

El flujo de la entrega de *software*

DevOps e Integración Continua.
Soluciones prácticas avanzadas
en desarrollo de *Software*



tech



CONTENIDO

1. Objetivos


 2. Identificando actores y artefactos

 3. Diseño del flujo de entrega de *software*

 4. Requisitos entre etapas

 5. Resumen

 6. Bibliografía

- 

OBJETIVOS

- Conocer las bases para diseñar un flujo de entrega de *software*.
- Identificar actores y artefactos del flujo de entrega de *software*.
- Comprender las necesidades y requisitos necesarios para definir etapas del flujo de entrega de *software*.
- Conocer conceptos de la filosofía DevOps orientados a la calidad del *software*.

IDENTIFICANDO ACTORES Y ARTEFACTOS

En este apartado se presentan los diferentes componentes principales que intervienen en el ciclo de entrega de *software* y los productores de los mismos. Es necesario conocerlos con antelación para entender las necesidades de negocio del producto y así poder diseñar un buen flujo de desarrollo de *software* que integre no sólo la generación de los artefactos finales, sino que también sea responsable de asegurarse de un determinado nivel de calidad.

Para ello, es necesario involucrar a todos los equipos intervinientes en cada etapa y fijar una serie de requisitos entre ellas [1]. Se detallará en siguientes apartados:

- Se refiere a actores a cualquier parte, ya sea automática o individuo que influya en la generación de un determinado objeto intermedio o final.
- Se refiere a artefactos de cualquier producto de un proceso automático o manual, de forma que pueda ser considerado una unidad independiente.

En otros términos, más sencillos, se entiende como artefacto la joya de una sortija, la cual sufre un determinado proceso para llegar a tener el nivel de pulido y pureza adecuado, eso es un artefacto, por otro lado, el anillo, compuesto de un determinado metal, al cual se le ha moldeado y pulido será otro artefacto, generado mediante un proceso independiente.

Ambos, joya y anillo metálico son artefactos independientes, generados en procesos independientes, y ambos, formarán una sortija única en un ensamblado futuro.

Para poder identificar estos componentes y sus productores, será necesario escudriñar el proceso existente, aunque esta tarea puede resultar especialmente complicada debido a las particularidades que actualmente se pueden encontrar en la mayoría de las compañías como [2]:

- La existencia de diferentes subsistemas, por lo tanto, es una complejidad agregada.
- Equipos de desarrollo distribuidos geográficamente.
- Falta de estandarización y de procesos comunes.

DISEÑO DEL FLUJO DE ENTREGA DE SOFTWARE

Obtener la información necesaria para diseñar un flujo de entrega de *software*, como se ha dicho, puede resultar complicado. Pero una vez obtenida, se podrá elaborar un flujo de entrega personalizado y adaptado para el caso particular, pero siempre se tenderá a usar estándares de la industria para así facilitar la comprensión y mantenimiento de la solución que se implemente.

El flujo de entrega de *software* ha de funcionar como una factoría de *software*, esta factoría deberá generar siempre el mismo resultado para cualquiera de sus etapas o en conjunto para la misma entrada.

Esta factoría estará influida por diferentes personas, equipos y roles, como se puede apreciar en la (figura 1).

Los requisitos de negocio siempre serán el comienzo de la planificación, independientemente de que la compañía trabaje con un *framework* de calidad o no (figura 1 y 2).

Tradicionalmente, la cultura DevOps, integra esta tarea en el proceso, la consecución de diversas etapas que se repiten de manera iterativa para conseguir artefactos finales entregables [3].

En este proceso los elementos o etapas primordiales son las siguientes:

- Planificación.
- Codificación o Desarrollo.
- Integración Continua.
- Despliegue (Continuo preferiblemente).
- Monitorización.

Para conseguir un proceso estable e idempotente, se apoya sobre una serie de herramientas. El mercado actualmente ofrece infinidad de ellas, con diversos modelos de licenciamiento, pero a lo largo de este módulo se hablará sobre las herramientas más extendidas actualmente en la comunidad.

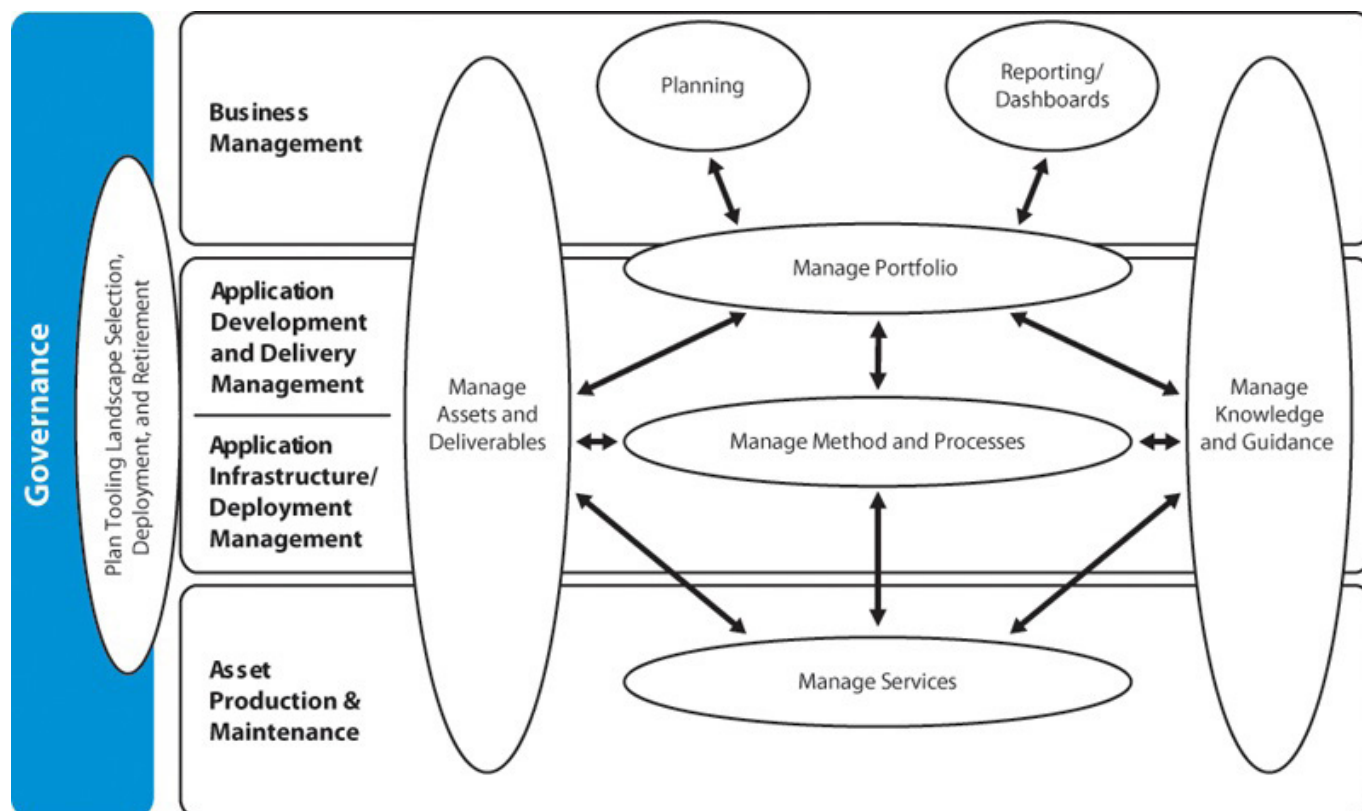


Figura 1. Flujo de entrega de un Governance.

The DevOps culture

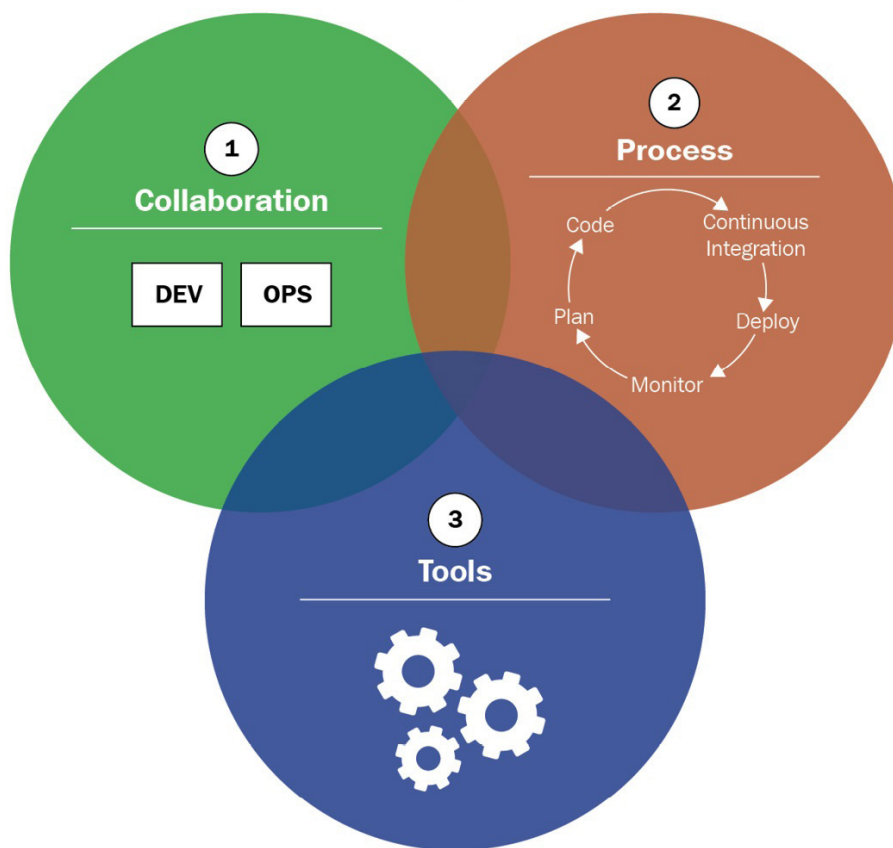


Figura 2. The devOps culture

Para conseguir un proceso estable e idempotente, se apoya sobre una serie de herramientas. El mercado actualmente ofrece infinidad de ellas, con diversos modelos de licenciamiento, pero a lo largo de este módulo se hablará sobre las herramientas más extendidas actualmente en la comunidad.

El flujo de entrega de *software* debe incluir comprobaciones sobre la calidad del artefacto generado. Usualmente se usan, y se recomienda encarecidamente, test unitarios para determinar un umbral de calidad apropiado.

No obstante, también se emplean test de integración, de rendimiento y de sistema, además debe haber presencia de todos ellos para asegurar la calidad del producto, pero siempre teniendo en cuenta la pirámide de test de Cohn (figura 3).

Tener en consideración la pirámide de test o no influirá notablemente en la complejidad del flujo de desarrollo de *software* a diseñar, de su coste y principalmente en el tiempo de ejecución necesario de este proceso.

El tiempo de ejecución completa del proceso de entrega de *software* ha de ser lo más reducido posible sin repercutir en la calidad final. Gracias a esto se ofrecerá la posibilidad de adaptarse rápidamente a los cambios solicitados.

Esto es particularmente importante en grandes equipos, donde diferentes equipos necesitan de los artefactos generados por una etapa anterior en el proceso de entrega de *software*.

Las diferentes etapas pueden ser manuales o automáticas, es bastante común observar procesos híbridos, sobre todo cuando se están comenzando a implantar las buenas prácticas de DevOps y se está modelando el proceso de entrega de *software*. Pero hay que tener en cuenta que esto solo debe ser un paso intermedio hasta conseguir un proceso completamente automático y que no dependa en absoluto de ninguna persona o acción externa al mismo.

Esto garantizará la calidad del proceso en sí ya que se podrá asegurar que, sin impactos de infraestructura, el resultado siempre será el mismo.

El principal requisito para comenzar con el diseño del ciclo de vida de *software* consiste en disponer del código del producto almacenado y versionado en cualquier gestor de versiones disponible, siendo actualmente Git el más utilizado.

El proveedor de este servicio puede ser on-Premise o gestionado, pero se tendrá en cuenta que normalmente los repositorios gestionados como GitHub, GitLab o BitBucket suelen ofrecer además soluciones propias para el desarrollo de Pipelines, el término utilizado para definir procesos de desarrollo de *software*.

Esto puede facilitar en gran medida la tarea, pero suelen tener asociado una fijación al proveedor en el que se desarrolle.

A partir del sistema de control de versiones, se configuran una serie de lanzadores automáticos que se ejecutarán con cada cambio que se efectúe en el código objetivo [3] (figura 4).

El proceso de entrega de *software* se suele escribir como código, dependiendo de las herramientas utilizadas para ello, pudiendo ser puramente scripts de sistema operativo, o pipelines de una herramienta en concreto.

Se debe tener cuidado con la implementación de cada etapa ya que deben ser robustas y tolerantes a posibles fallos de infraestructura que puedan ocurrir.

REQUISITOS ENTRE ETAPAS

Usualmente se especifican una serie de requisitos que se deben cumplir al finalizar una etapa para poder continuar con la siguiente. Es importante definirlas con claridad y deben estar basadas en una opinión unificada entre producto y desarrollo ya que determinará la calidad final del producto en cada iteración.

Prácticas muy extendidas en calidad de *software* y en DevOps son las siguientes:

- El código sigue las reglas de estilo definidas.
- Compilación exitosa sin errores graves.
- Los test unitarios se ejecutan sin fallos.
- El artefacto generado se puede desplegar sin problemas.
- El análisis de vulnerabilidades no arroja ninguna advertencia de seguridad.

En otros capítulos se verá en profundidad como implementar estos requisitos entre las etapas dentro del proceso de desarrollo de *software*, pero a modo de anticipo, se suelen denominar Quality Gates.

Estos requisitos deben ser de obligado cumplimiento, aunque en la práctica una selección de personas tiene permisos extendidos para poder saltarse estas comprobaciones entre etapas a demanda.

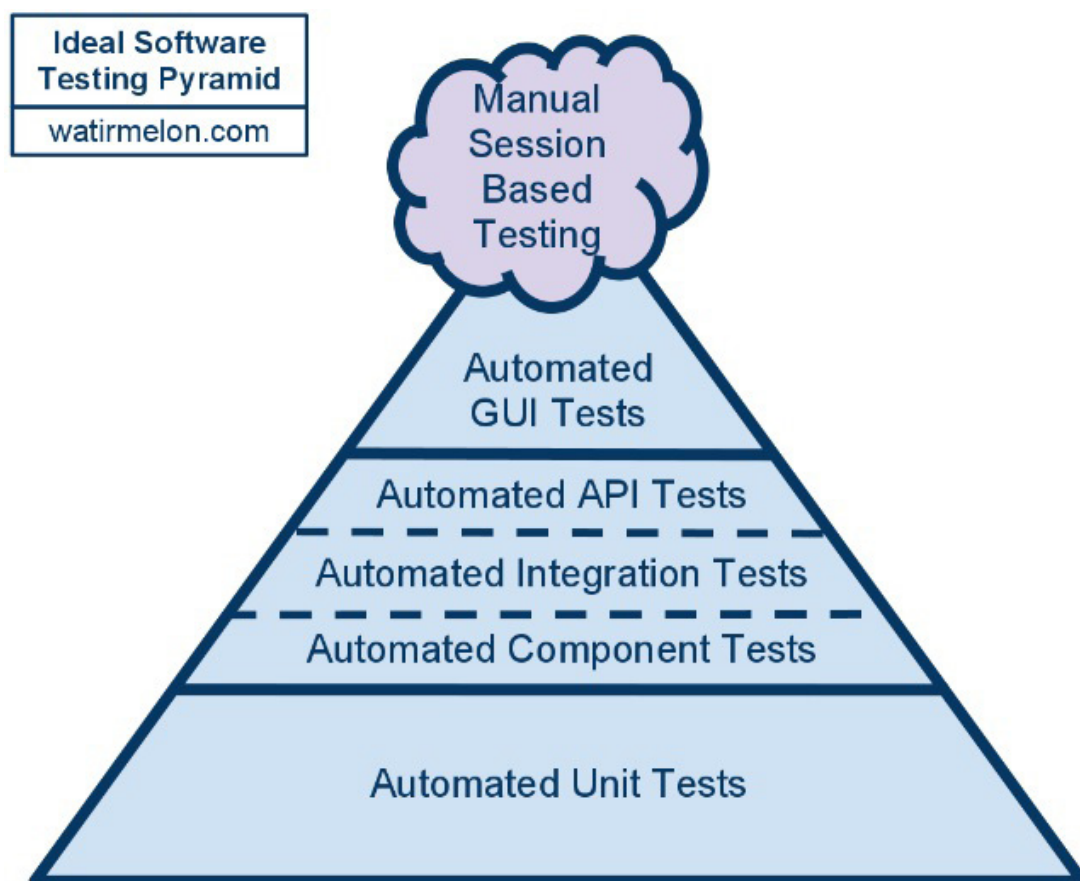


Figura 3.

The continuous integration (CI) pipeline

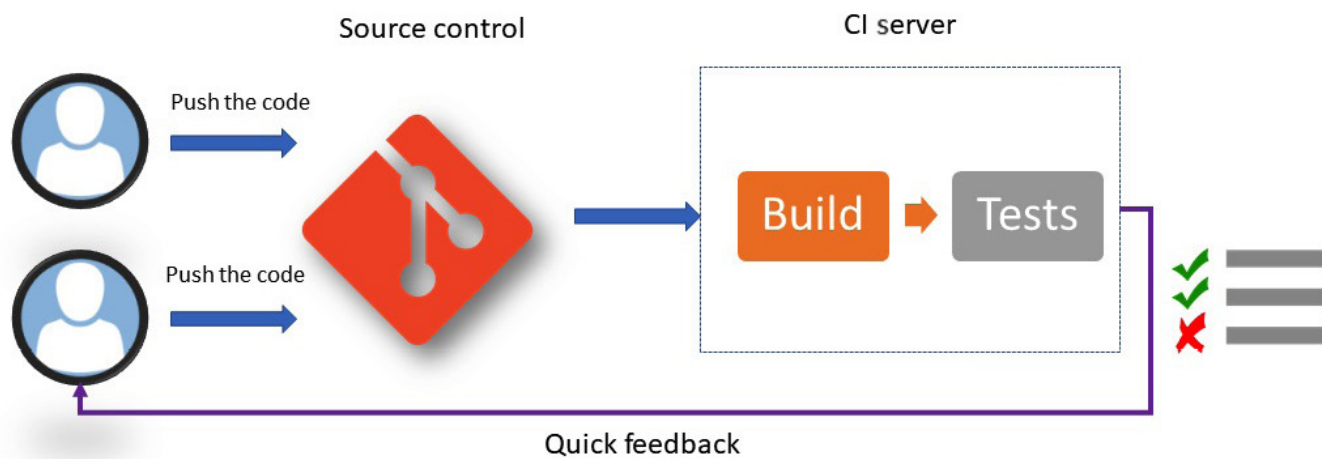


Figura 4.

Saltarse las restricciones de calidad no es una práctica recomendada y no debe usarse a la ligera. Puede llegar a ser útil para entregar cambios muy específicos bajo determinadas circunstancias, pero no debe hacerse como regla general.

RESUMEN

Este tema tiene como principal objetivo presentar los requisitos básicos que deben cumplirse a la hora de diseñar un flujo de entrega de *software*, centrándose en todos los aspectos del desarrollo de *software*, incluyendo a producto y a desarrollo, pero siempre centrándose en la calidad final del producto.

Para ello, se apoyará en la automatización de procesos y la definición de etapas, aunque éstas pueden no ser automáticas al principio.

Se explica la necesidad de comprender la complejidad del producto propio y colaborar con los diferentes equipos para poder establecer un flujo de desarrollo de *software* adecuado.

También se presentan los requisitos entre las diferentes etapas del flujo de entrega de *software* y cómo se suelen emplear en la industria del desarrollo de *software*.

BIBLIOGRAFÍA

- [1] S. W. Ambler y M. Lines, *Disciplined Agile Delivery: A Practitioner's Guide to Agile Software Delivery in the Enterprise*. Upper Saddle River, N.J: IBM Press, 2012.
- [2] A. W. Brown, *Enterprise software delivery : bringing agility and efficiency to the global software supply chain*, 1st edition. 2013.
- [3] M. Krief, *Learning DevOps: A Comprehensive Guide to Accelerating DevOps Culture Adoption with Terraform, Azure DevOps, Kubernetes, and Jenkins.*, 2nd ed. Birmingham: Packt Publishing, Limited, 2022.