Guía de desarrollo, Anexo 38

Normativa de versionado

Oficina de Calidad

GERENCIA DE INFORMÁTICA

JOSEFA VALCÁRCEL, 44
28027-MADRID

Índice General

Índice de Ilustraciones y Tablas

1 Introducción

1.1 Objetivo

Este documento describe las directrices a seguir para el versionado de aplicaciones software que deben seguir los proyectos de desarrollo de sistemas de información en la gerencia de informática de la Dirección General de Tráfico (DGT).

1.2 Audiencia

Este documento está dirigido a todas las personas que colaboren en labores relacionadas con la gestión, desarrollo, auditoría, implantación y explotación de los sistemas de información de la gerencia de informática de la Dirección General de Tráfico.

1.3 Glosario

Los términos y acrónimos que se utilizan en este documento y en el resto de documentos de la guía se encuentran recogidos por orden alfabético en el Anexo 30. Glosario con el objetivo de facilitar su lectura y comprensión.

1.4 Resolución de dudas

Para cualquier duda se debe realizar una consulta al Departamento de Calidad a través de la Herramienta de Gestión de servicios TI de la Gerencia de Informática.

2 Normativa de versionado del software en DGT

Con el objetivo de estandarizar el versionado en la DGT se va a seguir la especificación definida por el denominado versionado semántico apoyada en el uso de calificadores.

La documentación de la especificación definida por el denominado versionado semántico, puede encontrarse en la web http://semver.org/.

Para los calificadores se propone utilizar el orden y los calificadores definidos en la especificación de Maven, ya que los repositories de artefactos que se utilizarán se basan en esta tecnología (https://maven.apache.org/ref/3.8.6/maven-artifact/apidocs/org/apache/maven/artifact/versioning/ComparableVersion.html). Dicha especificación define una lista de calificadores reconocidos y que serán tratados con un orden determinado durante la comparación entre versiones:

Es importante recalcar que toda la información aquí recogida, trata de facilitar y orientar para el uso de versionado semántico, pero **debe ser el equipo de desarrollo quien determine la version de su software** en función de su planificación de liberación de versiones y para ello **se recomienda la lectura del contenido completo de la especificación.**

2.1 Versionado semántico

El **versionado semántico** está formado por tres dígitos separados por puntos x.y.z, cuya posición identifica un tipo de cambio dentro del software. Cada una de las posiciones x.y.z definen un cambio mayor (MAJOR), cambio menor (MINOR) o parche (PATCH).

El uso de las tres posiciones para la numeración es obligatorio x.y.z (el cambio de cualquiera de estos números depende del tipo de cambios que aporte la nueva version) y puede ir seguido de calificadores [-T] según la fase del ciclo de vida en la que se encuentre.

Inicialmente se asignan 3 números: major.minor.path que van incrementando conforme el desarrollo del software aumente y se requiera la asignación de un nuevo número único.

Además, se puede incluir un calificador asociado a la fase en la que se encuentra el desarrollo.

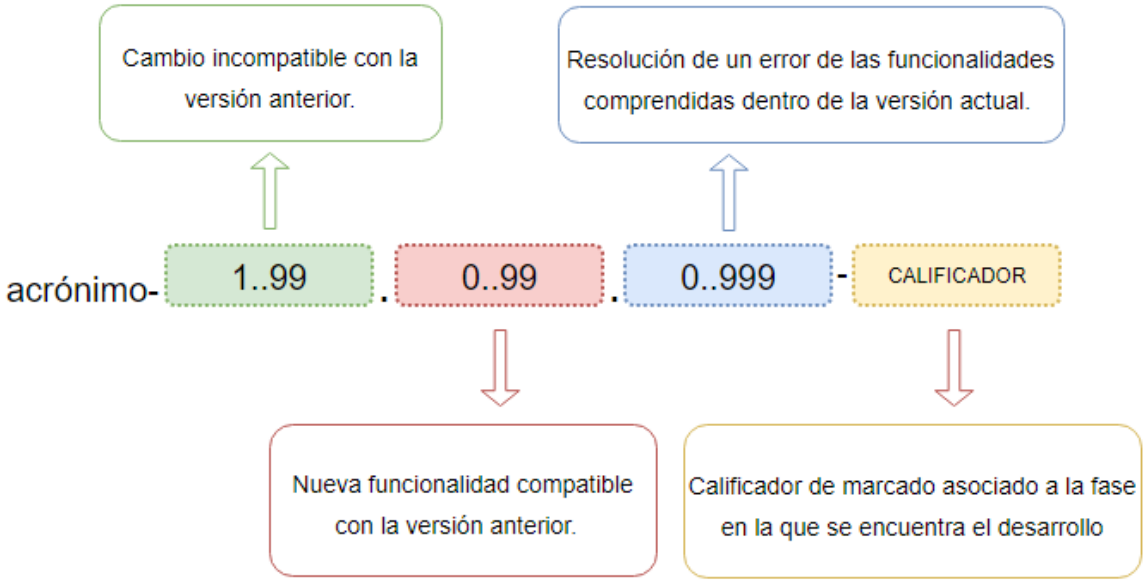

Ilustración 1. Versionado semántico

- **Major (x) –Número de version mayor**

El primer número (versión propiamente dicha) representa cambios mayores en el diseño del código que son incompatibles con la versión anterior (cambios disruptivos). Puede ser un número entero positivo y para la primera version cuando arranca en producción debe comenzar en 1.0.0, y aumentará cuando se añadan cambios significativos a la aplicación, asociados con modificaciones de diseño o implementación no retro-compatibles, es decir, la aplicación contendrá un cambio funcional que impedirá utilizarse como en versiones anteriores, por ejemplo, cambios significativos en servicios expuestos, o cambios sustanciales de su interfaz de usuario.

- **Minor (y) – Número de versión menor**

El segundo número representa modificaciones funcionales, es decir, se han añadido, eliminado o modificado funcionalidades al código compatibles con la versión anterior (cambios no-disruptivos). Aumenta cuando se añaden pequeños cambios y/o funcionalidades que no supongan un problema de compatibilidad con el uso realizado con anterioridad. Si la versión fuera 2.1.0, se trataría de la primera entrega con modificaciones funcionales de la versión 2 del software. Las versiones Minor y Patch DEBEN ser reseteadas a 0 cuando se incrementa la versión MAJOR.

- **Patch (z) - Número de revisión**

El tercer número significa correcciones de bugs o errores encontrados. Aumenta con los cambios realizados para solucionar errores detectados en las versiones liberadas, generalmente se asocia con soluciones hotfix de errores producidos en producción Si la versión fuera 1.4.3, se trataría de la tercera entrega sobre la versión 1.4 en la que se reparan los bugs o errores detectados. La versión Patch DEBE ser reseteada a 0 cuando se incrementa la versión MINOR.

- **[T] - Calificador - alpha, beta, rc, snapshot….**

Se utiliza para indicar el estado dentro del ciclo de vida del desarrollo. Se describen en el siguiente apartado.

2.2 Calificadores - Ciclo de vida asociado a versiones

Para aportar semántica a las versiones, es necesario utilizar distintos **calificadores** que se correspondan con la fase del ciclo de vida en el que se encuentra el Proyecto e indican el nivel de madurez del producto. Se incluyen detrás de la version separados por un guión.

Ejemplo 1.0.1-alpha o 1.0.1-rc1

Es importante recalcar que una versión **sin calificador** se entiende que es una versión **release** preparada para subir a producción. Aquellas versiones release, se marcarán a partir de la

inexistencia de cualquier calificador, de esta forma la versión se convertirá en final, estableciendo que su destino es el despliegue en entorno productivo, PRO. Por ejemplo: 1.0.0.

Para las versiones pre-release, se propone usar calificadores. Los calificadores más usados en DGT y en la comunidad de desarrollo de software son los siguientes:

- **snapshot** es recomendable su uso cuándo el código está en pleno desarrollo y es inestable o variable.
  - **Ejemplo:** 2.0.0-SNAPSHOT
- **rc (Release Candidate):** versión para pruebas de aceptación. Debería utilizarse para PRE hasta que se tenga la seguridad de que puede subir a producción como reléase definitiva. Podrá ser utilizado para marcar aquellas versiones en proceso de pruebas, cuyo destino sea el despliegue en el entorno PRE.
  - **Ejemplo:** 1.0.0-rc5
- **alpha, beta:** se utilizan como mecanismo adicional para proyectos complejos o de gran impacto respecto a su versión anterior.
  - alpha: versión para pruebas de integración. Ejemplo: 1.0.0-alpha
  - beta: versión para pruebas funcionales y de rendimiento. Ejemplo: 1.0.0-beta

Estos calificadores deben utilizarse con el objetivo de poder aportar de forma correcta la información de la fase del ciclo de vida y que tanto el lector de una versión, como las herramientas de automatización o repositorios de artefactos, la utilicen sin problemas. Siguiendo los calificadores reconocidos por maven/gradle el orden determinado sería el siguiente:

alpha < beta < milestone < rc < snapshot < (sin qualifier) | ga | final < sp

2.3 Versiones asociadas a desarrollo en entornos de Integración Continua

Existe un calificador especial llamado **SNAPSHOT**, que se define como un valor que identifica al último código disponible en una rama de desarrollo, marcándolo como código inestable o variable. Por el contrario, cualquier versión no marcada con el sufijo **SNAPSHOT** será inmutable. Es decir, una versión **SNAPSHOT** es la versión en desarrollo activo antes de su versión final o **release**.

En un entorno de integración continua, la versión SNAPSHOT juega un papel vital en el mantenimiento de la construcción diaria de la integración de un proyecto, minimizando la cantidad de reconfiguraciones que se requieren para integrarse con las últimas versiones de todas las dependencias de un proyecto Maven.

Las referencias hacia una versión en desarrollo activo, permiten recuperar la instancia desplegada más reciente de la dependencia SNAPSHOT en el momento de la construcción del proyecto. Es muy importante tener en cuenta que una versión snapshot puede cambiar constantemente, por lo que las dependencias SNAPSHOT serán recargadas en el repositorio local, en cada construcción, asegurando que las dependencias se actualizan y la integración se realiza con los últimos cambios de cada dependencia, sin la necesidad de cambiar las coordenadas de las dependencias del proyecto en proceso de integración.

Por lo general, una versión SNAPSHOT representa al artefacto desplegado más reciente, por lo que Maven utilizará la última versión disponible en el repositorio de artefactos local y/o remoto. Aunque el repositorio puede ser configurado para mantener un histórico rotativo con una serie de las implementaciones más recientes de un artefacto, este comportamiento posibilita marcar una dependencia con una versión snapshot determinada, por lo general sólo para solucionar algún problema puntual y únicamente de forma temporal.

Por ejemplo, para un artefacto desplegado en un repositorio con gestión de versiones SNAPSHOT, existiendo las versiones 1.2.0- 20160830.113234-1 y 1.2.0- 20160830.124911-2 asociadas a la versión 1.2.0-SNAPSHOT, un proyecto que declare la dependencia de ésta última, se integrará con la última versión disponible, es decir, la 1.2.0-20160830.124911-2. El almacenamiento de un histórico de implementaciones, marcado con la serie numérica 1 y 2, dan la posibilidad de fijar la dependencia con <version>1.2.0- 20160830.113234-1<version>, obligando a Maven a utilizar la versión indicada para realizar la integración, de esta forma se puede mantener estable las próximas integraciones y resolver algún problema puntual.

Para poder comprender en detalle qué es una versión **SNAPSHOT**, cómo es utilizada por los repositorios de artefactos, cómo se deben definir dependencias para su utilización, y cómo se deben utilizar en versionado de software, es recomendable estudiar en profundidad la

documentación proporcionada por la herramienta **Apache Maven**, disponible en la URL https://maven.apache.org/guides/getting-started/index.html#What_is_a_SNAPSHOT_version.

2.4 Ejemplos de versionado

A continuación, se muestran ejemplos de nombrado en función del tipo de modificación realizada sobre el proyecto:

- Si se comienza el desarrollo de una versión nueva y no retrocompatible, se incrementa la versión mayor, restableciendo las demás versiones:

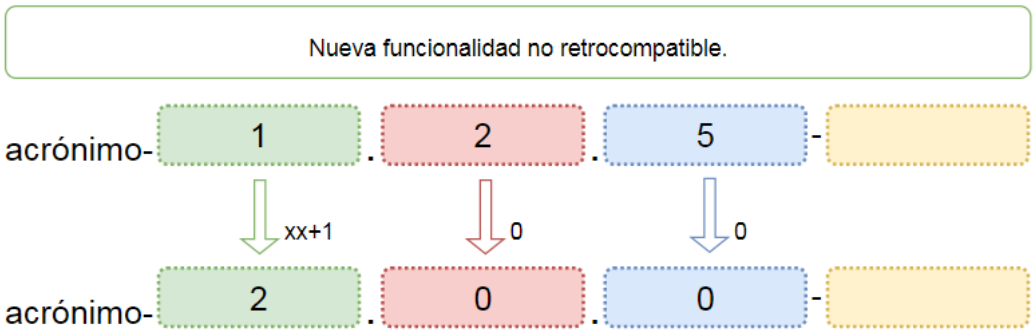

Ilustración 2. Nombrado de versión nueva no retrocompatible

- Si se cambia la funcionalidad en el contexto de la versión actual o se crea una nueva funcionalidad retrocompatible, se incrementa la versión menor:

Ilustración 3. Cambio de funcionalidad o nueva funcionalidad retrocompatible

- Si se resuelve un error de código, se incrementa la versión de revisión:

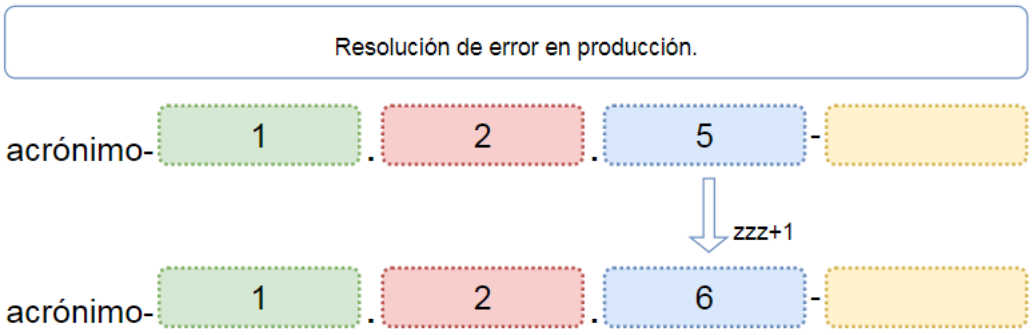

Ilustración 4. Error de código

- Si se indica que la madurez de la versión ha cambiado de fase:

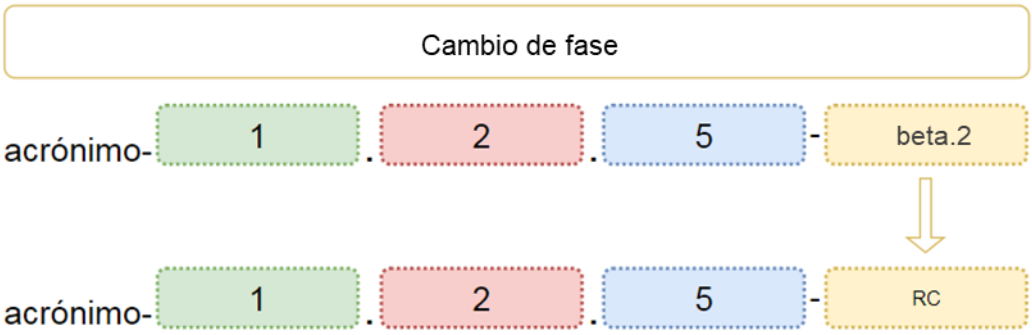

Ilustración 5. Madurez de la versión cambia de fase

3 Normativa de versionado de la documentación

Para poder identificar cada uno de los documentos involucrados en un proyecto de desarrollo se ha establecido la siguiente nomenclatura:

ACRO-SSSSSS-TD-NOMBREBREVE-xx.yy.zz, donde:

- **ACRO:** Acrónimo del sistema de información.
- **SSSSSS:** Acrónimo del subsistema.
- **TD:** Tipo de documento de entre los siguientes:
  - PLN: Plan
  - INF: Informe
  - REG: Registro
  - DES: Descripción
  - RES: Resultado
  - INS: Instrucciones (procedimiento)
  - MAN: Manual
- **NOMBREBREVE.** Por ejemplo: REQ para los requisitos.
- **xx.yy:** Código del documento. Este código se debería corresponder con las coordenadas major.minor de la versión de código con la que está relacionado el documento.
- **zz:** coordenada utilizada para identificar las diferentes revisiones que ha podido sufrir el documento, es independiente de la coordenada patch de la versión del código.