

Módulo 1. Software configuration management



Cuando se trata de temas de la mejora de procesos y su automatización, son muchos los caminos sobre los cuales las organizaciones transitan en busca de un único fin: poder replicar procesos exitosos, contar con trazabilidad y cumplir con los requerimientos de los requirentes y, por ende, con parte del propósito del negocio.

Desde esta perspectiva, las organizaciones crean estructuras organizacionales en las que cuentan con áreas de procesos, calidad o fusiones de estas que intentan canalizar los procedimientos y tareas que componen cada área, para lograr una perspectiva detallada que permitan a los encargados del desarrollo crear soluciones que mejoren dichos procesos con aplicativos mantenibles, adaptables y mejorables que fortalezcan los procesos que componen el negocio.

En este apartado de la creación de soluciones tecnológicas, se encuentra tanto lo teórico como práctico de estas disciplinas y las diferentes corrientes que de ella se derivan en torno a qué es mejor o peor, que agrega o no valor, y cuál será el mejor método con el cual se puede tener mejores resultados a la hora de sustituir procesos manuales por sistemas automatizados. Esto es todo un cúmulo de posibilidades que hacen difícil una única teoría; con mucha frecuencia, los procesos suelen estar pensados para abarcar todas las posibles situaciones y problemas, y, muchas veces, todo eso no es necesario para algunos proyectos, es decir, no siempre es mejor.

Si nos colocamos ahora del lado del desarrollo, más allá del proceso que se quiere automatizar, nos veremos en la necesidad de gestionar todos los procedimientos que se requieren para enfrentar los problemas y desafíos que se enfrentarán a la hora de establecer las prácticas necesarias en el desarrollo de una solución o producto de **software** (conjunto de programas) con todas las variantes que esto implica.

El hecho de que un producto de **software** sea íntegro puede depender de que se combinen tres tipos de disciplinas: desarrollo, gestión y control. Entre las disciplinas de control, está la GCS, que tiene como finalidad mantener la integridad de los componentes del producto **software**, hacer una evaluación y control sobre sus cambios, y favorecer la visibilidad del producto. El objetivo es aumentar la productividad y reducir al máximo los errores.

≡ Video de inmersión

≡ Unidad 1: Principios generales de software configuration management (SCM)

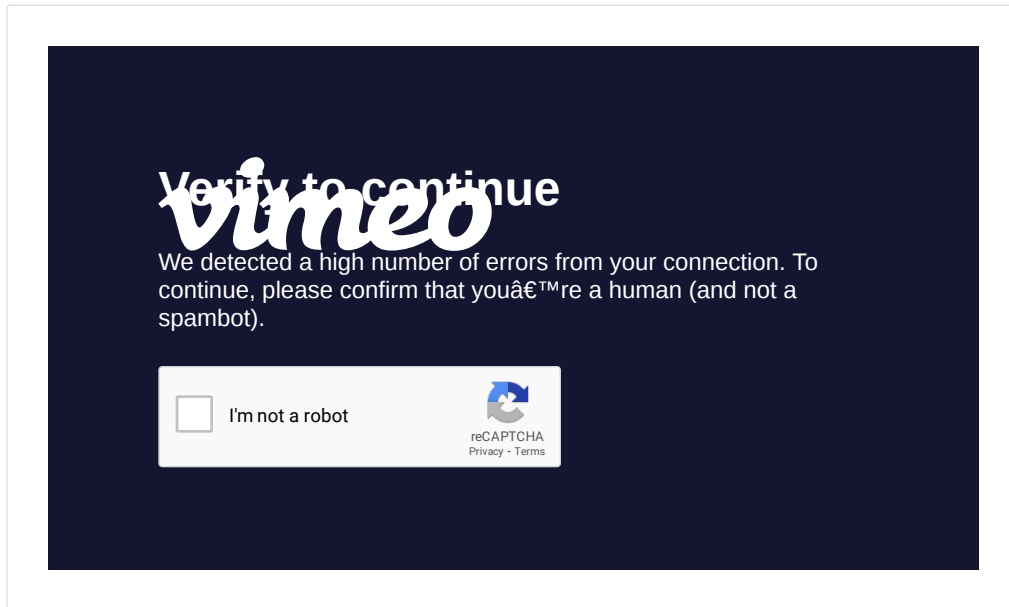
≡ Unidad 2: Control y seguimientos de cambios del software

≡ Glosario

≡ Video de habilidades

≡ Referencias

Video de inmersión



CONTINUAR

Unidad 1: Principios generales de software configuration management (SCM)

Tema 1: ¿Qué es el software configuration management (gestión de configuración de Software, SCM en adelante)?

Inicialmente, podemos decir que el SCM se define de la siguiente manera: “Es la disciplina cuyo objetivo es la identificación del **software** en ciertos momentos y el control sistemático de cambios a la configuración con el propósito de mantener la integridad, calidad y estandarización del **software**”. (Asamblea, 2010, <https://bit.ly/3ysAaJL>).

De acuerdo con Scarcella et al. (s. f.), el **software configuration management**:

[...] se refiere a las prácticas y herramientas transversales al desarrollo de **software** (a los requerimientos o el diseño en sí del programa), para atacar incumbencias como la gestión de versiones, gestión de cambios, gestión de requerimientos, gestión de incidencias, reproducibilidad de **release**, comunicación, interacción, coordinación e integración del trabajo de los diferentes miembros del equipo (<https://bit.ly/2UjwD1w>).

Para referirnos más cómodamente a este tema, en lo sucesivo lo llamaremos **trazabilidad**.

Tabla 1: Trazabilidad

Ventajas	Posibles desventajas	Actividades que se realizan en el GCS
<ul style="list-style-type: none">Resolución más rápida de los problemas.Gestión de cambios más eficiente.Reducción de costes.Control de licencias.Mayores niveles de seguridad.Mayor rapidez en la restauración del servicio.	<ul style="list-style-type: none">Una incorrecta planificación.Estructura inadecuada de la CMDB (configuration management data base – base de datos de la gestión de configuración, un concepto que introduce ITIL 4 – ISO/IEC 20000¹ para facilitar la gestión de los servicios TI).[1] ISO/IEC 20000. Information technology — Service management. (2011). International Organization for Standardization. Recuperado de https://bit.ly/3yXTuR.Herramientas inadecuadas.	<ul style="list-style-type: none">Planificación.Clasificación y registro.Monitorización y control.Realización de auditorías.Elaboración de informes.

	<ul style="list-style-type: none"> Falta de coordinación con la gestión de cambios y versiones. Falta de organización. Falta de compromiso. 	
--	--	--

Fuente: elaboración propia.

Modelo de actividades que se llevan a cabo en GCS

En la siguiente tabla, se recogen, de forma resumida, las actividades que conforman el proceso de gestión de configuración.

Tabla 2: Actividades que conforman el proceso de gestión de configuración

Actividad	Rol responsable	Descripción	Entradas	Salidas
Gestión del proceso de gestión de configuración	Gestor de configuración	Documentar el plan de gestión de configuración	Necesidades del proyecto. Plan de proyecto	Plan de gestión de configuración aprobado
Identificación de elementos de configuración	Gestor de configuración	Identificar elementos de configuración. Crear estructura del directorio de gestión de configuración	Productos del proyecto	Elementos de configuración identificados. Línea base. Estructura del directorio de gestión de configuración
Mantenimiento y control de la gestión de configuración	Responsable del elemento de configuración	Control de cambios sobre elementos de configuración y líneas base. Obtener aprobación de solicitudes de cambio sobre productos de trabajo de línea base	Peticiones de cambio	Registro de solicitud de cambio. Solicitud de cambio aprobada. Línea base
Informe de estado de la configuración	Gestor de configuración	Mantener actualizado y publicar el estado de los elementos de configuración	Elementos de configuración	Informe de estado de elementos de configuración

Verificación y auditoría	Gestor de configuración	Realizar auditorías de la gestión de configuración	Registros de la gestión de configuración. Línea base. Registros de cambios	Informe de auditoría de gestión de configuración
--------------------------	-------------------------	--	--	--

Fuente: Instituto Nacional de Tecnologías de la Comunicación, 2008, p. 11.

Características

1

Administración de concurrencia: se refiere a la edición simultánea de un archivo por más de una persona y su correcta integración.

2

Gestión de configuraciones:

- Identificación de configuración. El proceso de identificar los atributos que definen cada aspecto de un elemento de configuración. Un elemento de configuración es un producto que tiene el propósito de ser presentado al usuario.
- Control de configuración: es un grupo de procesos y etapas de aprobación requeridas para cambiar los atributos de un SCI y cambiar sus líneas de base.
- Auditoría de configuración: asegura que el funcionamiento de un SCI cumpla con las metas.
- Reporte de **status** (posición) de configuración: es la habilidad de grabar y reportar sobre las configuraciones de líneas de base asociadas con cada SCI en cualquier momento.

3

Control de versiones: se refiere al rastreo de revisiones de archivos, el cual hace posible recrear una versión anterior del archivo. Esto se logra creando una copia de cada archivo en el repositorio central.

4

Sincronización: para esto, cada miembro del proyecto baja periódicamente la versión más actual del proyecto y, luego de realizar el trabajo, se actualiza en el repositorio.

Elementos que componen el proceso de gestión de configuración



Versión: es una instancia de un elemento de configuración. El término se usa para señalar a un elemento de configuración del software que tiene un conjunto definido de características funcionales.



Revisión: se define como una versión que se construye sobre otra versión anterior. El término revisión generalmente se asocia a la noción de corrección de errores.



Variante: versión que es una alternativa a otra versión. Las variantes pueden permitir a un ECS satisfacer requerimientos en conflicto. Una variante es una nueva versión de un elemento que será añadida a la configuración sin reemplazar a la versión anterior.



Entrega o release: es una instancia del sistema distribuida a los clientes.



Línea base o baseline: es una especificación o producto revisado y aprobado formalmente, que sirve como base para el desarrollo posterior, y puede ser modificado solo a través de procedimientos formales de control de cambios.



Revisión técnica formal (RTF): corrección técnica del objeto de configuración que se ha modificado; se la debe realizar en casi la mayoría de los cambios triviales.



Repositorio: almacenamiento centralizado de los componentes de un mismo sistema, incluyendo las distintas versiones de cada componente. El repositorio permite ahorrar espacio de almacenamiento y evita guardar por duplicado elementos comunes a varias versiones o configuraciones; facilita el almacenar información de la evolución del sistema (historia), y no solo de los componentes en sí.

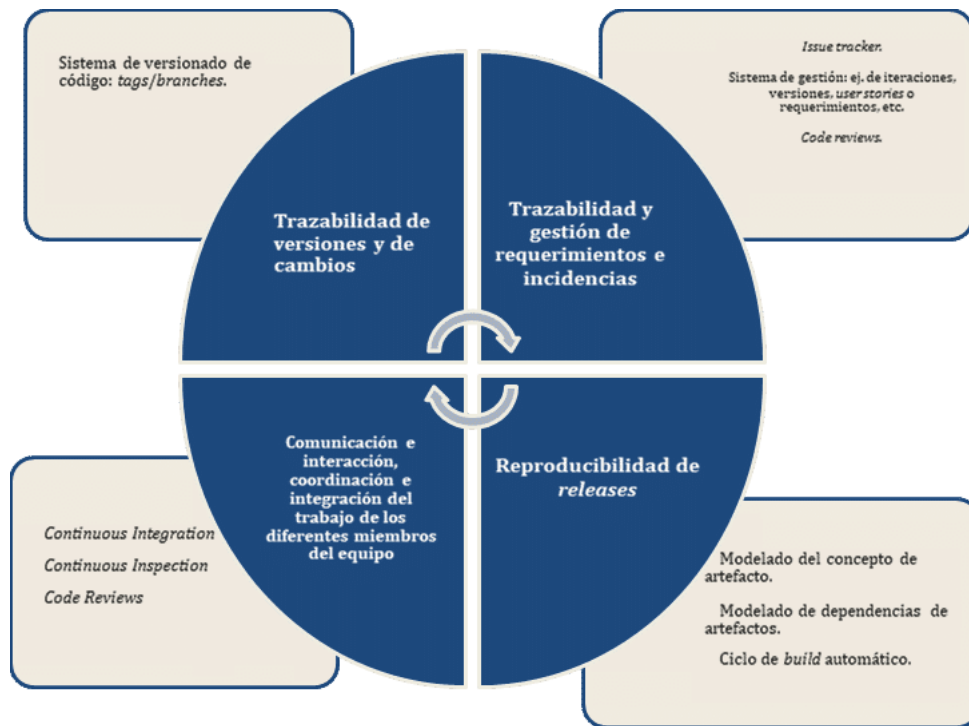
Tema 2: ¿Para qué sirve el *software configuration management* (SCM)?

El proceso de la gestión de la configuración del **software** tiene varios beneficios en la organización en la que se lo practique. Desarrolladores, **testers** (probadores), jefes de proyecto, personal encargado de la calidad del sistema y clientes pueden obtener beneficios del proceso de la gestión de la configuración del **software**. Entre los beneficios, pueden nombrarse los siguientes:

- Provee la habilidad de dar seguimiento a los cambios durante el desarrollo, sea este secuencial o en paralelo.
- Organiza las tareas y actividades que mantienen la integridad del **software**.
- Asegura la configuración correcta del **software**.
- Asegura que los ingenieros apliquen cambios correctamente en la línea base o en la versión del **software**.
- Ayuda a reducir el costo de la mantención del ciclo de vida del **software**, que puede ser fácilmente excedido.
- Provee información para reportes que pueden ser fácilmente generados.
- Permite realizar auditorías rápidas y fáciles.
- Ayuda en la producción de un **software** de mayor calidad.

Es común que existan diversas tendencias en torno al uso del GCS, algunas más enfocadas al control de la trazabilidad, otras, por el contrario, van más orientadas al control de cambios que se efectúan en los entregables y las combinaciones múltiples de ambas, además del uso de marcos de trabajo que pudieran dar un sentido de integración. Para efectos de este compendio, les mostraremos un entorno de trabajo que contemple la mayoría de los requerimientos de SCM, que se ajusten más a lo que pudieran necesitar. Este es, justamente, uno de los objetivos de esta materia, es decir, que aprendan a armar ese entorno de trabajo comprendiendo los problemas. Algunos tipos de herramientas que se necesitan para cada punto podrían ser las que se declaran en el gráfico siguiente.

Figura 1: Herramientas para SCM



Fuente: elaboración propia.

Tema 3: Herramientas para el desarrollo de software

Siguiendo lo establecido por Alvarado (2020):

Un sistema de gestión de configuración permite establecer y mantener la consistencia de la funcionalidad, rendimiento y atributos de configuración de un producto o servicio a lo largo de toda su vida operativa, estas tareas suelen aplicarse de forma sistemática a una infraestructura IT (tanto a configuraciones propias de servidores como a los componentes de **software** que se ejecutan sobre de ellos), las finalidades de un sistema de administración de configuraciones son:

- **Centralizar las tareas de configuración de servidores y software** en un repositorio de configuraciones común a toda la infraestructura que se desea administrar, reduciendo tiempos y errores.
- **Versionamiento y control de cambios** a lo largo del tiempo de vida de la infraestructura que se desea administrar, teniendo en cuenta que está sujeta a cambios y/o modificaciones con el tiempo que serían difícil e ineficiente seguir de manera manual, por otro lado, la respuesta ante un error o la necesidad de volver a una configuración anterior puede realizarse de manera automatizada.
- **Tiempos de configuración y aprovisionamiento.** La automatización controlada permite la configuración y aprovisionamiento de una gran y compleja infraestructura TI reduciendo los tiempos y riesgos que se tienen al realizar estas tareas de forma manual aplicándose a cada servidor o **software** a la vez. (<https://bit.ly/3xjOwMm>).

En el mercado, existen muchas **herramientas** para la configuración. Veamos algunas de ellas.

Herramientas para la gestión de configuraciones

Es posible desarrollar pruebas con las herramientas para la gestión de configuraciones a través de un experimento; cuatro de las herramientas más conocidas y utilizadas son Puppet, Chef, Ansible (Red Hat) y Salt.

Puppet

De acuerdo con Alvarado (2020):

Fue originalmente desarrollado por Luke Kanies con el objetivo de automatizar las tareas de un **sysadmin** y reducir los tiempos invertidos en la configuración, aprovisionamiento, soporte y mantenimiento en la operación de servidores. Utiliza una arquitectura agente/maestro, los agentes administran los nodos y solicitan información relevante a los maestros que controlan la información de configuración. (<https://bit.ly/3xjOwMm>).

¿Qué ofrece Puppet?

Siguiendo a Alvarado (2020):

- Orquestación de servicios y servidores.
- Automatización de configuraciones.
- Visualización y generación de reportes.
- Administración de código.
- Administración de nodos.
- Control de acceso basado en roles.

Pros:

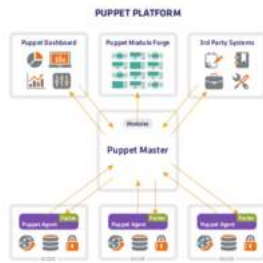
- poderosas herramientas para la generación de reportes.
- Interfaz de usuario completa y amigable.
- Tiene una documentación y comunidad bien desarrollada.
- Es una solución muy completa y estable.
- Arquitectura multimaster.

Contras:

- puede llegar a ser difícil para nuevos usuarios sin experiencia en Puppet DSL o Ruby.
- Las tareas avanzadas requieren del uso de la CLI.
- Se necesitan conocimientos de Ruby para obtener el mayor provecho.
- Requiere de la instalación de un agente.
- La curva de aprendizaje es elevada para obtener el mejor provecho. (<https://bit.ly/3xjOwMm>).

Figura 2: Puppet

Fuente: [imagen sin título sobre Puppet], s. f., <https://bit.ly/3hCRsgd>.



Chef

Dentro de esta herramienta, existen distintos productos para tareas específicas.

- **Chef Inspect**

Provee un lenguaje especificado para describir los requerimientos de seguridad y cumplimiento de reglas y normas que pueden ser compartidas y distribuidas entre equipos de la infraestructura y entre los equipos de ingenieros de soporte, operaciones e ingenieros de seguridad implicados en el manejo, soporte y administración de los servicios y servidores implicados. (Alvarado, 2020, <https://bit.ly/3xjOwMm>).

Siguiendo a Alvarado (2020):

Chef Inspect permite:

- La automatización de la seguridad de forma distribuida y simplificada.
- La automatización del cumplimiento de normas y requerimientos.
- La simplificación de la auditoría de seguridad manteniendo los datos relevantes siempre al día.

CHEF INSPECT es una herramienta de administración y estandarización de la auditoría y configuración de la seguridad TI.

- **CHEF HÁBITAT**

Automatiza la administración del ciclo de vida de las aplicaciones para ser desplegadas y ejecutadas a escala.

CHEF HÁBITAT permite la construcción, despliegue y ejecución sobre un centro de datos tradicional o microservicios sobre contenedores, tiene soporte incorporado para el descubrimiento de servicios, administración de la configuración, supervisión, monitoreo, verificaciones de seguridad, entre otros.

- **CHEF AUTOMATE**

Ofrece un **dashboard** (panel) de grado empresarial con un enfoque DevOps y visibilidad operacional que permite la colaboración entre equipos diversificados de una empresa (desarrolladores, operadores, DevOps, seguridad) así como un conjunto de herramientas de análisis que permiten entregar aplicaciones y cambios de infraestructura que responden a las necesidades y tiempos de negocio. Provee las herramientas e información para mejorar el rendimiento sobre múltiples centros de datos y/o proveedores de servicios en la nube

Pros:

- Plataforma y herramientas estables y orientadas a las necesidades de las empresas.
- Enfoque basado en el cumplimiento y la seguridad.
- Cuenta con una documentación y comunidad bien conformada y activa.

Contras:

- Curva de aprendizaje elevada con respecto a otras herramientas.
- La configuración inicial puede resultar complicada.
- Los cambios no se realizan de inmediato.
- Dependiendo de las necesidades es posible requerir más de una de las herramientas de CHEF.

¿Qué ofrece CHEF?

- Automatización de la infraestructura TI.
- Automatización de nube.
- Automatización para flujos de trabajo DevOps.
- Administración de «Compliance & Security» para cumplir con políticas y normativas.
- Flujo de trabajo automatizado para CD. (<https://bit.ly/3xjOwMm>).

Figura 3: Chef

Fuente: [imagen sin título sobre Chef], s. f., <https://bit.ly/3hihTsN>.



Ansible (red Hat)

Fue desarrollado para simplificar la complejidad de las tareas de orquestación y administración de la configuración, actualmente Ansible es propiedad de Red Hat teniendo un gran soporte por parte de la empresa sobre instalaciones Red Hat.

Ansible es una solución de administración de la seguridad que no necesita la instalación y manejo de agentes, la administración de los inventarios de servidores y grupos de servidores se realiza mediante archivos de texto plano fáciles de leer y mantener.

¿Qué ofrece Ansible?

- Aprovisionamiento simplificado.
- Administración de la configuración.
- Despliegue de aplicaciones.
- Flujos automatizados para CD.
- Integración de cumplimiento de políticas de seguridad dentro de procesos automatizados.
- Orquestación simplificada.

Pros:

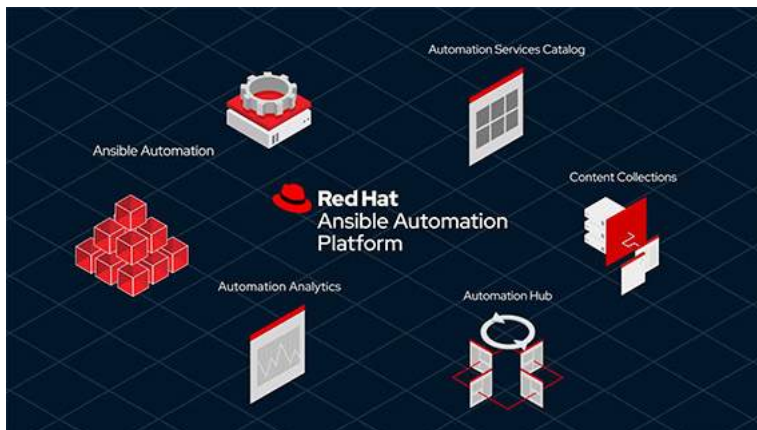
- No requiere de la instalación y administración de agentes.
- Ejecuciones remotas sencillas.
- Instalación y configuraciones sencillas.
- Orden de ejecución secuencial.
- Soporta modelos **push** (empuje) y **pull** (tire).

Contras:

- Su enfoque es mayor para la orquestación que para la configuración.
- La comunicación SSH puede ser un poco más lenta para infraestructuras grandes.
- Requiere acceso SSH y Python instalado en los equipos. (Alvarado, 2020, <https://bit.ly/3xjOwMm>).

Figura 4: Ansible (Red Hat)

Fuente: [imagen sin título sobre Ansible (Red Haut), s. f., <https://bit.ly/3xjOwMm>].



Salt

Es un sistema de administración de configuraciones y de ejecución remota que permite la ejecución de comandos en varios equipos de forma paralela escrito en Python, tiene un modelo de ejecución de comandos mediante protocolo SSH. Puede mantener una configuración "multimaster" en la que un nodo maestro (Salt Master) es capaz de administrar y controlar otros nodos esclavos.

¿Qué ofrece SaltStack?

- Orquestación y automatización para CloudOps.
- Automatización de ITOps.
- Automatización de SecOps.
- Monitoreo de aplicaciones e infraestructura con capacidades de *auto-healing* (aplicaciones con la posibilidad de auto-arreglarse).

Pros:

- Es una buena opción para una solución empresarial enfocada en el cumplimiento de normas y seguridad.
- Tolerante a fallos, gracias a su arquitectura multimaster.
- Altamente escalable, puede ser escalado hasta con 10 000 nodos Minion.
- Fácil de instalar.
- Seguro y ligero.

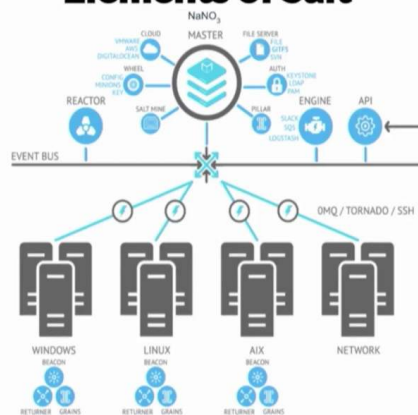
Contras:

- Tiene una comunidad y documentación menos abierta y desarrollada que las otras opciones.
- Su enfoque es mayormente SecOps e ITOps por lo que no ofrece una solución más equilibrada como otros.
- La interfaz web no ofrece muchas opciones y características.
- La plataforma no es tan madura en comparación con Puppet o Chef.

Figura 5: SALT

Fuente: [imagen sin título sobre SALT], s. f., <https://bit.ly/3xjOwMm>.

Elements of Salt



Dentro de las herramientas para testeo en la gestión de configuración del **software**, se encuentran las que se describen en los próximos párrafos.

Las herramientas para realizar las pruebas de gestión de configuraciones son herramientas **open source** (**software** de código abierto) que ofrecen diversos alcances y funcionalidades a la hora de gestionar los procesos de **testers**.

Un elemento importante a la hora de considerar aplicaciones de **testing** en los proyectos es la curva de aprendizaje que esta pueda tener, lo cual ofrece un desafío extra al equipo a la hora de la elección de la herramienta.

Para esta sección, elegimos cuatro alternativas; saber cuál es la mejor es difícil. En este sentido, Alvarado (2020) sostiene:

¿Cuál deberías elegir? ¿Cuál es mejor?

No existe una respuesta para estas preguntas, aunque si bien parecen funcionar indistintamente para los mismos fines, cada una de estas herramientas tiene ciertas funcionalidades y facilidades que las caracterizan y hacen únicas las unas de las otras, más sin embargo, es un hecho que dependiendo del entorno en el que se implementen y de la cultura organizacional de los equipos que se vean beneficiados de ellos una resultará sobresalir más que otras o incluso como suele pasar en algunos casos, las necesidades de los grupos podrían necesitar de más de una de ellas. (<https://bit.ly/3xjOwMm>).

CVS

Es una forma de trabajo relacionada con el desarrollo de **software**, que guarda el código fuente, almacena las versiones de todos los archivos y lleva un registro del acceso de los participantes; además, permite combinar el código de dos o más programadores que trabajen en un mismo archivo.

Subversión: es una herramienta de control de versiones, que utiliza un árbol de archivos en un repositorio que guarda todos los cambios que se realicen a los archivos y a los directorios, y además facilita realizar revisiones de las modificaciones de los componentes del sistema.

Figura 6: CVS

Fuente: [imagen sin título sobre CVS], s. f., <https://bit.ly/36bzNHg>.

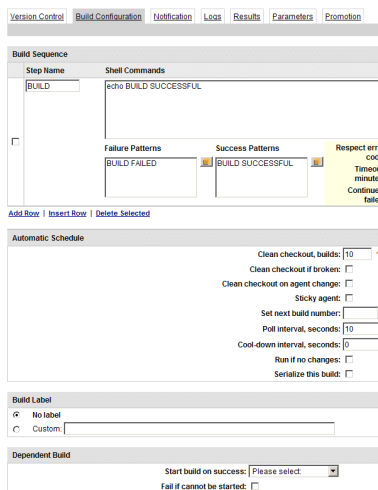


Mercurial

Es un sistema multiplataforma de control de versiones, desarrollado en su mayoría en lenguaje Python; facilita la gestión de una gran cantidad de archivos usando un repositorio distribuido, a diferencia de los sistemas de control de versiones tradicionales.

Figura 9: Mercurial

Fuente: Captura de pantalla de Mercurial (2005).



Bazaar

Esta herramienta fue diseñada principalmente para facilitar la contribución en grandes proyectos de **software** libre; soporta flujos de trabajos distribuidos o centralizados, y puede ser usada por un único programador o por varios a través de la red. Para evaluar las anteriores herramientas, se emplea el modelo de evaluación abierto para **software** libre y de código abierto, compuesto por cuatro fases, que a su vez están conformadas por actividades. Al ser un modelo abierto, las características de las herramientas a evaluar quedan a criterio del evaluador; de la misma manera, si se requieren evaluar características diferentes, puede hacerse.

Figura 10: Bazaar

Fuente: [Imagen sin título sobre Bazaar]. (s.f.)



Tema 4: Ítems de configuración

¿Qué es un ítem de configuración?

Es todo o parte de un producto de trabajo (por ejemplo, **un documento**, todos los casos de prueba o un componente de programa nominado).

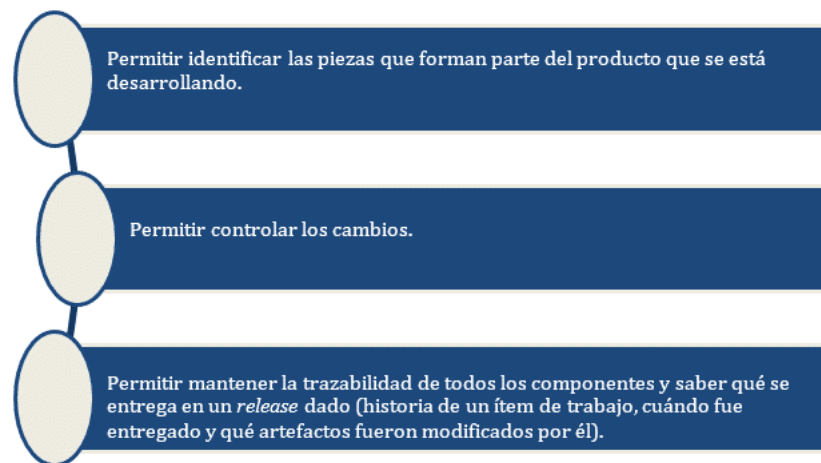
Los ítems de configuración se organizan para formar elementos que puedan catalogarse con un solo nombre en la base de datos del proyecto.

¿Qué es un elemento?

Un **elemento** es la parte del producto final que se selecciona para mantener bajo control toda su configuración; pero, a su vez, no todos los elementos deben mantenerse bajo control de configuración, se deben seleccionar cuáles se mantienen y cuándo se suman al proceso.

Para identificar estos elementos, debemos tener en cuenta los aspectos que se muestran en la figura 11.

Figura 11: Aspectos para identificar elementos



Fuente: elaboración propia.

En este sentido, puede decirse que el proceso de identificación de la configuración es el siguiente:

- la tarea de identificación empieza con la definición de los elementos de la configuración **software** representativos de los productos en cada línea base establecida. El formato, los contenidos y los mecanismos de control para toda la documentación son definidos para enlazar la información cuando la jerarquía de la configuración se despliega.
- Se asignan identificadores apropiados a todos los programas, documentos y periféricos, usando un esquema numerado que proporciona información sobre el elemento de la configuración **software**.
- Finalmente, la identificación debe facilitar el control de cambios, para acomodar actualizaciones y modificaciones.

La configuración **software** se mantiene durante la vida del sistema **software**.

Se establecen bibliotecas y ayudas de referencia como soportes a las configuraciones generadas. Pueden aplicarse tres enfoques fundamentales al control de la documentación:

- 1 todos los documentos **software** y otros elementos de cada configuración son mantenidos como parte de una biblioteca de esquema/documentación de ingeniería ya establecida.
- 2 Se establece una librería de **software** especial para todas las configuraciones **software**.
- 3 Se establece una librería de **software online**, soportada por un procesador de textos y facilidades de recuperación de documentos accedidos por terminales de computadora.

Independientemente del enfoque del control de la documentación, debe establecerse un sistema de referencia.

A continuación, describimos una guía para un sistema de numeración de documentos. En ella, cada documento es referenciado por un número único que contiene los siguientes elementos:

- 1 un identificador único de proyecto.
- 2 Un identificador del elemento de la configuración.
- 3 Un identificador del elemento de la configuración.
- 4 Un código del atributo.

A continuación, dejamos una plantilla de ejemplo, para documentar los ítems.

Tabla 3. N.º de referencia del documento: XXX-YYY-Z-RL-NNN

XXX-YYY	Es un identificador común para cada proyecto: XXX es el identificador de la empresa de software; YYY es el identificador del proyecto.

Z	<p>Es un identificador del elemento.</p> <p>P - Plan.</p> <p>R - Especificación de requisitos.</p> <p>D - Documento de diseño.</p> <p>S - Listado fuente.</p> <p>T - Documentación de prueba.</p> <p>U - Manual del usuario.</p> <p>I - Guía de instalación.</p> <p>M - Manual de mantenimiento.</p>
RL	Es el nivel de revisión.
NNN	Es un código de atributo (por ejemplo, la fecha) definido por el desarrollador del software para reflejar cierta información importante del elemento de la configuración.

Fuente: elaboración propia.

Los datos anteriores aparecen en cada elemento de la configuración y deben ser usados allá donde se hagan referencias cruzadas.

Tabla 4. Ejemplo

SPC-001-P-0-3/80	Este es el plan del proyecto 1 de la empresa Special Purpose Computer Center. Es el documento original puesto bajo control de cambios en marzo de 1980.
SPC-001-P-1-5/80	Esta es la revisión 1 al plan. Fue puesta bajo control de cambios en mayo de 1980.
SPC-005-R-3-9/81	Esta es la revisión 3 de la especificación de requerimientos para el proyecto número 5 de SPCC. Puesto bajo control de cambios en septiembre de 1981.

Fuente: elaboración propia.

CONTINUAR

Unidad 2: Control y seguimientos de cambios del software

Tema 1: Control de la configuración

El control de la configuración es la actividad de administrar los entregables o productos del proyecto y los documentos relacionados, durante todo el ciclo de vida del **producto**.

La gestión de configuración y control de cambios es la disciplina que se encarga de llevar a cabo las siguientes acciones:

- ☐ identificar los elementos del proyecto que deben estar bajo configuración.
- ☐ Restringir los cambios a dichos elementos.
- ☐ Auditar los cambios a estos elementos.
- ☐ Definir y gestionar la configuración de estos elementos.

¿Qué es el control de cambios?

Es una metodología **que** se utiliza para gestionar cualquier solicitud de **cambio que** afecte la línea de base de su proyecto. Es una forma de capturar ese **cambio** desde el punto en el **que** se ha identificado en cada paso del ciclo del proyecto. En este sentido, se establece lo siguiente:

Los métodos, procesos y herramientas utilizados para proveer este ambiente de configuración son parte esencial del proceso de desarrollo de **software**.

La gestión de configuración y control de cambios es esencial al momento de tener control sobre todos los elementos generados por los integrantes del equipo de proyecto. Este control ayuda a eliminar la posibilidad de confusiones que pueden resultar de alto costo para el proyecto y asegurar que no existan inconsistencias en el sistema desarrollado, generadas por los elementos que se describen a continuación:

- **actualizaciones simultáneas**

Cuando varios integrantes del equipo trabajan sobre un mismo elemento al mismo tiempo.

- **Problemas en la notificación de cambios**

Cuando un problema fue resuelto para algún elemento que es compartido por varios desarrolladores y alguno de ellos no fue notificado de dicho cambio.

- **Múltiples versiones**

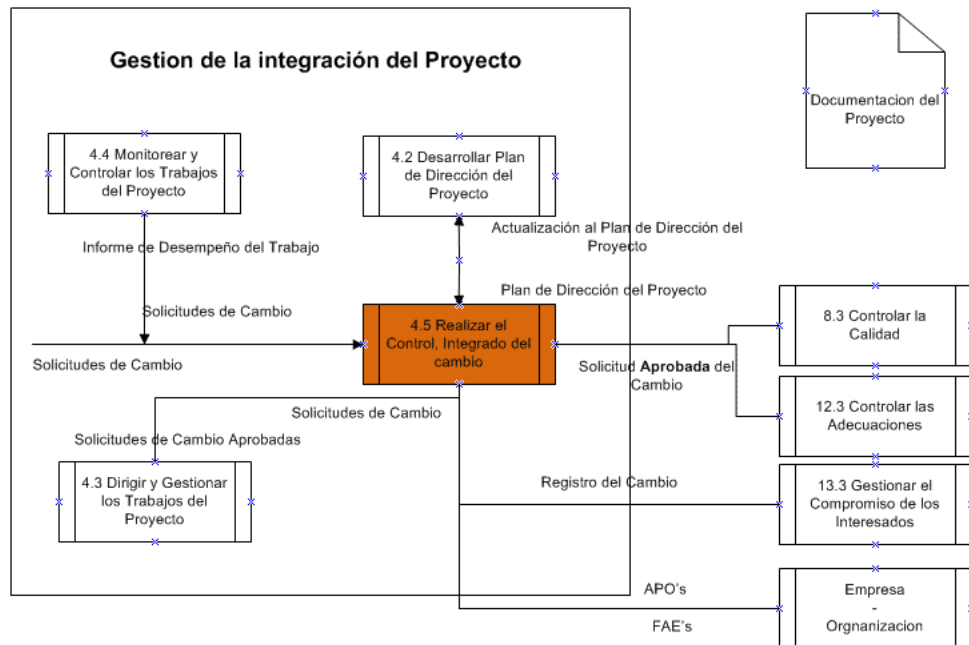
Usualmente se tienen varias versiones del producto en desarrollo, por ejemplo, una versión de desarrollo y otra de **test**, y se quiere que cuando haya un cambio en una este se vea reflejado en las demás versiones.

Algunos de los beneficios que se obtienen de la realización de una buena gestión de configuración y control de cambios son los siguientes:

- brindar apoyo a los métodos de desarrollo de **software**.
- Mantener la integridad del producto.
- Asegurar la completitud y correctitud de los elementos bajo configuración.
- Proveer un ambiente estable y controlado de trabajo.
- Restringir y controlar los cambios que se realizan.
- Proveer mecanismos de rastreo de por qué, cuándo y quién realizó un cambio. (Facultad de Ingeniería de la Universidad de la República del Uruguay, s. f., <https://bit.ly/3qPHVHa>).

Habiendo leído todo lo anterior, nos preguntamos cuáles serían los objetivos de la gestión de configuración y control de cambio.

Figura 12: Gestión de la integración del proyecto



Fuente: elaboración propia.

De acuerdo con Minmer Global (s. f.), es importante realizar lo siguiente:

[...] establecer un método progresivo para identificar y solicitar cambios a las líneas base del proyecto y para determinar el valor y la eficacia de esos cambios. Otro objetivo del control integrado de cambios del proyecto es proporcionar oportunidades de validar y mejorar el proyecto de manera continua. (<https://bit.ly/2TuPy9L>).

¿Cómo comunicar los cambios?

El control integrado de cambios del proyecto proporciona el mecanismo que permite al equipo de dirección del proyecto comunicar a los interesados, de manera sistemática, todos los cambios aprobados y rechazados en el proyecto.

Realizar correctamente la gestión de cambios

Es un elemento imprescindible en cualquier proyecto exitoso. La gestión integrada de cambios abarca diferentes acciones y áreas dentro del proyecto que deben ser revisadas y, si aplica, actualizadas con cada cambio realizado.

Metodología

Figura 13: Metodología ágil vs. tradicional

Tradicional	Ágil
<p>Su naturaleza es resistirse al cambio.</p> <p>No hay retroalimentación ante los problemas.</p> <p>Desfase entre lo solicitado y lo entregado.</p> <p>Desfase entre el producto y la situación real del mercado.</p> <p>Exceso de burocracia.</p>	<p>En cambio forma parte del proceso.</p> <p>Aprendizaje continuo.</p> <p>Se desarrolla en base a las necesidades.</p> <p>Siempre se agrega valor.</p> <p>Poco énfasis en la arquitectura.</p> <p>Punto equilibrado entre lo "excesivamente burocrático" y lo "anárquico y caótico".</p>

Fuente: elaboración propia.

Tipos de cambio

Según lo establece EduRed (s. f.):

Se pueden considerar fundamentalmente dos tipos de cambios:

- corrección de un defecto: los clientes tienden a clasificar todos los cambios en esta categoría.
- Mejora del sistema: los programadores, sin embargo, los suelen clasificar aquí.

Para solucionar el problema de determinar realmente de qué tipo es un cambio, es importante la trazabilidad de los requisitos. Por lo general, se establecen varios niveles de control de cambios:

- control de cambios informal: antes de que el elemento de configuración del **software** pase a formar parte de una línea base, aquel que haya desarrollado el elemento de configuración del **software** podrá realizar cualquier cambio justificado sobre él.
- Control de cambios al nivel del proyecto o semiinformal: una vez que el elemento de configuración del **software** pasa la revisión técnica formal y se convierte en una línea base, para que el encargado del desarrollo pueda realizar un cambio debe recibir la aprobación de parte de los siguientes grupos:
- el director del proyecto, si es un cambio local.
- El Comité de Control de Cambios, si el cambio tiene algún impacto sobre otros elementos de configuración del **software**.

- Control de cambios formal: se suele adoptar una vez que se empieza a comercializar el producto, cuando se transfieren los ECS a la Biblioteca Maestra. Todo cambio deberá ser aprobado por el Comité de Control de Cambios.

Es necesario establecer, de forma precisa, al comienzo de cada proyecto, cuál será el proceso de gestión de cambios que se va a utilizar. Para ello, será necesario definir:

- políticas a nivel organizativo que promuevan las actividades de control de cambios.
- Los estándares que se van a adoptar y a los que será necesario ajustarse.
- Los procedimientos que se van a utilizar para poner en práctica las políticas de gestión de configuración.

Las **políticas** de gestión de configuración deben reflejar la filosofía y las metas de la organización en cuanto a las actividades de gestión de configuración. Mientras que el objetivo de las políticas es definir el porqué de la gestión de configuración, deben establecer un marco para definir el qué, el cuándo, el cómo y el quién:

- las responsabilidades.
- El contenido de las líneas base.
- Los tipos de cambios que se van a controlar.
- Las funciones del comité de control de cambios.
- El flujo de documentación entre el solicitante de un cambio y el Comité de Control de Cambios.
- Los criterios para la valoración de las solicitudes de cambio, tanto de tipo técnico como de gestión.

Proceso de control de cambios

No hay ningún estándar para el control informal o interno de los cambios, aunque sí hay algunas recomendaciones (IEEE STD 1042 Guide to Software Configuration Management). En cuanto al control de cambios formal, se puede estructurar de muchas formas. Vamos a ver cuáles son las etapas típicas de un proceso formal, es decir, el proceso que habría que seguir para hacer un cambio sobre una línea base:

- iniciación del cambio: se presenta una solicitud de cambio, que puede venir provocada por un problema que se ha detectado o por un cambio en los requisitos.
- Clasificación y registro de la solicitud de cambio.
- Aprobación o rechazo inicial de la solicitud de cambio. De ello suele ser responsable el Comité de Control de Cambios.
- Evaluación de la solicitud de cambio, si ha sido aprobada, para calcular el esfuerzo técnico, los posibles efectos secundarios, el impacto global sobre otras funciones del sistema y el coste estimado del cambio. Como resultado se obtiene un informe de cambio.
- Se presenta el informe de cambio al Comité de Control de Cambios. Si se considera que el cambio es beneficioso, se genera una orden de cambio (también llamada orden de cambio de ingeniería), que describe el cambio a realizar, las restricciones que se deben respetar y los criterios de revisión y de auditoría. Esta orden de cambio es asignada a alguno de los ingenieros de **software** para que se encargue de llevarlo a cabo. En este momento, el objeto a cambiar se da de baja en la Biblioteca de Soporte al Proyecto.
- Se realiza el cambio, entrando en un proceso de seguimiento y control.
- Una vez finalizado el cambio, se certifica, mediante una revisión, que se ha efectuado correctamente el cambio y con ello se ha corregido el problema detectado o bien se han satisfecho los requisitos modificados. El objeto se devuelve a la Biblioteca de Soporte al Proyecto.
- Se notifica el resultado al originador del cambio.

Al definir este proceso, será también necesario:

- definir los mecanismos para solicitar cambios sobre los elementos de configuración.
- Definir los mecanismos para analizar y evaluar el impacto de las solicitudes de cambio.
- Definir los mecanismos para aprobar o rechazar las solicitudes de cambio.
- Definir los mecanismos para controlar la realización de los cambios aprobados.

Los procesos de alta y baja de la Biblioteca del Proyecto implementan dos elementos importantes del control de cambios: el control de acceso y el control de sincronización:

- El control de acceso se refiere a los derechos que tienen los diferentes miembros del equipo de desarrollo para acceder y modificar ECS concretos. Así, por ejemplo, hay que controlar el acceso del ingeniero de **software** que da de baja el ECS de la Biblioteca de Proyecto para realizar un cambio aprobado por una orden de cambio.
- El control de sincronización ayuda a asegurar que los cambios en paralelo, realizados por equipos o personas diferentes, no se sobrescriben. Así, cuando un ECS se da de baja de la Biblioteca de Soporte, el control de sincronización bloquea el objeto para que no se puedan hacer más actualizaciones sobre él hasta que se haya reemplazado con una nueva versión. El almacén de una herramienta de control de versiones se puede considerar como la Biblioteca de Soporte o de Proyecto. Estas herramientas ofrecen también de forma automática el control de acceso y control de sincronización. (<https://bit.ly/3dNsLNf>).

Figura 14: Control de versiones

Control de Versiones



Fuente: [Imagen sin título sobre control de versiones]. (s.f.).

Tema 2: Estado de cuenta

El estado de cuentas guarda contabilidad actual de una configuración y proporciona la trazabilidad de los elementos de configuración a lo largo de su desarrollo y funcionamiento. Los informes de estado regulares indicarán si las solicitudes de cambio se procesan de manera oportuna y pueden resaltar los productos que son objeto de frecuentes solicitudes de cambio o partes interesadas que son fuentes comunes de las solicitudes de cambio.

Informe del estado de los elementos de configuración

El informe del estado de la configuración reporta la información necesaria para gestionar de forma efectiva la configuración de software. En esta actividad, se deberá diseñar y operar un sistema para la captura y reporte de la información necesaria a medida que avanza el proceso de desarrollo.

Como en cualquier sistema de información, la información sobre el estado de la configuración que se quiere gestionar debe ser identificada, recogida y mantenida. Son necesarias diversas métricas e información para dar soporte al proceso de gestión de configuración. El tipo de información disponible incluye la identificación de configuración aprobada, así como el estado actual de la implementación de los cambios. Por ejemplo, información del siguiente tipo:

- un registro de documentación de configuración aprobada.
- La designación de un responsable de los elementos de configuración del proyecto.
- El estado de cambios propuestos y desviaciones de la configuración.
- La implementación del estado de los cambios aprobados.
- La configuración de todas las unidades de los elementos de configuración en el inventario.
- Resultados de las auditorías.

Para llevar a cabo estas actividades de obtención de datos y generación de informes, se hace necesario el soporte de una herramienta automatizada.

Tema 3: Auditorías de configuración

¿Qué es una auditoría de configuración? Una auditoría de configuración física (PCA) identifica los componentes de un producto que se van a desplegar desde el repositorio de proyectos.

Actividades de la auditoría de la configuración

Esta función a veces se considera fuera de la gestión de configuración y dentro de la garantía de calidad. También tiene relación con las actividades de validación y verificación. En realidad, es un punto de intersección entre todas ellas. Es la actividad de GCS más costosa. Requiere de personal experimentado con un gran conocimiento del proceso de desarrollo. Sin embargo, debe ser realizada por personal ajeno al equipo de desarrollo técnico para mantener la objetividad de la auditoría.

Se pueden diferenciar tres tipos de actividades:



revisiones de fase. Se realizan al finalizar cada fase del desarrollo y su objetivo es examinar los productos de dicha fase. Las revisiones propias de la gestión de configuración son aquellas en las que se establecerán las líneas base. El objetivo de esta revisión es descubrir problemas, no comprobar que todo está bien. Hay que ser capaz de desenmascarar los problemas ocultos y sutiles, no solo los que son obvios.



Revisiones de cambios: se realizan para comprobar que los cambios aprobados sobre una línea base se han realizado correctamente.





Auditorías: se realizan al final del proceso de desarrollo de software y su objetivo es examinar el producto en su conjunto.

Revisiones de la auditoría de la configuración

En lo que respecta a las revisiones de la auditoría de la configuración, se establece lo siguiente:

Las revisiones se deben realizar de forma continua, durante todo el proceso de desarrollo, y no solo al finalizar este, cuando los problemas ya no tienen solución.

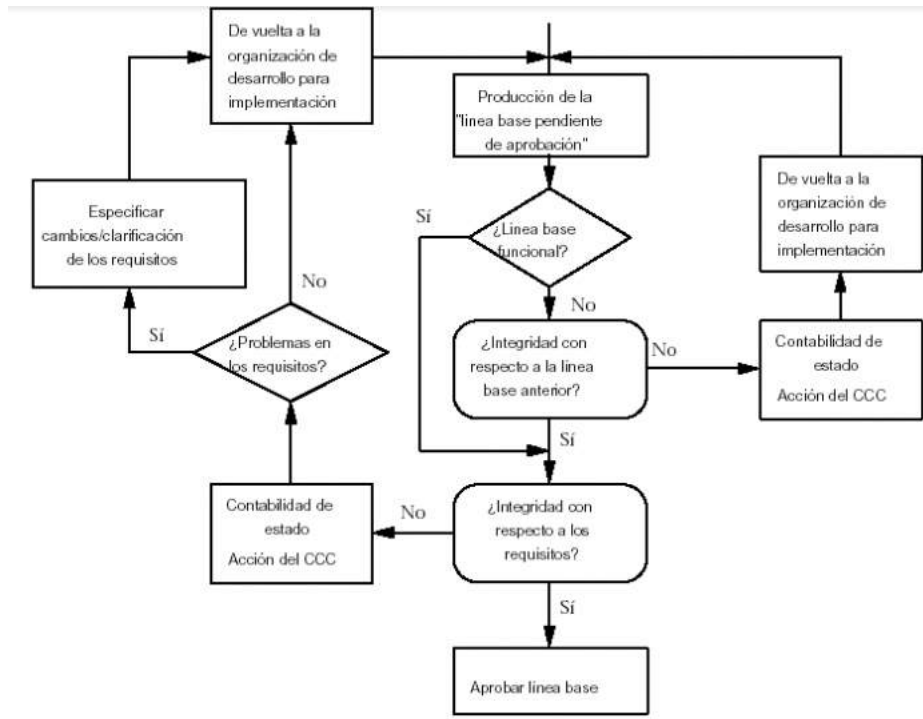
La tarea de revisión implica tres tipos de funciones:

- verificar que la configuración actual del **software** se corresponde con lo que era en fases anteriores. Debe haber correspondencia y trazabilidad entre los elementos de configuración que aparecen en una línea base y los que aparecen en la línea base que la preceden y que la siguen. La verificación se realiza con respecto a la línea base precedente.
- Validar que la configuración actual del **software** satisface la función que se esperaba del producto en cada hito del proceso de desarrollo. La validación se realiza con respecto a los requisitos del sistema.
- Valorar si una determinada línea base, teniendo en cuenta los resultados de la verificación y validación, y otro tipo de comprobaciones, se debe considerar aceptable o no.

El papel que juegan las revisiones en el proceso de gestión de configuración es el siguiente:

- los productos generados durante el proceso de desarrollo se agrupan, al llegar determinados hitos, en una «línea base pendiente de aprobación».
- Tiene lugar la revisión de fase.
- Si en la revisión se encuentran deficiencias en los ECS que componen la línea base, se generan los correspondientes informes de problemas, o informes de discrepancias, y se entregan al CCC, el cual recomienda ciertos cambios sobre la configuración.
- Una vez implementados los cambios propuestos, se pasa a una nueva revisión de cambios en la que se comprueba si los cambios se han implementado correctamente.
- Si en la revisión no se encuentra ninguna deficiencia, se declara «aceptable» la línea base pendiente de aprobación. Si el CCC está de acuerdo con la resolución de los revisores, la línea base se aprueba. (EcuRed, s. f., <https://bit.ly/3xInHYb>).

Figura 15: Revisiones en el proceso de gestión de configuración



Fuente: [Imagen sin título sobre Revisiones en el proceso de gestión de configuración]. (s.f.).

Tipos de auditorías de la configuración del *software*

En cuanto a las auditorías, se suelen distinguir dos tipos de auditorías de configuración:

- Auditoría funcional: [su] objetivo es comprobar que se han completado todos los **tests** (pruebas) necesarios para el elemento de configuración auditado, y que, teniendo en cuenta los resultados obtenidos en los **tests**, se puede afirmar que el elemento de configuración satisface los requisitos que se impusieron sobre él.
- Auditoría física: [su] objetivo es verificar la adecuación, completitud y precisión de la documentación que constituye las líneas base de diseño y de producto. Se trata de asegurar que representa el **software** que se ha codificado y probado. Tras la auditoría física, se establece la línea base de producto. Tiene lugar inmediatamente después de haber superado la auditoría funcional.

Y aún se puede considerar un tercer tipo de auditoría:

- revisión formal de certificación: [su] objetivo es certificar que el elemento de configuración del **software** se comporta correctamente una vez que este se encuentra en su entorno operativo. (EcuRed, s. f., <https://bit.ly/3xlnHYb>).

Cuando realizamos auditorías, debemos centrarnos en algunas cuestiones; para ello, debemos hacernos las siguientes preguntas:

1

¿Se hizo el cambio especificado en el orden de cambio de ingeniería (OCI)? ¿Se incorporaron modificaciones adicionales?

2

¿Se realizó una revisión técnica formal para comprobar la corrección técnica?

3

¿Se siguieron adecuadamente los modelos de ingeniería del **software**?

4

¿Se marcaron los cambios en el ECS? ¿Se especificaron la fecha y el autor del cambio?

5

¿Refleja los cambios en la identificación del ECS?

6

¿Se siguieron los procedimientos del GCS para señalar el cambio, registrarlo y divulgarlo?

7

¿Se han actualizado adecuadamente todos los ECS relacionados?

Verificación y auditoría

Siguiendo lo establecido por el INTECO (2008):

Una auditoría de **software** es una actividad llevada a cabo para evaluar, de forma independiente y objetiva, la conformidad de los productos y procesos **software** con respecto a las regulaciones, estándares, guías, planes y procedimientos aplicables (IEEE1028-97).

Las auditorías se llevarán a cabo de acuerdo a un proceso bien definido que detalla los roles y responsabilidades de los auditores. Cada auditoría debe ser cuidadosamente planificada, en función de la naturaleza del proyecto y de los requisitos. El uso de herramientas que ayuden en la planificación y realización de auditorías facilita en gran medida el proceso.

Estas auditorías incluirán:

- Objetivo: el objetivo de todas las auditorías es verificar que, en un momento dado, la línea base se compone de una colección consistente y bien definida de productos.
- Elementos de configuración bajo auditoría: se elegirán uno o más elementos de configuración de mayor prioridad en la línea base.
- Agenda de auditorías: antes de la liberación o actualización.
- Conducción: las auditorías serán dirigidas por el SCMR.
- Participantes: SCMR y los autores de los elementos de configuración a auditar.
- Documentos requeridos: documentos de SCR y reportes de estado de la configuración generados.
- Reportes de deficiencias y acciones correctivas: determinadas por los participantes.
- Criterio de aprobación: lo determina el SCMR.

La actividad de auditoría de configuración de **software** determinará en qué medida un elemento de configuración satisface sus características funcionales y físicas requeridas. Se pueden realizar auditorías de este tipo en puntos clave del proceso de desarrollo. El resultado satisfactorio de una auditoría se puede utilizar como prerrequisito para establecer una línea base del producto.

El objetivo de las auditorías de gestión de configuración es asegurarse de que:

- Los elementos de configuración se encuentran en el directorio apropiado.
- El estado actual de los elementos de configuración es consistente.
- La información de línea base se mantiene de forma correcta.

- Se verifica la conformidad con estándares y procedimientos aplicables a la gestión de configuración, por ejemplo, comprobando si se usa la versión correcta del documento de diseño para realizar la codificación.

Como resultado de la auditoría se deberá generar un informe donde se registren todas las no conformidades detectadas y así iniciar un plan de mejora para solucionarlas. Después de una auditoría de configuración exitosa se puede establecer una línea base del producto.

La verificación no se realiza sobre los propios productos, sino que consiste en comprobar que los productos que conforman una línea base están gestionados correctamente bajo el control de configuración, que todos los cambios realizados sobre estos productos han sido registrados y, por tanto, se puede establecer una trazabilidad entre cambios y productos afectados. (pp. 27-28).

Tema 4: Releases

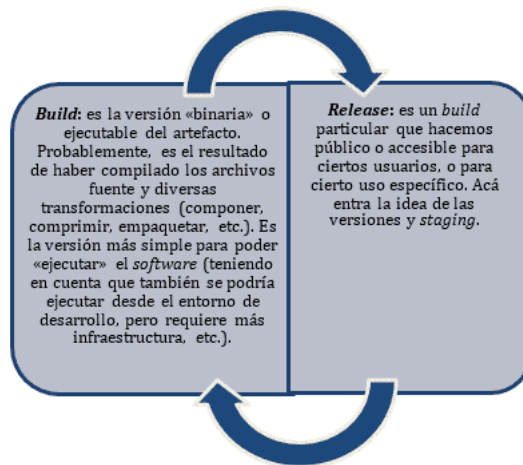
¿Qué permite la gestión de versiones?

La gestión de versiones se refiere al proceso de planificar, diseñar, programar, probar, implementar y controlar las versiones de **software**. [...] Ayudan a facilitar la transición a la entrega continua, y trabajan mediante la transformación digital una versión a la vez. Esta es la nueva normalidad de la TI. (Micro Focus, s. f., <https://bit.ly/3jHzNXz>).

Un **release** es una versión que se distribuye a los clientes. Cada **release** incluye nueva funcionalidad o está concebida para diferentes plataformas de **hardware**. Un **release** del sistema no es solo el código ejecutable del sistema. Antes de seguir hablando de **release**, debemos diferenciar qué es **build** (versión ejecutable) y qué es release. A veces, usamos estos términos indiferentemente para referirnos a conceptos distintos.

Sin embargo, está bueno distinguirlos.

Figura 16: Build y release



Fuente: elaboración propia.

Para alcanzar un **release**, es posible que pasemos por varios **builds** intermedios. En general, se denomina **release** al **build** y se cataloga como «Gold», es decir, el último más estable. Aunque a veces se denominan así a todas las versiones incluidas betas y RC. Lo importante para nosotros es entender la relación entre un **build** (que deberíamos poder generar en cualquier momento desde nuestras fuentes y con los procesos y herramientas SCM que tengamos) y la idea de generar una versión o entregable.

Generación de un *build*/release

Hay un conjunto de características que sería bueno que tengamos en nuestro proceso/mecanismo de **release**. Decimos «estaría bueno», pero en realidad queremos decir que estas son características necesarias si realmente uno aspira a tener procesos de SCM efectivos. Claro que hay contados ejemplos de desarrollos en los que los **builds** se generan a mano, con complejas tareas que pueden durar varias semanas; pero es eso justamente lo que «no queremos».

Independencia del entorno

La independencia del entorno se refiere a la capacidad de generar **builds/releases** desde cualquier máquina. Es necesaria una configuración previa mínima del entorno de trabajo como, por ejemplo, instalar cierto SDK o compiladores, herramientas, etc. Esto se hace de la misma forma en la que un desarrollador configura su ambiente una única vez y luego trabaja sobre ella.

Figura 17: Independencia del entorno



Fuente: Práctica de desarrollo (s.f.). [Imagen sin título sobre independencia del entorno]. Recuperado de <https://sites.google.com/site/practicadesarrollosoft/temario/scm/releases>

Lo ideal es que luego no requiera mantenimiento. El contraejemplo de esto es el esquema en el que la generación de los **builds** solo se puede realizar desde cierta máquina dedicada o porque requiere datos/archivos locales específicos, etc.

Idealmente, desde cualquier máquina uno podría bajarse las fuentes y generar un **build**. Es importante también que esto no requiera modificar el código para ajustarlo a la propia arquitectura o máquina en particular. Por ejemplo, que tengas que agregar cierta implementación de una librería para Linux en máquinas con este S. O. y desde otras bajarte otra para Windows.

Inmutabilidad

Esta característica es más evidente para los **releases** que para los **builds**, pero aplica igualmente. En los sistemas de versionado de código, cuando comentábamos un cambio, se generaba una nueva revisión.

Figura 18: Inmutabilidad

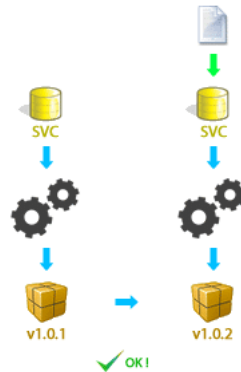


Fuente: Práctica de desarrollo (s.f.). [Imagen sin título sobre inmutabilidad]. Recuperado de <https://sites.google.com/site/practicadesarrollosoft/temario/scm/releases>

Antes, si uno necesitaba volver atrás ese cambio, debía realizar un nuevo **commit (consolidación)**. En los SVC, las revisiones son inmutables. Algo así como lo que sucedía con el «pasado» en la película *La máquina del tiempo*. Esto es porque ya sucedió y no se puede borrar todo rastro. Siempre se puede cambiar, pero a futuro, en una nueva revisión.

Esto es, en realidad, una consecuencia del mecanismo de trazabilidad estricto. Con los **releases** debería suceder lo mismo. El **release** v1.0.1 es único y no se puede modificar.

Figura 19: Mecanismo de trazabilidad estricto



Fuente: Práctica de desarrollo (s.f.). [Imagen sin título sobre mecanismo de trazabilidad estricto]. Recuperado de <https://sites.google.com/site/practicadesarrollosoft/temario/scm/releases>

En tal caso, si se requiere agregar o sacar algo, habrá que generar un nuevo v1.0.2., porque si lo modificamos, pasaríamos a tener dos v1.0.1. Luego, sería un caos saber cuál de los dos (tres o múltiples v1.0.1) tiene cada cliente en producción.

Para los **builds**, se suele agregar a la versión un número de **build**, que puede ser un auto numérico o un **timestamp** (marca temporal, como, por ejemplo, los **snapshots** (copia instantánea de volumen) en Maven).

Persistencia

Es importante que un **build o release** no se pierda por ahí o quede solo en la máquina de Pepito que lo generó. Más aún para **releases**, cuyo objetivo es publicar una versión para cierto target de personas. Esto es importante también para poder volver atrás y obtener el paquete de versiones anteriores, etc. Para esto, existen, en general, repositorios de binarios o paquetes de artefactos. Por ejemplo, en Maven hay varios productos y servicios **online (en línea)**.

Figura 20: Persistencia



Fuente: Práctica de desarrollo (s.f.). [Imagen sin título sobre persistencia]. Recuperado de <https://sites.google.com/site/practicadesarrollosoft/temario/scm/releases>

Estos repositorios pueden ser mucho más que un simple directorio compartido. Los productos, por ejemplo, tienen interfaces web de administración y permiten funcionalidades como, por ejemplo, las siguientes:

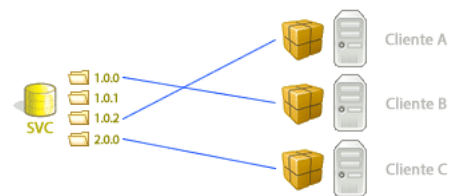
- administración de permisos.
- Gestión de repositorios (crear varios repositorios para diferentes equipos, por ejemplo).
- Gestión de **mirrors (espejos)**:
 - copias locales de repositorios extensos.
 - En general, para componentes **3rd party** (de tercera parte, esto podría servir para mantener cierto control de licencias).

Vamos a trabajar en algunas aclaraciones o comentarios a tener en cuenta: en general, si tenemos un proceso de **release** bien armado, el repositorio de binarios no es un factor crítico. En cuanto a que, si se destruye por accidente o por cualquier motivo, debería ser posible regenerar todas las versiones nuevamente, ya que tenemos trazabilidad con las fuentes. En particular, para esto sí es crítico el sistema de versionado de código con sus **tags** (etiquetas) y revisiones. Esto nos lleva al siguiente punto.

Trazabilidad

Los **builds**, y especialmente los **releases**, deberán proveer un mecanismo para poder relacionarlos de alguna forma con la revisión de código fuente del cual salieron.

Figura 21: Trazabilidad



Fuente: Práctica de desarrollo (s.f.). [Imagen sin título sobre trazabilidad]. Recuperado de <https://sites.google.com/site/practicadesarrollosoft/temario/scm/releases>

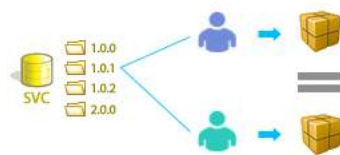
Así, dado un cliente o una plataforma que tenga instalado el producto, podremos saber exactamente qué código es el que se utilizó para generarlo. Por ejemplo, para poder arreglar un **bug**. De esto, se desprende la idea de que todos los **releases** siempre surgen a partir de un **tag**; en este sentido, generar un **release** involucra:

- preparar el código fuente.
- **Taggear (etiquetar).**
- Preparar el código fuente de la siguiente versión.

Consistencia

Por otra parte, la consistencia también se desprende de lo anteriormente dicho. Debería suceder que, si yo genero dos veces o desde diferentes máquinas un **release**, ambos sean equivalentes, que sean consistentes.

Figura 22: Consistencia



Fuente: Práctica de desarrollo (s.f.). [Imagen sin título sobre consistencia]. Recuperado de <https://sites.google.com/site/practicadesarrollosoft/temario/scm/releases>

Podría pasar, lógicamente, que los **releases** tengan distinto **hash**, pero a nivel de funcionalidad y trazabilidad (como dijimos antes) deberían ser iguales. Esto es lo que a veces se denomina reproducibilidad del **build**, es decir, la posibilidad de regenerar un **build/release** nuevamente, en caso de perderlo o necesitarlo por algún motivo.

Automatización

Idealmente, invertiremos tiempo inicialmente para configurar y crear un mecanismo de **release/build** automático, para que luego cada **release** tenga un costo mínimo de esfuerzo.

También está bueno que el mecanismo conste de una versión completamente automática, que no requiere intervención humana. Esto nos va a permitir generar **builds** automáticos, relacionados con la práctica de **continuous integration** (integración continua) y **continuous deployment** (entrega continua) (que veremos en las siguientes unidades). La idea es reducir el **overhead** (sobrecoste), manteniendo igualmente todas las características que nombramos antes. Un mecanismo que requiere mucho esfuerzo va a terminar por coartar la práctica de generar releases frecuentes o bien va a terminar forzando un trabajo manual repetitivo y, a veces, de acuerdo con la cultura de la empresa, la idea de trabajar horas extras, etc. En algunas empresas, en lugar de invertir en automatización, se invierte (de forma de renta continua) en complejas estructuras organizacionales, con áreas de **deployment y build**. Es decir, personas que están pura y exclusivamente dedicadas a generar **builds**, ya no para un solo producto, sino para varios. Así hay que generar pedidos, lo cual significa largas esperas, etc. Eventualmente, la burocracia en estos casos termina generando un canal de **releases** informales que termina por violar todas las características de trazabilidad, consistencia, etc., entonces, resulta ser peor el remedio que la enfermedad.

Ahora vamos a describir con un ejemplo cuál sería una opción de identificación de versiones. El esquema sugerido para la identificación de las distintas versiones del producto es un esquema numérico conformado de la siguiente manera:

<major_version> . <minor_version> . <patch_number>.

1

Major version: indica un cambio de funcionalidad sustancial a otras versiones del sistema.

2

Minor version: indica que el sistema es funcionalmente idéntico, pero es distinto desde el punto de vista no funcional a otras versiones.

3

Patch version: indica el número de un parche aplicado al sistema, que significa la corrección de un defecto encontrado.

¿Qué significa el código *release*?

Un *release* es una versión que se distribuye a los clientes. Un *release* del sistema no es solo el código ejecutable del sistema. Las entregas también incluyen archivos de configuración.

Resumen de este tema

- Problemas que motivan la gestión de la configuración.
- Definiciones terminológicas.
- Qué es la trazabilidad y para qué sirve.
- Implantación de planes para la gestión de la configuración.
- Identificación de versiones: árboles de características.
- Proceso de gestión de cambios: **check-in** (entrada), **check-out** (salida).
- Concepto de auditoría.
- Concepto entregas y aspectos esenciales de la gestión de entregas.
- Problema de la construcción de entregas.

Conclusiones del módulo

En ingeniería de software, la **gestión de configuración de software (SCM)** es un proceso para gestionar, organizar y controlar sistemáticamente los cambios en los documentos, códigos y otras entidades durante el ciclo de vida del desarrollo de **software**. El objetivo principal es aumentar la productividad con un mínimo de errores. SCM es parte del campo interdisciplinario de la gestión de la configuración y puede determinar con precisión quién realizó qué revisión.

Razones para configurar el **software**

- Hay varias personas que trabajan en un **software** que se actualiza continuamente.
- Puede ser un caso en el que múltiples versiones, sucursales, autores están involucrados en un proyecto de configuración de **software** y el equipo está distribuido geográficamente y trabaja al mismo tiempo.
- Los cambios en los requisitos del usuario, la política, el presupuesto y el cronograma deben adaptarse.
- El **software** debería poder ejecutarse en varias máquinas y sistemas operativos.
- Ayuda a desarrollar la coordinación entre las partes interesadas.
- El proceso SCM también es beneficioso para controlar los costos involucrados en realizar cambios en un sistema.

Tareas de CSM

- ☐ Identificación de configuración.
- ☐ Líneas de base.
- ☐ Cambio de control.
- ☐ Contabilidad del estado de configuración.
- ☐ Revisiones y auditorías de configuración.

El control de cambios es un método de procedimiento que garantiza la calidad y la coherencia cuando se realizan cambios en el objeto de configuración. En este paso, la solicitud de cambio se envía al administrador de configuración de **software**.

La contabilidad del estado de configuración rastrea cada versión durante el proceso de SCM. Esta etapa implica rastrear lo que tiene cada versión y los cambios que conducen a esta versión.

Las auditorías de configuración de **software** verifican que todo el producto de **software** satisfaga las necesidades básicas. Asegura que lo que se construye es lo que se entrega.

Cualquier **software** de gestión de cambios debe tener las siguientes tres características claves:

Gestión de concurrencia. El momento en el que dos o más tareas están sucediendo al mismo tiempo se conoce como operación concurrente. La concurrencia en contexto con SCM significa que el mismo archivo está siendo editado por varias personas al mismo tiempo.

Control de versiones: SCM utiliza un método de archivo o guardar todos los cambios realizados en el archivo. Con la ayuda de la función de archivo o guardado, es posible volver a la versión anterior en caso de problemas.

Sincronización: los usuarios pueden retirar más de un archivo o una copia completa del repositorio. Luego, el usuario trabaja en el archivo necesario y comprueba los cambios en el repositorio. Puede sincronizar su copia local para mantenerse actualizado con los cambios realizados por otros miembros del equipo.

En conclusión, podemos afirmar lo siguiente:

- las mejores prácticas de gestión de la configuración ayudan a las organizaciones a gestionar, organizar y controlar sistemáticamente los cambios en los documentos, códigos y otras entidades durante el ciclo de vida del desarrollo de **software**.
- El objetivo principal del proceso SCM es aumentar la productividad con un mínimo de errores.
- La razón principal detrás del proceso de gestión de la configuración es que hay varias personas trabajando en el **software** que se actualiza continuamente. SCM ayuda a establecer simultaneidad, sincronización y control de versiones.
- Una línea de base es una versión aceptada formalmente de un elemento de configuración de **software**.
- El control de cambios es un método de procedimiento que garantiza la calidad y la coherencia cuando se realizan cambios en el objeto de configuración.
- La contabilidad del estado de configuración rastrea cada versión durante el proceso de SCM.
- Las auditorías de configuración de **software** verifican que todo el producto de **software** satisfaga las necesidades básicas.
- El gerente de proyecto, el gerente de configuración, el desarrollador, el auditor y el usuario son participantes en el proceso de SCM.
- La planificación del proceso de SCM comienza en las primeras fases de un proyecto.
- Git, Team foundation Server y Ansible son algunas de las herramientas de SCM más populares.

Las siguientes preguntas no buscan que las respondas. Las proponemos con el objetivo de que logres pensar con ellas, a modo de repaso, las diferentes unidades del presente módulo.

Preguntas de reflexión

☐

En función de lo abordado a lo largo del módulo, ¿cuáles serían los aspectos claves que identificarías respecto al propósito, la necesidad e importancia del GCS?

☐

¿Por qué usarías el GCS en tus proyectos? Reflexioná sobre tu respuesta.

☐

Considerando la metodología GCS, te invito a pensar en los roles y beneficios que trae aparejado su accionable. ¿Por qué los considerarás importantes?

☐

Debés pensar en los diversos ejes de importancia del control de cambios e identificar los elementos que lo componen. Este ejercicio te servirá de ayuda para instancias evaluativas.

CONTINUAR

Glosario

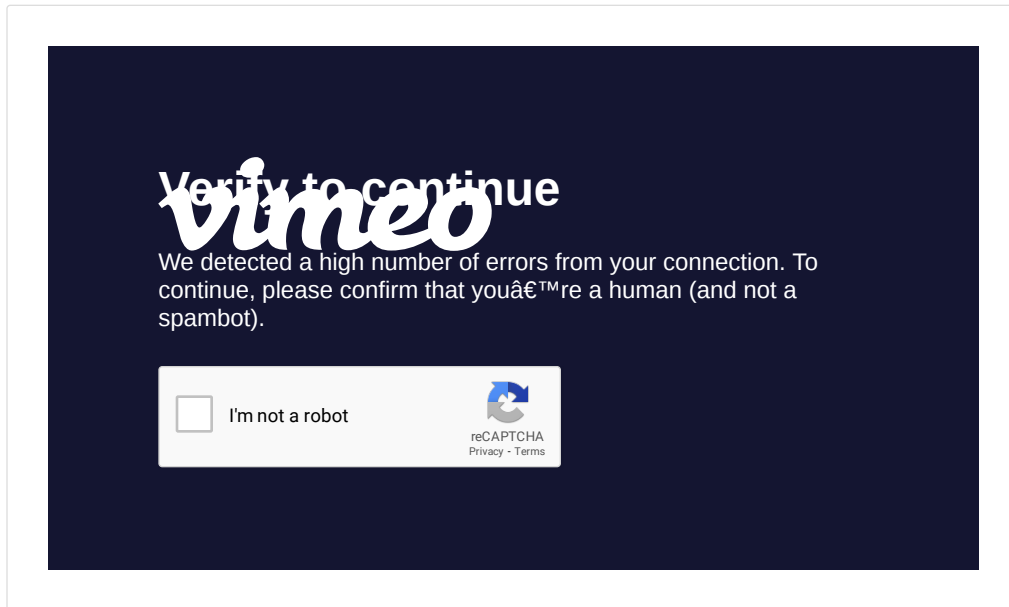
Término	Descripción
Agilidad	Entendemos por agilidad la habilidad de cambiar la posición del cuerpo de manera eficaz, controlando todas sus partes y moviéndolas con rapidez y soltura. Requiere la integración de las habilidades ya nombradas, combinando equilibrio, coordinación, velocidad, reflejos, fuerza y resistencia.
build	Build es una versión ejecutable y funcional del código de un juego. En aquellos juegos en los que es posible personalizar las características y habilidades del personaje, una build es cada una de las diversas combinaciones posibles.
CloudOps	Se entiende por CloudOps a las operaciones de gestión, entrega y consumo de software en un entorno en Cloud en el que la visibilidad de la infraestructura subyacente de una aplicación es limitada o inexistente.
commit	El comando git commit captura una instantánea de los cambios preparados en ese momento del proyecto. Las instantáneas confirmadas pueden considerarse versiones «seguras» de un proyecto: Git no las cambiará nunca, a no ser que se lo pidas expresamente.
Continuous integration y continuous deployment	La CI/CD es un método para distribuir aplicaciones a los clientes con frecuencia mediante el uso de la automatización en las etapas del desarrollo de aplicaciones. Los principales conceptos que se atribuyen a la CI/CD son la integración continua, la distribución continua y la aplicación continua.
deployment	Es un término famoso entre los desarrolladores web. Puede significar muchas cosas, dependiendo del ambiente y de la tecnología usada. Sin embargo, los significados que más se refieren a la práctica y pueden resumir su función son implantar, colocar en posición, habilitar para uso o, simplemente, publicar.
DevOps	Es una combinación de los términos ingleses development (desarrollo) y operations (operaciones); designa la unión de personas, procesos y tecnología para ofrecer valor a los clientes de forma constante.
DSL	DSL (línea de suscriptor digital) es una tecnología de banda ancha que proporciona transmisión de información a alta velocidad (hasta 7.1 Mbps) a través de una línea telefónica existente. Las velocidades son hasta 50 veces más rápidas que las de un módem por dial-up estándar de 28.8 Kbps.
IEEE 1028-97 ²	El propósito de esta norma de calidad es definir revisiones sistemáticas y auditorías

[2] IEEE 1028-1997. IEEE Standard for Software Reviews. (1997). The Institute of Electrical and Electronics Engineers. Recuperado de https://bit.ly/2VdXooz .	aplicables a la adquisición, suministro, desarrollo, operación y mantenimiento de software . Esta norma describe cómo realizar una revisión.
IT	El acrónimo IT son las siglas en inglés de information technology , cuyo significado en español es tecnología de información. Especialistas en analizar las necesidades informativas de tecnología, desarrollar y probar software , realizar tareas de la computadora y solucionar problemas.
ITOps	ITop significa IT Operational Portal. Es una aplicación web de código abierto que brinda una solución web unificada y personalizable para centros de servicio IT. Constituye una herramienta colaborativa que ofrece la capacidad de responder mejor y más rápido
OCI	Oracle Cloud Infrastructure (OCI) es una IaaS que ofrece potencia de procesamiento de alto rendimiento en las instalaciones locales para ejecutar cargas de trabajo de TI de la empresa y nativas de la nube.
Release	Un release es una versión que se distribuye a los clientes. Cada release incluye nueva funcionalidad o está concebida para diferentes plataformas de hardware . Un release del sistema no es solo el código ejecutable del sistema .
SCMR	El SCMR debe proveer la infraestructura y el entorno de configuración para el proyecto.
SecOps	SecOps , también llamado DevSecOps, es un enfoque de gestión que conecta los equipos de seguridad y operaciones, de forma similar a cómo DevOps unifica a los desarrolladores de software y profesionales de operaciones.
stagin	Con el objeto de minimizar al máximo nivel los posibles errores o problemas en la fase de carga de los procesos ETL, normalmente se reserva un área de disco para poder recuperar los datos por etapas.
tag	Los tags , a veces llamados etiquetas en español, son los «comandos» que los programas navegadores leen e interpretan para armar y dar forma a las páginas de Internet. Además del nombre, dentro de < y > el tag puede recibir o requerir ciertos argumentos.
Trazabilidad	Conjunto de aquellos procedimientos preestablecidos y autosuficientes que permiten conocer el histórico, la ubicación y la trayectoria de un producto o lote de productos a lo largo de la cadena de suministros en un momento dado, a través de unas herramientas determinadas.
Software	Conjunto de programas
Software configuration management	Gestión de configuración de Software
Configuration management data base	Base de datos de la gestión de configuración

Status	Posición
Baseline	Línea base
Testers	Probadores
Dashboard	Panel
Push	Empuje
Pull	Tire
Auto-healing	Aplicaciones con la posibilidad de auto-arreglarse
Open source	Software de código abierto
Tests	Pruebas
Timestamp	Marca temporal
Snapshots	Copia instantánea de volumen
Online	En línea
Mirrors	Espejos
3rd party	Tercera parte
Tags	Etiquetas
Taggear	Etiquetar
Overhead	Sobrecoste
Check-in/ check-out	Entrada/salida
Plugin	Aplicación


CONTINUAR


Video de habilidades



1. Primero debemos ingresar a la siguiente página web www.github.com.
2. De acuerdo a las actividades sugeridas en el video, debemos crear un directorio dónde realicemos los pasos descritos.
3. Una vez creado, debemos subir un documento como, por ejemplo, un desarrollo o un documento funcional.
4. Para cerrar la actividad, configurará en forma de imágenes el README.

Luego de hacer la actividad, verificá que hayas realizado correctamente las consignas. Descargá el ítem de resolución a continuación:

**Resolución M1.docx.pdf**
76.2 KB



CONTINUAR

Referencias

Alvarado, J. (2020). 4 Herramientas de Administración de Configuraciones que debes conocer. Recuperado de <https://medium.com/bolillodigital/4-herramientas-de-administraci%C3%B3n-de-configuraciones-que-debes-conocer-407dbfa668d2>.

Assembla. (2010). Software Configuration Management (SCM). Recuperado de [https://app.assembla.com/wiki/show/almejoLisp/Software_Configuration_Management_\(SCM\)](https://app.assembla.com/wiki/show/almejoLisp/Software_Configuration_Management_(SCM)).

EcuRed. (s. f.). Control de cambios. Recuperado de https://www.ecured.cu/Control_de_cambios.

Facultad de Ingeniería de la Universidad de la República del Uruguay. (s. f.). Gestión de Configuración y Control de Cambios. Recuperado de <https://www.fing.edu.uy/inco/cursos/ingsoft/pis/proceso/MUM/dat/dimmod/gesconcc.htm>.

IEEE 1028-1997. IEEE Standard for Software Reviews. (1997). The Institute of Electrical and Electronics Engineers. Recuperado de http://profs.etsmtl.ca/claporte/english/enseignement/cmu_sqa/travaux/TP_Reviews/IEEE%201028-2002%20-%20Software%20Reviews.pdf.

[Imagen sin título sobre Ansible (Red Haut)], (s. f.). Recuperado de <https://medium.com/bolillodigital/4-herramientas-de-administraci%C3%B3n-de-configuraciones-que-debes-conocer-407dbfa668d2>.

[Imagen sin título sobre Chef], (s. f.). Recuperado de <https://dev.classmethod.jp/articles/chef-server-install/>.

[Imagen sin título sobre CVS], (s. f.). Recuperado de <https://slideplayer.com/slide/10288986/>.

[Imagen sin título sobre Puppet], (s. f.). Recuperado de <https://www.linux-magazine.com/Online/News/Puppet-Labs-Announces-Puppet-Enterprise>.

Instituto Nacional de Tecnologías de la Comunicación (INTECO). (2008). Guía avanzada de gestión de configuración. Recuperado de <https://1library.co/document/y49j6d0z-guia-avanzada-de-gestion-de-configuracion-Incs.html>.

ISO/IEC 20000. Information technology — Service management. (2011). International Organization for Standardization. Recuperado de <https://www.iso.org/standard/70636.html>.

Micro Focus. (s. f.). ¿Qué es la Gestión de lanzamientos? Recuperado de <https://www.microfocus.com/es-es/what-is/release-management>.

Minmer Global. (s. f.). Gestión de auditoría y documentación de procesos logísticos. Recuperado de <https://www.minmerglobal.com/gestiondeauditoria>.

Rundeck y Jenkins. (s. f.). Rundeck and Jenkins are a great duo. Recuperado de <https://www.rundeck.com/jenkins>.

Scarcella, N., Passerini, N., Lombardi, C., Fernandes, J., Fernández, C., De Haro, P., Álvarez, A. (s. f.). Práctica del Desarrollo del Software. Recuperado de <https://sites.google.com/site/practicadesarrollosft/temario/scm>.

CONTINUAR
