TEMA 100. LA MIGRACIÓN DE APLICACIONES EN EL MARCO DE PROCESOS DE AJUSTE DIMENSIONAL Y POR OBSOLESCENCIA TÉCNICA. GESTIÓN DE LA CONFIGURACIÓN Y DE VERSIONES. GESTIÓN DE ENTORNOS.

Actualizado a 09/10/2020

1. LA MIGRACIÓN DE APLICACIONES EN PROCESOS DE AJUSTE DIMENSIONAL Y POR OBSOLESCENCIA

1.1.AJUSTE DIMENSIONAL

El ajuste dimensional, consiste en aligerar la carga de trabajo que soporta un gran ordenador (*mainframe*) trasladándola a otro tipo de ordenadores de menor coste.

1.2.DOWNSIZING, UPSIZING Y RIGHTSIZING

DOWNSIZING

Proceso de mover una aplicación o parte de ella de un *mainframe* a una plataforma menos costosa.

Ventajas: No hay dependencia de un solo sistema (*mainframe*). Mayor portabilidad e interoperabilidad, reducción de costes, facilidad para optimizar y balancear *hardware* y *software*, facilidad para implementar alta disponibilidad, mayor flexibilidad, reducción del espacio físico requerido.

Inconvenientes: posibles carencias funcionales si la migración no se realiza correctamente, mayor heterogeneidad de equipamiento, mayor necesidad de monitorización y actualización de servidores, al estar distribuidos.

UPSIZING

Integración en entornos de red de aplicaciones y ordenadores que estaban aislados, de forma que se permita la compartición de datos. Ej integración de BBDD aisladas en un servidor único de BD.

RIGHTSIZING

Elección de la plataforma más apropiada para las actividades informáticas de una organización (*mainframe*, arquitectura cliente-servidor, etc).

1.3.OBSOLESCENCIA DE APLICACIONES Y MIGRACIÓN

Las aplicaciones obsoletas o con riesgo de obsolescencia suelen denominarse aplicaciones *legacy*, legadas, o heredadas.

Causas de obsolescencia que pueden llevar a la necesidad de una migración: obsolescencia hardware, final de soporte del fabricante, código fuente no adaptable, complejidad del código, por reducción de costes al pasar de aplicación propietaria a software libre.

1.4.TIPOS DE MIGRACIÓN DE APLICACIONES

Modalidades de migración de aplicaciones:

- **Refronting o refacing**: modificar únicamente la interfaz gráfica de una aplicación sin necesidad de reescribir la aplicación completa.
- **Replacement** (sustitución): sustituye la aplicación completas por una o varias aplicaciones.

- **Rehosting** (re-alojamiento): mover la aplicación desde un entorno *hardware legacy* a un entorno más moderno, sin modificar el código de la aplicación o su arquitectura.
- **Rearchitecting** (re-arquitectura o reescritura): desarrollar de nuevo la aplicación utilizando nuevos paradigmas de programación, (pj.: desarrollo web, orientación a objetos, etc).
- **Interoperation** (interoperabilidad) o wrapping: encapsular la aplicación o algunos de sus componentes para que sean utilizados por otras aplicaciones o infraestructura con tecnología más moderna.
- **Retirada**.
- **Migración a la nube**: engloba en mayor o menor medida a varias de las anteriores, pj.: un *re-hosting* a un entorno *cloud*, o una sustitución por una aplicación SaaS.

2. GESTIÓN DE LA CONFIGURACIÓN

Proceso de la ingeniería del software que tiene como objetivo la calidad de los productos, mediante el control de los cambios sobre los activos *sw* y *hw*. Para ello, mantiene una imagen detallada de la situación de los activos y de sus relaciones a lo largo del tiempo. Pueden definirse dos aproximaciones a la gestión de la configuración: la visión orientada al desarrollo del software, cuyo objetivo es controlar el estado del código y sus modificaciones (**Interfaz GC de Métricav3**) y la visión orientada al gobierno de los sistemas de información o a la gestión de los servicios IT(**práctica gestión de la configuración ITIL 4**).

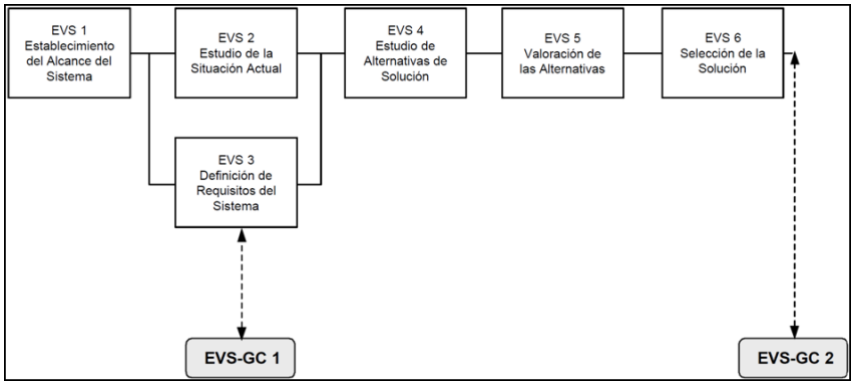
2.2. MÉTRICA V3 – INTERFAZ GC

La Gestión de la Configuración es una de las cuatro interfaces de la Metodología Métrica v3.

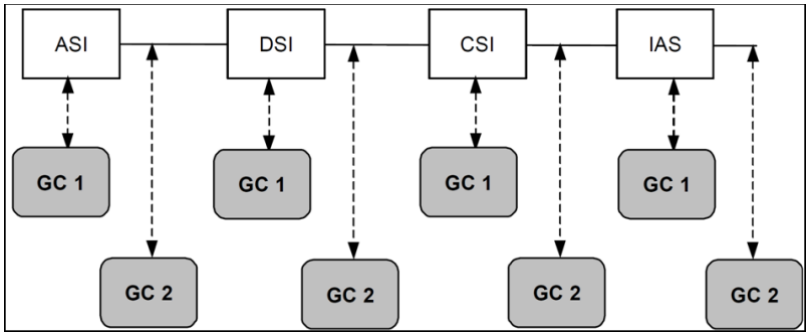
Objetivo de la interfaz: "mantener la integridad de los productos que se obtienen a lo largo del desarrollo de los sistemas de información, garantizando que no se realizan cambios incontrolados y que todos los participantes en el desarrollo del sistema disponen de la versión adecuada de los productos que manejan".

Existen tres escenarios en los que entra en juego la Gestión de la Configuración en Métricav3:

- **Estudio de Viabilidad del Sistema (EVS)**: Tras el proceso EVS se obtiene el Plan de Gestión de la Configuración. Durante la actividad EVS3 – Definición de Requisitos del Sistema, tiene lugar la actividad **EVS-GC1 – Definición de los requisitos de Gestión de la Configuración** (véase Figura 2). En esta actividad se definen los requisitos generales relativos a la gestión de la configuración y los procesos de control que se llevarán a cabo como el control de versiones, control de cambios, etc. La actividad **EVS-GC2 – Establecimiento del plan de Gestión de la Configuración** tiene lugar tras la tarea EVS6 – Selección de la Solución. Durante esta actividad se establece el plan y también se define el entorno tecnológico (herramientas) que se emplearán para llevar a cabo el plan.



- **Análisis, diseño, construcción e implantación y aceptación del sistema de información:** Durante los procesos de ASI, DSI, CSI e IAS el principal cometido de la Gestión de la Configuración es realizar actividades de identificación y registro previstas en el Plan de Gestión de Configuración



La actividad **GC 1 – Identificación y registro de productos** tiene como objeto identificar los productos que se obtienen en los procesos.

La actividad **GC 2 - Identificación y registro del producto global** tiene como finalidad identificar y registrar en el sistema de gestión de la configuración los productos globales.

- **Mantenimiento del Sistema de Información (MSI):** En este caso, se velará por la integridad en el mantenimiento correctivo y evolutivo. La actividad MSI-GC 1 – Registro del cambio en el sistema de gestión de la configuración, registra la nueva versión del producto modificado y la nueva versión del sistema o sistemas impactados por los cambios.

2.1.ITIL 4 – PRÁCTICA GÉSTIÓN DE LA CONFIGURACIÓN

La gestión de la configuración en ITIL 4 es una de las 34 prácticas incluidas en el Sistema de Valor del Servicio, incluida dentro de la categoría “Prácticas de Gestión de Servicios”.

El **propósito de la práctica de gestión de la configuración** del servicio es garantizar que disponemos de información precisa y confiable sobre la configuración de servicios, y de los elementos de configuración (CI o Configuration Item) que los respaldan.

Un **CI** es cualquier componente que requiera administración y sea necesario para proporcionar un servicio de TI: hardware, software, redes, edificios, personas, proveedores y documentación. Los servicios también se tratan como elementos de configuración.

La gestión de la configuración proporciona información sobre los CI que contribuyen a cada servicio y sobre sus relaciones: cómo interactúan, cómo se relacionan y si dependen unos de otros para crear valor para los clientes y usuarios.

Los requisitos de la práctica de gestión de la configuración deben basarse en los objetivos de la organización y en cómo esta gestión puede contribuir a la generación de valor. El valor obtenido a través de la gestión de la configuración es indirecto, pero permite a muchas otras prácticas trabajar de manera eficiente y efectiva. El tipo y cantidad de información registrada para cada tipo de CI debe basarse en el valor de esa información, en el coste de mantenerla y en cómo se utilizará esta información.

Es necesario conocer quién necesitará la información sobre la configuración, cómo se utilizará, cuál es la mejor manera de obtenerla y cómo se mantendrá y actualizará la información. La información de configuración se puede almacenar y publicar en una sola base de datos de gestión configuración (CMDB) para toda la organización, sin embargo, es más frecuente que se encuentre distribuida.

Las herramientas que se utilizan para registrar incidentes, problemas y gestionar los cambios necesitan acceso a los registros de configuración.

Algunos **términos relevantes**:

- **CMS**: Sistema de Gestión de la configuración. Grupo de herramientas, datos e información que se utiliza para apoyar la práctica de gestión de configuración del servicio.
- **CMDB**: Base de datos usada para guardar los registros de configuración a lo largo de su ciclo de vida. Mantiene también las relaciones entre los registros de configuración.
- **Línea base de configuración**: Es la configuración de un producto, servicio o infraestructura que se ha revisado formalmente. Sirve como base para otras actividades como las de desarrollo y planificación.

Los **factores clave** en esta práctica son:

- Asegurar que la organización dispone de la información de configuración relevante sobre sus productos y servicios.
- Asegurar que el coste de proveer la información de configuración se optimiza de forma continua.

2.2.GESTIÓN DE LA CONFIGURACIÓN EN ESTÁNDARES Y METODOLOGÍAS

Algunos estándares y metodologías generales relativos a la gestión de la configuración:

- CMMI – Configuration Management.
- ISO 10007:2017 Quality management - Guidelines for configuration management.
- 828-2012 IEEE Standard for Configuration Management in Systems and Software Engineering.

3. GESTIÓN DE VERSIONES

Existen dos aproximaciones a la gestión de versiones: la orientada al desarrollo y la orientada al gobierno de sistemas o gestión del servicio IT (relacionada con los conceptos ITSM e ITIL).

El término versión puede referirse tanto a versión como a entrega. La gestión de versiones (*version control*) suele asociarse al control de los cambios en el código durante el proceso de desarrollo, mientras que la gestión de entregas (*release management)* suele referirse a la prueba, empaquetado y puesta en producción de código.

3.1.ITIL 4 – PRÁCTICA GESTIÓN DE LIBERACIONES O VERSIONES

La gestión de versiones es una práctica de ITIL 4.

El propósito de esta práctica es hacer los servicios nuevos y modificados adecuados y disponibles para su uso por los usuarios. Hay que tener en cuenta que una versión de un servicio (o elemento de

configuración) puede involucrar cambios en muchos otros componentes y procesos, bien proporcionados por la propia organización o por terceras partes e integrados por la organización proveedora.

Las versiones pueden tener diferentes tamaños o complejidad, es necesario acordar con el cliente una correcta planificación de las mismas, estableciendo tiempos para la misma así como otros aspectos relacionados con la post implantación de la versión que permitan a la organización aprender y mejorar la satisfacción de los clientes. En muchos casos el grueso del trabajo de esta práctica se lleva a cabo antes del despliegue de la versión, momento en el que la nueva funcionalidad se hace disponible.

La práctica de gestión de versiones puede adoptar un enfoque tradicional, donde la gestión de versiones y despliegue se combinan en un único proceso, o bien un enfoque ágil, donde el despliegue se hace en pequeños incrementos de la versión

En un entorno DevOps, la gestión de versiones está integrada con la integración continua y las herramientas para la entrega continua. Toda nueva versión tiene un determinado registro en la herramienta de gestión, enlazada con los elementos de configuración (CI´s) y registros de cambios relacionados con la misma.

3.2.GESTIÓN DE VERSIONES Y RELEASES EN EL DESARROLLO SOFTWARE

La gestión de versiones en el ámbito del desarrollo software se refiere al conjunto de actividades relativas a gestionar, planificar y controlar las modificaciones de código a través de las diferentes etapas y entornos (desarrollo, integración, pre-producción, etc).

TERMINOLOGÍA COMÚN EN LA GESTIÓN DE VERSIONES

- **Repositorio**: servidor de versiones y ficheros que conforman el código fuente de la aplicación.
- **Check-out**: descargar una copia del código fuente a un equipo local desde el servidor.
- **Update**: actualizar la versión local del código con la última versión disponible en el servidor.
- **Commit (o check-in):** convertir cambios temporales en definitivos, consolidando en el servidor de versiones los cambios realizados en local. En caso de conflictos de código, pej. por múltiples modificaciones sobre una misma línea, se deberá resolver, en general, de forma manual.
- **Diff (o Delta):** diferencias entre dos versiones (código añadido, eliminado, modificado).
- **Delta compression:** se almacenan en el repositorio únicamente las diferencias entre versiones, en vez de una copia de cada versión, con el fin de reducir la necesidad de almacenamiento.
- **Branch (rama):** una familia de modificaciones al código fuente independiente de otra, de tal forma que se pueda trabajar de forma aislada en la evolución de una determinada parte del código sin tener que modificar todo el código cada vez que se actualiza. Cuando se crea una nueva rama, se duplica la versión actual del código fuente, y esta se modifica de forma independientemente. Pueden existir múltiples ramas a la vez. Una rama se considerará la rama principal o tronco (trunk).
- **Merge**: fusionar una rama con otra rama (generalmente con la rama principal).
- **Fork (bifurcación):** crear una nueva rama que nunca volverá a ser fusionada con la principal, por ejemplo, para crear un producto con una evolución diferente.
- **Push:** enviar el código a otros servidores tras hacer el commit en local.
- **Pull**: descargar una copia de código de un servidor y sustituir a la local.
- **Fetch:** descargar una copia de código de un servidor al equipo local pero sin sustituir la copia local de código.

INTEGRACIÓN CONTINUA

Aproximación relacionada con Agile o Extreme Programming, en la que se realizan integraciones de código muy frecuentes. Se entiende integración de código como el *check-in* del mismo, compilado si procede y ejecución de casos de prueba. Permite obtener versiones de la aplicación con mejoras incrementales en periodos cortos de tiempo y reducir la incertidumbre propia de integraciones de código. Hace uso de herramientas de automatización de la compilación y pruebas como Jenkins o Hudson, las cuales a su vez pueden integrarse con herramientas de gestión de versiones.

3.3.HERRAMIENTAS PARA GESTIÓN DE CONFIGURACIÓN Y DE VERSIONES

HERRAMIENTAS ORIENTADAS AL GOBIERNO DE LOS SISTEMAS DE INFORMACIÓN (ITIL)

Herramientas orientadas a la gestión de la configuración de los activos TIC de una organización, desde el punto de vista del IT governance o IT Service Management.

Forman parte de *suites* de aplicaciones que incluyen más funcionalidad, como gestión de incidencias, gestión de cambios, etc. Las herramientas más populares en este ámbito son de licencia propietaria, entre las que cabe destacar:

- **BMC Remedy**: Cuenta con funcionalidad completa de CMDB (BMC Atrium) y otros módulos ITSM adicionales.
- **EasyVista Service Manager:** La CMDB forma parte de una suite ITSM.
- **ServiceNow**: Igualmente la CMDB como parte de la suite de módulos ITSM.

HERRAMIENTAS ORIENTADAS AL DESARROLLO

Herramientas orientadas a la gestión de la configuración y versiones en proyectos de desarrollo. Facilitan la gestión del código, coordinación entre equipos de desarrollo, control de cambios y versiones a lo largo del ciclo de vida de desarrollo.

Existen dos aproximaciones en herramientas de gestión de versiones:

- La **gestión centralizada o modelo cliente-servidor**: almacena el código en un servidor central del cual los clientes descargan una copia local (*check-out*), trabajan sobre ella, y la suben de nuevo al servidor (*check-in*).
- El **modelo distribuido:** es más reciente, clona el repositorio en todos los equipos de los desarrolladores (todo el historial de cambios y metadatos asociados). Esto permite aplicar cambios rápidamente en local, y una vez que se han realizado todos, publicarlos al resto de repositorios.

Algunas herramientas populares de gestión de versiones:

- **CVS** (Concurrent Versions System): Escrito en C y con licencia GPL. Herramienta de gestión de versiones en modelo cliente-servidor.
- **SVN** (Apache Subversion): Escrito en C y con licencia Apache. Más moderno que CVS, adopta también un modelo cliente-servidor.
- **IBM Rational Clear Case**: Licencia propietaria, modelo cliente-servidor.
- **Perforce Helix**: Licencia propietaria, modelo cliente-servidor.
- **Git**: Escrito en múltiples lenguajes, licencia GPL y LGPL. Es más reciente que los dos anteriores y goza también de gran popularidad. Es un sistema distribuido.
- **Mercurial**. Escrito en Python y C, licencia GPL. Representa otro ejemplo popular de gestión de versiones distribuida, al igual que Git.

- **GNU Bazaar**: Escrito en Python y C, licencia GPL. Permite modo distribuido y modo cliente-servidor.

4. GESTIÓN DE ENTORNOS

La gestión de entornos da soporte al ciclo de vida de desarrollo del software, proporcionando diferentes entornos de ejecución de la aplicación, adaptados a cada una de las fases del ciclo de vida. Tipos de entornos:

- **Entorno local**: equipo local del desarrollador donde trabaja en el código fuente de la aplicación.
- **Entorno de desarrollo**: entorno de trabajo del desarrollador o equipo de desarrollo. Puede ser igualmente el entorno local. En este entorno se ejecutarían las pruebas unitarias.
- **Entorno de integración**: entorno en el que se realiza el merge de las distintas copias del código en la que trabajan los desarrolladores. Permite realizar pruebas de integración.
- **Entorno de pruebas:** entorno con características más parecidas al de producción. Orientado a la ejecución de pruebas extremo a extremo, pruebas de interfaces, aseguramiento de la calidad, etc.
- **Entorno de pre-producción** (staging): orientado en general a la realización de pruebas por parte de usuarios. Debe ser lo más similar posible al de producción.
- **Entorno de producción**: entorno en el que se ejecuta la aplicación.

Según la organización, y el tipo de desarrollo, se podrá simplificar el número de entornos o hacerlo más complejo.

Actividades típicas de la gestión de entornos:

- Reserva de entornos.
- Limpieza de entornos.
- Coherencia entre entornos: disponer de imagen fiable del entorno de Producción.
- Gestión de la capacidad de los entornos: capacidad de proceso y recursos suficientes.