Integración Continua
Conceptos claves e importancia de su aplicación
Herramientas para su utilización

Achával Vinuesa, Julio Guillermo; Liendo Loker, Tomas; Monteros, Facundo; Ovejero, Miguel; Revello, Juan Manuel; Velasco, Joaquín.
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL CÓRDOBA
{julioachavalvinuesa, TomasLLoker, facundomonteros24, miguelovejero50, juanmanuelrevello, joacovelascob}@gmail.com

Resumen

Para llevar a cabo un proyecto de software bajo los lineamientos de las metodologías ágiles, es imprescindible la aplicación de la CI, en la que los miembros de un equipo integran sus desarrollos de forma frecuente, siendo cada integración validada exhaustivamente de forma automática para detectar de manera temprana cualquier error que pueda ocurrir en el proceso.

Como cualquier otra práctica de la Ingeniería del Software, integrar componentes de software, es una actividad compleja. En este contexto, la CI contribuye con el desarrollo ágil, brindando un esquema que posibilita la realización de integraciones a medida que se lleva a cabo el desarrollo, generando incrementos pequeños y mostrando los resultados obtenidos.

Palabras clave: *software, ágil, integración, frecuente.*

Introducción

En el marco de las Metodologías Ágiles para la gestión de proyectos de software, surge la necesidad de citar los conceptos de CI/CD. Ver Figura 1.

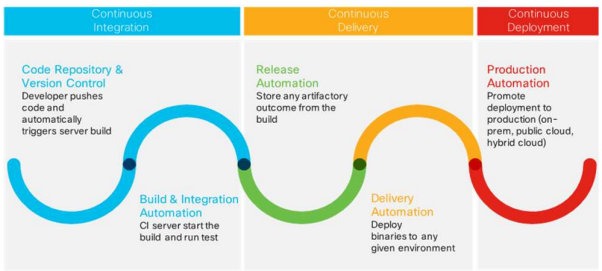

Figura 1. Continuous Integration, Delivery and Deployment

El presente informe tiene por fin profundizar sobre la Integración Continua, analizando en qué consiste la misma y cuál es su importancia para el desarrollo del software.

Los principios ágiles que brindan contexto a la implementación de esta práctica son principalmente los siguientes:

Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.

Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.

Anteriormente, los desarrolladores programaban de manera independiente y al finalizar se realizaba una integración de todo el código. Esta tarea, que naturalmente es compleja, se complejiza aún más cuando la integración comprende a una gran cantidad de código y la misma se realiza de forma manual. La consecuencia de llevar a cabo esta práctica es un incremento en la probabilidad de ocurrencia de errores al momento de integrar, lo que implica a su vez una pérdida de tiempo para resolverlos, lo cual en el contexto del desarrollo ágil es cuestionable.

Desarrollo

¿Qué es la Integración Continua?

La Integración Continua es una práctica, propuesta inicialmente por Martin Fowler, que consiste en hacer integraciones frecuentes en un proyecto para así poder detectar fallos de forma temprana.

Fowler la define como: "*...una práctica de desarrollo de software en la cual los miembros de un equipo integran su trabajo con frecuencia, por lo general cada persona integra su trabajo por lo menos una vez al día dando lugar a múltiples integraciones por día. Cada integración es verificada por una build automática (incluyendo pruebas) para detectar errores de integración lo más rápidamente posible. Muchos equipos encuentran que este enfoque conduce a la reducción de manera significativa de los problemas derivados de integración y permite a los equipos desarrollar software con mayor rapidez*." [1].

Prácticas de la Integración Continua

La Integración Continua está basada fundamentalmente en tres buenas prácticas: mantener un repositorio de código fuente único, realizar pruebas automáticas y comprometerse con integrar los cambios del software de manera frecuente.

Los proyectos de software, usualmente, implican varios archivos que deben organizarse en forma conjunta para conformar a un producto. Una de las tareas más costosas a la hora de desarrollar software es gestionar estos archivos y por consiguiente el código fuente de este. Para ser más eficientes, en la actualidad es indispensable la utilización de herramientas que permitan gestionar el código fuente y el versionado del producto.

Una de las características de los sistemas de control de versiones es que permiten crear múltiples "ramas", para manejar diferentes flujos de desarrollo. Generalmente se tiene una línea o rama principal, que se corresponde con la rama de desarrollo actual del proyecto. Los desarrolladores implementan nuevas funcionalidades en ramas paralelas, y una vez que las mismas se finalizan (fueron desarrolladas y testeadas) se integran a la rama principal. Ver figura 2.

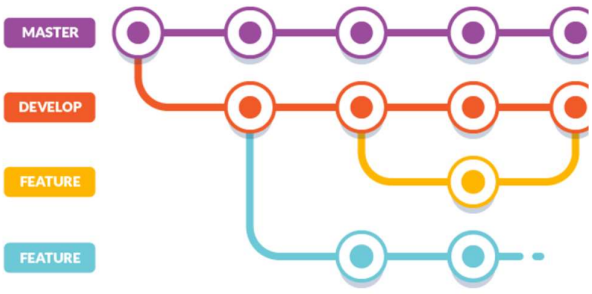

Figura 2. Ramas de desarrollo en paralelo

Una buena forma de detectar errores de forma temprana y eficiente es incluir pruebas unitarias en el proceso de compilación. Una prueba unitaria es simplemente un fragmento de código que llama a un método (el método a probar) con un entrada predefinida y comprueba si el resultado es el esperado. Si el resultado es correcto, informa de éxito, de lo contrario informa error. La prueba unitaria, como el nombre lo indica, prueba unidades de código pequeñas y aisladas.

La compilación automatizada es otra práctica que incluye la Integración Continua, la cual es implementada mediante servidores de automatización como Jenkins. Si al menos una prueba unitaria falla en el proceso de compilación, entonces la compilación falla.

La integración se trata principalmente de comunicación, la cual permite a los desarrolladores informar a otros desarrolladores sobre los cambios que han realizado. Como se puede denotar a continuación, en la Figura 3.

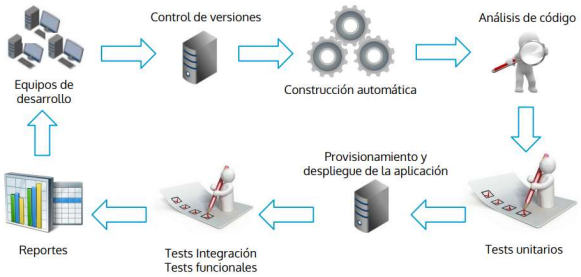

Figura 3. Ciclo de integración continua

Condiciones previas para implementar una correcta Integración Continua

A la hora de implementar CI se debe tener en cuenta ciertos criterios previos, entre ellos:

- Concientizar al equipo de realizar integraciones con cierta frecuencia, esto se relaciona directamente con esencia de las metodologías Ágiles.
- Realizar pruebas unitarias completas y correctas a cada porción de código que se integrará.
- La gestión de Configuración y la Gestión de despliegue documentada y divulgada deben estar perfectamente incorporadas al equipo de trabajo para que la integración continua se pueda desarrollar correctamente y para que el proceso que implica la misma, pueda, vaya la redundancia integrarse y estandarizarse dentro de los procesos de la organización y de los equipos de trabajo.
- Los integrantes del equipo de trabajo deben tener conocimientos en Arquitectura, altos estándares de codificación, formación y vocación en metodologías Ágiles con capacidad para el trabajo colaborativo y proactivos y predispuestos a

solucionar errores y problemas en cambios y funcionalidades que no desarrollaron.

- Se debe definir quien o quienes se encargarán de la configuración del Servidor de Integración Continua y de cada uno de sus componentes, como también el manejo de grupos, roles y usuarios con sus permisos en la herramienta elegida.

Beneficios de la Integración Continua

1. Mejorar la Calidad del código.
2. Detección de Errores más rápida y fácil.
3. Reduce Tareas repetitivas y manuales.
4. Puede crear Versiones de prueba en cualquier momento.
5. Completa Visibilidad del proyecto.
6. Mayor Confianza y seguridad del equipo de trabajo.

Tal como se pueden observar en la figura 4.

Figura 4. Principales Beneficios de la Integración Continua

1. Mejorar la calidad del código

Cuando se utiliza CI se prevé que los cambios insertados en el código poseen mayor calidad e integridad con el resto del software. El servidor CI facilita las tareas de automatización, despliegue, ejecución de pruebas y análisis de código. Existen varias alternativas en el mercado Open source y de licencia comercial integrados con los diferentes ambientes de desarrollo. Estos servidores ejecutan tareas programadas que garantizan la calidad del código y facilitan la detección de errores.

2. Detección de errores más rápida y fácil

Integrar de forma periódica permite encontrar los errores de forma temprana y el tratamiento de los mismos es más simple e insume menos tiempo.

Al aplicar CI se reducen los riesgos (uno de los beneficios más importante para Martin Fowler) y tiempos, ya que son menos los cambios que se incluyen en cada integración por ende se requiere menos tiempo y son menos los errores los que pueden surgir, ya que se detectan en un tiempo temprano evitando que se combinen con errores que pueden surgir al agregar cambios posteriores, generando más tiempo y complicaciones para solucionar los mismos. Por otro lado, al realizar integraciones constantemente, los programadores recuerdan y tienen en mente los cambios que se están integrando y, por ende, los errores que pueden surgir y cómo solventarlos de manera más simple.

Según Martin Fowler: "*Como resultado, los proyectos con integración continua tienden a tener dramáticamente menos errores, tanto en producción como en proceso. Sin embargo, debo enfatizar que el grado de este beneficio está directamente relacionado con lo bueno que es su conjunto de pruebas*".

Llevar a cabo esta práctica no garantiza la inexistencia de errores al momento de integrar, pero si estamos reduciendo la probabilidad de que estos ocurran, ya que la tendencia es integrar código que está funcionando, porque los builds y los tests pasaron exitosamente.

Podemos concluir que para llegar a un nivel bajo de errores implica trabajar y mejorar de manera constante en el conjunto de pruebas a aplicar.

3. Reduce tareas repetitivas y manuales

Los procesos de instalación manuales, cuando se realizan de manera repetitiva, son una pérdida de tiempo y constituyen una fuente de equivocaciones. Las personas nos equivocamos, los scripts de instalación no. Además, tener scripts que «saben» como hacer la integración continua (y añadidos posteriores, como despliegues, testeo, etc.) evita dependencias de personas.

Los procesos manuales repetitivos son lentos y susceptibles a cometer diversos errores, por lo que, poder automatizarlos implica un gran beneficio para asegurarnos que los procedimientos se completen de manera correcta.

4. Puede crear Versiones de prueba en cualquier momento

Permite generar versiones de software en cualquier momento y de manera transparente, clara y entendible para todos los integrantes del proyecto de desarrollo; principalmente sin errores y generalmente con los últimos cambios realizados. Además, dicha versión será correcta, funcional y completa, siempre y cuando la integración se haya realizado correctamente. Por otra parte, otorga independización de una persona específica que realice la integración para recién poder generar una versión del software.

5. Completa Visibilidad del proyecto

Al aplicar continuamente la integración, sabemos frecuentemente el estado del proyecto. No se espera al final para detectar funcionalidades incorrectas o errores de carácter grave, que generarían grandes demoras y complicaciones para solucionarlos.

Analizando cómo se va desarrollando la integración continua, se pueden obtener estadísticas y tendencias del estado del desarrollo y plantear mejoras en el proceso de desarrollo, como así también solucionar problemas que surgen en el mismo.

6. Mayor Confianza y seguridad del equipo de trabajo

Según Javier Garzás: "*La experiencia demuestra que un equipo que utiliza integración continua logra una mayor confianza interna, al estar comprobando constantemente el estado del desarrollo. Y no es solamente la confianza en el equipo, es también del entorno (usuarios, gestores, etc.).*

Un equipo de desarrollo de software que se siente seguro es un equipo más motivado y productivo" [2].

CI mejora ampliamente el rendimiento y la interacción entre los participantes, ya que no se requiere llegar al final del proyecto para realizar una integración y verificar fehacientemente lo que cada uno de los integrantes desarrolló; si realmente está sin errores, evita peleas y disputas por los cambios que puedan perjudicar a los integrantes.

Herramientas para Integración Continua

La integración continua (CI) permite a los desarrolladores de software evitar una larga y problemática fase de integración al final de un proyecto. En lugar de compilar todos los componentes al final, con la CI se van implementando todas las novedades directamente en el código base. Esto requiere disciplina y un proceso eficiente, pues de lo contrario la CI obstaculizará más de lo que ayudará. El proceso se puede facilitar además con software específico.

A veces de forma totalmente autónoma y otras veces en combinación con otras aplicaciones, las herramientas de integración continua (CI tools) ayudan en la creación de un repositorio, en la ejecución de las pruebas y en la compilación, así como en el control de versiones y, por supuesto, en la propia integración continua.

En la actualidad, Internet ofrece una gran variedad de herramientas para la integración continua. Todas tienen como objetivo ayudar al desarrollador en la implementación de esta metodología, y lo hacen de diferentes modos y con la ayuda de características distintas. Pero estas herramientas no solo se diferencian unas de otras en cuanto a sus características, sino que también existe una gran variedad en lo que respecta a precios y licencias. Mientras que muchas de ellas son de código abierto y se encuentran disponibles de forma gratuita, otros fabricantes ofrecen herramientas comerciales.

Entre las herramientas que priorizamos para mostrar, se encuentran:

Jenkins, una de las herramientas de integración continua más conocidas del mercado. Este software escrito en Java ha continuado desarrollándose constantemente desde el año 2005 y cuenta en la actualidad con numerosas funciones que asisten no solo en la integración continua, sino también en el despliegue y la entrega continua. Jenkins CI no es más que un sistema desplegado en un servidor que nos ayuda en la tarea de hacer integración continua y programar tareas automáticas cuando ocurra una determinada acción.

Travis CI, un servicio de integración continua que se utiliza para construir y probar software alojado en GitHub, ya que se integra a la perfección y realiza automáticamente los pipelines definidos en cada push o pull request. La herramienta es gratuita en proyectos Open Source, aunque cobra una comisión si se utiliza con repositorios privados. Permite testear y buildear aplicaciones escritas en Ruby, Node, Objective-C, Go, Java, C# y F#, entre otras (siempre que corran en Linux). Permite combinarse con Azure para facilitar los deploys automáticos.

A diferencia de Jenkins no utiliza tareas programadas, no tiene costos de mantenimiento (Ya que está alojado como servicio) y posee un setup más simple.

Conclusión

En base a lo investigado y expuesto en el presente, no quedan dudas que la CI es una práctica fundamental que contribuye con el desarrollo ágil de cualquier producto de software. Es indispensable su aplicación, ya que facilita al equipo de desarrollo a integrar sus cambios y resolver posibles conflictos lo antes posible. Integrar periódicamente, encontrar errores de forma temprana y corregirlos en el momento, permite optimizar la forma de trabajo, lo cual contribuye al valor que se le entrega al cliente final. Por esto, consideramos que la CI no es una práctica que beneficia únicamente a los desarrolladores, sino también a todas las personas involucradas en el proceso de desarrollo del software.

La elaboración de pruebas unitarias, que permitan validar porciones pequeñas de código, es una inversión de tiempo necesaria para llevar a cabo la CI. La compilación automática y la ejecución de estas pruebas permiten detectar conflictos en los últimos cambios integrados por el equipo de desarrollo. La calidad con la cual se elaboran las pruebas unitarias es un factor para tener en cuenta, ya que luego

facilita las correcciones a aplicar para solucionar los errores que puedan surgir de la ejecución automática de las mismas.

Al igual que la Gestión de Configuración y otras disciplinas de la Ingeniería del Software, la CI es una práctica cuya responsabilidad de llevarla a cabo no recae sobre una única persona, sino más bien, es una actividad colaborativa entre los diferentes miembros del equipo de trabajo. Por otro lado, disponer de un repositorio único y de servidores para realizar compilaciones automáticas no garantiza que realmente se esté implementando la CI. Más aun sabiendo que existen diferentes herramientas gratuitas para su implementación, el verdadero desafío de los equipos de desarrollo en la actualidad es interiorizar esta práctica, generar la costumbre y asumir el compromiso de integrar frecuentemente sus cambios.

Por último, queremos destacar el factor humano y comunicativo que componen a esta práctica. La coordinación y el compromiso requerido para llevarla a cabo contribuye y fortalece la sinergia del equipo de trabajo, siendo además esta práctica la base para implementar la entrega y el despliegue continuo, que en definitiva responden a los principios ágiles planteados al comienzo del informe.

Bibliografía

[1] Martín Fowler, "Continuous Integration", 01 mayo 2006. https://www.martinfowler.com/articles/continuousIntegration.html

[2] Javier Garzás, "Pero, realmente… ¿Qué beneficios aporta la integración continua?", 25 septiembre 2014. https://www.javiergarzas.com/2014/09/beneficios-integracion-continua.html

[3] Sander Rossel, Editorial Packt, "Continuous Integration, Delivery, and Deployment", 2017.

[4] Gisell Chávez, "Integración Continua". https://www.smartnodus.cl/integracion-continua/

[5] José Ripla, Universitat de Valéncia, Capgemini, "Testing + Integración Continua". https://www.uv.es/capgeminiuv/documents/Presentacion_Testing_IC_2015_11_05.pdf

[6] Hernández A., Universidad de las Ciencias Informáticas, "Aplicación del proceso de integración continua en el centro de telemática de la universidad de las ciencias informáticas", junio 2014. https://www.researchgate.net/publication/302010909_APLICACION_DEL_PROCESO_DE_INTEGRACION_CONTINUA_EN_EL_CENTRO_DE_TELEMATICA_DE_LA_UNIVERSIDAD_DE_LAS_CIENCIAS_INFORMATICAS

[7] Alicia Salamon, Patricio Maller, Alejandra Boggio, Natalia Mira, Sofia Pérez, Francisco Coenda, "La Integración Continua Aplicada en el Desarrollo de Software en el Ámbito Científico – Técnico". https://rdu.iua.edu.ar/bitstream/123456789/264/1/IC%20Aplicada%20en%20el%20Desarrollo%20de%20Software%20C-T.pdf

[8] Pedro Galván Kondo, "Integración Continua". https://sg.com.mx/revista/54/integraci-n-continua

[9] J. Camilorada, "Integración continua, asegurando la calidad en el desarrollo de software", 10 junio 2014. https://camilorada.wordpress.com/2014/06/10/integracion-continua-asegurando-la-calidad-en-el-desarrollo-de-software/

[10] Travis CI User Documentation. https://docs.travis-ci.com/

[11] Principios de las Metodologías Ágiles, "Manifiesto Ágil". https://agilemanifesto.org/iso/es/principles.html

[12] Julio Alberto Fernández Guerrero, "Implantación de un Sistema de Integración Continua en una metodología consolidada", Trabajo de Fin de Grado, Universidad de Castilla - La Mancha, Escuela Superior de Informática, Junio 2017. https://ruidera.uclm.es/xmlui/bitstream/handle/10578/15411/TFG_Julio_Alberto_Fernandez_Guerrero.pdf?sequence=1&isAllowed=y

[13] Ana María García Orozco, "La integración continua y su aporte al aseguramiento de la calidad en el ciclo de vida del desarrollo de software", Proyecto de Grado para optar al título de especialista en Proyectos Informáticos, Universidad Distrital Francisco José de Caldas, Facultad de Ingeniería, Especialización Proyectos Informáticos, Bogotá, D.C., 02, diciembre 2015.

Glosario

- CI: Continuous Integration / Integración Continua.
- CI/CD: Continuous Integration, Delivery y Deployment / Integración, Entrega y Despliegue Continuo.
- SW: Software.
- CI tools: Herramientas de Integración continua.