



GESTIÓN DEL SOFTWARE

# DEFINICIÓN Y APROBACIÓN DE LÍNEAS BASE EN PROYECTOS DE SOFTWARE

# DEFINICIÓN Y APROBACIÓN DE LÍNEAS BASE EN PROYECTOS DE SOFTWARE

La línea base es un hito clave en la gestión de la configuración de proyectos de software. Su función es establecer un conjunto de artefactos aprobados que sirvan como punto de referencia estable para el desarrollo, prueba y mantenimiento del sistema. Este proceso permite controlar cambios, asegurar la trazabilidad y mantener la integridad de los entregables en cada etapa del ciclo de vida. El presente documento describe paso a paso cómo identificar los elementos a incluir, verificar su consistencia, gestionar versiones, obtener la aprobación formal y mantener la línea base mediante herramientas especializadas y buenas prácticas.


## Proceso para la definición y aprobación de líneas base

El establecimiento de una línea base en un proyecto de software es una práctica esencial dentro de la gestión de la configuración, ya que define puntos de control que permiten evaluar y validar el avance del proyecto con base en artefactos estables y aprobados. Este proceso sigue una serie de etapas claramente definidas que garantizan que sólo los elementos verificados, trazables y consensuados sean congelados como referencia (Pérez Martínez, 2015).

A continuación se describe cada una de las etapas clave del proceso de definición y aprobación de una línea base:


### 1. Identificación de los elementos a incluir

El proceso comienza con la selección de los artefactos que se someterán a línea base. Estos pueden ser requisitos, modelos de diseño, documentos técnicos, código fuente, scripts de prueba, entre otros. La elección depende del momento del ciclo de vida del software y de los objetivos específicos del control (Guillamón Morales, 2013).

 **Ejemplo:** Durante la fase de análisis, se identifican el documento de especificación de requisitos, el modelo de casos de uso y el diccionario de datos como candidatos para la línea base de requisitos.

### 2. Verificación de integridad y consistencia

Una vez identificados los elementos, se realiza una verificación formal para asegurar que estén completos, libres de ambigüedades, y alineados entre sí. Esta etapa puede involucrar revisiones por pares, pruebas de consistencia o validaciones cruzadas (Chacon & Straub, 2014).

 **Ejemplo:** El equipo de QA revisa que cada requisito está vinculado a un caso de prueba correspondiente antes de permitir su inclusión en la línea base.

### 3. Documentación y control de versiones

Los artefactos deben ser correctamente documentados y gestionados a través de un sistema de control de versiones que permita identificar claramente su estado, historial de cambios y responsables. Esto proporciona trazabilidad y asegura que cualquier modificación posterior pueda ser registrada (Guillamón Morales, 2013).

✎ **Ejemplo:** El diseño arquitectónico se sube al repositorio de documentación bajo la versión 1.0, con comentarios sobre las decisiones técnicas tomadas y validaciones realizadas.

#### 4. Solicitud de revisión formal

El equipo responsable de la configuración solicita una revisión oficial por parte de los interesados clave: líderes técnicos, responsables de calidad, usuarios representantes y gestores de proyecto. Esta revisión tiene como objetivo garantizar que los elementos cumplen con los criterios predefinidos de inclusión.

✎ **Ejemplo:** Antes de congelar la línea base de pruebas, se agenda una reunión con el área de control de calidad y el cliente para revisar los criterios de aceptación de cada caso de prueba.

#### 5. Aprobación y registro

Una vez realizada la revisión, los elementos son aprobados formalmente mediante firmas físicas o electrónicas, y registrados en un repositorio o sistema de gestión de configuración. En este punto, la línea base se considera oficial y no puede ser modificada sin pasar por un proceso de control de cambios.

✎ **Ejemplo:** El comité de configuración firma electrónicamente el acta de aprobación de la línea base de requisitos funcionales, quedando estos congelados para su desarrollo posterior.

#### 6. Comunicación a los equipos implicados

Finalmente, se comunica la aprobación de la línea base a todos los miembros del proyecto, asegurando que trabajen con los mismos artefactos y versiones. Esta etapa es vital para prevenir desviaciones o confusiones durante el desarrollo.

✎ **Ejemplo:** El líder técnico envía un comunicado por correo y actualiza la wiki del proyecto indicando que la versión 2.0 del código fuente ha sido establecida como línea base para la próxima fase de pruebas.

### Herramientas de soporte para la gestión de líneas base

La gestión de líneas base dentro de un proyecto de software requiere de herramientas especializadas que permitan controlar, registrar y supervisar los artefactos versionados de forma segura, trazable y eficiente. Estas herramientas actúan como soporte esencial para garantizar que los elementos configurados permanezcan estables una vez aprobados, y que cualquier cambio posterior sea correctamente gestionado bajo criterios formales.

#### 1. Sistemas de control de versiones

Los sistemas de control de versiones son fundamentales para gestionar líneas base. Permiten almacenar versiones de archivos, mantener un historial de cambios y generar marcas (tags) para congelar una versión específica que representa la línea base (Chacon & Straub, 2014).

✎ **Ejemplo:** Git, en conjunto con plataformas como GitHub o GitLab, permite crear tags para marcar versiones aprobadas del código fuente. La versión v1.0 puede usarse como línea base para el inicio de pruebas de integración.

## 2. Herramientas de gestión de configuración

Estas herramientas proporcionan funcionalidades específicas para identificar, controlar, auditar y reportar elementos de configuración. Permiten asociar documentos, versiones, aprobaciones y cambios en un mismo entorno.

✎ **Ejemplo:** IBM Engineering Workflow Management (anteriormente RTC) permite definir líneas base de requisitos, diseño y código dentro de un entorno de desarrollo colaborativo, manteniendo relaciones entre artefactos.

## 3. Plataformas de gestión del ciclo de vida de aplicaciones (ALM)

Las herramientas ALM integran funciones de gestión de requisitos, pruebas, versiones y configuración. Ayudan a establecer líneas base en distintas fases del proyecto y facilitan el seguimiento desde la planificación hasta la entrega.

✎ **Ejemplo:** Polarion ALM permite definir líneas base en documentos de especificación de requisitos y trazarlas hacia los casos de prueba, integrando visibilidad total del avance del proyecto.

## 4. Sistemas de documentación colaborativa con control de cambios

En proyectos donde se gestionan documentos técnicos como artefactos de línea base, es común el uso de plataformas que permitan edición colaborativa, seguimiento de versiones y aprobación formal de documentos.

✎ **Ejemplo:** Confluence, integrado con Jira, permite generar versiones de documentos funcionales y establecer controles para marcarlos como “aprobados” y parte de una línea base específica.

## 5. Herramientas de automatización de entregas (CI/CD)

Estas herramientas permiten establecer líneas base no solo a nivel de código, sino también de entregables ejecutables, integrando automáticamente el versionado de compilaciones estables.

✎ **Ejemplo:** Jenkins permite etiquetar una compilación como estable tras pasar exitosamente las pruebas, estableciendo esa versión del artefacto como la línea base de despliegue.

## 6. Repositorios centralizados de configuración

Algunas herramientas están diseñadas para centralizar el almacenamiento y control de todos los elementos de configuración, incluyendo código, documentación, configuraciones y librerías externas (Pérez Martínez, 2015).

✎ **Ejemplo:** Apache Subversion (SVN), aunque más tradicional, permite crear snapshots de directorios completos, sirviendo como línea base de un módulo completo del sistema.


Las herramientas de soporte para la gestión de líneas base desempeñan un papel crucial en la integridad y trazabilidad de los proyectos de software. Facilitan el control riguroso de versiones, promueven la colaboración entre equipos y reducen riesgos asociados a cambios no autorizados. La selección adecuada de estas herramientas debe responder a las necesidades específicas del proyecto, su tamaño, nivel de criticidad y grado de madurez en prácticas de gestión de configuración.

## Seguimiento y mantenimiento de líneas base

El seguimiento y mantenimiento de líneas base constituye una actividad crítica dentro de la gestión de configuración de software, orientada a preservar la integridad y consistencia de los componentes aprobados de un sistema a lo largo de su evolución. Una línea base no es un punto final, sino un hito controlado que debe ser monitoreado constantemente para detectar desviaciones, aplicar cambios autorizados y garantizar que el sistema permanezca alineado con los objetivos técnicos y funcionales definidos (Pérez Martínez, 2015).


### 1. Propósito del seguimiento de líneas base

El seguimiento permite verificar que todos los elementos que conforman la línea base (documentos, código, requisitos, entregables) se mantengan estables, trazables y sin alteraciones indebidas. Se aplican mecanismos de auditoría, inspección y control para asegurar la integridad de los artefactos, tanto en entornos de desarrollo como en producción.

 **Ejemplo:** Un equipo de desarrollo revisa semanalmente si la versión del código marcado como “Release 1.0” se ha modificado de forma no autorizada, usando herramientas como Git para comparar el estado actual con la línea base registrada.


### 2. Control de modificaciones posteriores

Aunque las líneas base son puntos de referencia estables, en la práctica pueden requerir ajustes por errores detectados o cambios evolutivos. Para ello, se implementan procesos formales de gestión de cambios que regulan cualquier modificación posterior a la aprobación de una línea base.

 **Ejemplo:** Si se detecta una vulnerabilidad en un módulo aprobado en la línea base, se emite una solicitud de cambio, se evalúa el impacto, se aprueba formalmente y se actualiza la línea base con la versión corregida.


### 3. Trazabilidad de versiones

Mantener un registro claro de todas las versiones que derivan de una línea base es fundamental. Esto permite comparar el estado actual del producto con versiones anteriores, analizar regresiones y asegurar que las mejoras o correcciones aplicadas estén documentadas adecuadamente.

 **Ejemplo:** Una herramienta como Jira, enlazada con Git, permite trazar cada cambio en el código fuente con una tarea específica del backlog, facilitando la comparación entre la línea base original y la versión desplegada.


#### 4. Auditorías y validaciones periódicas

El mantenimiento implica la realización de auditorías para comprobar que el estado de los componentes sigue alineado con lo que se aprobó inicialmente. Estas auditorías pueden ser internas o externas, y deben validar tanto los contenidos como las relaciones entre los elementos de la línea base.

 **Ejemplo:** En un proyecto regulado por normas de calidad como ISO 12207, se realiza una auditoría trimestral que revisa si los requisitos aprobados en la línea base inicial coinciden con la funcionalidad implementada.


#### 5. Revisión y actualización controlada de líneas base

Cuando se requieren cambios significativos o una nueva fase del proyecto inicia, es posible establecer una nueva línea base. Este proceso debe realizarse de forma controlada, asegurando que las versiones previas queden correctamente archivadas y que se mantenga la trazabilidad histórica.




 **Ejemplo:** Tras el cierre de la fase de pruebas del sistema, el equipo define una nueva línea base para la versión candidata a producción, incorporando todos los cambios validados y rechazando aquellos no autorizados.

#### 6. Herramientas de soporte al mantenimiento

Plataformas como Git, Subversion, IBM EWM o Azure DevOps facilitan el mantenimiento de líneas base al ofrecer control de versiones, etiquetado, seguimiento de cambios, historial de revisiones y soporte para trazabilidad.

 **Ejemplo:** En un entorno DevOps, Jenkins se configura para generar automáticamente reportes de compilación basados en la línea base activa, indicando si se han introducido modificaciones no autorizadas en el pipeline de integración continua.

### BIBLIOGRAFÍA

-  Chacon, S., & Straub, B. (2014). Pro Git (2.<sup>a</sup> ed.). Apress.  
<https://git-scm.com/book/es/v2>
-  Guillamón Morales, A. (2013). Manual desarrollo de elementos software para gestión de sistemas. Editorial CEP.  
<https://elibro.net/es/lc/tecnologicadeloriente/titulos/50603>
-  Pérez Martínez, E. (2015). Desarrollo de aplicaciones mediante el Framework de Spring. RA-MA Editorial.  
<https://elibro.net/es/lc/tecnologicadeloriente/titulos/107207>