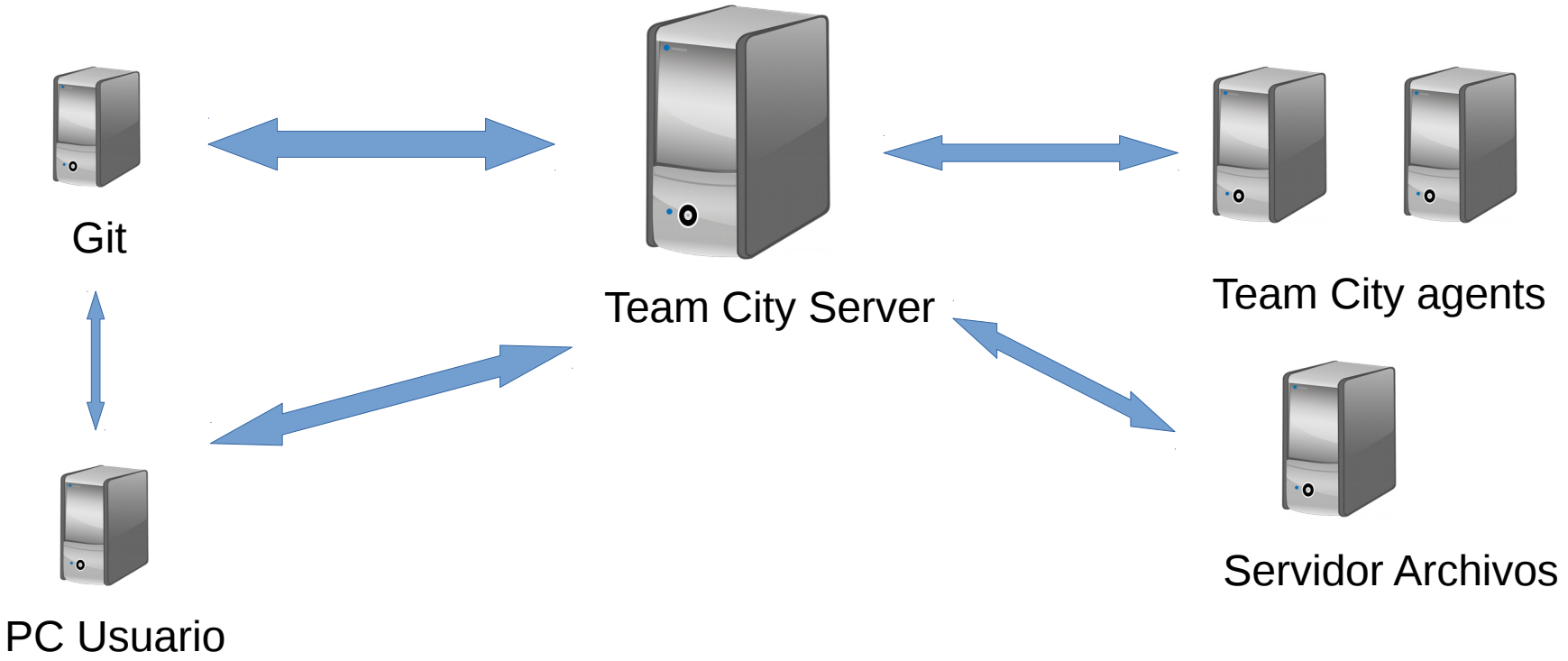
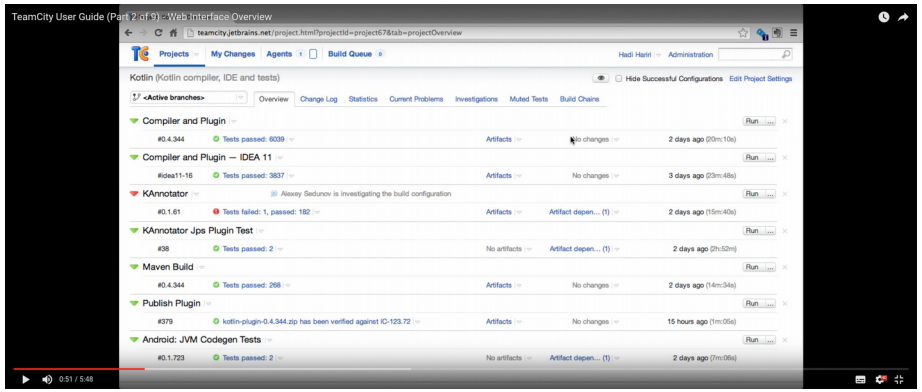
Supuesto 1

Elementos Sistema Integración Continua



Configuración de los proyectos en Teamcity



- Para cada tipo de build de un proyecto, se creará un subproyecto en Teamcity. Por ejemplo, para el supuesto 1, tendremos al menos el subproyecto que genera el binario con información de debug, y el subproyecto con la versión release
- En cuanto al tratamiento del repositorio de código, basado en gitflow. Branch master contiene el código de producción, develop el de integración, “fixes” y “features” los desarrollos relativos a arreglar un defecto o a desarrollar una nueva funcionalidad.

1.- Inicio proceso build

- El branch “master” contiene el código de producción, el branch “development” el de integración, los branches de trabajo son merges del branch de “development”
- El commit se realiza en un branch “feature xxx” o “fix defect yyy”. Branch que es regularmente sincronizado por el / los desarrolladores con el branch “development” por medio de un “merge”.
- Seguiremos el proceso de construcción a partir de la rama principal “development”
- Proceso de construcción se inicia con un “merge request” aceptado de un desarrollador. El proceso de construcción está configurado para ver si hay cambios en el repositorio de forma regular, por ejemplo, cada 20 minutos, o bine puede configurarse para lanzarse siempre a una hora determinada.

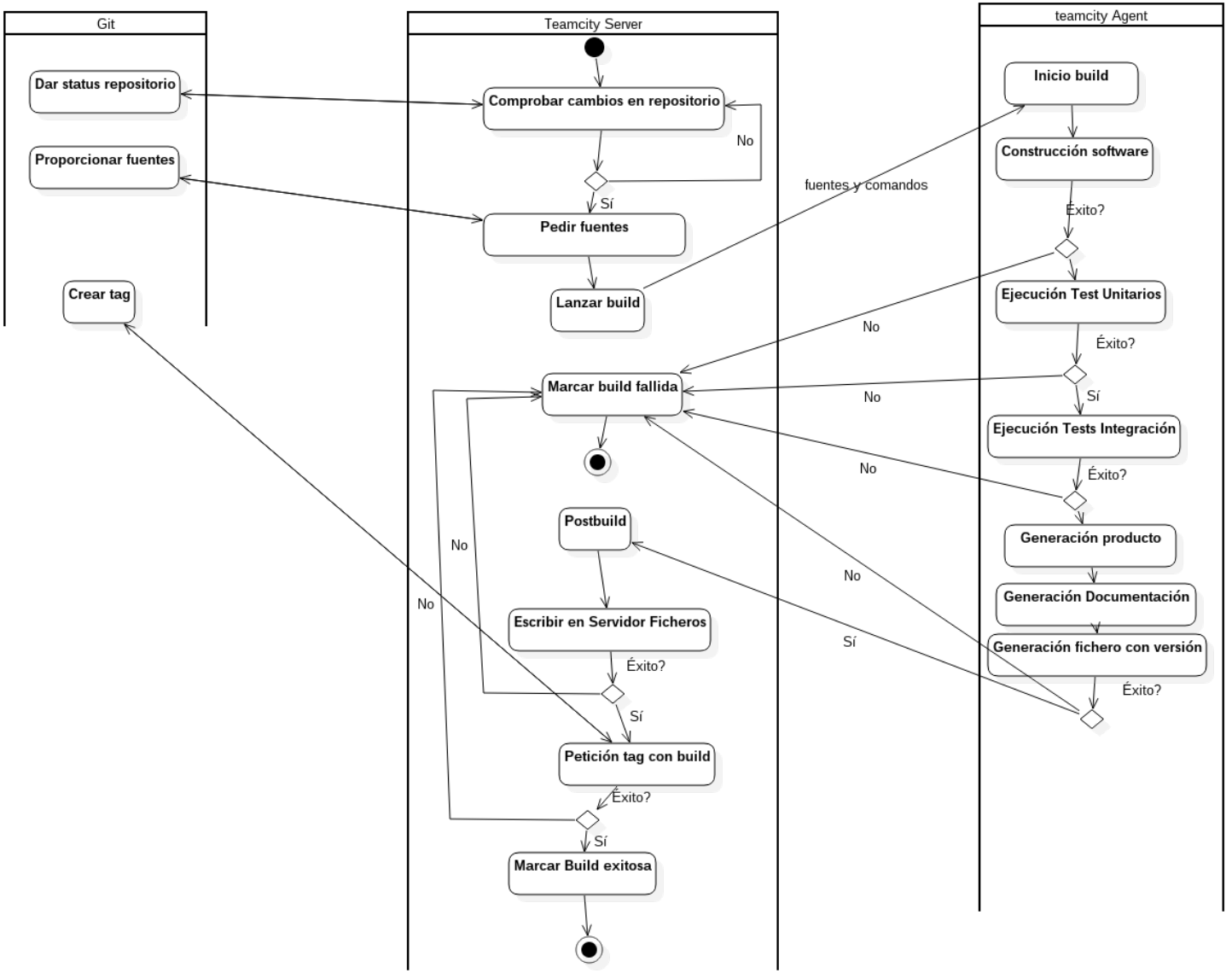
2 Build y ejecución tests

- Teamcity Server lanza en el agente adecuado (Agent requirements) el proceso de build. El agente contiene todo el entorno necesario para el proceso y la ejecución de los tests unitarios
- Si el proceso de build y ejecución de los tests unitarios tienen éxito, se lanzan los tests de integración
- Si se diera el caso de el agente no estuviera preparado para la ejecución de los tests de integración, por medio de “Artifact dependency”, se lanzaría automáticamente una build en el agente configurado para los tests de integración.
- El grado de cobertura de los tests queda reflejado en el Teamcity Server

3 Postbuild

- Si la construcción y los tests se han pasado con éxito, se genera la documentación y el binario comprimido mediante un build step “Command line”.
- Por medio del plugin “Deployer” se lanza el proceso de escritura en el servidor de ficheros.
- Finalmente, por medio de “VCS Labeling” se escribe un tag en el servidor git que nos de la trazabilidad del build.

Diagrama 1.1


Git
Dar status repositorio
Proporcionar fuentes
Crear tag
Teamcity Server
Comprobar cambios en repositorio
No
Sí
Pedir fuentes
Lanzar build
fuentes y comandos
Marcar build fallida
Postbuild
Escribir en Servidor Ficheros
Éxito?
Sí
No
Petición tag con build
Éxito?
Sí
Marcar Build exitosa
teamcity Agent
Inicio build
Construcción software
Éxito?
Ejecución Test Unitarios
No
Éxito?
Sí
Ejecución Tests Integración
No
Éxito?
Generación producto
No
Generación Documentación
Sí
Generación fichero con versión
Éxito?

Supuesto 1.2

- La historificación ya está lograda en cuanto a los fuentes por medio del tag en git
- En cuanto al servidor de ficheros, cada build puede guardarse en un subdirectorio distinto, con nombre con formato parecido al fichero “aa.bb.cc.dd”.
- Podría usarse un servidor Nexus alimentado en el proceso de PostBuild con el paquete del build (comprimido binario y documentación), para mayor facilidad de acceso y mantenimiento que un simple servidor de ficheros

Supuesto 1.3

- Para la detección de memory leaks habría que añadir un subproyecto en el proyecto de teamcity que generara builds con información de debug, y configurarlo para que pasara una herramienta estilo valgrind.
- Añadiendo el plugin Sonarqube al teamcity, configurando el build para que alimentara al sonarqube, y escribiendo la salida de los tests en un formato compatible con sonarqube (que acepta varios estándares del mercado), conseguiríamos todos los tipos de análisis que proporciona sonarquebe, que incluyen los citados en 1.3