Tema 8: Despliegue, Evolución y Mantenimiento del Software

BLOQUE V: MANTENIMIENTO DE LOS SISTEMAS SOFTWARE

Ingeniería del Software

Grado en Ingeniería Informática

Curso 2024/2025

Índice

Índice

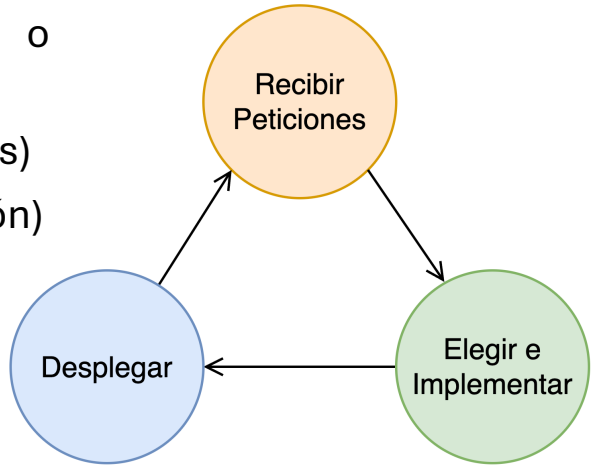
Introducción - ¿Qué sucede después de las pruebas?

- Una vez que el software ha sido desarrollado y probado exhaustivamente, el trabajo no termina
- Este es el inicio de actividades clave del ciclo de vida del software para garantizar su utilidad y sostenibilidad:
  1. Puesta en marcha: incluye la instalación, configuración en el entorno operativo y la capacitación de usuarios o administradores
  2. Incorporación de nuevas funcionalidades: desarrollo de características adicionales y mejora de las existentes
  3. Corrección de errores no detectados: solución de fallos derivados de casos imprevistos o limitaciones en las pruebas iniciales
  4. Adaptación a nuevos entornos: ajustes para garantizar la compatibilidad con tecnologías emergentes o cambios en la infraestructura
  5. Readaptación por cambios del negocio: adaptación a nuevos procesos, normativas o expectativas del mercado
- Todas estas actividades forman parte del ciclo de vida del software y deben abordarse adecuadamente con:
  - Metodologías que faciliten la labor de los ingenieros
  - Herramientas de soporte para la gestión y automatización de tareas

Índice

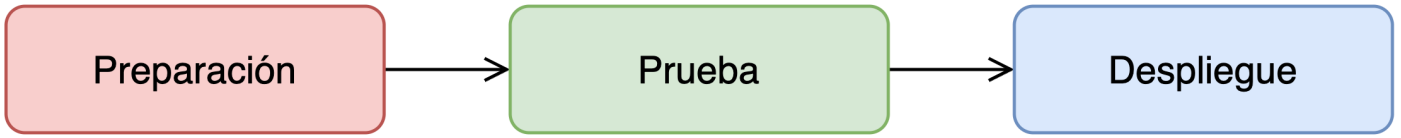
Proceso de Despliegue

- El despliegue consiste en poner el software o sus actualizaciones a disposición de los usuarios en un entorno real. Este proceso incluye:
  - Preparación y lanzamiento de versiones
  - Configuración e instalación del sistema
  - Monitoreo del rendimiento tras la puesta en marcha
- Las técnicas de despliegue combinan procesos manuales y automáticos para facilitar la distribución eficiente de actualizaciones y parches
- El despliegue consta de tres pasos:
  1. Recibir peticiones (pull requests o actualizaciones)
  2. Elegir e implementar peticiones (o aceptarlas)
  3. Hacer los cambios disponibles (nueva versión)



Proceso de Despliegue

- El despliegue varía según el sistema y su entorno operativo. Cada organización debe establecer un flujo de trabajo que incluya actividades, prácticas y herramientas adaptadas a sus necesidades
- Estados esenciales en el despliegue
  1. Preparación
     - Recopilación del código y recursos necesarios (configuraciones, librerías, etc) para la integración
  2. Prueba
     - Evaluación del código en un entorno de pruebas (*staging*)
     - Uso de pruebas de regresión para prevenir errores
  3. Despliegue
     - Integración en el entorno de producción
     - Disponibilidad de nuevas funcionalidades junto con las previas



Proceso de Despliegue

- Beneficios de un proceso estructurado de despliegue
  - Incorporación segura de funcionalidades: Responde rápidamente a las necesidades del negocio y asegura una evolución positiva del sistema
  - Optimización de operaciones: Mejora procesos y aumenta la eficiencia general de la organización
  - Monitoreo y mejora continua: Permite recolectar datos valiosos para ajustar y perfeccionar el despliegue
  - Automatización eficiente: Acelera las pruebas, hace los cambios transparentes y aumenta la confianza en el sistema
- Despliegue Continuo (CD): Sistema automatizado para desplegar nuevas funcionalidades con alta frecuencia
- Integración Continua (CI): Cambios en el código integrados continuamente en la rama principal
- Es habitual combinarlos en el modelo CI/CD

Proceso de Despliegue

- Prácticas recomendadas para CI/CD
  - Mantener un repositorio único de código
  - Automatizar la construcción (*build*)
  - Pruebas automáticas
  - Todo el mundo hace commits diarios a la rama principal
  - Los commits deben ser correctamente integrados en la versión actual
  - Todo “bug” corregido debe ir acompañado del caso de prueba que lo encontró
  - El proceso de construcción debe permanecer rápido
  - Haz pruebas en un entorno específico (staging)
  - Acceso rápido a nuevas funcionalidades para facilitar las pruebas
  - Todo el mundo debe poder ver si la construcción falla y por qué

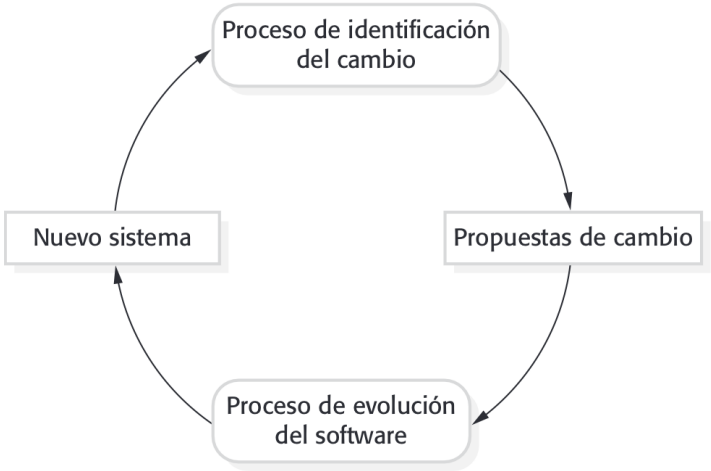
Índice

Evolución del Software

- La evolución del software es un proceso inevitable y esencial para mantener su relevancia y utilidad en un entorno dinámico
- Los cambios pueden ser debido a:
  - Nuevas situaciones en el negocio
  - Cambios en las expectativas de los clientes (nuevos requisitos)
  - Necesidad de adaptarse a nuevo hardware o software
  - Mejoras en el rendimiento o de otros aspectos no funcionales (seguridad, escalabilidad..)
- La frecuencia de versiones se ha incrementado notablemente:
  - Antes: lanzamientos cada varios años
  - Ahora: frecuencias de semanas, días o incluso diarias (CI/CD)

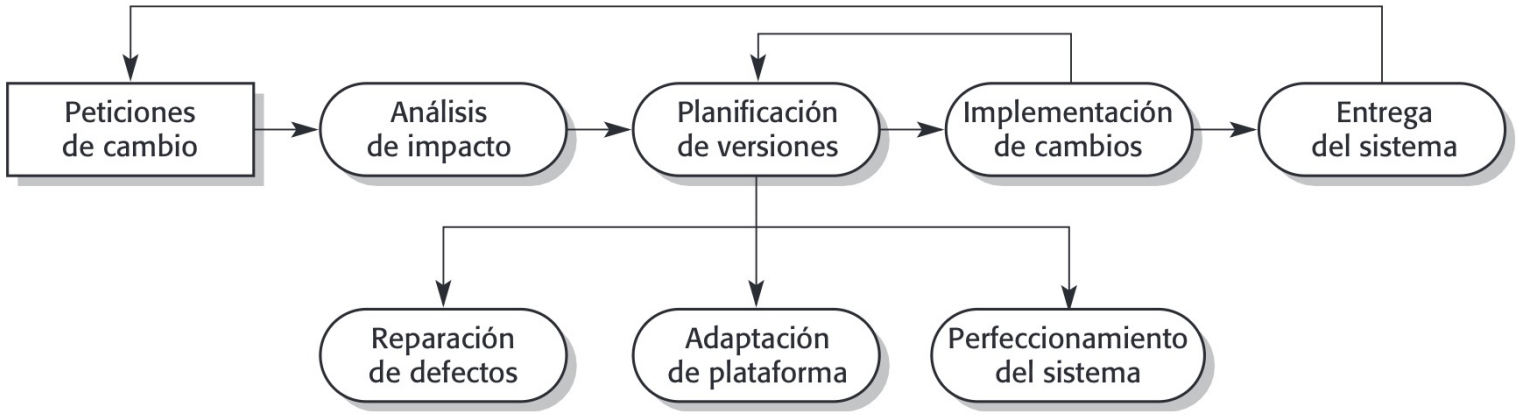
Evolución del Software

- Es muy importancia en sistemas empresariales, donde cambios en un componente pueden afectar significativamente otros sistemas interconectados
- El nivel de formalidad del proceso depende del tipo de sistema:
    1. Identificación de cambios: clientes, usuarios o el equipo de desarrollo proponen modificaciones necesarias
    2. Propuesta de cambios: las solicitudes son evaluadas para determinar su viabilidad e impacto
    3. Implementación de modificaciones: el equipo desarrolla y prueba los cambios aprobados
    4. Generación de nueva versión



Evolución del Software

- Una vez aceptada la propuesta de cambios:
  - Se deben identificar las partes del sistema que se verán afectadas
  - Se debe analizar el coste y el impacto del cambio
  - Se planifica la nueva versión con los cambios a considerar (reparación de defectos, adaptaciones y nuevas funcionalidades)
  - Se implementan los cambios y se valida la nueva versión del sistema
  - Se entrega la nueva versión del sistema
- En situaciones como vulnerabilidades de seguridad o fallos graves, el proceso puede simplificarse para acelerar los cambios

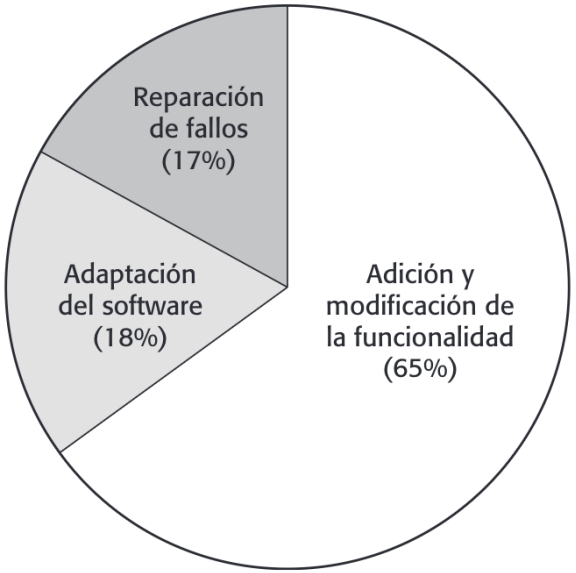


Índice

Mantenimiento del Software

- El mantenimiento comprende los procesos para realizar cambios en un sistema tras su entrega. Es crucial en sistemas a medida, donde los equipos de desarrollo y mantenimiento suelen ser distintos
- Los tipos de cambios en el mantenimiento pueden ser:
  - Correcciones simples: Solución de errores en el código
  - Modificaciones profundas: Rediseños o incorporación de nuevos requisitos

- Existen una serie de actividades comunes:
  - Reparación de fallos para corregir errores y vulnerabilidades
  - Adaptación a nuevas plataformas o entornos
  - Incorporación de nuevas funcionalidades o soporte para nuevos requisitos



Mantenimiento del Software

Tipos de Mantenimiento

1. Mantenimiento Correctivo
   - Corregir errores o fallos (como bugs o vulnerabilidades) después del despliegue, asegurando que el sistema siga funcionando correctamente
2. Mantenimiento Adaptativo
   - Modificar el software para garantizar su compatibilidad con nuevos entornos, plataformas o tecnologías, como actualizaciones de sistemas operativos o hardware
3. Mantenimiento Perfectivo
   - Mejorar el rendimiento o agregar nuevas funcionalidades en respuesta a cambios en los requisitos del negocio o la demanda de nuevas características
4. Mantenimiento Preventivo
   - Anticipar problemas futuros y evitar errores, mediante refactorización de código, mejora de la documentación y revisión de la arquitectura para mantener la eficiencia y sostenibilidad

Mantenimiento del Software

Costes del Mantenimiento

- Añadir nuevas funcionalidades es más costoso que reparar fallos
- Los cambios pueden requerir análisis y rediseño
- El equipo de mantenimiento debe entender bien el sistema
- El código original puede ser difícil de mantener, aumentando los costes

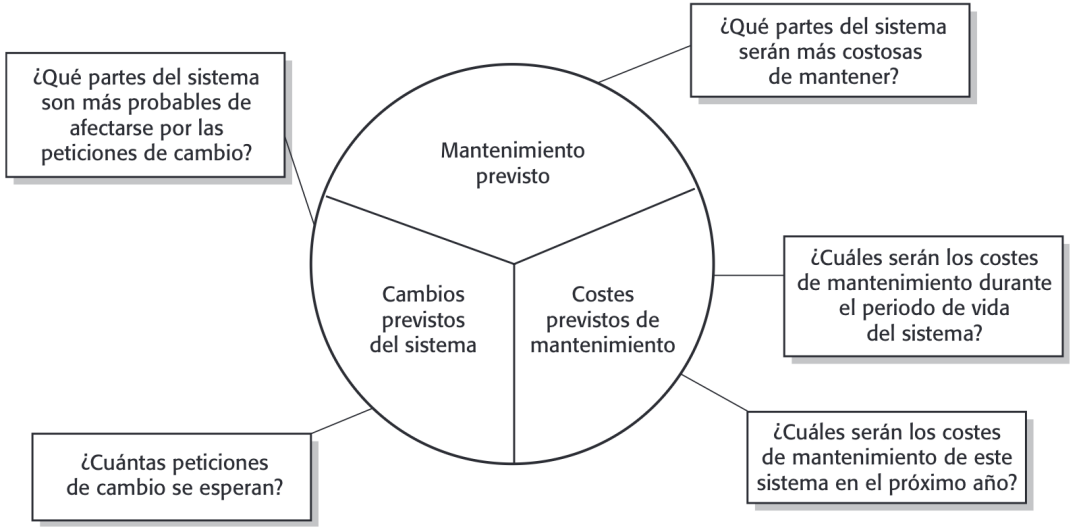
Desafíos del Mantenimiento del Software

- El mantenimiento se ve como menos creativo, y a menudo lo hace personal menos experimentado
- Las tecnologías obsoletas complican el proceso
- A medida que el sistema crece, se vuelve más difícil de modificar
- La documentación desactualizada dificulta las modificaciones

Mantenimiento del Software

Predicción del Mantenimiento

- Anticiparse a los problemas ayuda a evitar costes inesperados
- Las predicciones clave son:
  - Aceptar o rechazar un cambio depende de la facilidad de mantenimiento de los componentes afectados
  - Los cambios pueden degradar la estructura del sistema, reduciendo su mantenibilidad a largo plazo
  - Los costes de mantenimiento aumentan con el número de cambios



Índice

Proceso de Reingeniería

- La reingeniería del software se aplica a sistemas heredados (legacy systems) que, debido a su antigüedad, son difíciles de actualizar
- Este enfoque es útil cuando realizar cambios directos sobre el sistema no es viable o reemplazarlo por completo resulta costoso o riesgoso
- Incluye una serie de beneficios:
  - Reducción del riesgo: minimiza los riesgos al modernizar un sistema existente en lugar de implementar uno nuevo
  - Reducción de costes: reutiliza componentes existentes, evitando los gastos de desarrollar un sistema desde cero
- El propósito de la reingeniería es mejorar y reestructurar el sistema, optimizando su rendimiento, adaptabilidad y sostenibilidad, según las necesidades específicas del negocio

Proceso de Reingeniería

- Las actividades del proceso de reingeniería son:
  1. Traducción de código: Migrar el código a un lenguaje más moderno y adecuado a las necesidades actuales
  2. Ingeniería inversa: Analizar el código para entender su estructura y funcionalidades, especialmente cuando falta documentación
  3. Mejora de la estructura del programa: Reorganizar y refactorizar el código para hacerlo más claro y mantenible
  4. Modularización del programa: Dividir el sistema en módulos independientes y coherentes para mejorar la flexibilidad y escalabilidad
  5. Reingeniería de datos: Optimizar las estructuras de bases de datos para mejorar el rendimiento y compatibilidad con tecnologías actuales