Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 12, No. 10, Mes Octubre, 2019
ISSN: 2306-2495 | RNPS: 2343
Pág. 102-116
http://publicaciones.uci.cu

Tipo de artículo: Artículo original
Temática: Gestión de Proyectos
Recibido: 10/06/2019 | Aceptado: 10/10/2019 | Publicado: 22/10/2019

La Integración Continua pilar fundamental en el proceso de desarrollo de software

Continuous Integration is a fundamental pillar in the software development process

Marbelis Rojas Rodríguez [1*], Yannia Moreira Gamboa [2], Suaryne Torres López [3]

1 Dirección de Producción de Software, Universidad de las Ciencias Informáticas, La Habana, Cuba. marbelis@uci.cu
[2] Dirección de Producción de Software, Universidad de las Ciencias Informáticas, La Habana, Cuba. ygamboa@uci.cu
[3] Departamento de Gestión de Proyectos, Universidad de las Ciencias Informáticas, La Habana, Cuba. storres@uci.cu

[*] Autor para correspondencia: marbelis@uci.cu

Resumen
Obtener productos de alta calidad en tiempos cada vez más cortos y con la documentación necesaria, se ha convertido en la meta principal para las empresas de desarrollo de software debido a la fuerte competencia en el mercado. Durante el desarrollo del trabajo se analizan diferentes aspectos de la integración continua como apoyo a la gestión de proyectos y al aseguramiento de la calidad de software, señalando como aspectos fundamentales sus ventajas y la utilización de herramientas de uso gratuito que permiten su automatización. El resultado de la investigación permitió realizar un estudio del estado del arte de la práctica de integración continua en el mundo y se menciona el entorno que se utilizan en la Universidad de las Ciencias Informáticas para la integración continua.

Palabras clave: integración continua, gestión de proyectos, aseguramiento de la calidad, desarrollo de software

Abstract
Get high quality products in increasingly short times and with the necessary documentation, it has become the main goal for software development companies due to strong competition in the market. During the development of the work, different aspects of continuous integration are analyzed as support for project management and software quality assurance, pointing out its advantages and the use of free-use tools that allow its automation as fundamental aspects. The result of the specified research conducts a study of the state of the art of the practice of continuous integration in the world and mentions the environment that will be used at the University of Computer Science for continuous integration.

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 12, No. 10, Mes Octubre, 2019
ISSN: 2306-2495 | RNPS: 2343
Pág. 102-116
http://publicaciones.uci.cu

Keywords: *continuous integration, project management, quality assurance, software development*

Introducción

Frecuentemente las empresas de desarrollo de software enfrentan contratiempos para garantizar la calidad de sus productos y lograr una correcta gestión de los proyectos. Uno de los principales problemas que existen a la hora de desplegar una versión donde se encuentren involucrados varios desarrolladores, es que no todos los desarrolladores realizan las pruebas unitarias documentadas, código inestable, integraciones fallidas, entre otros. Por esta razón se hace necesario detectar errores en momentos del desarrollo en los que cueste menos solucionarlos a través de la Integración Continua (IC) como práctica compatible con las metodologías ágiles de desarrollo de software.

La Universidad de las Ciencias Informáticas (UCI) es una de las entidades dedicada a la producción de software en Cuba que mantiene compromisos tanto a nivel nacional como internacional, de ahí la importancia de garantizar la gestión de los proyectos y asegurar la calidad de sus productos y servicios. No practicar la IC en algunos proyectos es uno de los problemas que afectan la calidad de los productos que son liberados en la UCI, ya que el 20% de los proyectos no utilizan esta técnica.

Mediante esta investigación se propone exponer los beneficios de aplicar la IC utilizando herramientas de uso gratuito, presentando sus ventajas y la forma de implementarse, como apoyo al mejoramiento de los procesos de gestión de proyectos y aseguramiento de la calidad.

Materiales y métodos o Metodología computacional

En el desarrollo de la presente investigación se utilizó el método analítico-deductivo y como técnica de recolección de datos el análisis documental. Se realizó el análisis de la literatura especializada publicada a nivel nacional e internacional sobre la IC en el desarrollo de software. También se consultaron líderes de proyectos de software de la UCI que permitieron plasmar en este trabajo el entorno que utilizan para la IC. Se obtuvieron además un conjunto de conclusiones que permiten identificar los elementos fundamentales que brinda la IC en los procesos de gestión de proyectos y aseguramiento de la calidad.

Integración Continua

El concepto de IC surgió en la comunidad de Extreme Programming, XP y patrocinada por Martin Fowler. Kent Beck fue el primero quién escribió acerca de la IC alrededor del año 1999. El término de IC fue creado por Martin Fowler: La IC es una práctica de desarrollo de software en la cual los miembros de un equipo integran su trabajo

frecuentemente, como mínimo de forma diaria. Cada integración se verifica mediante una herramienta de construcción automática para detectar los errores de integración tan pronto como sea posible. Muchos equipos creen que este enfoque lleva a una reducción significativa de los problemas de integración y permite a un equipo desarrollar software cohesivo de forma más rápida. [1]

La IC facilita la integración en un proyecto de una forma incremental y en cortos espacios de tiempo, pues permite identificar errores en el software en etapas tempranas y brinda funcionalidades para la automatización de tareas agilizando de esta forma el proceso de desarrollo de software. Entre las tareas que se pueden automatizar se encuentran: compilación de los componentes, integración de todo el desarrollo, ejecución de pruebas unitarias, ejecución de pruebas de integración, ejecución de pruebas de aceptación, obtención de métricas de calidad de código y despliegues automáticos.

En este momento existen modelos definidos para la implementación de la IC [2], los cuales son básicamente para el proceso de codificación de software, donde:

1. Un desarrollador realiza un cambio (commit) sobre el Software Configuration Management (SCM).
2. Después del cambio el administrador de integración detecta el cambio, toma del repositorio las últimas versiones y ejecuta los scripts que forman todo el software.
3. El integrador de administración continua informa por correo electrónico sobre los resultados de la compilación a los miembros del grupo que ya fueron configurados previamente.
4. El administrador realiza las inspecciones con una frecuencia determinada.

Para la implementación de un entorno de IC es importante tener en cuenta los siguientes aspectos como punto de partida:

- ✓ Contar con un repositorio para llevar el historial de todos los cambios del código y que permita obtener la última versión del proyecto.
- ✓ Sensibilizar al equipo de desarrollo para que realice integraciones con frecuencia.
- ✓ El equipo de desarrollo debe estar capacitado para documentar y realizar pruebas unitarias y asumirlas como una práctica constante durante la implementación.
- ✓ Automatizar las pruebas unitarias.
- ✓ Realizar la revisión estática de código.
- ✓ Definir los roles del equipo de desarrollo con el fin de determinar la configuración de los avisos que va a arrojar el servidor de IC.
- ✓ Definir el administrador de la configuración del servidor de IC y de cada uno de sus componentes.

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 12, No. 10, Mes Octubre, 2019
ISSN: 2306-2495 | RNPS: 2343
Pág. 102-116
http://publicaciones.uci.cu

- ✓ Realizar integración diaria en un servidor de IC.

Principios

A continuación, se mencionan los principios de la IC identificados por otros autores [3]:

- ❖ Mantener un único repositorio de código fuente: Todo se debe incluir en el repositorio. Se debe ser capaz de construir un proyecto a partir del código descargado del repositorio. También es útil incluir recursos de configuración del código. Se recomienda no subir todo aquello que se pueda construir.
- ❖ Automatizar la construcción del proyecto: La construcción de un proyecto implica la compilación del código fuente, mover ficheros de un sitio a otro, cargar esquemas en base de datos, resolver dependencias del código con librerías de tercero, entre otros aspectos. Todas estas tareas deberían automatizarse para evitar errores y ganar tiempo.
- ❖ Autodiagnóstico de la construcción: Dentro del proceso de construcción de un proceso se deberían realizar las pruebas unitarias y de integración del código, que pueden implementarse mediante herramientas de la familia XUnit. Estas pruebas pueden detectar errores en etapas tempranas del desarrollo haciendo más ágil su detección y resolución.
- ❖ Entregar los cambios diariamente a la línea base: La integración permite que los desarrolladores informen unos a otros de los cambios que han hecho. Si esto se hace de forma frecuente, todo el mundo sabrá rápidamente los cambios que se han producido. Haciendo esto de forma frecuente, lo desarrolladores encuentran rápidamente si hay un conflicto y lo arreglan tan pronto lo detectan, cuando todavía es fácil de arreglar. Como mínimo, los desarrolladores deberían entregar sus cambios una vez al día.
- ❖ Construir la línea base tras cada entrega: A pesar de las entregas diarias de los desarrolladores, todavía se pueden producir errores de integración debido a falta de disciplina del equipo o diferencias ambientales entre las máquinas de los desarrolladores. Para evitar estos errores hay que asegurarse de que se construya el proyecto en el ambiente destinado para la integración tras cada entrega a la línea base (commit). El desarrollador que está haciendo la entrega es el responsable de que esta construcción se realice de forma correcta. La construcción en el ambiente de integración se puede hacer de forma manual o mediante un servidor de IC como Jenkins, CruiseControl, Continuum, etc.
- ❖ Mantener una ejecución rápida de la construcción del proyecto: Puesto que en un proceso de IC lo que cuenta es obtener resultados del proceso tan pronto como sea posible, es preciso que el proceso de construcción no se demore excesivamente.

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 12, No. 10, Mes Octubre, 2019
ISSN: 2306-2495 | RNPS: 2343
Pág. 102-116
http://publicaciones.uci.cu

- Probar en una réplica del entorno de producción: Una práctica común es mantener un ambiente de pre-producción lo más similar posible al ambiente de producción real, para realizar allí pruebas de integración.
- Fácil obtención del último ejecutable del proyecto: Todo el que esté involucrado en el desarrollo de un determinado software, como los desarrolladores mismos o el equipo de gestión de proyecto o el de calidad, debería ser capaz de obtener fácilmente el último ejecutable y de ejecutarlo, facilitando las demostraciones, las pruebas y las revisiones de los últimos cambios.
- Publicar el estado del proyecto: La IC considera que la comunicación es muy importante, así que hay que asegurarse que todo el equipo involucrado pueda ver fácilmente el estado del sistema y los cambios que se han realizado.
- Automatizar el despliegue: Para llevar a cabo la práctica de IC eficazmente se necesitan varios entornos o ambientes, como el de desarrollo, el de prueba y el de producción. Ya que se deben mover ejecutables entre múltiples entornos varias veces al día, por lo que es mejor hacerlo de forma automática, siendo necesario tener scripts que permitan el despliegue de aplicaciones entre entornos de forma sencilla.

Resultados y discusión

Herramientas existentes para realizar la Integración Continua

Existen varias herramientas involucradas en la IC para diferentes plataformas tecnológicas. Algunas propietarias, pero existen herramientas Open Source que proveen la automatización en diferentes fases de la IC, siendo integrables con otras herramientas Open Source multiplataforma que trabajan con lenguajes de programación como C#, Java y .Net.

En el siguiente análisis comparativo se identificaron los procesos en los que interviene la práctica de IC con un conjunto de herramientas que permiten controlarlos y ejecutarlos.

- **Sistemas de Control de Versiones**

Los Sistemas de Control de Versiones (SCV) permiten trabajar de forma colaborativa en el desarrollo de proyectos gestionando los cambios que se realizan, descargando la versión completa del código, actualizando cambios y subiendo cambios.

Los SCV se pueden clasificar en 2 grandes grupos:

1. Centralizados (Cliente-Servidor): Funciona de manera que los desarrolladores usan un repositorio de código fuente central al que acceden mediante un cliente desde su máquina.
2. Distribuido: Opera de forma que cada desarrollador trabaja directamente con un repositorio de código fuente local y los cambios que realice se publican en otros repositorios posteriormente.

Los principales beneficios de utilizar una herramienta SCV son [4]:

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 12, No. 10, Mes Octubre, 2019
ISSN: 2306-2495 | RNPS: 2343
Pág. 102-116
http://publicaciones.uci.cu

1. Cualquier revisión almacenada de un archivo puede ser recuperada para visualizarse o modificarse.
2. Pueden desplegarse las diferencias entre distintas versiones.
3. Las correcciones pueden ser creadas automáticamente.
4. Múltiples desarrolladores pueden trabajar simultáneamente en el mismo proyecto o archivo sin pérdida de datos.
5. Los proyectos pueden ser divididos para permitir el desarrollo simultáneo en varias versiones, estas divisiones pueden ser fusionadas para alcanzar el objetivo principal del desarrollo.
6. El desarrollo distribuido, es soportado a través de las redes de datos con diferentes mecanismos de autentificación.

Para comparar las herramientas ver Tabla 1 y Tabla 2, se obtuvieron las características siguientes:

- ✓ Licencia: Open Source (Código Abierto) y propietaria (con costo, pago por licencia).
- ✓ Clasificación: Cliente-Servidor y Distribuido.
- ✓ Etiquetas para revisiones: Creación de etiquetas para identificar o diferenciar versiones.
- ✓ Integración: Capacidad de interacción y plugins para ser integrado con otras plataformas utilizadas y Sistemas Operativos con los que son compatibles.
- ✓ Administración de Archivo: Renombrar y/o fusionar versiones sin perder la historia del cambio.
- ✓ Historia del cambio: Changeset si almacena únicamente el conjunto de los nuevos cambios realizados o snapshot si almacena una instantánea del árbol de directorios antes y después del cambio.
- ✓ Alcance del cambio: Árbol o ficheros individuales.

Característica / Herramientas	Licencia	Clasificación	Etiquetas para revisiones	Integración
Concurrent Versions System (CVS)	Open Source	Cliente-Servidor		Visual Studio, Eclipse, Windows, Unix, Linux, Mac OSX, BeOS
Subversion	Open Source	Cliente-Servidor	x	BBEdit, Eclipse, NetBeans, Visual Studio, Emacs, IntelliJ IDEA, KDevelop, Komodo IDE
AccuRev	Propietaria	Cliente-Servidor		Visual Studio, Eclipse, IntelliJ IDEA, Windows, Unix, Linux, Mac OSX,

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 12, No. 10, Mes Octubre, 2019
ISSN: 2306-2495 | RNPS: 2343
Pág. 102-116
http://publicaciones.uci.cu

Característica / Herramientas	Licencia	Clasificación	Etiquetas para revisiones	Integración
				BeOS
Clear Case (IBM)	Propietaria	Cliente-Servidor	x	Emac, Eclipse, Visual Studio, Kdevelop, IntelliJ IDEA, MS Windows, Unix, Clientes TSO para z/OS
Bazaar	Open Source	Distribuido	x	Visual Studio, Eclipse, Text Mate, Komodo IDE, Bazaar Explorer, Tortoise Bzr, Windows, Linux, Mac OS o Solaris.
Team Foundation Server	Propietaria	Cliente-Servidor	x	Visual Studio, IntelliJ IDEA, Windows, Linux, Mac OS, Solaris, SourceAnyWhere for VSS
GIT	Open Source	Distribuido	x	Aptana 3 Beta, Eclipse, Emacs, Text Mate, Gitk, git-gui, tig, TortoiseGit, qgit, gitg, git-cola, Tower, Source Tree, GitX, Windows, Linux, Mac OS o Solaris.
Fossil	Open Source	Distribuido	x	Eclipse Team Provide for Fossil, Windows, Linux, Mac OS o Solaris.
Mercurial	Open Source	Distribuido	x	Visual Studio, Eclipse, IntelliJ IDEA, Windows, Linux, Mac OS o Solaris.
Microsoft Visual SourceSafe (VSS)	Propietaria	Cliente-Servidor	x	Visual Studio

Tabla 1: Comparación de las herramientas de control de versiones

Características / Herramientas	Administración de Archivo	Historia del cambio	Alcance del cambio

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 12, No. 10, Mes Octubre, 2019
ISSN: 2306-2495 | RNPS: 2343
Pág. 102-116
http://publicaciones.uci.cu

Características / Herramientas	Administración de Archivo	Historia del cambio	Alcance del cambio
Concurrent Versions System (CVS)	x	Snapshot	Fichero
Subversion	x	Snapshot/Changeset	Árbol
AccuRev	x	Changeset	Árbol
Clear Case (IBM)	x	Changeset	Fichero
Bazaar	x	Snapshot	Árbol
Team Foundation Server	x	Snapshot	Fichero
GIT		Snapshot	Árbol
Fossil	x	Snapshot	Árbol
Mercurial	x	Changeset	Árbol
Microsoft Visual SourceSafe (VSS)	x	Snapshot	Fichero

Tabla 2: Comparación de las herramientas de control de versiones. Gestión de cambios.

Las comparaciones que se muestran en las tablas anteriores facilita a las entidades la selección de las herramientas a utilizar acordes con sus características.

❖ **Pruebas Automáticas**

La automatización de las pruebas funcionales disminuye significativamente el esfuerzo dedicado a las pruebas de regresión. Entre las prácticas y herramientas que los desarrolladores pueden utilizar con este fin se encuentran:

- ✓ Automatización de pruebas unitarias: Las pruebas unitarias buscan conocer el funcionamiento de las unidades de software de forma temprana, mediante el uso de marcos de trabajo como NUnit, JUnit, PHPUnit, entre otras.
- ✓ Automatización de pruebas de sistema: Estas tienen como objetivo verificar que las funcionalidades implementadas satisfacen los requerimientos establecidos por el cliente.
- ✓ Automatización de pruebas de funcionalidad: Para la automatización de las pruebas funcionales son principalmente utilizadas las herramientas de ejecución de las pruebas de captura y reproducción. Estas herramientas posibilitan al probador capturar y grabar pruebas, para luego editarlas, modificarlas y

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 12, No. 10, Mes Octubre, 2019
ISSN: 2306-2495 | RNPS: 2343
Pág. 102-116
http://publicaciones.uci.cu

reproducirlas en entornos diferentes, como ejemplos de estas herramientas se encuentran Selenium (para Web) y Abbot (para GUI). El impacto de automatizar dichas pruebas se evidencia en negocios donde las pruebas son repetitivas o requiere un gran esfuerzo para hacerlas manualmente.

En la Tabla 3 se describen algunas herramientas que se utilizan para ejecutar las pruebas unitarias.

Herramienta	Licencia	Plataforma	Lenguaje	Integración
JUnit	Open Source	Windows/ Linux	Eclipse, NetBeans	Jenkins Hudson, Ant, Maven
PHPUnit	Open Source	Windows/ Linux	PHP	Jenkins
SimpleTest	Propietario	Windows/ Linux	Eclipse	Jenkins
TestNG	Open Source	Windows/ Linux	Java	Jenkins, Maven
CPPUNit	Open Source	Windows/ Linux	C/C++	Jenkins
Nunit	Open Source	Windows/ Linux	.NET	Jenkins

Tabla 3: Herramientas para realizar pruebas unitarias.

De forma general, estas herramientas cuentan con documentación para guiar su uso y se integran casi todas con el servidor de IC Jenkins.

En la Tabla 4 se describen algunas herramientas que se utilizan para ejecutar las pruebas funcionales:

Herramientas	Simulación exacta del escenario	Utiliza diferentes navegadores	Sistema operativo	Integración con herramientas de gestión de pruebas	Lenguajes soportados en la creación de scripts
Unified functional testing (HP)	x	x	Windows	x	Visual Basic

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 12, No. 10, Mes Octubre, 2019
ISSN: 2306-2495 | RNPS: 2343
Pág. 102-116
http://publicaciones.uci.cu

Herramientas	Simulación exacta del escenario	Utiliza diferentes navegadores	Sistema operativo	Integración con herramientas de gestión de pruebas	Lenguajes soportados en la creación de scripts
Selenium By SeleniumHQ	x	x	Windows, MacOS y Linux	x	Visual Basic, Java, .NET
Eggplant by TestPlant	x	x	Windows, MacOS y Linux	x	Sense Talk
Ranorex	x	x	Windows	x	Visual Basic, Java, .NET

Tabla 4: Herramientas para realizar pruebas funcionales.

En las tablas 3 y 4 se mencionan las herramientas que se consideran las más destacadas, el criterio para seleccionar la que se podría utilizar depende en buen grado de los lenguajes soportados en la creación de scripts, ya que excepto Selenium las demás son propietarias.

❖ **Construcción y despliegue automatizados**

Garantizar la calidad del código se hace necesario para los desarrolladores y esto se logra a través de las pruebas estáticas de código. Para la realización de estas pruebas es esencial contar con una herramienta que facilite el encontrar errores como: código muerto, código quemado, código no documentado, ciclos mal construidos, errores de sintaxis, entre otros.

En la Tabla 5 se describen algunas herramientas que se utilizan para la construcción del código fuente:

Herramienta	Forma de Adquisición	Plataforma	Lenguaje	Integración
PHPLint	Propietario	Windows/ Linux	PHP	Jenkins
YASCA	Open Source	Windows/ Linux	Java, .NET, PHP, HTML, CSS, etc.	ClamAV, Cpp Check, PHPlint,

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 12, No. 10, Mes Octubre, 2019
ISSN: 2306-2495 | RNPS: 2343
Pág. 102-116
http://publicaciones.uci.cu

Herramienta	Forma de Adquisición	Plataforma	Lenguaje	Integración
				FindBugs
PMD	Propietario	Windows/ Linux	Java	No se Integra
FindBugs	Open Source	Windows/ Linux	Java, Eclipse, NetBeans, IntelliJ IDEA	Jenkins
SonarQube	Open Source	Windows/ Linux	Java, .NET, PHP, HTML, CSS, Eclipse, NetBeans, IntelliJ IDEA	Maven, Ant Jenkins, Hudson

Tabla 5: Comparación de herramientas para la construcción de código fuente.

Para seleccionar la herramienta a utilizar se tendría en cuenta principalmente el lenguaje y la herramienta con la que se integre.

Construir el sistema a partir del código que se ha desarrollado y lograr su despliegue es de vital importancia. La construcción de un proyecto significa la compilación del código fuente, mover ficheros de un sitio a otro, cargar esquemas en base de datos, resolver dependencias del código con librerías de tercero, entre otros. Esto es posible mediante el uso de las herramientas que se describen a continuación:

- Ant:
  - ✓ Herramienta gratuita utilizada en la compilación y creación de proyectos Java.
  - ✓ La ejecución de tareas se modela en un fichero XML.
  - ✓ Es multiplataforma, ya que está escrito en Java. Licencia Apache 2.0.
- Maven:
  - ✓ Herramienta para la gestión y comprensión de proyectos Java. Licencia Apache 2.0.
  - ✓ Se encarga de la gestión de dependencias, construcción de los artefactos, generación de la documentación, entre otros.
  - ✓ Una vez instalado y configurado, no es necesario mantenerlo.
  - ✓ Es rápido el acceso cuando los paquetes están descargados.
  - ✓ Independencia de la conexión a internet, sólo cuando se necesita un paquete nuevo.

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 12, No. 10, Mes Octubre, 2019
ISSN: 2306-2495 | RNPS: 2343
Pág. 102-116
http://publicaciones.uci.cu

❖ **Servidores de Integración Continua**

Los servidores de IC posibilitan verificar que cada cambio que se haga en el código de una aplicación no genere errores al desplegar el sistema en los ambientes de pruebas o productivo. A continuación, se muestra en la Tabla 6 las principales herramientas para la IC.

Herramientas	Licencia	Plataforma	Lenguajes	Integración
Cruise Control	Open Source	Windows, MacOS y Linux	J2EE, .NET	Ant, Maven, Subversion, CVS
Jenkins/Hudson	Open Source	Windows, MacOS y Linux	J2EE, .NET	CVS, Subversion, Git, Ant, Maven, Jenkins
Continuum	Open Source	Windows y Linux	Java	Subversion, Ant
Bamboo	Propietario	Windows, MacOS y Linux	Java	Mercurial, Git

Tabla 6: Herramientas para la IC.

❖ **Ventajas de la IC**

A continuación, se muestra un resumen de los beneficios que aporta la integración, entrega y despliegue continuo, según Humble y Farley [5]:

- ✓ Aumentar la calidad del producto: Al integrar el código de forma continua, el riesgo de que existan errores y el tiempo en solucionarlos, si los hay, disminuye. Esto se traduce en un aumento de la calidad del producto, ya que hay más probabilidades de éxito a la hora de solucionar los fallos de programación.
- ✓ Automatizar procesos repetitivos: Los procesos manuales, cuando se realizan frecuentemente, ocasionan una pérdida de tiempo y puede convertirse en una fuente inagotable de problemas. Automatizar estos procesos mediante scripts hace que las personas no necesiten perder el tiempo en tener que realizarlos a mano.
- ✓ Lograr un aumento de la productividad y la motivación del equipo: Al tener un mayor control sobre el desarrollo y el estado del proyecto, la productividad del equipo aumenta puesto que los desarrolladores solo tienen que ocuparse de programar. El entorno ya se encarga de integrar, compilar, testear y medir la calidad de forma automática, evitando que ningún miembro del equipo ocupe su tiempo en realizar estas tareas de forma manual.

Son disímiles las ventajas que nos brinda la IC en el proceso de desarrollo de software, a continuación, se destacan las siguientes [2]:

- ✓ Una metodología de desarrollo definida, divulgada y aplicada.

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 12, No. 10, Mes Octubre, 2019
ISSN: 2306-2495 | RNPS: 2343
Pág. 102-116
http://publicaciones.uci.cu

- ✓ Estándares de generación de código definidos, divulgados y aplicados.
- ✓ Revisión de código diario, lo que da certeza al punto anterior.
- ✓ Ambientes estables de desarrollo y pruebas.
- ✓ Equipo de trabajo de alta calidad (disciplina, conocimiento e innovación).
- ✓ Equipo de trabajo con aptitudes colaborativas.
- ✓ Tablero de control en tiempo real del estado del proyecto.
- ✓ Roles establecidos para la ejecución de tareas.
- ✓ El control de la complejidad del código reduce los riesgos del desarrollo de software.
- ✓ Control total de los despliegues de aplicaciones.
- ✓ Comunicación asertiva en cada etapa del ciclo de vida de desarrollo del software.
- ✓ Capacidad de reacción a los errores.
- ✓ Capacidad de reacción a los cambios en los requerimientos del proyecto.
- ✓ Clientes satisfechos con los tiempos de ejecución del proyecto.
- ✓ Clientes satisfechos con la calidad del producto.

❖ **Entorno de IC en la UCI**

Para exponer el entorno de IC más utilizado en la UCI se analizaron 47 proyectos, de ellos 9 que habían cerrado. Es necesario destacar que la metodología de desarrollo para la actividad productiva de la UCI que se utiliza es la AUP-UCI, la cual tiene como eje principal la aplicación de técnicas ágiles. La UCI como entidad desarrolladora de software también se beneficia de las prácticas referentes a la IC. A continuación, se mencionan las herramientas que más se utilizan:

- ✓ Como servidor de IC Jenkins aunque su uso todavía no se ha generalizado pues existen proyectos que no utilizan ningún servidor de IC. (Jenkins es una plataforma de automatización de código abierto que es ampliamente utilizada para la construcción y despliegue de software. Es altamente extensible [6]).
- ✓ Entre las herramientas que más se utilizan para las pruebas unitarias se encuentran: JUnit y PHPUnit, ambas se integran con Jenkins. (JUnit es un marco de pruebas unitarias para el lenguaje de programación Java. Juega un papel importante en el desarrollo guiado por pruebas y es parte de la familia de marcos de pruebas unitarias conocido como xUnit que se originó con JUnit. JUnit se utiliza para configurar los datos de prueba para un fragmento de código antes de que se implemente. Esto aumenta la productividad del programador y la estabilidad del código, y reduce el tiempo dedicado a la depuración [7]).
- ✓ Para la automatización de las pruebas funcionales se utiliza Selenium.

- ✓ Para la gestión y construcción de proyectos Java la herramienta que se utiliza es Maven. (Es una herramienta creada en el año 2001, de código abierto, para simplificar el proceso de construcción de proyectos. Maven describe el proyecto de software a construir, sus dependencias de otros módulos, y el orden de construcción de los elementos. Viene con objetivos ya definidos para realizar ciertas tareas, como la compilación del código y su empaquetado [8]).
- ✓ Como repositorio de código se utiliza principalmente las herramientas Git y Gitlab que se integran con el Jenkings. (GitLab: plataforma para desarrollar proyectos de software utilizando un control de versiones distribuido y la funcionalidad de gestión de códigos fuentes de Git [9]).
- ✓ Se identificó que pocos proyectos utilizan herramientas para realizar el análisis estático de código, entre las herramientas que se utilizan se encuentra SonarQube. (SonarQube es la plataforma de código abierto para gestionar la calidad del código fuente. Inicialmente pensado para Java, pero acepta otros lenguajes mediante extensiones. Se integra con Eclipse, además de otras plataformas [10]).

Conclusiones

Concluida la investigación se evidencian los aspectos fundamentales de la IC como apoyo a la gestión de proyectos y al aseguramiento de la calidad de software, a partir de los elementos que a continuación se mencionan:

- ✓ El uso de las herramientas que posibilitan la IC garantizan contar con la información en tiempo real de la ejecución del proyecto, así como identificar y solucionar los problemas en el desarrollo de software con mayor rapidez, facilitando la gestión de los proyectos.
- ✓ Implantar la IC conlleva un cambio en la forma de trabajo del equipo que debe ser aceptado poco a poco. Para la implementación de la IC se debe capacitar al personal para que sea viable la aceptación al cambio en la forma de trabajo.
- ✓ Para el proceso de pruebas funcionales los probadores, deben poseer un alto nivel de conocimiento, para que programen sus pruebas automatizadas, interpreten los resultados que arroje la herramienta y realicen las pruebas que no se pueden automatizar para asegurar la calidad del software.
- ✓ Se expuso el uso de las prácticas de la IC en la UCI que evidenció poca madurez en estos procesos. Dentro de las potencialidades de la IC en la UCI se encuentra que utiliza el repositorio de código Git lo que permite controlar el cambio del proyecto con el paso del tiempo, facilitando la toma de decisiones en la gestión de proyectos.

Referencias

[1] Fowler, M. (2006): Continuous Integration. 2006.

[2] OROZCO, A. M. G. (2018): La Integración Continua y su Aporte al Aseguramiento de la Calidad en el Ciclo de Vida del Desarrollo de Software. 2018.

[3] OROZCO, G. y RIGHI, L. G. (2018): Integración continua: solución a los problemas de productividad y calidad del código en un entorno ágil testigo. 2018.

[4] Tello, L.E. y SOSA, C. M. (2012): Revisión de los Sistemas de Control de Versiones Utilizados en el Desarrollo de Software. s.l. : Revista Ingenierias USBmed, 2012. Vol. 3, 1.

[5] Farley, J. y Humble, D. (2010): Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation. 1. s.l. : Addison-Wesley Professional, 2010. ISBN 0321601912, 9780321601919.

[6] Sampedro, J. Z., Holt, A. y Hauser, T. (2018): Continuous Integration and Delivery for HPC. 2018.

[7] Patel, D. (2018): Test Utility for Live and Online Testing of an Anti-Phishing Message Security System. 2018.

[8] Martínez, I. y Gil, F. P. (2018): GameCraft: Una plataforma de integración continua para videojuegos basada en microservicios. 2018.

[9] Segrelles, D., Moltó, G. y Miranda, F. (2018): Portafolios Docentes de Programación en la Nube para la Evaluación de Competencias. 2018.

[10] Montero, T. P. (2015): Un entorno software para el aprendizaje de la programación. 2015.