Asignatura: Entornos de programación
Gestión de configuración

Control de versiones, configuración y cambios

1. Introducción

En este tema se describen las actividades básicas de gestión de configuración y las técnicas y herramientas utilizadas para ello. La terminología empleada para referirse a esta actividad varía según los casos. Por ejemplo:

- VCS: Version Control System
- SCM: Software Configuration Management
- CMS: Configuration Management System

Control:

- Se dispone de medios para realizar materialmente una tarea de manera segura.

Gestión:

- Hay criterios establecidos que hay que seguir en la realización de una tarea.

2. Evolución del software

La necesidad de gestionar la configuración surge del hecho de que el software evoluciona con el tiempo:

- Durante el desarrollo
  - El desarrollo del software siempre es progresivo, incluso en el ciclo de vida en cascada
  - El desarrollo evolutivo consiste, precisamente, en una evolución controlada (ciclo de vida espiral, prototipos evolutivos)
- Durante la explotación
  - Durante la fase de mantenimiento se realizan modificaciones sucesivas del producto
- En todos los casos
  - Suele ser necesario recuperar versiones antiguas, aunque sea sólo para consulta
  - Para ello hace falta tener organizado el almacenamiento de versiones anteriores

3. Conceptos generales

- Control de versiones
  - Evolución de un elemento.
- Control de configuración
  - Evolución de un conjunto de elementos.
- Control de cambios
  - Metodología de desarrollo evolutivo.
- Revisiones y variantes
  - Configuraciones alternativas.
  - *Branch-merge*.
- Repositorio
  - Almacén de información organizado.

3.1 Control de versiones

Utilizaremos este término para referirnos a la evolución de un único elemento, o de cada elemento por separado.

- Concepto de versión:
  - Desde el punto de vista de la evolución, es la forma particular de un objeto en un instante o contexto dado. Se suele denominar "revisión" cuando se refiere a la evolución en el tiempo.
  - También hay que contemplar la posibilidad de que coexistan versiones alternativas en un instante dado.
  - Hay que disponer de un método para designar las diferentes versiones de manera sistemática u organizada.

3.2 Control de configuración

Con este término nos referiremos a la evolución de un conjunto de elementos.

- Concepto de configuración
  - Un sistema software comprende distintos componentes, que evolucionan individualmente.
  - Hay que garantizar la consistencia del conjunto del sistema.
  - Una 'configuración' es una combinación de versiones particulares de los componentes que forman un sistema consistente.
  - Desde el punto de vista de la evolución en el tiempo, es el conjunto de las versiones de los objetos componentes en un instante dado
  - Una configuración cambia porque se añaden, retiran o modifican elementos. También hay que contemplar la posibilidad de que los mismos elementos se reorganicen de forma diferente, sin que cambien individualmente.
  - Hay que disponer de un método para designar las diferentes configuraciones de manera sistemática u organizada.

3.3 Control de cambios

Es un concepto relacionado con la metodología de desarrollo de software. Se trata de hacer el desarrollo de forma evolutiva, mediante cambios sucesivos realizados de una manera disciplinada.

- Línea base
  - Denominaremos así a una configuración operativa del sistema software, a partir de la cual se puede desarrollar un cambio.
  - La evolución del sistema puede verse como evolución de la línea base.
- Concepto de cambio
  - Es el paso de una versión de la línea base a la siguiente.
  - Puede incluir modificaciones del contenido de algún componente.
  - Puede incluir modificaciones de la estructura del sistema, añadiendo, eliminando o reorganizando componentes.

3.4 Variantes

- Configuraciones alternativas
  - Un sistema software puede adoptar distintas formas (configuraciones) dependiendo del lugar donde se instale. Por ejemplo, dependiendo de la plataforma (máquina + S.O.) que la soporta, o de las funciones opcionales que haya de realizar o no.
  - Una variante es una versión de un componente (o de la configuración global) que evoluciona por separado.
  - Las *variantes* representan una *variación espacial*, mientras que las *revisiones* representan una *variación temporal*.

3.5 Repositorio

- Es un almacén general de versiones:
  - Es habitual centralizar el almacenamiento de los componentes de un mismo sistema, incluyendo las distintas versiones de cada componente. Este almacén común se denomina REPOSITORIO.
  - El repositorio permite ahorrar espacio de almacenamiento, evitando guardar por duplicado elementos comunes a varias versiones o configuraciones.
  - Para conseguir ese ahorro hay que disponer de un sistema de representación especializado para las versiones.
  - El repositorio facilita el almacenar información de la evolución del sistema (historia), y no sólo de los componentes en sí (*datos* + *metadatos*).

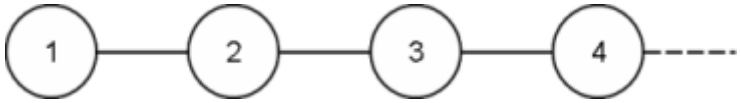
4. Control de versiones

Como se ha dicho, se refiere a la evolución de un único elemento, o de cada elemento por separado si son varios.

La evolución puede representarse gráficamente en forma de grafo, en el que los nodos son las versiones y los arcos corresponden a la creación de una nueva versión a partir de otra ya existente.

4.1 Grafo de evolución simple

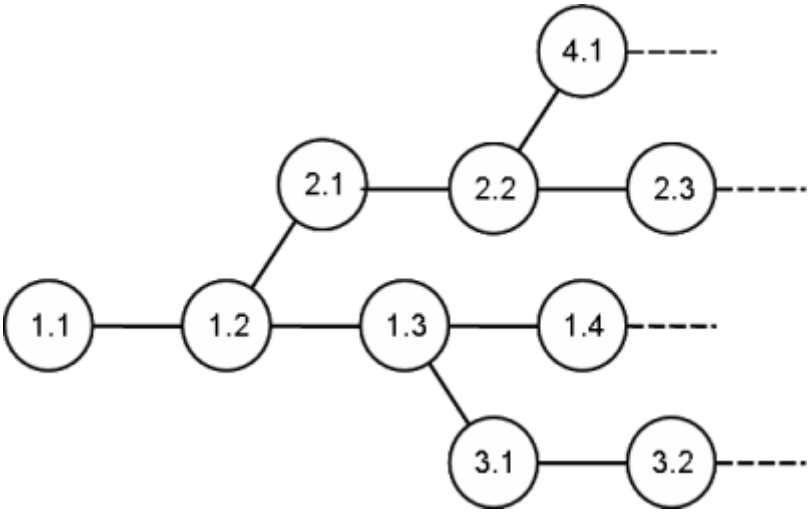
Las revisiones sucesivas de un componente dan lugar a una simple secuencia lineal.



Esta forma de evolución no presenta problemas desde el punto de vista de organización del repositorio. Las versiones se pueden designar simplemente mediante números correlativos, como en la figura.

4.2 Variantes

Cuando hay variantes, es decir, cuando existen simultáneamente varias versiones del componente, el grafo de evolución ya no es una secuencia lineal, sino que adopta la forma de un árbol. Si queremos seguir numerando las versiones se necesitará ahora una numeración a dos niveles. El primer número designa la variante (línea de evolución), y el segundo la versión particular (revisión) a lo largo de dicha variante.



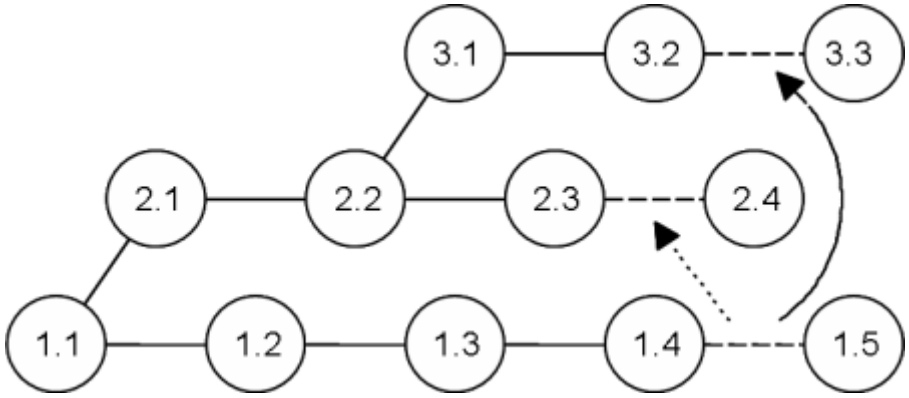
La terminología usada para referirse a los elementos del grafo es la propia de un árbol:

- TRONCO (*trunk*): Es la variante principal, p.ej. 1.1-1.2...
- CABEZA (*head*): Es la última versión del tronco, p.ej. 1.4
- RAMAS (*branches*): Son las variantes secundarias, p.ej: 2.1..., 3.1...
- DELTA (*delta*): Es el cambio de una revisión respecto a la anterior. El nombre delta puede aplicarse a varios conceptos. Ejemplo, Delta 3.2 puede representar:

- El paso de una versión a otra, es decir, un arco del grafo: (3.1 → 3.2)
- Los cambios (diferencias) entre una versión y otra: (3.2 - 3.1)
- La versión resultante, en sí misma: (3.2)

4.3 Propagación de cambios

Cuando se tienen variantes que se desarrollan en paralelo suele ser necesario aplicar un mismo cambio a varias variantes. Podemos empezar por realizar el cambio en una rama y luego propagarlo a las otras.



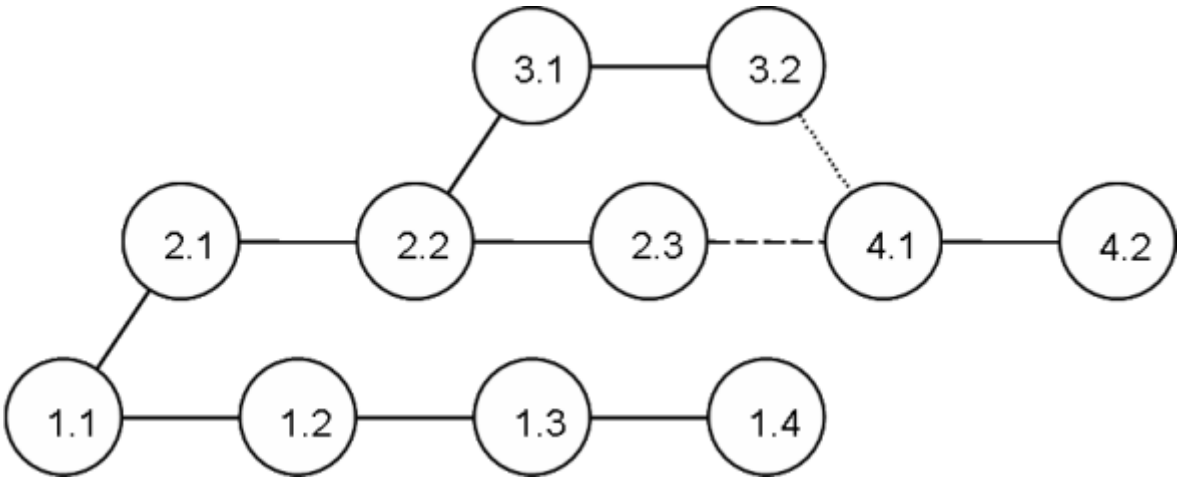
Hay herramientas concretas que permiten automatizar la propagación del cambio. Se denominan “Diff-Merge”. Usando una notación matemática ("-" para la diferencia entre versiones y "+" para la aplicación de un cambio) podríamos escribir:

```
2.4 = 2.3 + (1.5 - 1.4)
3.3 = 3.2 + (1.5 - 1.4)
```

La nueva versión se obtiene a partir de otras tres. Esta acción se denomina mezcla de tres vías (*three-way merge*).

4.4 Fusión de variantes

En determinados momentos puede dejar de ser necesario mantener una rama independiente. En este caso se puede fundir con otra (MERGE), y el árbol de evolución pasa a ser un grafo convencional.



Para fundir variantes se puede operar de forma similar a la propagación de cambios, aplicando a una rama los cambios independientes hechos en la otra. Por ejemplo:

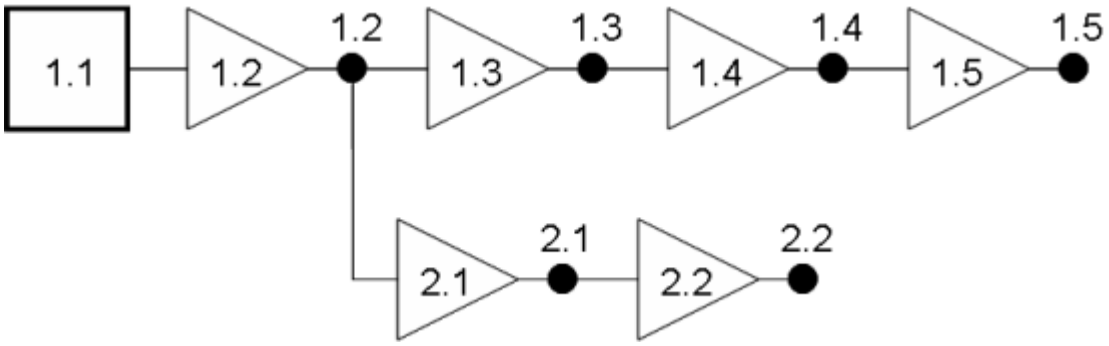
```
4.1 = 2.3 + (3.2 - 2.2),  o bien
4.1 = 3.2 + (2.3 - 2.2)
```

También se pueden combinar manualmente o mediante alguna herramienta las dos versiones finales para crear la nueva versión común. Esta acción se denomina mezcla de dos vías (*two-way merge*).

4.5 Técnicas de almacenamiento

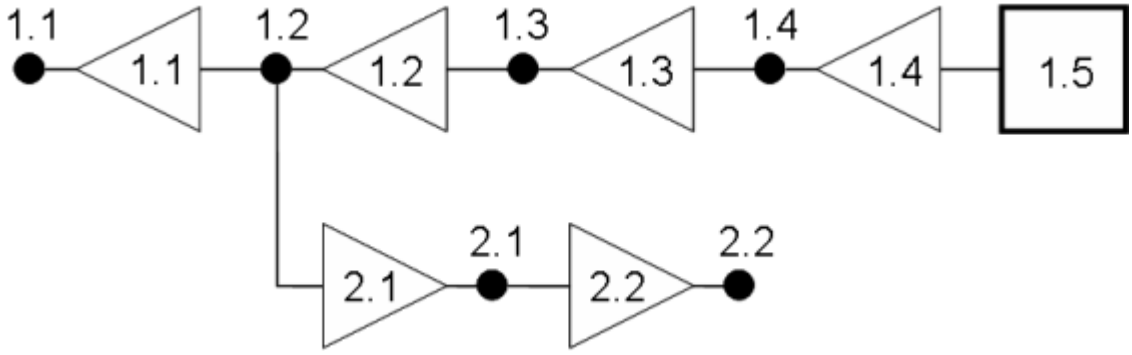
En la mayoría de los casos las distintas versiones tienen en común gran parte de su contenido. Almacenar cada versión completa por separado desaprovecha bastantes recursos, en comparación con la posibilidad de almacenar cada fragmento de información distinto sólo una vez aunque aparezca repetido en diferentes versiones. Existen distintas técnicas para organizar el almacenamiento combinado del conjunto de versiones de una manera eficiente. Se apoyan en herramientas del tipo *diff* o *diff-merge*.

- **Deltas directos**: Se almacena completa la primera versión, y luego los cambios mínimos necesarios para reconstruir cada nueva versión a partir de la anterior.



Ventajas: Es sencillo de implementar y resulta bastante intuitivo.
Inconvenientes: Es más costoso recuperar las últimas versiones (lo más frecuente) que las primeras (menos frecuente).

- **Deltas inversos** (RCS): Se almacena completa la última versión del tronco y los cambios necesarios para reconstruir cada versión anterior a partir de la siguiente. En las ramas se mantiene el uso de deltas directos.



Ventajas: Es menos costoso recuperar las últimas versiones que las primeras, pero sólo en el tronco o rama principal.

Inconvenientes: En las otras ramas es más costoso recuperar las últimas versiones que si se aplicaran sólo deltas directos.

- **Marcado selectivo** (SCCS): Se almacena el texto refundido de todas las versiones como una secuencia lineal, marcando cada sección del conjunto con los números de versiones a los que corresponde. Usando una notación simbólica tendríamos, por ejemplo:

```
   x x x x x
   x x x x x
<<1.3,1.2
   y y y y
>>
<<1.2
   z z z z z z
   z z z z z z
>>
   x x x x x
<<1.3
   t t t
>>
   x x x x x
   x x x x x
```

Ventajas: Cuesta lo mismo recuperar cualquier versión, tanto reciente como antigua y de cualquier rama.
Inconvenientes: A medida que aumenta el número de versiones aumenta también el costo de recuperar cualquier de ellas.

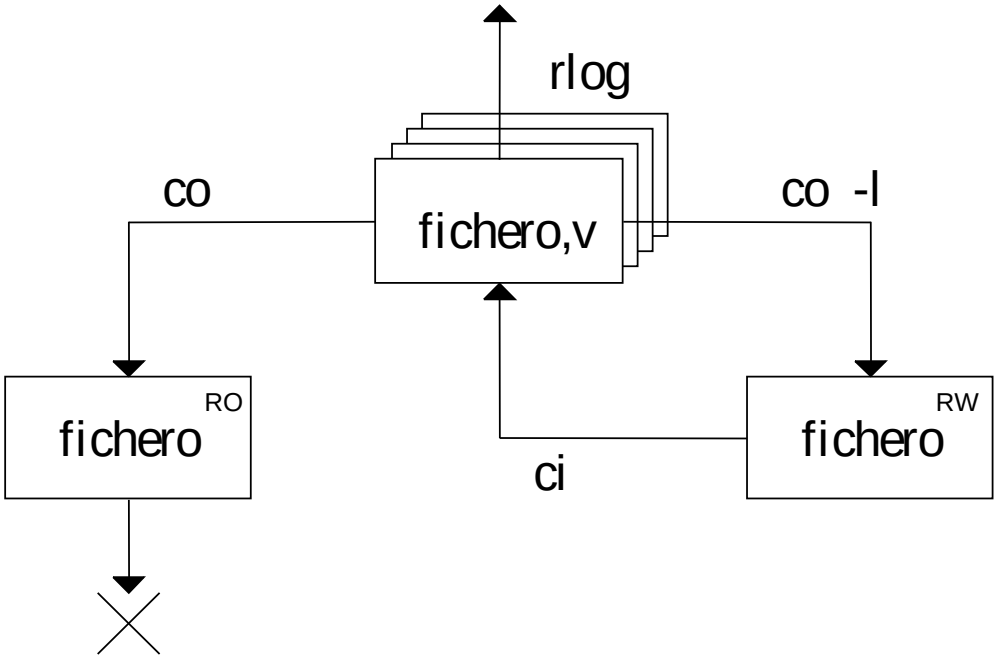
4.6 Herramientas de control de versiones

Como ejemplo de herramientas de control de versiones se pueden citar:

- SCCS (Source Code Control System)
  - Control básico de versiones, original de UNIX
- RCS (Revision Control System)
  - Herramienta similar, parte del proyecto GNU

En realidad estas herramientas ya no se usan, y en su lugar normalmente se trabaja con sistemas de control de configuración, que manejan conjuntos de ficheros y no cada uno por separado.

4.7 Ejemplo: herramienta RCS



- El archivo "*fichero*,v" almacena todas las versiones de "*fichero*"
- Los operaciones principales son `ci` (*check-in*) y `co` (*check-out*)
- Hay otras operaciones, como `rlog` (historia de cambios)

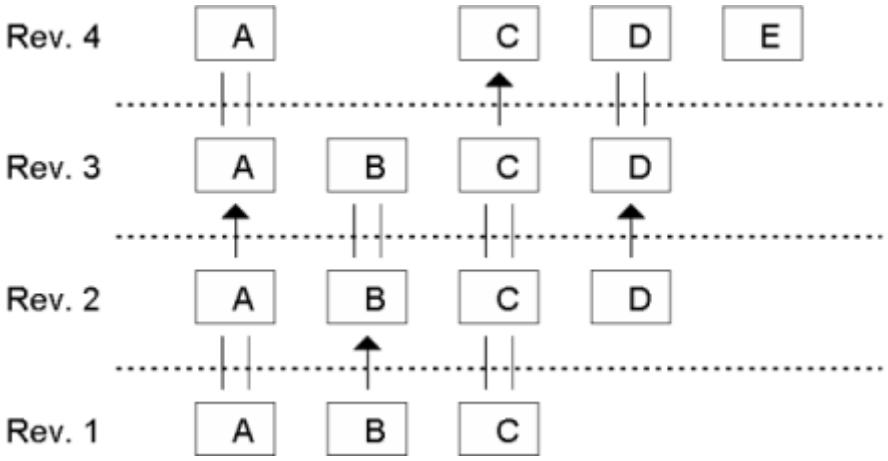
5. Control de configuración

Como se ha dicho antes, aplicaremos esta denominación al control de la evolución de un conjunto de elementos.

- La evolución del sistema consiste en:
  - Añadir componentes
  - Suprimir componentes
  - Modificar componentes
  - Reorganizar la estructura
- Evolución temporal: *revisiones*
  - Son cambios a lo largo del tiempo
- Evolución espacial: *variantes*
  - Son versiones (configuraciones) simultáneas

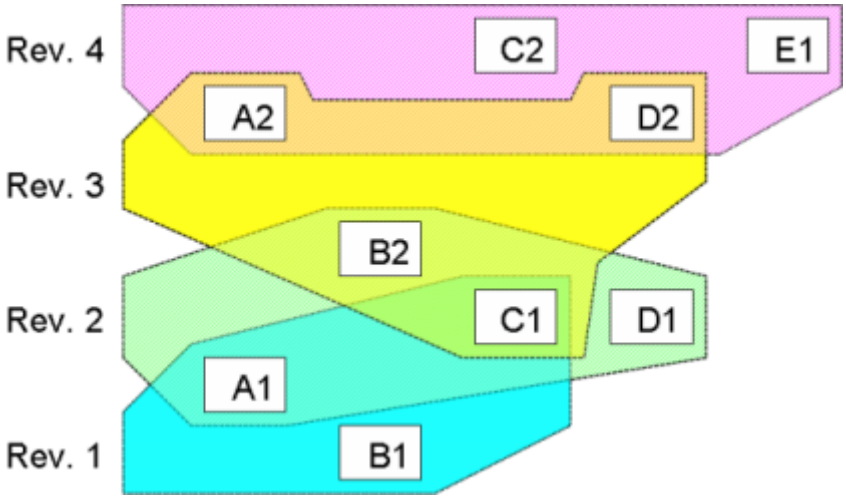
5.1 Ejemplo de evolución de una configuración

Se presenta un ejemplo de evolución simple (secuencia temporal lineal). Las revisiones del conjunto se han numerado correlativamente (Rev.1, Rev.2, ...). Cada configuración contiene una colección de elementos, no siempre los mismos. Pueden crearse o eliminarse elementos entre una revisión y la siguiente. Los cambios individuales de un componente se indican con flechas. Los componentes que se mantienen sin cambios se marcan con dos líneas paralelas (como el signo = en vertical).



5.2 Problema de coherencia de versiones

La primera dificultad del control de configuración, respecto al control de versiones, es cómo nombrar las versiones de los componentes individuales. Si se numeran las versiones de los componentes con independencia de la evolución del conjunto tendríamos lo siguiente:



O bien, dibujando repetidas las versiones que se mantienen sin cambios en cada revisión del conjunto:

Como puede verse la numeración de las versiones individuales no tiene una relación sencilla con la numeración de las versiones del conjunto. Hace falta mantener una tabla o índice que asocie cada versión del conjunto con las versiones individuales de sus componentes.

5.3 Herramientas de control de configuración

Hay diversos ejemplos de herramientas de software libre para control de configuración:

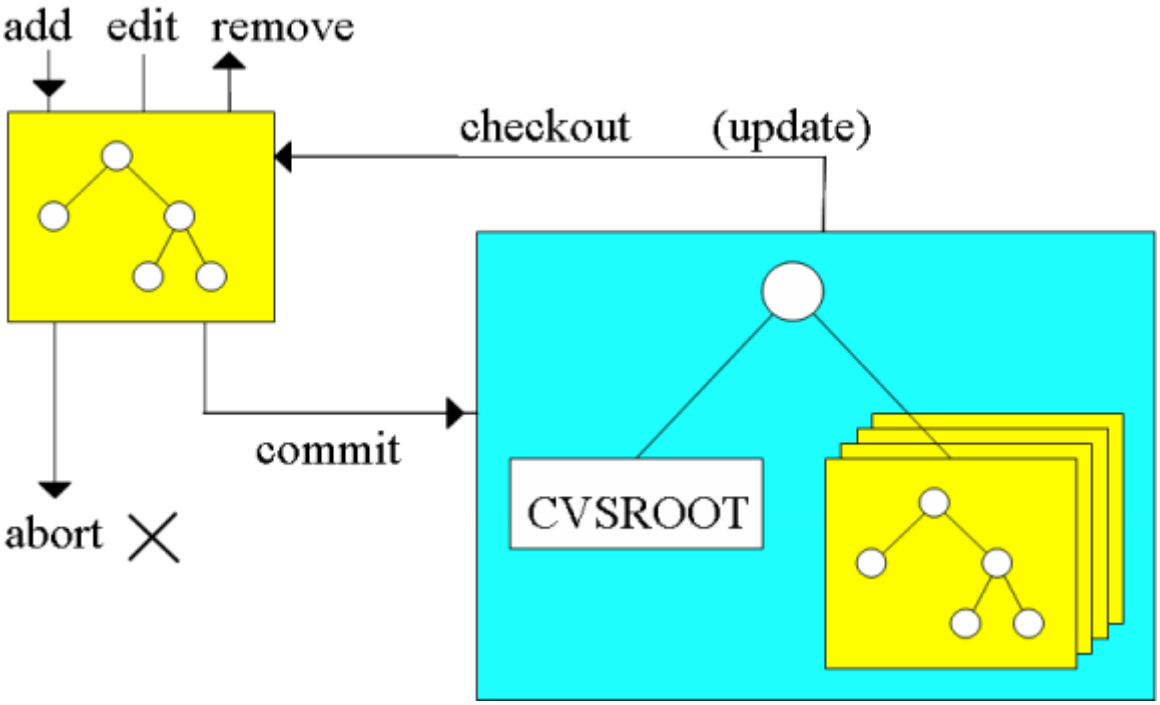
- CVS (Concurrent Version System):
  - Control de configuración, con cambios simultáneos.
  - La más antigua, usada todavía.
- Subversion:
  - Similar a la anterior, más moderna.
  - Permite trazar renombrado o movimiento de ficheros.
  - Ha desplazado a CVS.
- Gnuarch, Bazaar, etc.:
  - Su implantación es irregular. De momento no compiten con las anteriores.

- Git:
  - Es la herramienta de moda en la actualidad. Más compleja que las anteriores.
  - Facilita la creación y fusión de ramas (*branch-merge*).

5.4 Ejemplo: herramienta CVS

La figura muestra las órdenes e intercambio de datos entre:

- Un directorio de trabajo (a la izquierda), y
- El repositorio (a la derecha)
- Puede haber varias copias de trabajo simultáneas, conectadas al mismo repositorio.



Las órdenes principales son:

- Sobre la configuración en su conjunto:
  - *checkout*: obtiene una copia de trabajo para operar con ella
  - *update*: actualiza la copia con cambios recientes en el repositorio
  - *commit*: almacena la copia modificada en el repositorio
  - *abort*: abandona los cambios en la copia de trabajo
- Sobre ficheros individuales:
  - *add*: añade nuevos ficheros a la lista de la configuración
  - *remove*: elimina algunos ficheros de la lista de la configuración
  - *edit*: autoriza modificaciones en un fichero (si el *checkout* se hizo en modo sólo lectura)

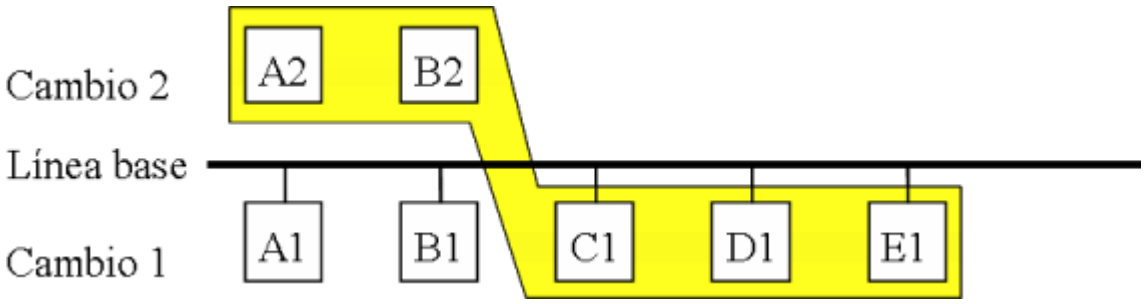
6. Desarrollo mediante cambios sucesivos

Las herramientas de control de configuración facilitan desarrollar software de manera evolutiva, mediante cambios sucesivos aplicados a partir de una configuración inicial (que puede ser vacía) hasta llegar a una versión final aceptable del producto. En lo que sigue se planteará el desarrollo como una evolución simple (secuencia de revisiones) de la línea base. En la práctica suele ser necesario contemplar también la gestión de variantes.

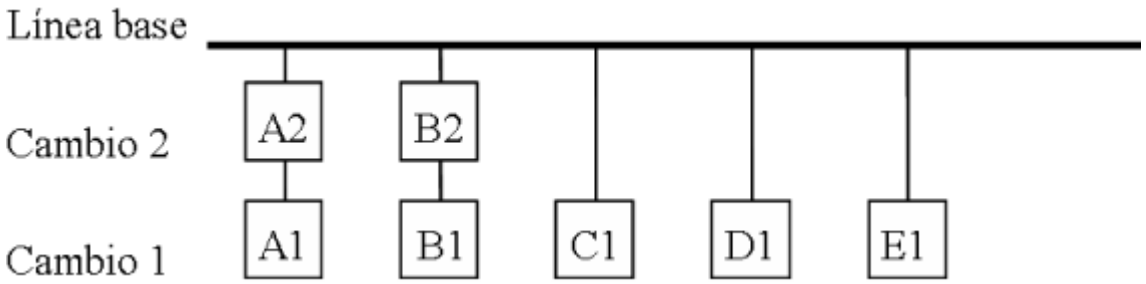
6.1 Cambios sucesivos, no simultáneos

Si los cambios se realizan estrictamente uno tras otro, entonces no hay ningún problema para realizarlos. En el siguiente ejemplo se realizan cambios sobre copias de trabajo de ciertos componentes, que luego se almacenan como parte del repositorio, actualizando la línea base. Partiremos de la situación en que el primer cambio ha generado ya una línea base no vacía.

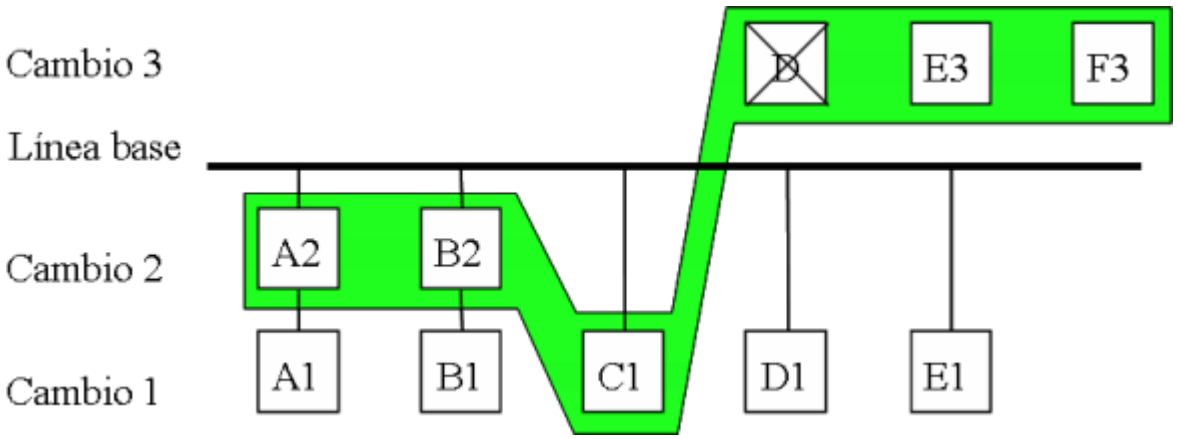
- Desarrollo del Cambio 2: se modifican A y B



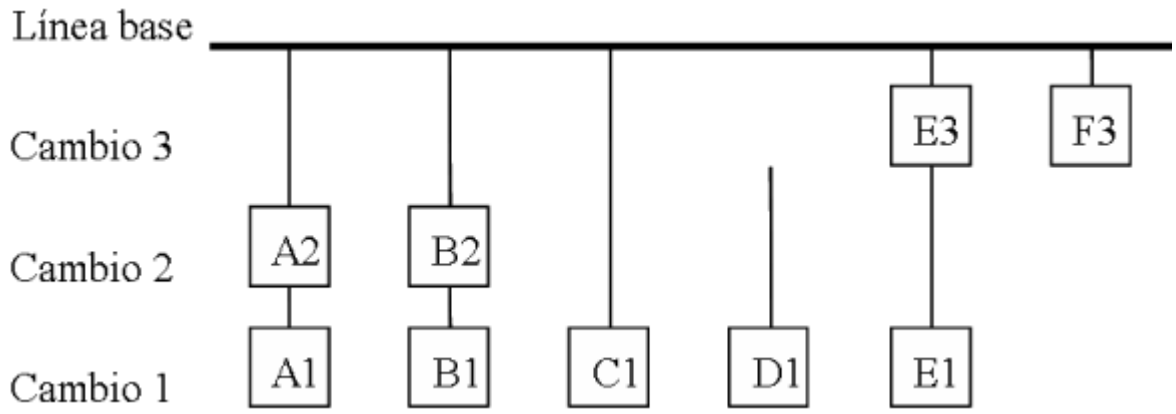
- Integración del Cambio 2 en el repositorio



- Desarrollo del Cambio 3: se elimina D, se modifica E y se añade F



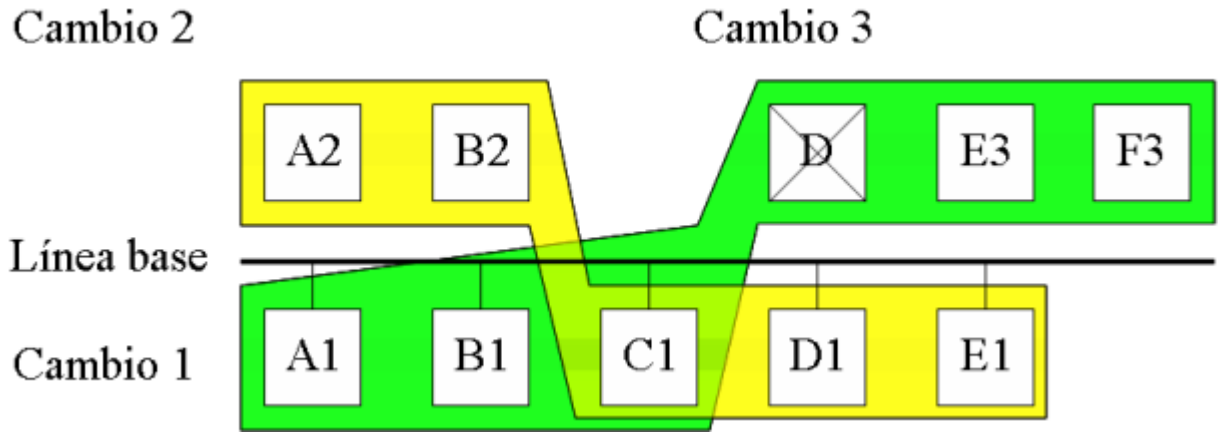
- Integración del Cambio 3



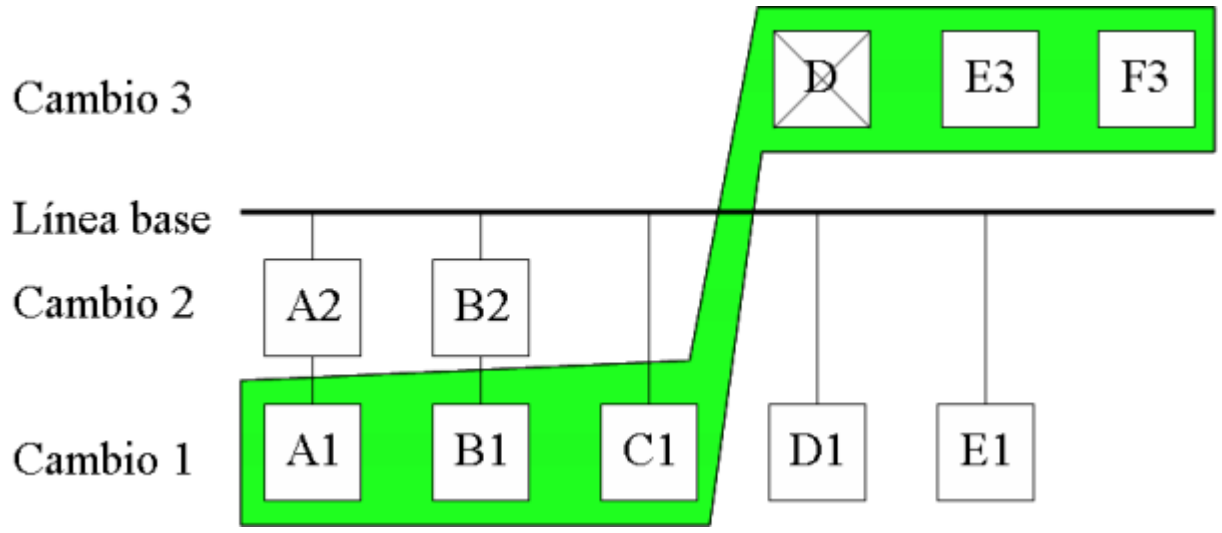
6.2 Desarrollo simultáneo de cambios

En la práctica no es posible esperar a que termine un cambio para empezar el siguiente. El trabajo en equipo exige desarrollar simultáneamente más de un cambio a la vez. Para evitar complicaciones lo que sí se suele hacer es integrar los cambios en el repositorio de uno en uno según se van completando, con independencia de cuándo se haya iniciado cada uno.

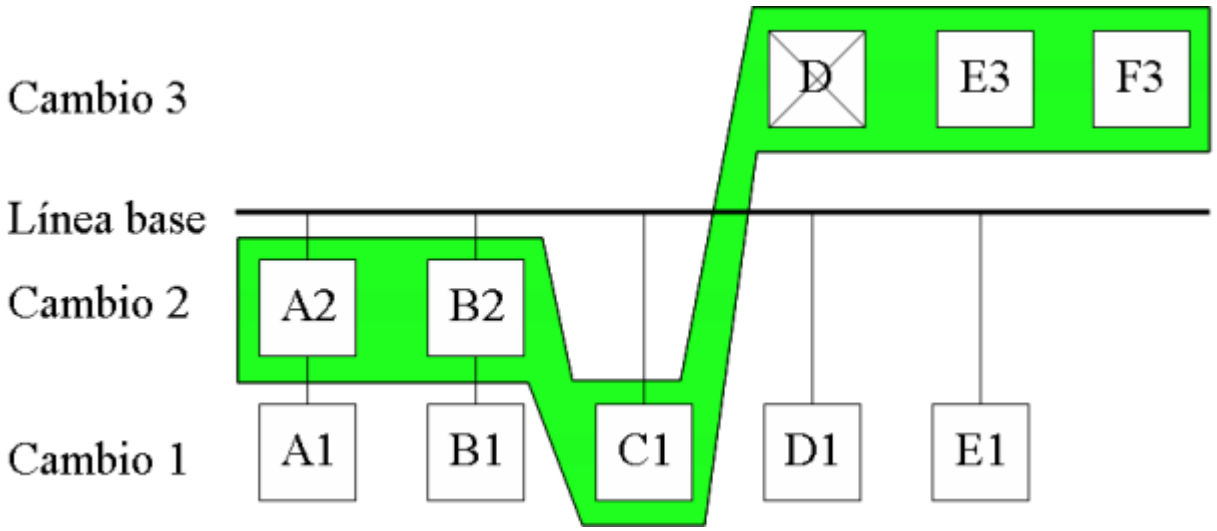
- Cambios 2 y 3 en desarrollo



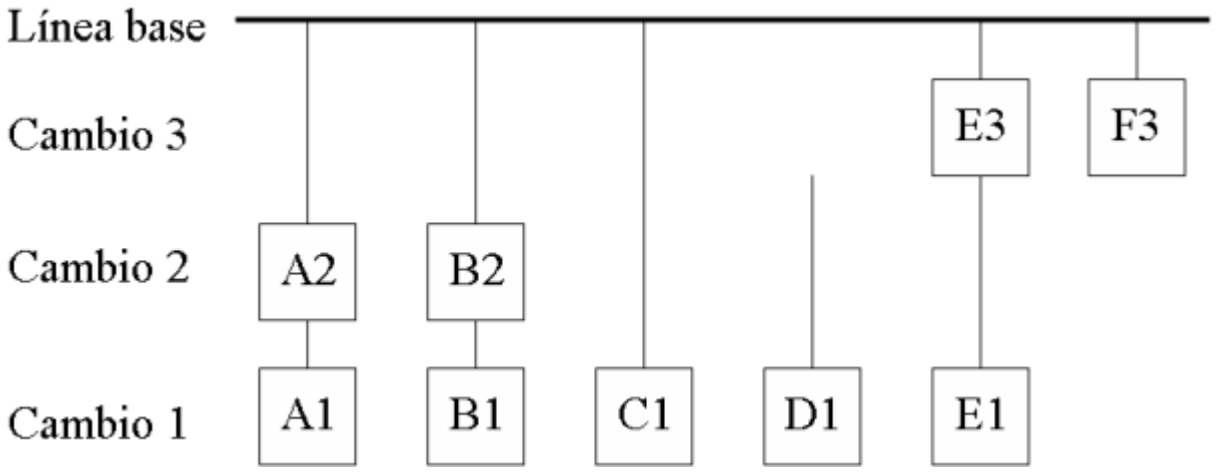
- Integración del Cambio 2



- Actualización del Cambio 3



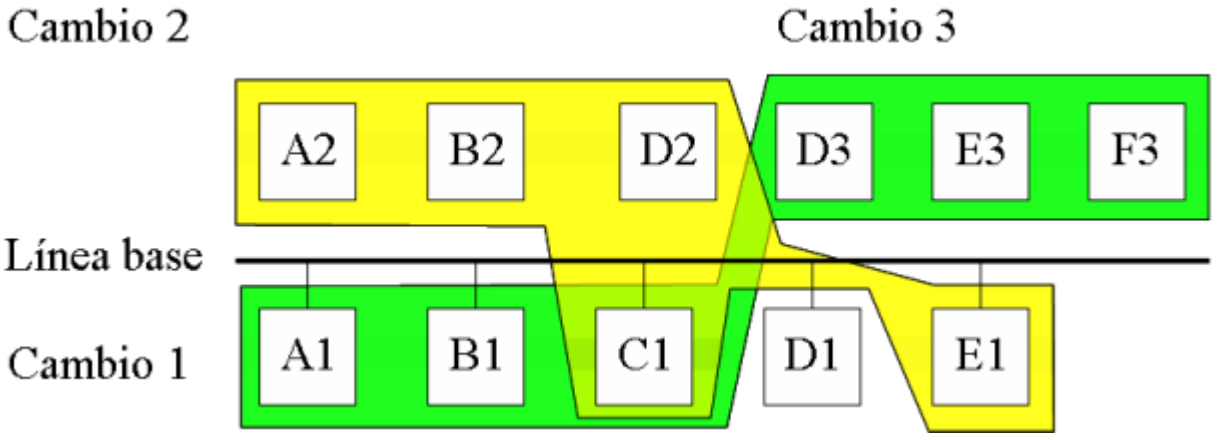
- Integración del Cambio 3



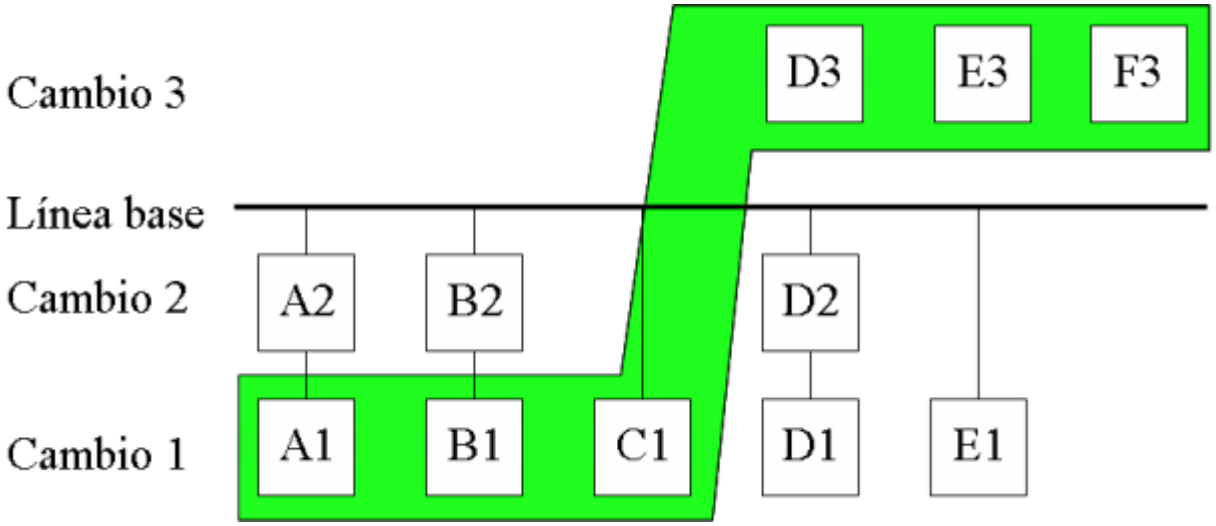
6.3 Cambios simultáneos de un mismo componente

Las cosas se complican cuando dos cambios simultáneos modifican un mismo componente. Tras integrar el primer cambio hay que propagar las modificaciones del componente común al otro cambio, todavía en desarrollo.

- Desarrollo de los Cambios 2 y 3

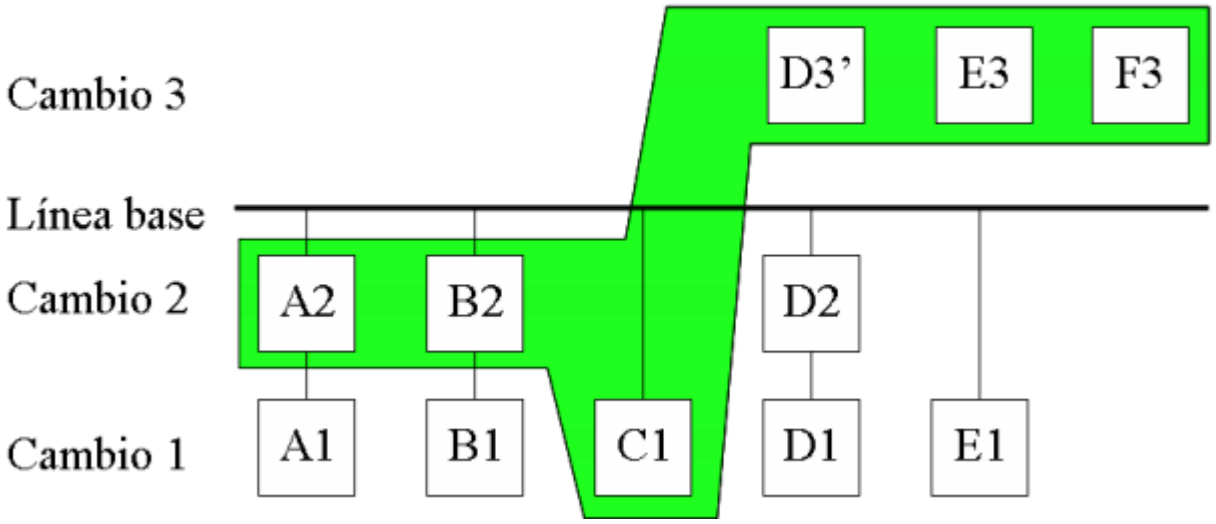


- Integración del Cambio 2

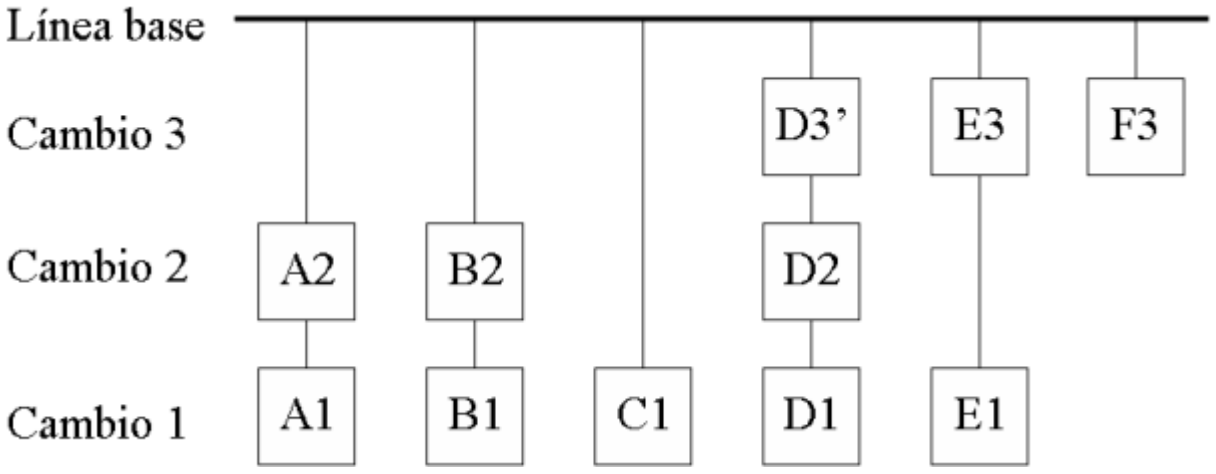


- Actualización del Cambio 3: mezcla de cambios en el componente D

D3' = D3 + D2 - D1



- Integración del Cambio 3

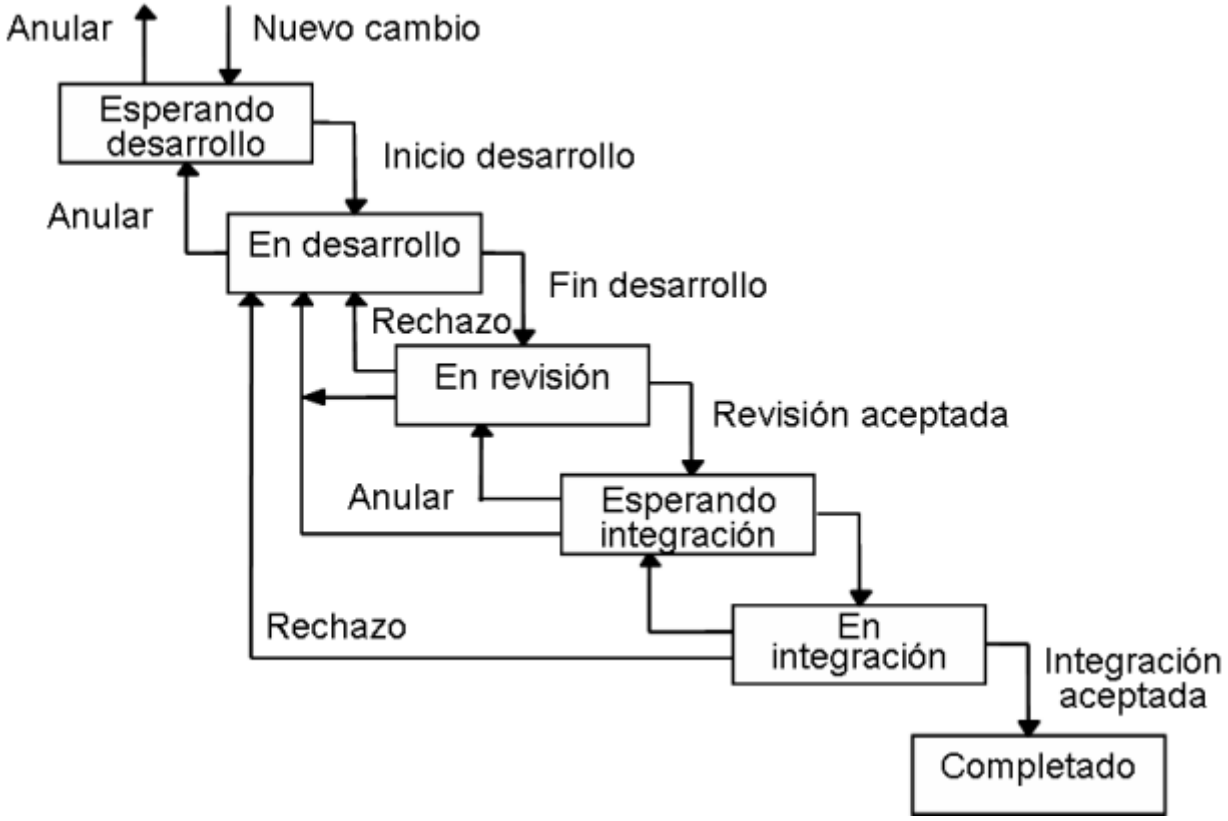


7. Control de cambios

La ingeniería de software recomienda realizar el desarrollo de manera disciplinada. Las herramientas de control de versiones no garantizan un desarrollo razonable si cualquier miembro del equipo puede realizar los cambios que quiera e integrarlos en el repositorio sin ningún tipo de control.

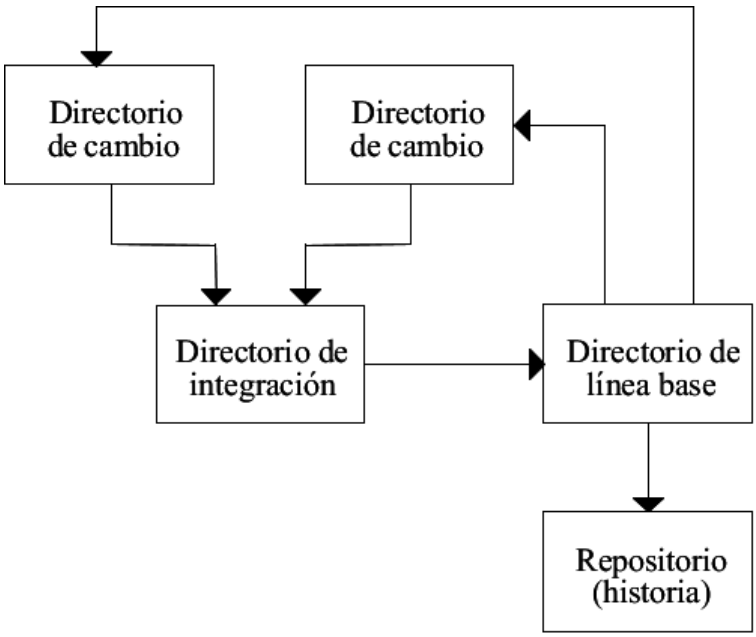
7.1 Ciclo de vida de cambios

Para garantizar que siempre disponemos de una línea base adecuada para continuar el desarrollo es necesarios aplicar controles al desarrollo e integración de cambios. El siguientes esquema muestra el ciclo de vida de cambios soportado por la herramienta Aegis, que fuerza esta disciplina de desarrollo.



7.2 Ejemplo de herramienta: Aegis

A continuación se presenta un esquema de organización del trabajo bajo la herramienta Aegis, usando directorios separados para cada parte de la actividad de cambio.


Directorio de cambio
Directorio de cambio
Directorio de integración
Directorio de línea base
Repositorio (historia)