Unidad 8
Integración, pruebas y despliegue de software

Guadalupe Ibargüengoitia
Hanna Oktaba

Entrada a esta unidad

Condiciones

- Construcción y pruebas unitarias de componentes del software terminados

Productos de trabajo

- Código documentado, probado, revisado y resguardado en las sub-ramas de cada desarrollador del repositorio compartido
- *Tablero* de la iteración actualizado

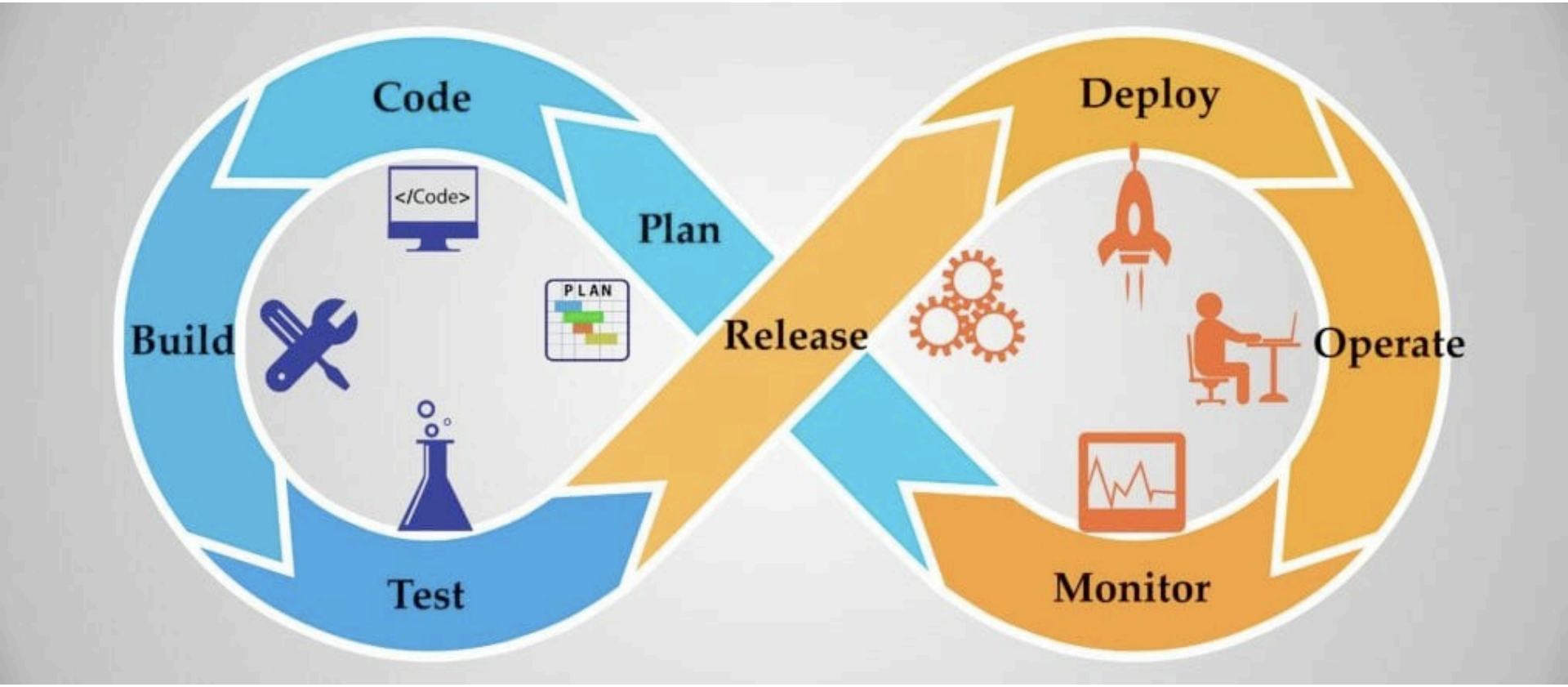
Objetivos

- Integrar componentes, compilar y probarlos de manera automatizada, corregir defectos hasta eliminarlos.

- Generar el código binario de la nueva versión y desplegarlo automáticamente en un servidor.

DEFINICIONES DE CONCEPTOS

DevOps


Code
Plan
Build
Test
Release
Deploy
Operate
Monitor
</Code>
PLAN

Integración de software

- El objetivo de la integración del software es comprobar que los componentes construidos en la iteración funcionan bien juntos y también, cuándo se incorporan al software de iteraciones anteriores.

Integración continua de software
Continuos Integration (CI)

- En el desarrollo de software iterativo, la integración de software es continua conforme se van generando los componentes.
- El orden de la integración de los componentes se basa principalmente en la arquitectura.
- Lo que implica que conforme se van obteniendo componentes probados de cada nivel de abstracción, se van integrando. (SWEBOK 2014)

Actividades para la integración del software

- Preparar la integración del software
- Realizar la integración y pruebas del software

Preparar la integración del software

- Para efectuar la integración de la nueva versión del software se incluyen los componentes construidos y probados individualmente en la iteración.
- Se establecer una estrategia de integración: en qué orden se van incorporando el códigos de los componentes en un solo código.
- Asegurarse que estén disponibles en la sub-rama correspondiente los componentes en el repositorio compartido para su integración.

Realizar la integración y pruebas del software

- Siguiendo la estrategia establecida por el equipo, se van incorporando uno por uno los componentes al código previamente integrado del software.
- La nueva versión del código se compila, se corrigen los defectos de compilación.
- Se aplican todas las pruebas unitarias acumuladas, incluyendo las del componente nuevo, al código integrado y si sea necesario, se corrigen los defectos hasta eliminarlos.

Evaluación de calidad de software

- Pruebas de calidad estática del código (para requerimiento de mantenibilidad)
- Pruebas de regresión
- Pruebas para el cumplimiento de los requerimientos funcionales y no funcionales.
- Métricas de calidad de software

Pruebas de calidad estática del código

- Se aplican de manera automática revisando el código fuente:
  - Cumplimiento de estándares de codificación
  - Complejidad ciclomática
  - Documentación del código

Prueba de regresión

- Se vuelven a aplicar las mismas pruebas para asegurarse que el defecto fue eliminado y que su corrección no ha inyectado otros defectos.

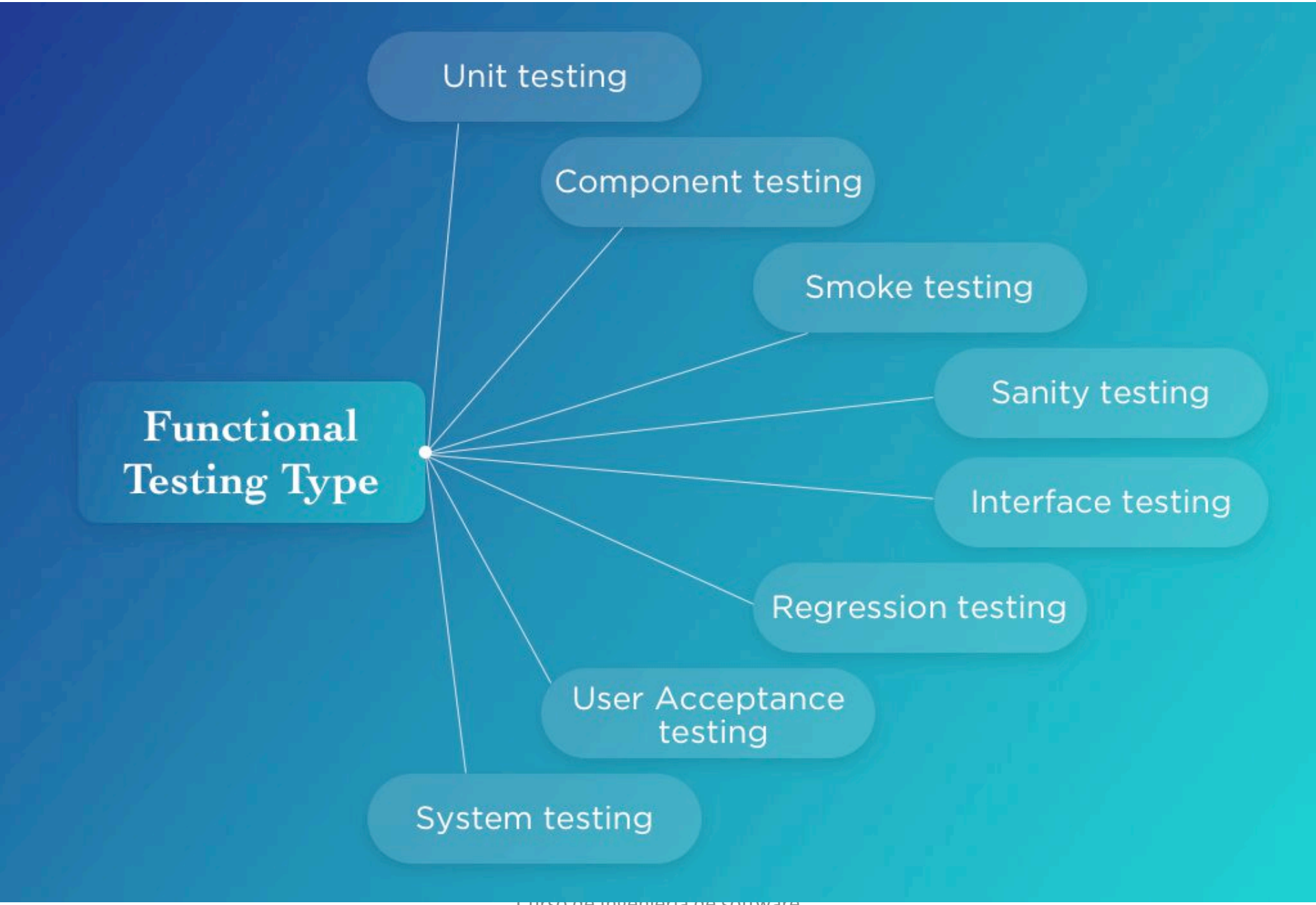
- Las *pruebas de regresión* sirven para detectar defectos al hacer cambios a componentes ya probados o detectar defectos causados por cambios como efectos laterales o por correcciones mal hechas.

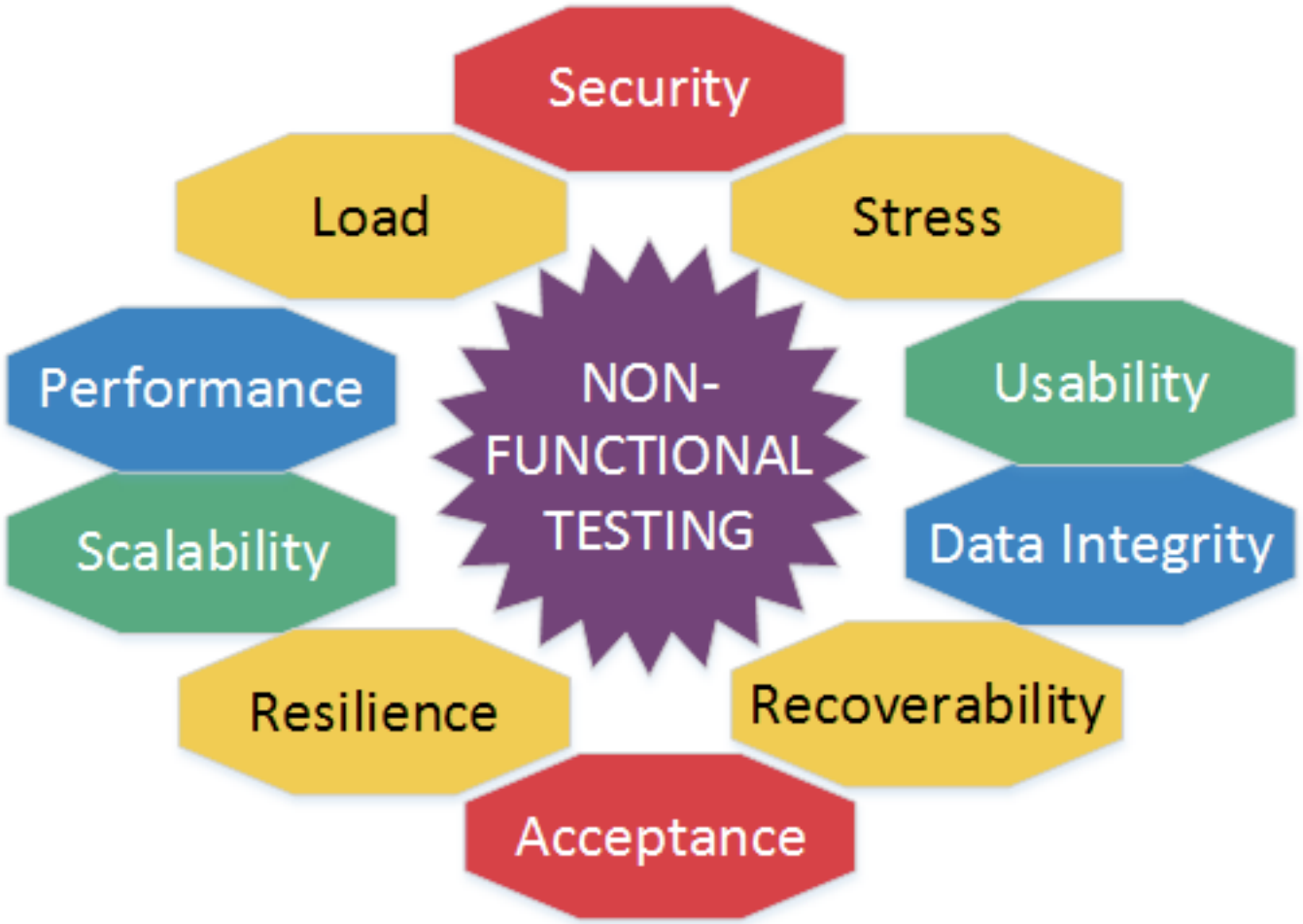
Pruebas para el cumplimiento de los requerimientos funcionales y no funcionales

- En la prueba del sistema integrado de software, conocida también como prueba del sistema, se trata de comprobar que el sistema integrado de software cumple con todos los requerimientos establecidos tanto los funcionales como los no funcionales.

Aspectos a probar

- **Funcionalidad**. Que cumpla con los requerimientos funcionales especificados.
- **Usabilidad**. Si los usuarios finales lo encuentran útil para apoyarlos en sus actividades y qué tan fácilmente se recupera de errores de usuario.
- **Eficiencia**. La capacidad de respuesta es adecuada.
- **Seguridad**. Cumpla con sus requerimientos de seguridad.
- **Estrés**. Ejecutar el software a su máxima capacidad.
- **Recuperación**. Que tan fácilmente se recupera el software de algún error.


Functional Testing Type
Unit testing
Component testing
Smoke testing
Sanity testing
Interface testing
Regression testing
User Acceptance testing
System testing


Security
Load
Stress
Performance
NON-
FUNCTIONAL
TESTING
Usability
Scalability
Data Integrity
Resilience
Recoverability
Acceptance

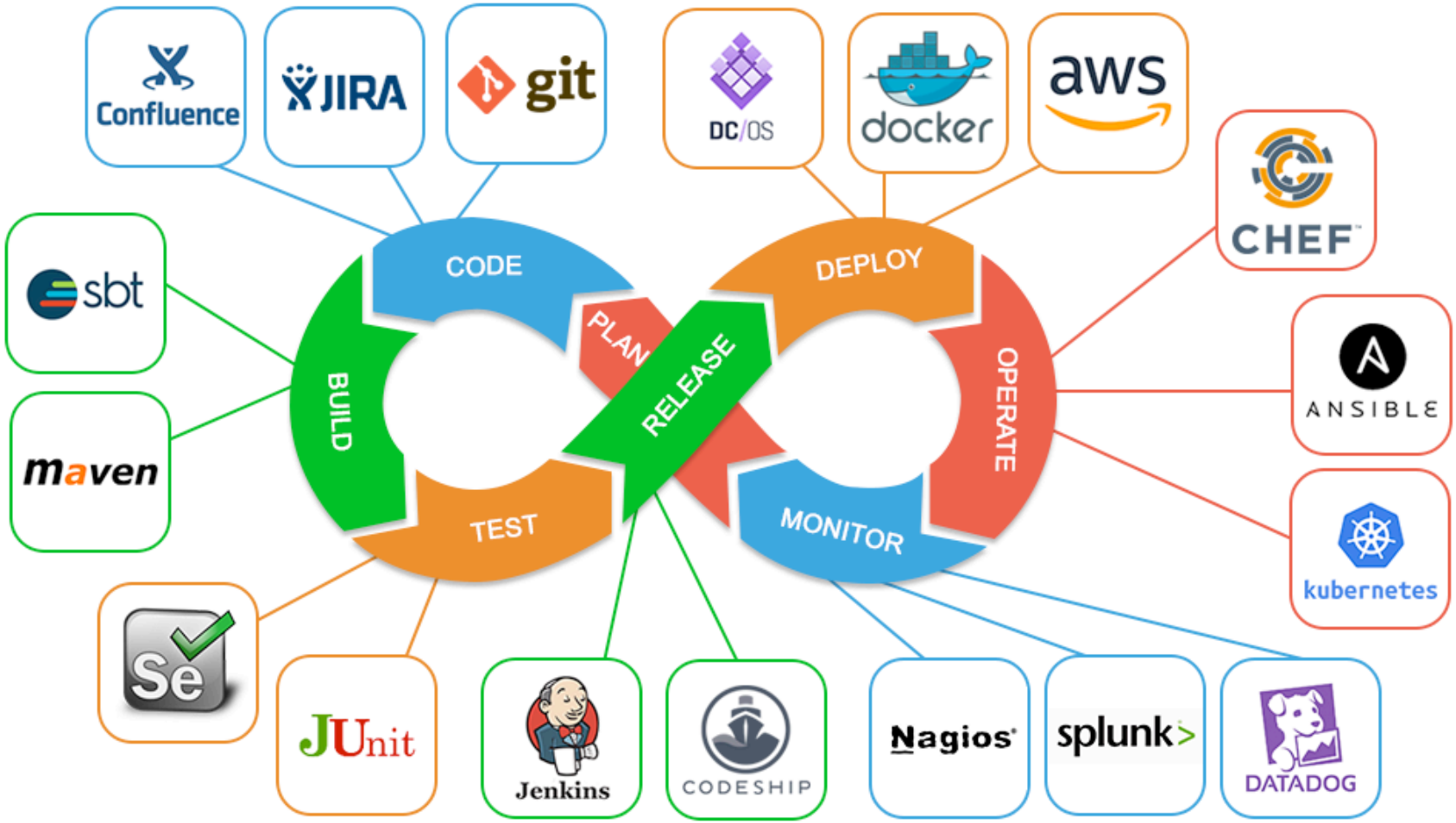
Actividades de pruebas de sistema

- **Preparar las pruebas a realizar.** Decidir qué se probará, en qué orden y preparar el ambiente para efectuar las pruebas.
- **Ejecutar las pruebas.** Se efectúan las pruebas guiadas por **casos de prueba** y se identifican los defectos.
- **Corregir los defectos**. Se asigna al responsable de corregir los defectos y se asegura que se corrijan.
- **Verificar la corrección de defectos**. Se realizan las **pruebas de regresión**.

Métricas de calidad de software

- Medir el número de defectos encontrados y corregidos ayudan a entender la calidad del software en términos de la densidad de defectos por medida de tamaño.

- Medir el número de defectos fugados al ambiente productivo (de operación) por medida de tamaño.


Confluence
JIRA
git
DC/OS
docker
aws
CHEF
sbt
maven
ANSIBLE
kubernetes
Se
JUnit
Jenkins
CODESHIP
Nagios
splunk>
DATADOG
CODE
PLAN
BUILD
TEST
RELEASE
DEPLOY
OPERATE
MONITOR

¿Qué hemos aprendido?

- ¿Cuáles son las fases de la integración de software?
- ¿Qué es una prueba de regresión y cuál es su objetivo?
- ¿Qué se hace para cuidar la calidad de software antes del despliegue?
- ¿Cuáles son las métricas de de calidad de software?

PRACTICA PD4 INTEGRACIÓN, PRUEBAS Y DESPLIEGUE DE SOFTWARE

PD4 Integración, pruebas y despliegue de software

Objetivos

Integrar componentes construidos, probar el código en todas las etapas de la integración, evaluar su calidad, generar el código binario y realizar el despliegue en un servidor.

PD4 Integración, pruebas y despliegue de software

Condiciones

- Construcción y pruebas unitarias de componentes del software terminados

Productos de trabajo

- Código documentado, probado, revisado y resguardado en las sub-ramas de cada desarrollador del repositorio compartido
- *Tablero* de la iteración actualizado

PD4 Integración, pruebas y despliegue de software

Actividades

1. Se integran componentes y se prueba código integrado.
2. Se evalúa la calidad del código.
3. Se versiona el código binario
4. Se realiza el despliegue del código.

TÉCNICAS PARA REALIZAR LAS ACTIVIDADES DE PD4

A1. Se integran componentes y se prueba código integrado

1. El equipo define la estrategia de integración: primero

se integra el código, que pueda ser verificado en su despliegue, y así se continua hasta integrar todo el microservicio.

2. El equipo reparte las tareas de integración equitativamente y lo refleja con tarjetas en el tablero.

3. El autor original del componente dispara en GitLab el “Merge" hacia “Develop" de la sub-rama validada para integrar su código con el ya integrado, según la estrategia. Incluyendo: número de versión, rutas de almacenamiento donde se van a guardar recursos en el sistema de archivos, parámetros de conexión a la BD en la nube, parámetros de conexión al servicio de correos electrónicos, IP del servidor de despliegue.

4. GitLab de manera automática le notifica a Jenkins que la rama de “Develop" tiene nuevo código.

A1. Se integran componentes y se prueba código integrado

4. Jenkins recupera el código de la rama “Develop" y:

4.1. Compila el código

4.2 Si se produce un defecto, envía notificación electrónica a todos los miembros del equipo para que estos se enfoquen en eliminar el defecto repitiendo las actividades correspondientes del ciclo de construcción.

4.3. Cuando pasa la compilación, ejecuta TODAS las pruebas unitarias anteriores e incluye las nuevas asociadas al código nuevo.

4.4. Si se produce un defecto, envía notificación electrónica a todos los miembros del equipo para que estos se enfoquen en eliminar el defecto repitiendo las actividades correspondientes del ciclo de construcción.

4.5 En ambos casos 4.2 y 4.4 Jenkins vuelve a compilar y probar el código corregido hasta no encontrar defectos.

A2. Se evalúa la calidad del código

5. Cuando ya no se detectan errores en pruebas Jenkins evalúa la calidad del código generando estadísticas sobre:

- 5.1 Apego a estándares de codificación
- 5.2 Complejidad ciclomática
- 5.3 Cobertura de pruebas unitarias
- 5.4 Documentación Javadoc
- 5.5 Documentación de doble barra //
- 5.6 % de log (error, info, warning y debug)

6. Se presentan las estadísticas de calidad en el tablero de control de calidad para su consulta.

A3. Se versiona el código binario

6. Jenkins genera el binario de despliegue, con la versión definida al hacer “Merge”, y lo almacena en Nexus en el espacio para guardar las bibliotecas binarias con la finalidad de su uso para el despliegue, reutilización y/o recuperación de las versiones anteriores.

El estándar de asignar versiones:
Major.Minor.Patch

https://semver.org/lang/es/

A4. Se realiza el despliegue del código

8. Jenkins sustituye el binario de despliegue anterior por el binario nuevo en el contenedor asociado al servicio en el servidor de despliegue,

cuyo IP fue indicado en los parámetros del "Merge".

Con esto inicia la ejecución del microservicio.

PD3 Construcción de software

Resultados

Condiciones

- Integración, pruebas y despliegue del software terminado

Productos de trabajo

- Código integrado, documentado, probado, evaluado y desplegado en un servidor
- *Tablero* de la iteración actualizado

Resultados de esta unidad

- **Condiciones**
  - Integración, pruebas y despliegue del software terminado

- **Productos de trabajo**
  - Código integrado, documentado, probado, evaluado y desplegado en un servidor
  - *Tablero* de la iteración actualizado

Bibliografía

Binder R.V. (2000). *Testing Object-Oriented Systems. Models, patterns and Tools.* Addison Wesley.

Dusrin E., T. G. (2009). *Implementing Automated Software Testing.* Addison Wesley.

DevOps https://www.redhat.com/es/topics/devops#?

Ruiz B. (2013). Marcando la pauta para las pruebas ágiles. *Software Gurú, no. 38*, 18-20.

SWEBOK 3.0. (2014). *Guide to the Software Engineering Body of Knowledge v3.0.* IEEE Computer Society.