Ecosistemas Software de soporte a la Integración Continua

Juan José Guerra

Abril 2011

Introducción

Para dotar de las herramientas necesarias en un **proyecto software**, en general una organización dispone de las siguientes alternativas:

- Herramientas "al uso": entornos de desarrollo, servidores web, herramientas de testing, herramientas de gestión, etc.
- Infraestructuras o plataformas de desarrollo "en la nube" (Cloud Computing). Ejemplos: Azure, Vmforce, etc.
- Ecosistemas Software.

Las opciones no son excluyentes ya que pueden complementarse. Dependerá de las necesidades concretas de cada proyecto: entorno tecnológico, herramientas de desarrollo y de soporte necesarias, etc.

La tendencia actual es adoptar las 2 últimas opciones: Entornos de desarrollo "en la nube" propietarios y Ecosistemas Software.

Nos centraremos en los Ecosistemas Software, primero desde un enfoque general y luego, enfocaremos exclusivamente las herramientas de Integración Continua.

Ecosistemas Software

¿Qué es un Ecosistema Software?

- Un Ecosistema Software es una plataforma o infraestructura integrada de herramientas de soporte a los procesos de desarrollo y de gestión de proyectos software de una organización.
- Otra definición (www.manuelrecena.com): Un Ecosistema Software es un espacio de trabajo en el que conviven una serie de herramientas que, acompañadas de unas buenas prácticas, permiten a un equipo de desarrollo modelar una metodología de trabajo.
- Las herramientas del Ecosistema Software nos ayudan a controlar el desarrollo de software, asegurar la calidad de los productos software, y a ser mas productivos.
- Dependiendo del alcance del Ecosistema Software que se implante, las herramientas del mismo darán soporte a las tareas de desarrollo del software (análisis, diseño, implementación, testing, etc.) y a las de gestión del proyecto.

Ecosistemas Software

Evolución

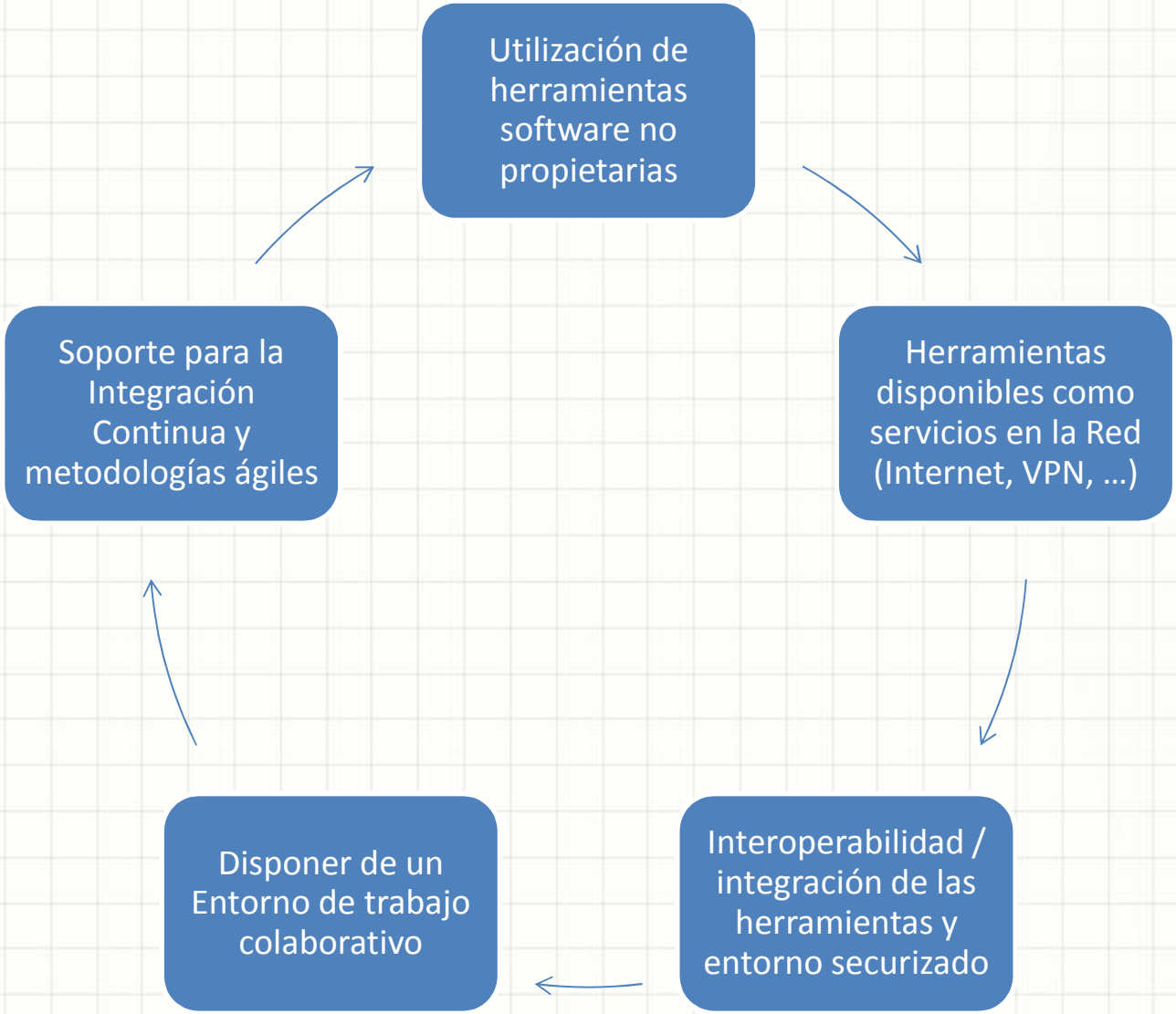
Tradicionalmente cada proyecto requería de su propio Ecosistema Software, que era definido y puesto en marcha “al uso” en el inicio del proyecto. Algunas de las herramientas podrían estar ya disponibles.

En función de las necesidades comunes de los proyectos se seleccionan y/o implantan una serie de herramientas para su uso en los proyectos, disponibles como servicios en la Red o como máquinas virtuales.

Se crean Ecosistemas para cubrir mas tipologías de proyectos, se amplían sus capacidades, se cubren los aspectos de seguridad e integración con otros sistemas “corporativos”, se disponen en la “nube”, etc.

Ecosistemas Software

Tendencias (I)



Ecosistemas Software

Tendencias (II)

El concepto de Ecosistema Software puede ser mas amplio y modelarse como:

El conjunto de infraestructuras hard, soft y de comunicaciones disponibles en una organización que dan soporte a la mayor cantidad de proyectos software posibles

- Entornos tecnológicos: Java, .NET, PHP, etc.
- Tipologías de herramientas: gestión de proyectos y equipos, desarrollo, calidad y testing, etc.
- Entornos de despliegue: bases de datos, servidores web, etc.

Las herramientas soft estarán disponibles …

- Como servicios de Red y/o en entornos virtualizados. Convergencia de Ecosistemas Software hacia modelos de Cloud Computing.
- Otras en cambio requieren de instalación. Por ejemplo, las herramientas de desarrollo.
- Aunque la virtualización puede incluso llegar a los herramientas de desarrollo: Máquinas virtuales con el entorno de herramientas necesarias para un tipo de proyecto. Ejemplo: VM para desarrollo de software para móviles (Android, iOS).

Ecosistemas Software

Herramientas (I)

Software Development	• Herramientas análisis y diseño, construcción software, testing, bbdd, runtime, utilidades, etc.
Source Code Management (SCM)	• Gestión del código fuente, perfiles de configuración, documentos y ficheros en general. Control de versiones.
Repository Manager	• Administración de repositorios locales y externos de artefactos software.
Build Tool	• Sistema para la compilación y empaquetado del software.
Continuous Integration & Build	• Soporte a la integración continua y la ejecución de labores automáticas de verificación y validación del código.
Source Code Quality	• Verificación y validación del código fuente. Métricas y reportes de análisis del código.

Ecosistemas Software

Herramientas (II)

Project Management	• Soporte tareas de planificación, seguimiento y control, gestión de incidencias y problemas, gestión del cambio, gestión de la configuración, gestión del testing, etc.
Content Management System	• Gestor documental y de ficheros en general, wikis, etc.
System/App. Management & Infraestructure monitoring	• Gestión y monitorización de sistemas, aplicaciones e infraestructuras.
Entorno / herramientas colaborativas de trabajo	• Web proyecto, wikis, foros, mensajería interna, listas de correo, etc.
Entornos de despliegue	• Físicos o virtualizados . Para testing, maquetas y prototipos, explotación, etc.
Otras herramientas	• Single Sign On, Integración LDAP, Seguridad y Respaldo, etc.

Integración Continua

El problema de la integración en proyectos software

Los proyectos de software implican un gran número de ficheros que necesitan ser integrados juntos para construir uno o varios productos.

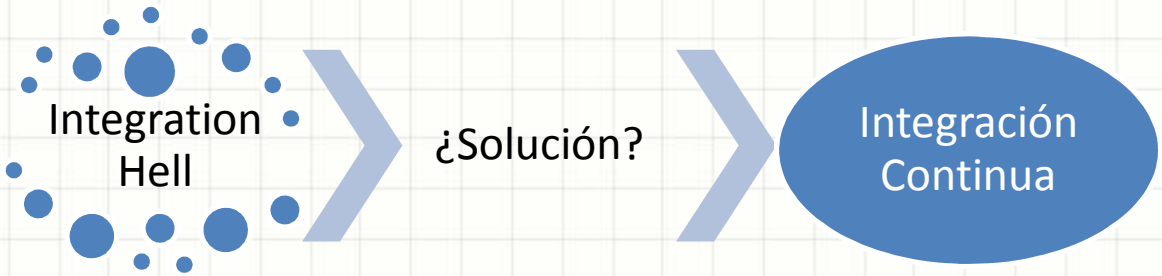
El proceso de integración puede representar un tiempo valioso para nuestro proyecto. Además realizar esta tarea es repetitiva.

La integración puede entenderse como cualquier tarea relacionada con la construcción y entrega de los productos software: sincronización de ficheros, compilación, testing, empaquetado, generación de informes y documentación, despliegue / publicación

En cada una de las tareas de integración pueden surgir problemas. Por ejemplo, el código desarrollado por un desarrollador puede causar conflictos sobre otro código que ya esté implementado.

Cuanto más tiempo se tarda en integrar, más aumenta el riesgo de que existan problemas de integración.

Los problemas de integración afectan al coste y al tiempo en los proyectos software.



Integración Continua

Introducción a la IC

La **Integración Continua** (IC) consiste en la practica de automatizar tareas que son repetitivas y que éstas sean ejecutadas lo más a menudo posible y de manera automática para así poder detectar fallos cuanto antes.

De esta forma, nos permite estar siempre informado sobre el estado del software y controlamos la evolución del mismo de forma automatizada.

La IC sustituye las pesadas fases de integración con otras pequeñas y frecuentes.

El objetivo final es minimizar el esfuerzo de rehacer el trabajo manteniendo el proceso de desarrollo funcionando, y con ello reducir el coste y tiempo del proyecto.

La IC está asociada con las metodologías de testing continuo (TDD), programación extrema (XP) y metodologías ágiles en general (Scrum, AM, AUP, etc.).

Está centrada en disminuir la carga de trabajo a los desarrolladores. Para ello, se utiliza al menos un servidor específico (**Servidor IC**).

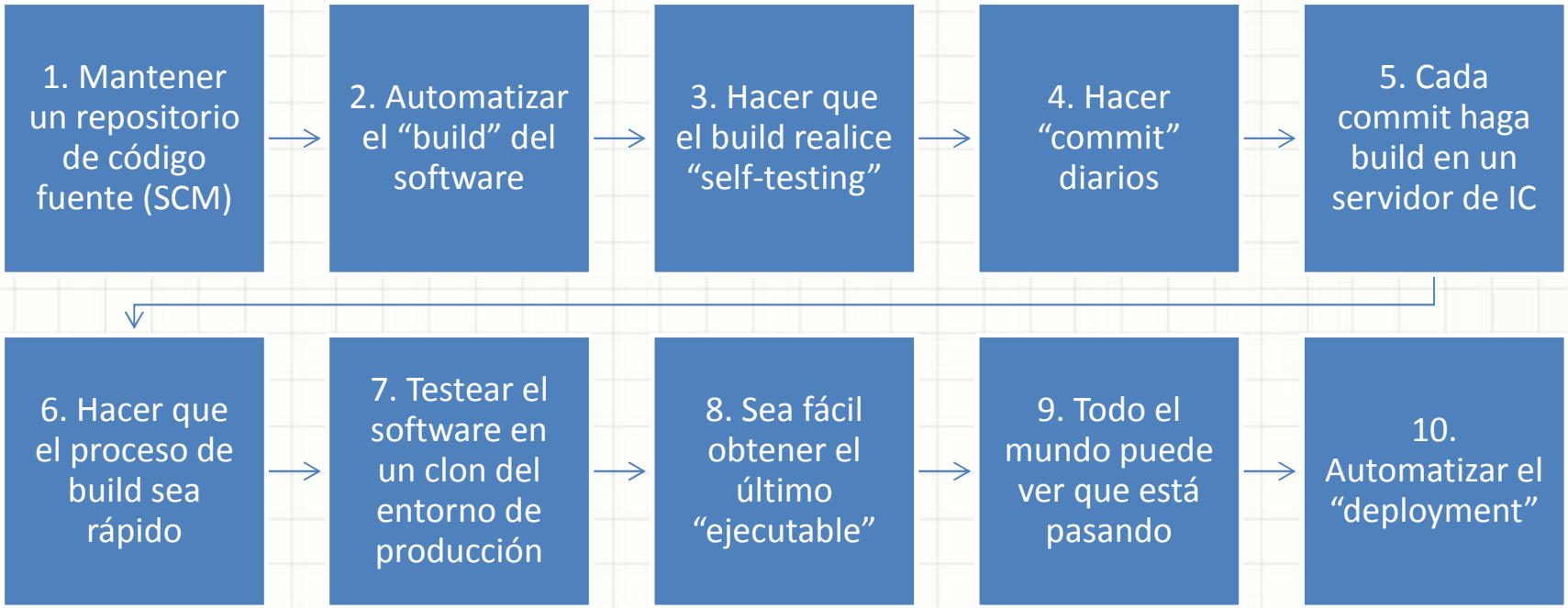
El Servidor de IC es la clave

Integración Continua

10 prácticas de IC según Martin Fowler

- Según Martin Fowler, escritor del libro "Continuous integration – Improving software quality and reducing risk", el concepto de integración continua se define como sigue:

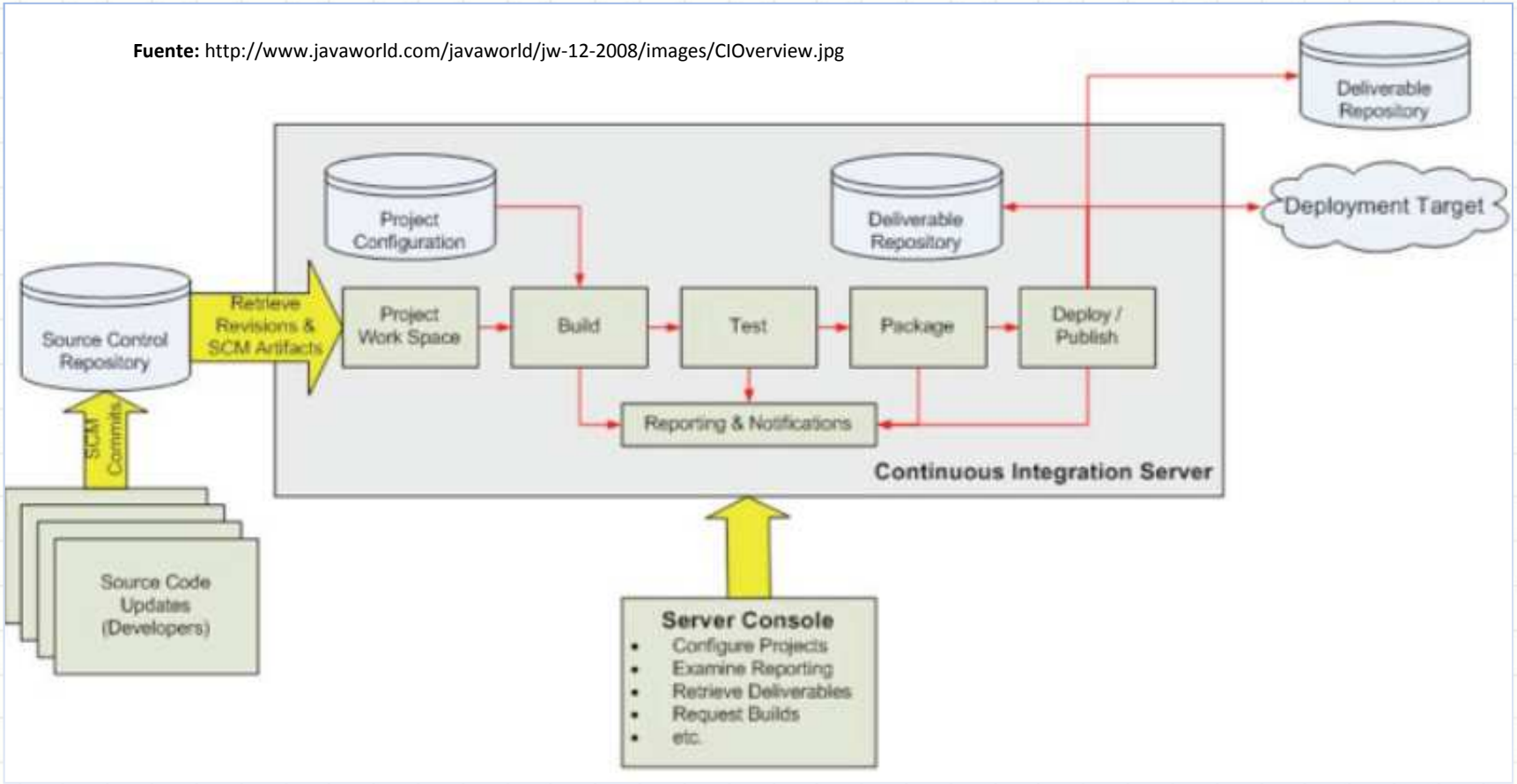
  "Es una práctica de software donde los miembros del equipo de trabajo integran su código de manera frecuente, dando así multiples integraciones por día. Donde cada integración forma parte de un Build (Integración, Construcción, Pruebas, Despliegue, entre otras cosas)."



Integración Continua

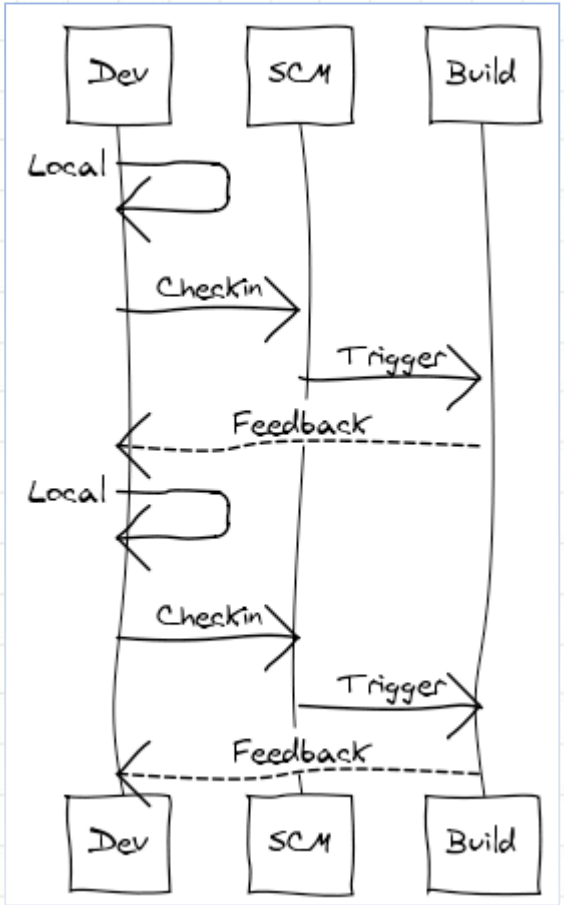
Esquema general

- Generalmente la IC consta de las siguientes partes: Source Control, Build, Test, Package, Deploy / Publish.



Integración Continua

Flujo clásico de IC para el desarrollador

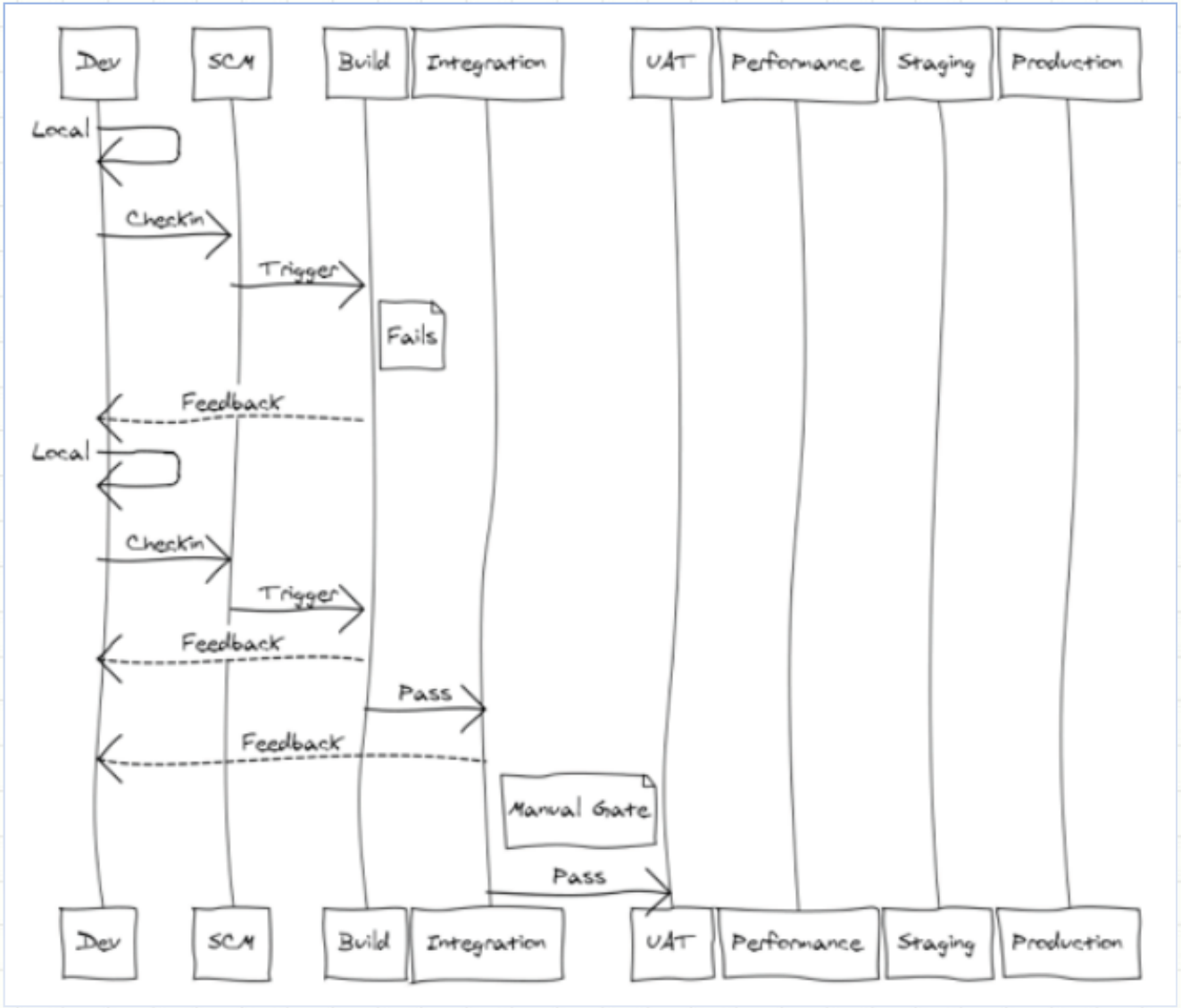


1. Checkout / Update desde SCM.
2. Implementar nueva funcionalidad o cambio en el software.
3. Hacer “build” en la máquina local del desarrollador. Repetir pasos 2-3 hasta que pasen los tests.
4. Hacer “merge” con los últimos cambios en SCM. Arreglar posibles problemas de “merge” y hacer “rebuild” hasta que pasen los tests.
5. Hacer “commit” al SCM.
6. Hacer “build” en una máquina de pruebas. Corregir errores y problemas de integración.

Fuente: http://www.slideshare.net/ChristopherRead/continuous-integration-build-pipelines-and-continuous-deployment

Integración Continua

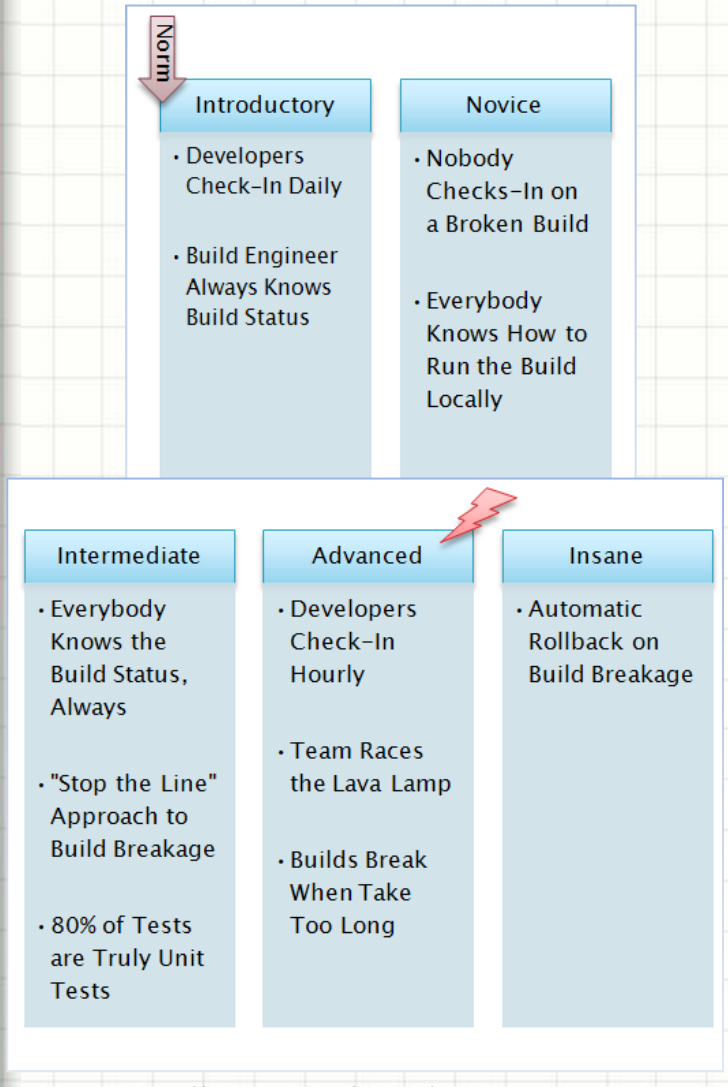
Flujo de IC general: desarrollador → tests de aceptación (UAT)



Fuente: http://www.slideshare.net/ChristopherRead/continuous-integration-build-pipelines-and-continuous-deployment

Integración Continua

Modelo de madurez de IC



Fuente: http://pauljulius.com/images/ci-maturity-culture.PNG

Fuente: http://www.urbancode.com/html/resources/elements-enterprise-ci.html

Integración Continua

Ventajas e inconvenientes

Ventajas

- Reducción del tiempo de integración y detección de errores lo más pronto posible.
- Pruebas inmediatas tras un cambio en el código.
- Disponibilidad del código para test, demos, etc.
- Monitorización continua de las métricas de calidad del software.

Desventajas

- Necesidad de al menos un servidor dedicado para la IC.
- El impacto inmediato al subir código erróneo provoca que los desarrolladores no hagan tantos commits como sería conveniente.

Conclusiones

- La IC está enfocada a disminuir el riesgo y a la detención y solución temprana de problemas.
- La IC nos brindará información en todo momento.
- El éxito de la IC esta fuertemente ligada con las serie de pruebas (Cobertura) que se tiene en el proyecto.
- Permite una rápida retroalimentación de nuestro proyecto.

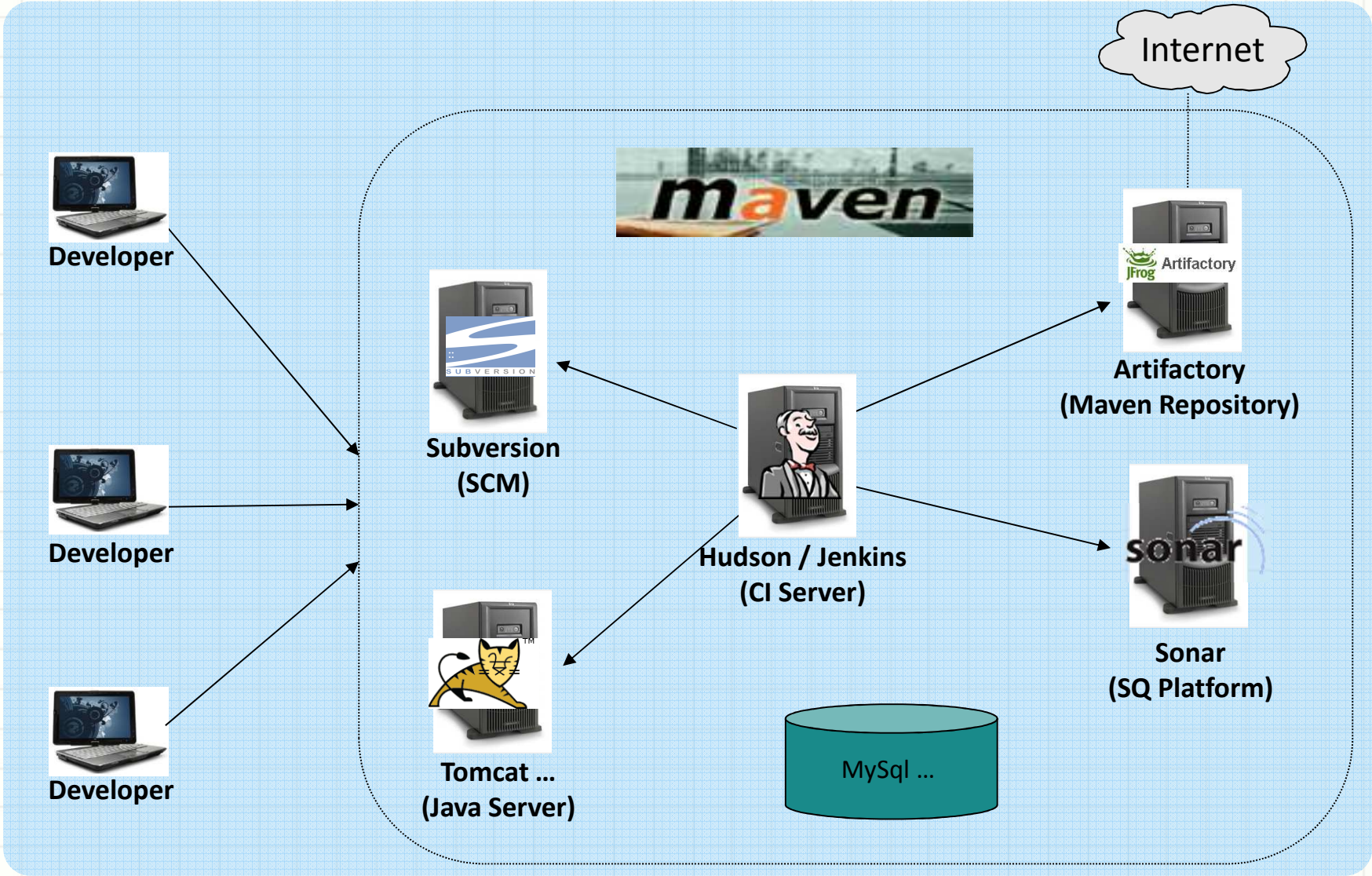
Integración Continua

¿Qué ecosistema software se necesita?

Source Code Management (SCM)	Build Tool & Testing	Repository Manager	Software Quality Platform	Continuous Integration Server
Subversion, CVS, Git, Mercurial, VSS, TFS	**Maven,** Ivy, Gradle, Ant, Make, Rake, MSBuild, Scripts Junit, Nunit, MSTest	**Artifactory**, Nexus, Archiva	**Sonar,** Plugins Maven, VS-ALM	**Hudson / Jenkins**, Continuum, Cruise Control, Bamboo

Un modelo de IC para proyectos Java

Esquema general



Un modelo de IC para proyectos Java

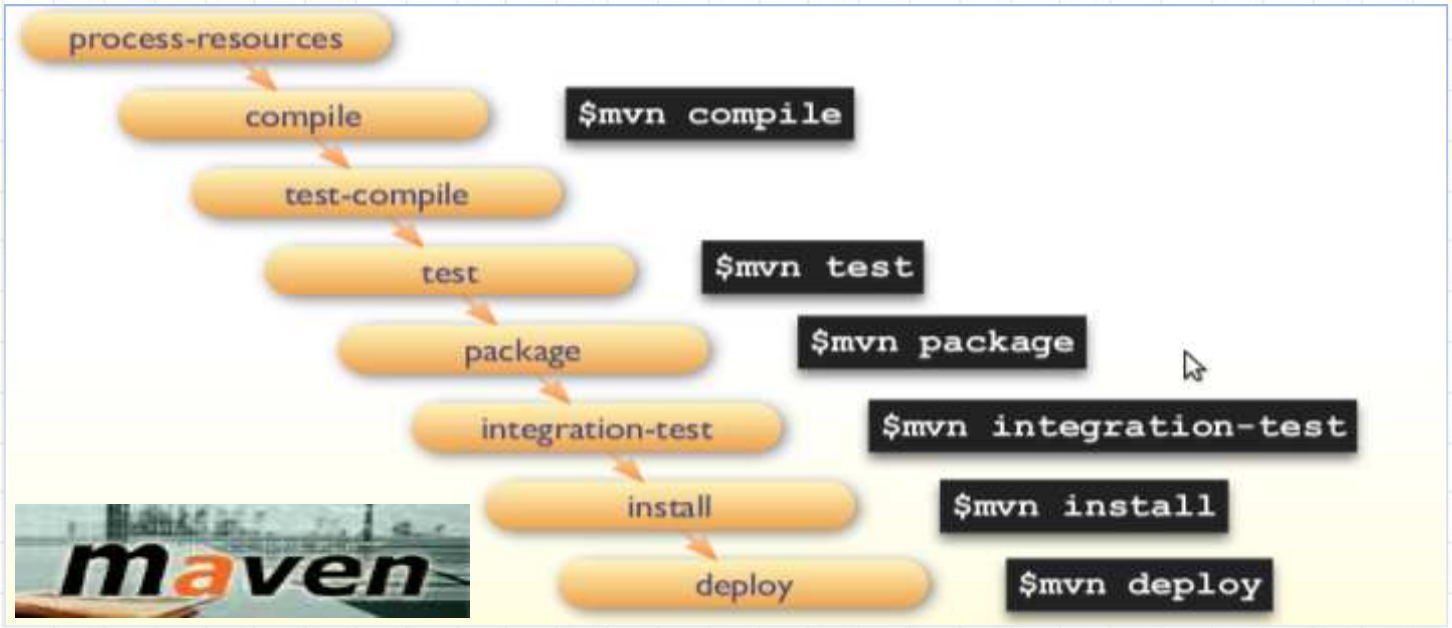
Subversion

- Subversion es un sistema de control de versiones (SCM) y es software libre.
- Nos permite tener nuestro código centralizado, descargar, actualizar y subir código que se encuentra en este repositorio.
- Hacer frecuentes commits acelerará la construcción del proyecto, además de encontrar rápido errores.
- Se le conoce también como svn por ser el nombre de la herramienta utilizada en la línea de órdenes. Existe una variedad de clientes de Subversion adicionales (Tortoise, Eclipse plugin, Netbeans plugin, etc.).
- Está soportado por el servidor de IC Hudson / Jenkins.

Un modelo de IC para proyectos Java

Maven

- Herramienta para la gestión y construcción de software Java.
- Es la base que se tiene para trabajar para la IC, ya que cuenta con comandos de compilacion, deploy, test, etc. Con un solo comando puede construir un proyecto.
- Permite hasta publicar el artefacto en un repositorio local (install) o remotamente en un servidor (deploy).
- Utiliza un Project Object Model (POM) para describir el artefacto de software a construir, sus dependencias de otros módulos y componentes externos, y el orden de construcción de los elementos.



Un modelo de IC para proyectos Java

Artifactory

- Artifactory es una herramienta open source que funciona como gestor de repositorios de artefactos software (Repository Manager).
  - Por un lado, actúa como proxy entre las “Build Tools” (Maven, Ant, Ivy, Gradle) y los repositorios externos, gestionando las dependencias entre software de terceros.
  - Por otro, sobre Artifactory se pueden desplegar las dependencias propias y de terceros que necesiten los proyectos (en Maven por ejemplo) para su ciclo de vida.
- De esta forma, los clientes Maven (y otros) no tienen que resolver las dependencias directamente sobre los repositorios remotos, ya que Artifactory realiza esa tarea, con lo que únicamente deben conectarse a Artifactory.
- Plenamente soportada por Hudson / Jenkins y Maven.

Un modelo de IC para proyectos Java

Sonar

- Sonar es una plataforma open source de calidad del software.
- Permite la verificación de la calidad del código Java, calculando y mostrando en formato web las principales métricas de calidad del software de nuestro proyecto.
- Herramientas como PMD, CheckStyle, FindBugs, Cobertura, etc; ya vienen embebidas.
- Cubre los 7 ejes de la calidad: Diseño de la arquitectura, Comentarios, Bugs potenciales, Reglas de codificación, Complejidad, Pruebas unitarias y Duplicidad.
- Está soportada por Hudson / Jenkins y Maven.

sonar

Un modelo de IC para proyectos Java

Hudson / Jenkins

- Herramienta open source que nos ayuda a definir y monitorizar la ejecución de tareas repetitivas de Integración Continua en proyectos software.
- Soporta herramientas de control de versiones como CVS, Subversion, Git y Clearcase y puede ejecutar proyectos basados en Ant, Maven, Ivy y Gradle.
- Dispone de una gran cantidad de plugins que se pueden utilizar: Artifactory, Sonar, Redmine, Trac, sftp, Selenium, Tomcat, Jboss, etc…
- La administración se realiza totalmente usando una interfaz web lo cual facilita enormemente su configuración.
- Gran cantidad de reportes para JUnit y TestNG.
- Soporta notificación via IM, e-mail y RSS.
- Existen 2 variantes: Hudson (Oracle) y Jenkins (desarrolladores Hudson original).

HUDSON
Extensible continuous integration server

Jenkins

Un modelo de IC para proyectos Java

Tomcat, MySQL y entornos de desarrollo

- El servidor java Tomcat y la base de datos MySQL servirán tanto para la ejecución desde Hudson / Jenkins de los tests que requieran de un contenedor servlet (tests de integración), como para el despliegue de aplicaciones para demos, pilotos, etc.
- En caso de que un proyecto requiera un entorno operacional específico, como el de réplica del de producción, y en dicho entorno se utilicen otros servidores java (Jboss, glassfish, …) y de base de datos (postgres, oracle, …), existen plugins en Hudson / Jenkins para configurar las tareas de IC sobre una gran cantidad de servidores y herramientas.
- Sobre los entornos IDE de desarrollo Java, actualmente existen plugins para desarrollar con proyectos Maven y servidores Hudson / Jenkins desde Eclipse, Netbeans, etc.

Un modelo de IC para proyectos Java

Ejemplo: VM de IC

- Máquina virtual con S.O. Windows Server o Linux
- Software de base:
  - Java SDK: JDK 5, 6 y 7
  - Build Tools: Maven 2 y 3, Ant
  - LDAP Server: Apache Directory Server
  - Web Server: Apache HTTP Server
- Software de IC:
  - Software Configuration Management (SCM): Subversion
  - Repository Manager: Artifactory
  - Continuous Integration Server: Hudson / Jenkins
  - Quality Management Platform: Sonar
- Software adicional:
  - Java Servlet/JSP Container: Tomcat 6 y 7
  - Database Server: MySQL
- El Servidor LDAP permite el control de acceso centralizado de los usuarios a las herramientas de Integración Continua.

¿SUGERENCIAS?

Juan José Guerra
http://es.linkedin.com/in/guerram
guaose@gmail.com