Guía de desarrollo, Anexo 10.00

Normativa de versionado, estructura y nombrado en SVN

Oficina de Calidad

GERENCIA INFORMÁTICA
JOSEFA VALCÁRCEL, 44
28027-MADRID

Índice General

Índice de Ilustraciones y Tablas

1 Introducción

1.1 Objetivo

Subversion (SVN) es un sistema de control de versiones libre y de código fuente abierto. Se trata de un sistema de gestión de la configuración que gestionará en DGT las fuentes de las aplicaciones y la documentación del proyecto, facilitando el control de las modificaciones que se van realizando a lo largo de todo el ciclo de vida del producto y del proyecto.

El objetivo de este documento es describir la estructura de SVN y las directrices que deben seguir todos los proyectos de construcción de aplicaciones software 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, análisis, diseño, desarrollo, auditoría, pruebas, 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.

2 Estructura de subversion

Subversion es el sistema de gestión de la configuración que gestionará las fuentes de las aplicaciones y la documentación del proyecto de la DGT mientras no se migre a Git.

La estructura que se ha definido para el SVN, tiene como objetivo dar soporte a los diferentes ciclos de vida existentes en el desarrollo de aplicaciones software, entre los que se identifican: el ciclo de vida de la gestión del proyecto, el ciclo de vida de la documentación y el ciclo de vida del software. Es por ello que tanto la gestión del proyecto, como la documentación y código de la aplicación contarán con un directorio o repositorio (dependiendo de la aplicación) independientes.

Actualmente conviven en DGT dos estructuras diferentes dependiendo del momento del alta del acrónimo en subversión. Una está basada en directorios que cuelgan de la ruta existente en el SVN (por ejemplo Horizontales) y otra está basada en repositorios independientes que cuelgan directamente de subversión, pero ambas comparten por debajo la misma estructura de carpetas y de tags.

2.1 Estructura basada en directorios

Es la estructura usada por los proyectos más antiguos. Cada acrónimo tiene un directorio con una estructura definida que cuelga del grupo funcional al que pertenece.

https://subversion.trafico.es/subversion/Grupo funcional/acronimo/

La estructura interna definida está formada por los directorios que se definen a continuación:

- **Acro,** Este directorio será el específico para la **documentación asociada a la gestión del proyecto**. Además de las actas y los informes de seguimiento contendrá la documentación de arranque del proyecto como puede ser el pliego, el Plan de proyecto, Acuerdos y ANS establecidos, etc. **y la documentación común a todos los subsistemas**.

Nota: Para los proyectos ya existentes se permite también tener un directorio por cada proyecto (acro-proyxx),

- **acro-[subsistema]-dev,** Este es el directorio específico para el **código de la aplicación y la documentación necesaria para el despliegue**, en el caso de que existieran subsistemas habrá un directorio de este tipo por cada uno de ellos.

- **acro-[subsistema]-doc,** Este es el directorio específico para la **documentación técnica de la aplicación cuyo ciclo de vida es independiente del código**; en el caso de que existieran subsistemas habrá un directorio de este tipo por cada uno de ellos, a excepción del subsistema de base de datos para el que no existiría este directorio.

- **acro-bbdd** Este directorio es específico para la base de datos y contendrá los scripts a ejecutar (dev/scripts) y el modelo de datos correspondiente (doc).

- **Vxx,** donde se encontrará el histórico de las versiones anteriores de la aplicación.

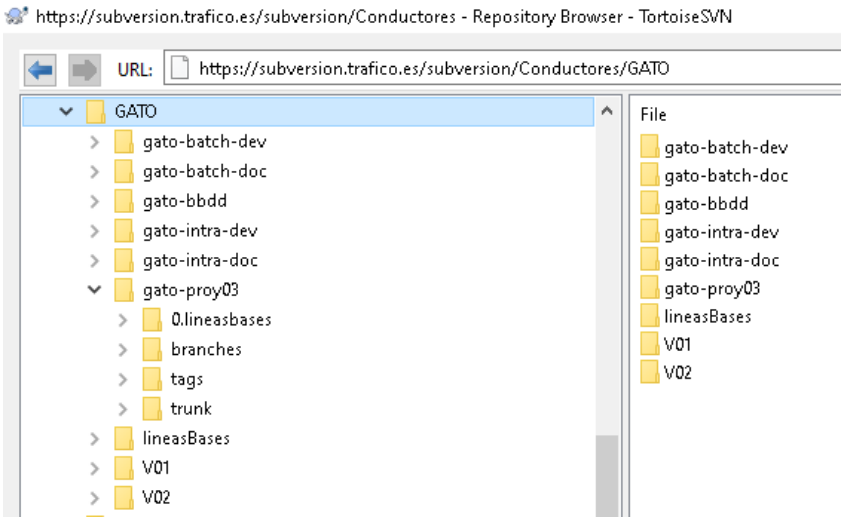

Ilustración 1. Ejemplo de estructura basada en directorios

La estructura interna de cada uno de los directorios se detalla en el apartado Estructura de carpetas.

2.2 Estructura basada en repositorios

Para las aplicaciones más recientes creadas en DGT se ha definido una estructura muy similar a la anterior pero basada en la creación de repositorios independientes. Como mínimo habrá los siguientes:

https://subversion.trafico.es/subversion/acronimo/

https://subversion.trafico.es/subversion/acronimo-[subsistema1-]dev/

https://subversion.trafico.es/subversion/acronimo-[subsistema1-]doc/

https://subversion.trafico.es/subversion/acronimo-bbdd /

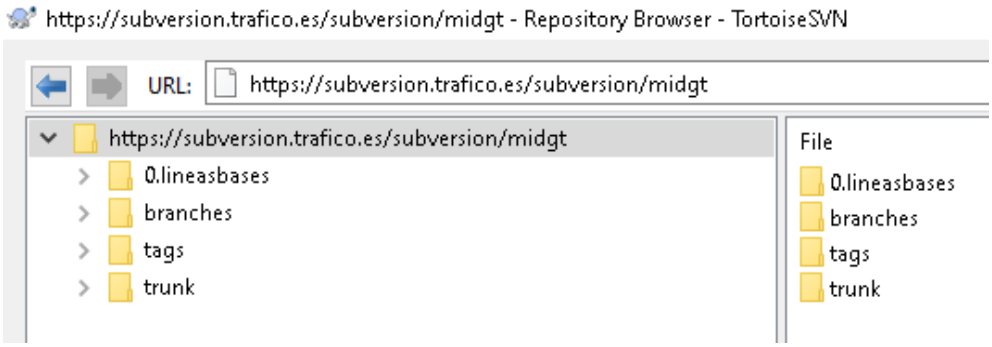

Ilustración 2. Ejemplo de estructura basado en repositorios

La estructura interna de cada uno de los repositorios se detalla en el apartado Estructura de carpetas.

2.3 Estructura de carpetas

Todos los directorios/repositorios descritos anteriormente, contarán a su vez con la estructura:

- **0.lineasbases**: espacio reservado para los equipos transversales: oficinas de calidad y pruebas, departamento de arquitectura y departamento de sistemas. En él se localizará la información compartida entre los diferentes equipos transversales y el equipo de desarrollo, así como líneas

base de los tags de pre (aquellos que no se han subido a PRO) y de pro (aquellos que se han subido a PRO). En el caso de 0.lineasbases de acro o acro-proyxx, contendrá la carpeta intercambio subdividida en proyectos y/o hitos (h0x).

- **branches:** donde se encontrarán las diferentes líneas de desarrollo, con su nomenclatura correspondiente, al margen de la línea principal de documentación que se encuentra en trunk.
- **tags:** donde se encontrarán las etiquetas con las entregas de la aplicación.
- **trunk:** se utiliza para alojar la línea base del desarrollo.

2.4 Estructura de tags

Dentro de cada etiqueta y dentro del trunk se incluirán los directorios dev, doc y ges. De forma generalizada:

- En “dev” se localizará el código fuente necesario para generar los artefactos a desplegar. No se incluye documentación.
- En “doc” se localizará la documentación técnica de acuerdo a la fase en la que se elabora:
  - **doc/02.ASI:** Documentación de la fase de análisis: Maqueta, Especificación de Requisitos, Documento de análisis, etc.
  - **doc/03.DSI:** Documentación de la fase de diseño. Ejemplo, Diseño de arquitectura, Interfaz de entrada, etc.
  - **doc/04.CSI** Documentación de la aplicación: Manual de Usuario, Manual CAU, Manual de integración, Descripción de Registros de Auditoría etc.
  - **doc/05.IAS/** Documentación de implantación (documentación detallada en el anexo 33) y documentos de resultados de las pruebas agrupados en carpetas por tipo de pruebas. Ej. Carpeta\pruebas\seguridad

- En “ges” se podrá incluir información y documentación de gestión agrupada por tipo de documento.

  - **ges/actas/** Actas de reunión.

  - **ges/contrato/** Memorias y contrato (MT, PPT, ...).

  - **ges/informes seguimiento/** Copia de los correos enviados por el proyecto al Jefe de proyecto DGT notificando riesgos del proyecto (si procede), Informe Seguimiento aaaa_mes_Proveedor, etc.

  - **ges/correos/** Copia de los correos relevantes del proyecto intercambiados entre los participantes del proyecto que necesiten ser almacenados para referencias posteriores.

  - **ges/solicitudes/** Solicitudes enviadas por parte del proyecto a los distintos actores y funciones transversales de DGT.

  - **ges/trazabilidad/** Matrices de trazabilidad y matriz de configuración.

  - **ges/informes y acuerdos/** Evidencias de auditorías. Informes de auditoría resultantes de la realización de las auditorías internas de los proyectos.

  - **ges/validaciones/** Correos de validación de entregables. Copia de los correos de confirmación enviados por el usuario, CAU, almacén de datos, etc. tras revisar los entregables que necesitan su firma.

- Además, para poder manejar toda la documentación adicional no sujeta a la metodología DGT útil para el proyecto y con el fin de no perderla una vez terminado el mismo, se podrán crear nuevos directorios con nombre “documentacion adicional” dentro de las carpetas “doc” y “ges” siempre que no se modifique la estructura básica del repositorio.

3 ¿Cómo solicitar el alta de un acrónimo en subversion?

Como primera consideración a tener en cuenta, el acrónimo de los proyectos podrá tener hasta 10 caracteres. No se podrá reutilizar un nombre de acrónimo ya existente. Además, se recomienda el uso de vocales para facilitar su identificación.

Para solicitar el alta de un acrónimo en subversion y la creación de los nuevos repositorios hay que enviar un correo a operaciones.tic indicando el acro, los repositorios a crear y los usuarios que deben tener permiso.

Para los proyectos nuevos, es decir, para aquellos cuyo acrónimo no existe en el SVN de la DGT, se recomienda la estructura de SVN basada en repositorios, aunque si no fuese posible, debido a los problemas de espacio y permisos que hay actualmente, se podrá optar por la estructura basada en directorios.

4 Gestión de la configuración

Todas las entregas de código fuente y documentación que se realicen, deberán llevarse a cabo utilizando etiquetas (tags). Todas las etiquetas se deberán generar en subversion en la carpeta “tags”, con la nomenclatura que se define en el siguiente punto. Las ramas (branches) también utilizarán la nomenclatura designada.

4.1 Nomenclatura de etiquetas (tags)

La nomenclatura de las etiquetas puede variar dependiendo de si es una etiqueta de documentación del proyecto, una entrega de código o de documentación:

- La nomenclatura de etiqueta a usar para las **carpetas de gestión del proyecto** creadas en proyxx/tags será: **acronimo-proyxx_[hyy]-x.y.z**

- La nomenclatura de etiqueta a usar para las carpetas de **código** creadas en acronimo-[subsistema]-dev/tags será: **acrónimo-[subsistema]-x.y.z[-T]**.

- La nomenclatura de etiqueta a usar para las carpetas de **documentación técnica** creadas en acronimo-[subsistema]-doc/tags será: **acrónimo-[subsistema]-x.y.z** o **acrónimo-[subsistema]-doc-x.y.z**

Nota1: Las coordenadas x.y.z (versión) deben seguir las especificaciones de versionado semántico descritas en el anexo 38.

Nota 2: Dentro del proceso de versionado y liberación del proyecto, es necesario tener en cuenta que la documentación y el código fuente, deberían seguir un versionado paralelo, es decir, **las versiones asignadas a las etiquetas de código y de su documentación, deben tener la misma versión mayor y menor**, dejando el número de revisión y el calificador para poder disponer de un ciclo de vida independiente entre código y documentación.

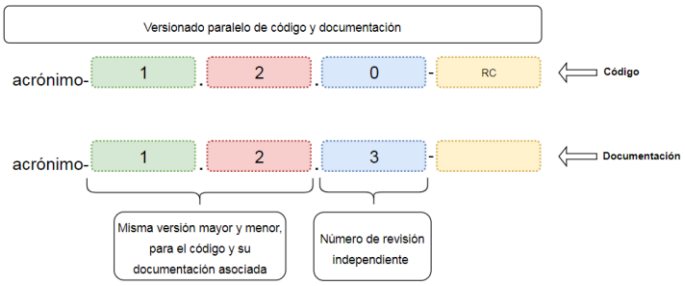

Ilustración 3. Versionado paralelo de código y documentación

4.2 Nomenclatura de ramas (branches)

Se generará una rama en la carpeta “branches” de la estructura de SVN, siguiendo la nomenclatura designada para las ramas. Todas las copias/ramas son versiones que en un futuro se pueden borrar una vez cumplido su cometido.

Las ramas deben utilizar la semántica de los siguientes prefijos:

- **hotfix**: para marcar una rama destinada a la resolución de incidencias en producción, encontradas en código liberado. Ejemplo: hotfix-acrónimo-1.2.5.
- **feature**: para marcar una rama destinada a la implementación a largo plazo, y de forma paralela, de una nueva funcionalidad. Ejemplo: feature-descripción-nueva-funcionalidad
- **release**: para marcar una rama de mantenimiento o desarrollo paralelo a la rama principal del repositorio. Ejemplo: release-acrónimo-1.2.x.

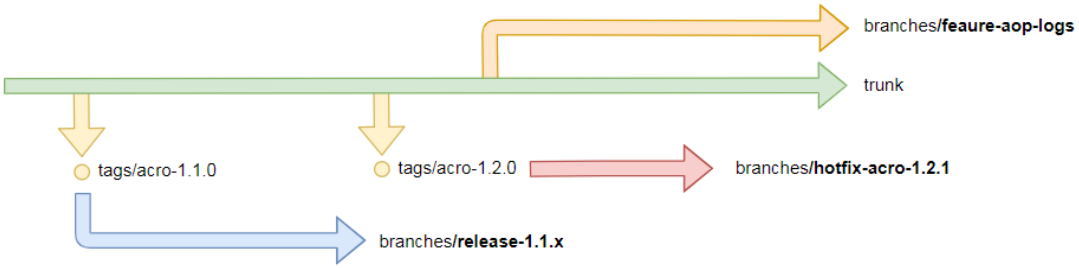

Ilustración 4. Nombrado de ramas

4.3 Nomenclatura de proyectos, módulos, ficheros y artefactos

- **Nombres de proyectos maven/gradle y módulos/subproyectos:** los nombres de proyectos maven/gradle y módulos/subproyectos deben ser generados a partir de palabras en minúsculas y separados por guiones. Ejemplo: pojo-acro-local-salida
- **Nombres de ficheros:** el nombre de los ficheros debe seguir la nomenclatura definida por el lenguaje de programación, herramienta o software al que esté asociado. En el presente documento no se detallarán las buenas prácticas asociadas a todas las tecnologías, pero si se identificarán nomenclaturas o notaciones que se deban seguir de forma obligatoria en aquellos apartados de la normativa que hagan referencia a su uso.

- **Nombres de artefactos**: los nombres de los artefactos deben coincidir con el nombre del proyecto maven/gradle o módulo/subproyecto que lo genera, por lo que seguirán su misma nomenclatura, añadiéndole la versión del software (siempre que aplique). Ejemplo: pojo-acro-1.0.1-beta.jar.

5 Ciclo de vida de desarrollo con subversion

Atendiendo a la estructura del control de versiones para el código fuente:

- **trunk (línea base):** se utiliza para alojar la línea base del desarrollo. Es la carpeta en la que se actualizarán los cambios continuos con los que implementar las nuevas versiones.

- **branches (ramas):** puede contener las copias/ramas. En el caso en que se deban mantener varias versiones en paralelo, se utilizarán ramas para gestionar las distintas líneas de desarrollo concurrentes.

- **tags (etiquetas):** contiene las etiquetas asociadas con la liberación del producto mediante versiones release o release candidate (despliegue de pruebas en PRE).

La línea base, o trunk, es la línea principal de desarrollo con evolución constante, siendo el equipo de desarrollo el encargado de mantener el código fuente del proyecto sin errores durante la compilación y ejecución de pruebas unitarias.

Las etiquetas son instantáneas de los archivos de la línea base o rama en un momento determinado del ciclo de vida del software, utilizadas para liberar versiones finales o destinadas a procesos de pruebas y aprobación.

Las etiquetas acrónimo-[subsistema]-x.y.z-RC representan aquellas versiones de las aplicaciones que están siendo sometidas a pruebas de integración en el entorno de pre-producción. En el transcurso de las pruebas pueden surgir defectos a reparar, volviendo a empezar el ciclo.

Cuando una versión de la aplicación, pasa las pruebas de integración (entorno de pre-producción), puede ser promovida y marcada como liberada, eliminado cualquier tipo de calificador o fijando el calificador adecuado para el contexto de su liberación.

5.1 Desarrollo correctivo

En el mantenimiento correctivo se corrigen los defectos detectados después de la entrega del producto; es la forma básica de mantenimiento y consiste en localizar averías o defectos y corregirlos.

Durante el ciclo de desarrollo correctivo, se localiza el defecto y corrige generando una nueva versión en SVN que es desplegada en el entorno de PRE para su testeo. Si todo es correcto, se genera y despliega en Producción cumpliendo con los procedimientos establecidos de despliegue entre entornos.

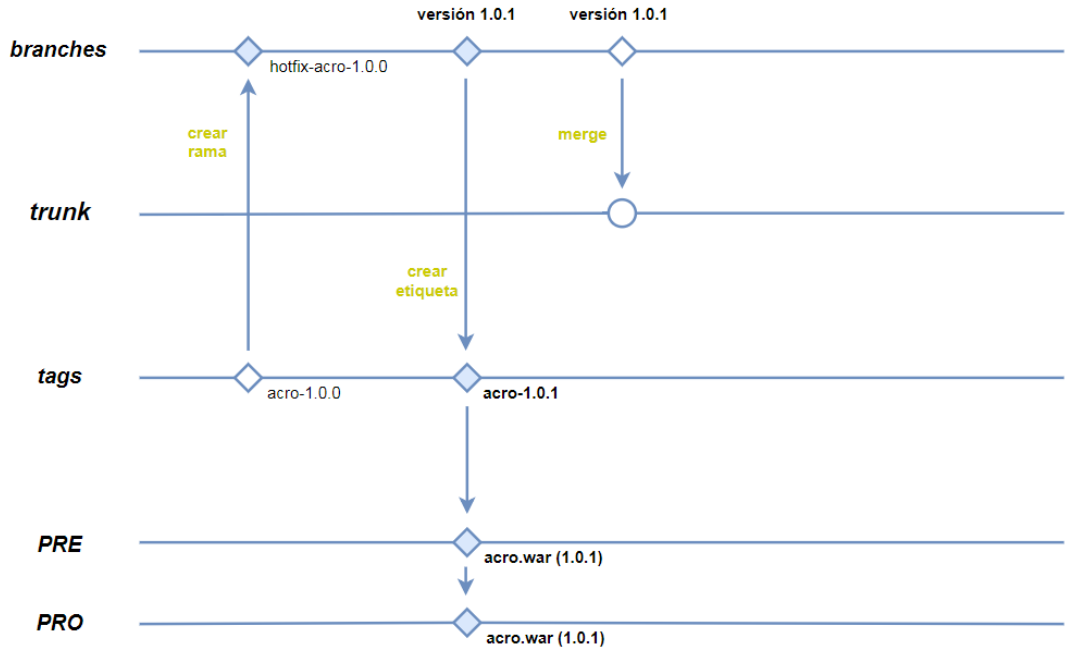

Ilustración 5. Desarrollo correctivo

5.2 Desarrollo evolutivo

Un mantenimiento evolutivo es aquel que pretende modificar algo que funciona y es correcto, con el objeto de aumentar, disminuir o cambiar las funcionalidades del sistema, ya sea por las necesidades del usuario o por otras causas, por ejemplo, cambios normativos.

Todos los desarrolladores trabajan sobre la rama trunk, en la que realizarán commit diarios de sus implementaciones, siempre que no contengan errores detectados por SonarLint (ver Anexo 39 Guia tecnica de uso de SonarLint), ni fallen las pruebas unitarias lanzadas en local. De esta forma se dispondrá de una rama continuamente actualizada y probada, de la que saldrán las versiones con las que generar los tags.

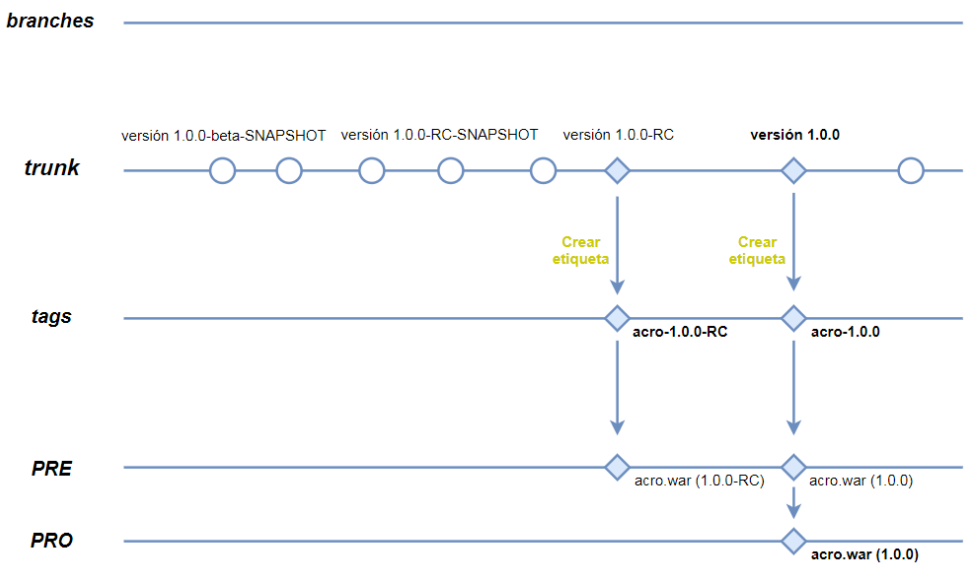

Ilustración 6. Desarrollo evolutivo

5.3 Desarrollo evolutivo y correctivo

En desarrollos evolutivos más largos pueden surgir cambios en la línea principal y tener que realizar desarrollos correctivos, éstos una vez terminados se integrarán desde la línea base a la rama de la nueva funcionalidad.

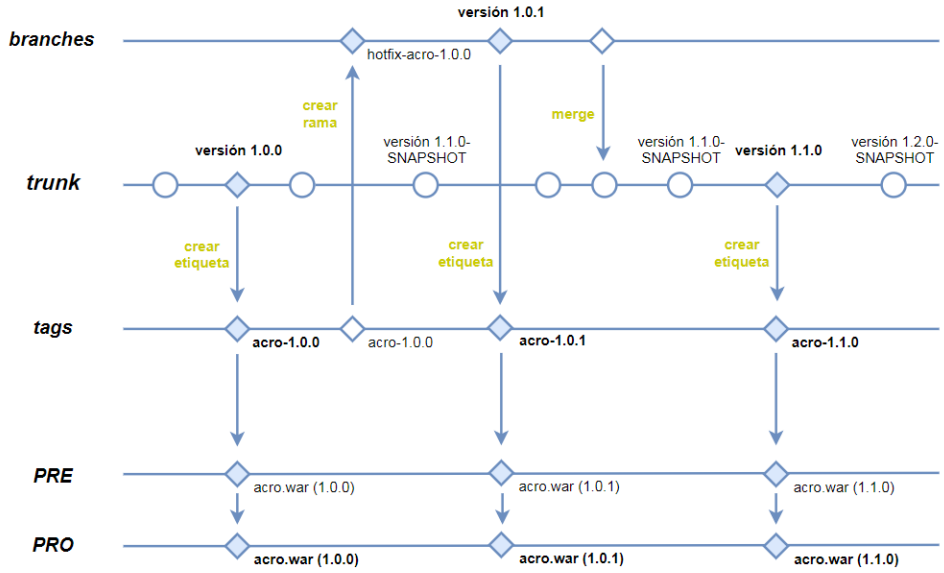

Ilustración 7. Desarrollo evolutivo y correctivo

6 Plantillas

- En el directorio de distribución se proporciona un fichero comprimido en formato RAR "EstructuraCarpetasVacias.rar" donde está generada la estructura de carpetas del SVN para facilitar la tarea de generación del repositorio del proyecto.
- En la figura siguiente se muestra un ejemplo de estructura de SVN para una **aplicación sin subsistemas** incluyendo ejemplos de nombrado de branches y tags.

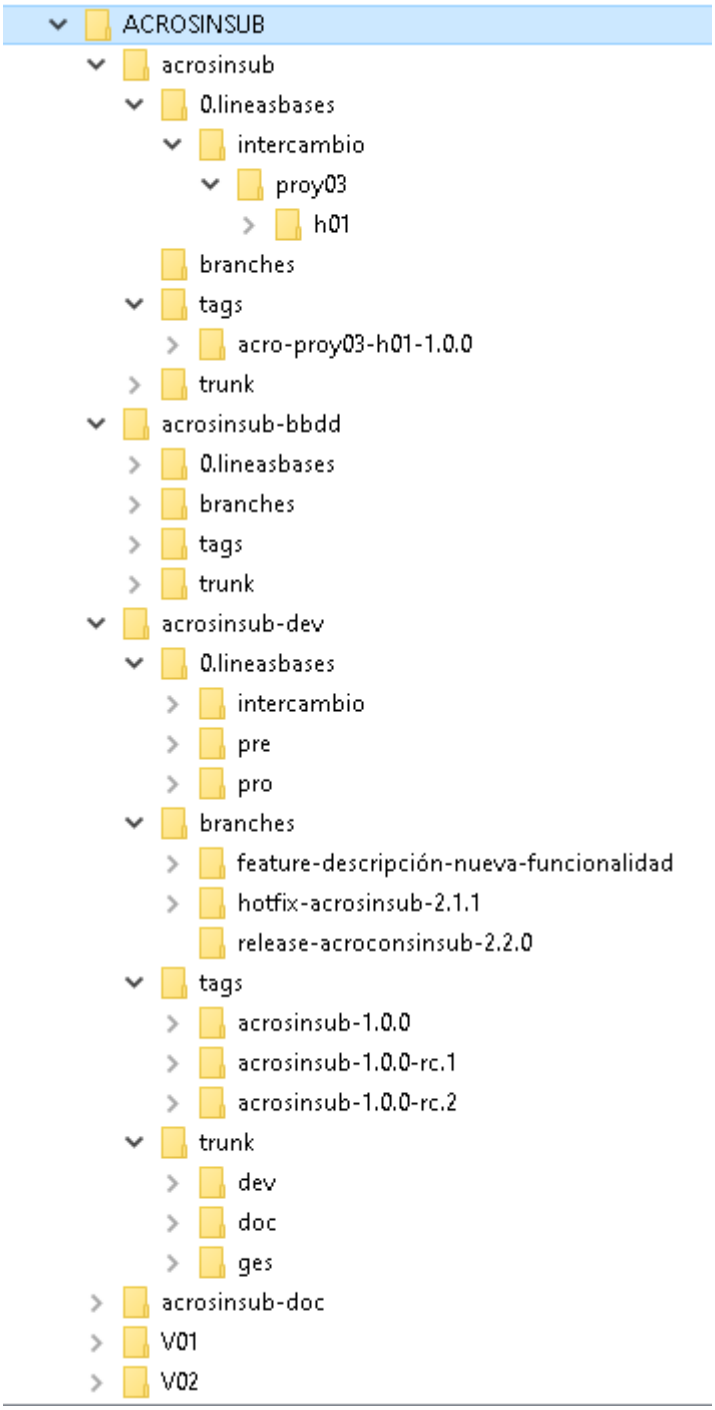

Ilustración 8: Ejemplo de estructura sin subsistemas