks that developers perform in preparing their applications for deployment to runtime environments. Tasks include:

- Version control integration
- Dependency resolution
- Code quality analysis
- Compiling source code into binary code
- Packaging the binary code into deployable archives
- Running tests
- Deployment to different environments
- File system manipulation (creating, copying, and deleting files and directories)
- Creating documentation and or release notes

Automated Build Tools *(continued…)*

Tool	Platform	Description
Ant	Java	Most widely-used build tool for Java; extensive functionality; uses XML configuration; run from command line or various build servers and IDEs http://ant.apache.org/
Maven 1	Java	A software "project management" tool; manages project build, reporting, and documentation from a central piece of information; integrated dependency management http://maven.apache.org/maven-1.x/
Maven 2/3	Java	Same as Maven 1 but significantly enhanced with many new features and better performance; support for transitive dependencies; improved plug-in architecture http://maven.apache.org/
Gant	Groovy/Java	An alternative way of scripting things with Ant, uses the Groovy programming language rather than XML to specify the rules http://gant.codehaus.org/
Nant	.NET	.NET build tool. In theory it is like *make* without *make's* "wrinkles." In practice it is a lot like Ant, including using XML as a scripting syntax http://nant.sourceforge.net/
setup.py	Python	The standard setuptools module included in Python http://pypi.python.org/pypi/setuptools
Pip	Python	A modern replacement for the standard Python package installer (easy_install) with native support for most version control systems http://pip.openplans.org/
virtualenv	Python	Provides virtualized Python installs so you can have multiple Python projects on the same system without the possibility of interference or needing administration rights to install software http://pypi.python.org/pypi/virtualenv
Buildout	Python	A Python-based build system for creating, assembling and deploying applications from multiple parts, some of which may be non-Python-based. It lets you create a buildout configuration and reproduce the same software later http://www.buildout.org/

This list is never complete

Automated Test Tools: Unit Testing

Unit testing tools are ubiquitous. This is a small selection of the better-known products. For a comprehensive listing in a multitude of languages, see: http://en.wikipedia.org/wiki/List_of_unit_testing_frameworks.

Tool	Platform	Description
Junit	Java	The *de facto* standard for unit testing in Java http://junit.org/
TestNG	Java	Designed to overcome many of the perceived deficiencies of JUnit; supports test groups, dependencies, and parallel testing; integrates well with IDEs and build tools and servers http://www.testng.org
Cactus	Java	Cactus is a simple test framework for integration unit testing server-side java code (Servlets, EJBs, Tag Libs, Filters, etc. http://jakarta.apache.org/cactus/
DbUnit	Java	A JUnit extension targeted at database-driven projects that, puts the database into a known state between test runs; has the ability to export and import your database data to and from XML datasets http://www.dbunit.org/
Nunit	.NET	Includes GUI, command line, integrates into VisualStudio with ReSharper http://www.nunit.org
PyUnit	Python	The Python language version of JUnit; part of Python's standard library http://pyunit.sourceforge.net/
Nose	Python	A discovery-based unittest extension; includes many options including code test coverage http://somethingaboutorange.com/mrl/projects/nose/
MXUnit	ColdFusion	Unit Test Framework and Eclipse Plugin for CFMX http://mxunit.org/
CFUnit	ColdFusion	CFUnit is a unit testing framework for ColdFusion modeled after the JUnit framework http://cfunit.sourceforge.net/
CFCUnit	ColdFusion	A full-fledged framework for unit testing ColdFusion code based on JUnit http://www.cfcunit.org/cfcunit/

Automated Test Tools: Performance Testing

Most compilations of software performance testing tools on the Internet do not differentiate among the different types, so tools for load testing, stress testing, stability testing, etc., are lumped together under “performance testing.” For example, see http://www.opensourcetesting.org/performance.php.

Tool	Platform	Description
JMeter	Java	An open source desktop application designed to load test functional behavior and measure performance; 100% Java; originally designed for testing Web Applications but has since expanded to other test functions; extensive documentation describes how to use its many features and provides many examples; a special Ant task allows for automation by executing load tests while providing a way to pass in optional parameters and properties (see Duvall, 2008). http://jakarta.apache.org/jmeter/
Funkload	Python	Automated functional testing and detailed performance analysis http://funkload.nuxeo.org/
PyLot	Python	PyLot: performance and scalability testing, with correct result validation http://www.pylot.org/gettingstarted.html

Automated Test Tools: Web Application Functional Testing

Among the many compilations of links to functional test sites found was the one at SoftwareQATest.com (http://www.softwareqatest.com/qatweb1.html#FUNC).

Tool	Platform	Description
Selenium Remote Control (RC) Selenium Integrated Development Environment (IDE)	Web/Java	Created in 2004, Selenium RC is a test tool that allows the tester to write automated web application UI tests in any programming language against any HTTP website using any mainstream JavaScript-enabled browser. Selenium RC comes in two parts. •A server that automatically launches and shuts down browsers and acts as a HTTP proxy for web requests from them. •Client libraries for the preferred computer language (Java, Python, Perl, Ruby, PHP, C#, etc.). Selenium IDE is a complete integrated development environment for recording, editing, and running Selenium tests. It is implemented as a Firefox plug-in. Tests written with Selenium IDE may be run later on Selenium RC, since both products use the same test execution engine, known as Selenium Core. http://seleniumhq.org/projects/remote-control/ http://seleniumhq.org/projects/ide/
Windmill	Web/Python	Windmill is a web testing tool designed to automate testing and debugging web applications http://www.getwindmill.com/

Automated Test Tools: Web Application Functional Testing *(continued…)*

Tool	Platform	Description
WebTest	Web/Java	Canoo WebTest is a free open source tool for automated testing of web applications that has existed since 2001. WebTest is plain Java and runs everywhere as long as there is a Java development kit installed on the operating system. WebTest scripts are Ant scripts, which means an easy integration path to CI servers such as CruiseControl and Hudson. Simple and fast, no display is needed (uses HtmlUnit), although Javascript support is not as good as in "normal" browser. Easily extended using Java or Groovy http://webtest.canoo.com/
twill	Web/Python	twill is a simple scripting language that allows users to browse the Web from a command-line interface http://twill.idyll.org/
Funkload	Web/Python	Automated functional testing and detailed performance analysis http://funkload.nuxeo.org/

Automated Test Tools: Security Testing

SoftwareQATesting.com also lists many projects in the subject area of Web site security testing (see http://www.softwareqatest.com/qatweb1.html#SECURITY).

Tool	Platform	Description
ratproxy	Web	Ratproxy is a web application security auditing tool. It is a passive proxy, watching data sent between a browser and a web application. Detects and prioritizes broad classes of security problems, such as dynamic cross-site trust model considerations, script inclusion issues, content serving problems, insufficient XSRF and XSS defenses, and much more. When you test a web application, either manually or using Selenium, ratproxy can watch what you are doing and report any potential security holes it sees. http://code.google.com/p/ratproxy/
Paros	Web	Another proxy with active scanning support and varying degrees of automation http://www.parosproxy.org/
WebScarab	Web	Another proxy with active scanning support and varying degrees of automation; available through the Open Web Application Security Project (OWASP), a worldwide free and open community focused on improving the security of application software http://www.owasp.org/index.php/Category:OWASP_WebScarab_Project

Code Inspection Tools

There are many tools that provide code inspection via static and dynamic analysis of the source code. A partial listing is shown:

Tool	Platform	Description
Checkstyle	Java	Checkstyle is a development tool to help programmers write Java code that adheres to a coding standard. It automates the process of checking Java code to spare humans of this boring (but important) task. http://checkstyle.sourceforge.net/
JavaNCSS	Java	JavaNCSS is a simple command line utility that determines the lengths of methods and classes (a metric for complexity) by examining the Java source code. http://www.kclee.de/clemens/java/javancss/
JCSC	Java	JCSC is a powerful tool to check source code against a highly definable coding standard and potential bad code; similar to Checkstyle. http://jcsc.sourceforge.net/
PMD	Java	PMD scans Java source code and looks for potential problems such as possible bugs, dead code, suboptimal code, overcomplicated expressions, and duplicated code. http://pmd.sourceforge.net/
FindBugs	Java	A program which uses static analysis to look for bugs in Java code; looks for bug patterns, inspired by real problems in real code. http://findbugs.sourceforge.net/
Sonar	Java	Sonar leverages the existing ecosystem of quality open source tools (e.g., Checkstyle, PMD, Maven, Cobertura, etc.) to offer a fully integrated solution to development environments and continuous integration tools. http://sonar.codehaus.org/
Clover	Java	Clover measures code coverage generated by system tests, functional tests or unit tests, allowing you to improve test quality and find bugs sooner. (Commercial). http://www.atlassian.com/software/clover/
Cobertura	Java	Cobertura is a free Java tool that calculates the percentage of code accessed by tests. It can be used to identify which parts of a Java program are lacking test coverage. It is based on jcoverage. http://cobertura.sourceforge.net/

Code Inspection Tools (continued…)

Tool	Platform	Description
EMMA	Java	Another free Java code coverage tool. http://emma.sourceforge.net/
FxCop	.NET	FxCop is an application that analyzes managed code assemblies and reports information about the assemblies, such as possible design, localization, performance, and security improvements. http://msdn.microsoft.com/en-us/library/bb429476(VS.80).aspx
Ncover	.NET	NCover is a test coverage tool for the .NET platform; (Commercial). http://www.ncover.com/
Pylint	Python	Pylint is a python tool that checks if a module satisfies a coding standard; checks line-code length, variable name standard compliance, and more. http://www.logilab.org/857
Pyflakes	Python	Pyflakes is a program that analyzes Python programs and detects various errors. It works by parsing the source file rather than importing it, so it is safe to use on modules with side effects. http://freshmeat.net/projects/pyflakes/?branch_id=60662&release_id=208696
Clone Digger	Python	Reports code in Python or Java programs which appears to be very similar (even with slight variations); a good place to look for bugs and bad practice. http://clonedigger.sourceforge.net/
JSLint	Python	The popular JavaScript lint utility from jslint.com (http://jslint.com) can be run from the command-line using Rhino. http://www.jslint.com/rhino/index.html
CSS Validator	Python	Can be run as a command-line tool as of earlier this year. http://jigsaw.w3.org/css-validator/DOWNLOAD.html
Coverage.py	Python	Provides annotated code coverage reports and has been integrated into several of the common Python test suites, such as Nose. http://nedbatchelder.com/code/modules/rees-coverage.html
QueryParam Scanner	ColdFusion	qpScanner is a simple tool that scans the ColdFusion codebase checking to see if there are any CFML variables in queries that are not contained within a cfqueryparam tag. http://qpscanner.riaforge.org/
varScoper	ColdFusion	varScoper is a code scanning tool that can be used to identify variables that are not explicitly scoped to be local or global to a ColdFusion function. http://varscoper.riaforge.org/

Introducción

Escenarios

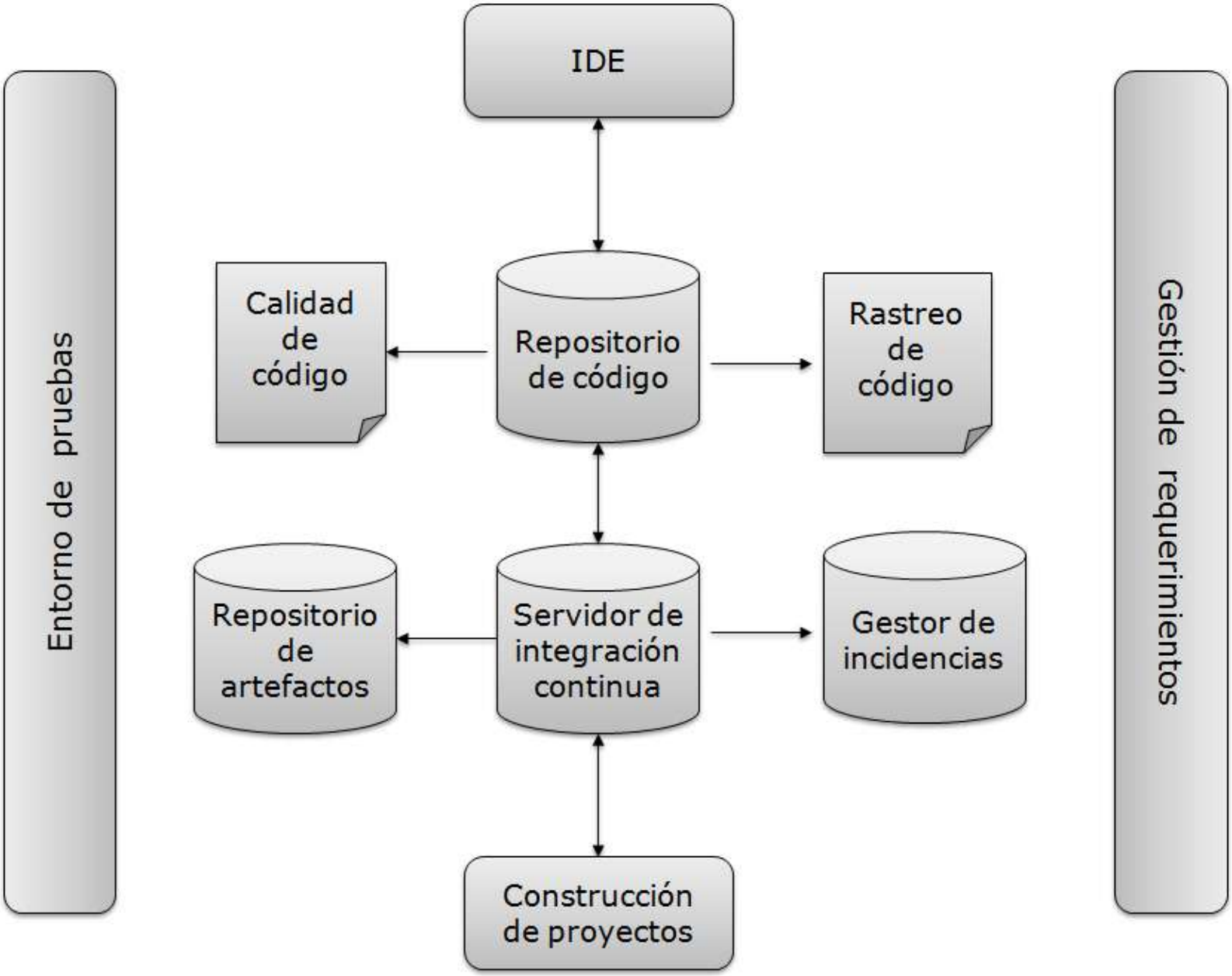
Integración continua

- Definición
- Arquitectura
- Builds
- Servidor de CI
- Pruebas y CI
- Inspección
- Feedback
- Pros/cons
- Herramientas
- Lo esencial

Resumen

Bibliografía

Application Lifecycle Management



Índice

Introducción

Escenarios

Integración continua

Resumen

Bibliografía

Resumen

- ¿Qué hemos aprendido?
  - Construir/ensamblar software es algo más que "compilar"
  - La integración es un problema que hay que tratar
  - La integración continua aboga por hacerlo desde el primer momento
  - Automatizar la construcción se convierte en fundamental
- ¿Qué veremos en las siguientes lecciones?
  - Gestión de entregas, liberaciones, cambios…
  - En práctica herramientas de construcción, servidores de integración continua

Índice

Introducción

Escenarios

Integración continua

Resumen

Bibliografía

Bibliografía