Tema 1: Evolución y Mantenimiento de Software

Evolución y Adaptación de Software

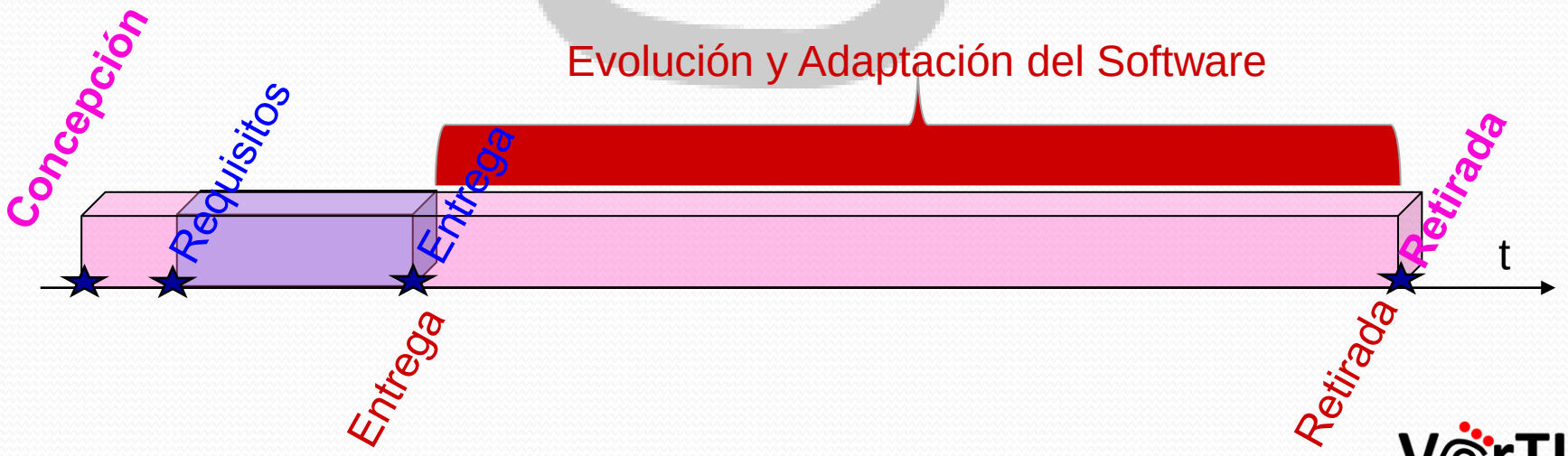
Carlos E. Cuesta, ETSII, URJC

Universidad
Rey Juan Carlos

Índice

0. Ubicación

- Ciclo de desarrollo
  - Requisitos, análisis, diseño, implementación y pruebas.
  - Desde los requisitos hasta la entrega.
- Ciclo de vida
  - Adquisición, suministro, desarrollo, explotación y mantenimiento del sw (IEEE 1074 e ISO 12207)
  - Desde la concepción hasta la retirada.
- Nuestra asignatura



1. Evolución del software: Definiciones (I)

Históricamente, el **mantenimiento** se puede considerar como la primera forma de **evolución** de los sistemas. La mayoría de las empresas que realizaron sus grandes desarrollo hace 25 años, necesitan, sobre todo si la aplicación sigue resultando efectiva, la realización de estas actividades.

Definición de mantenimiento **ANSI/IEEE**:

Las modificaciones de los productos software después de su entrega para corregir fallos, mejorar rendimiento u otros atributos, o adaptar el producto a un cambio de entorno.

Otra definición dada por **ISO/IEC**:

Un producto software soporta una modificación en el código y su documentación asociada para la solución de un problema o por la necesidad de una mejora. Su objetivo es mejorar el software existente manteniendo su integridad.

1. Evolución del software: Definiciones (II)

La **evolución** del software es la dinámica de crecimiento del software.

Una definición atribuida a Meir M. **Lehman** y Juan F. Ramil dice que la evolución del software es ***todas las actividades de programación que se orientan a generar una nueva versión de un software a partir de una versión anterior operativa***.

Chapin, N. et al. (2001) lo definió como ***la aplicación de las actividades y procesos de mantenimiento del software que generan una nueva versión operativa de un software, con una funcionalidad de usuario o propiedades cambiadas, a partir de una versión anterior […] junto con los procesos y actividades de garantía de calidad y con la gestión de esos procesos***.

De estas definiciones se desprende que **la evolución cubre el ajuste a funcionalidades adicionales.**

La guía **SWEBOK** considera que *la causa del mantenimiento está tanto en la necesidad de **cambios** como de **evolución** en el software*.

Home del SWEBOK: http://www.computer.org/portal/web/swebok

1. Evolución del software: Definiciones (III)

Los requisitos cambian durante el desarrollo y los grandes sistemas cambian a lo largo del tiempo:

Mantenimiento: proceso de mejora y optimización del software después de su entrega al usuario final, así como también la corrección y prevención de los defectos.

Evolución: todas las actividades de generación de software que se orientan a generar una nueva versión de un software a partir de una versión anterior operativa. Incluye cambios en los requisitos.

Conservación: uso continuo de medios extraordinarios para mantener operativo un sistema viejo.

2. Importancia del mantenimiento

Los sistemas grandes y complejos tienen un **largo período de vida** (ciclo de vida). **Hay que realizar cambios en el software** para corregir errores en los requisitos iniciales del sistema original y para implementar nuevos requisitos que surgen.

Existen empresas que se acercan a porcentajes del **95% de los recursos** dedicados **al mantenimiento**, con lo cual se hace imposible el desarrollo de nuevos productos software. Esta situación se conoce como ***Barrera de Mantenimiento.***

Algo de información en cuanto a costes y porcentajes:
http://mundoerp.com/blog/porque-pagar-mantenimiento-software-erp-crm-bi/

2. Importancia del mantenimiento (II)

Cuestiones asociadas al coste en la evolución del sistema/software:

- **Analizar los cambios desde la perspectiva técnica y de negocio**. Los cambios tienen que contribuir a los objetivos del negocio, no simplemente a objetivos técnicos.

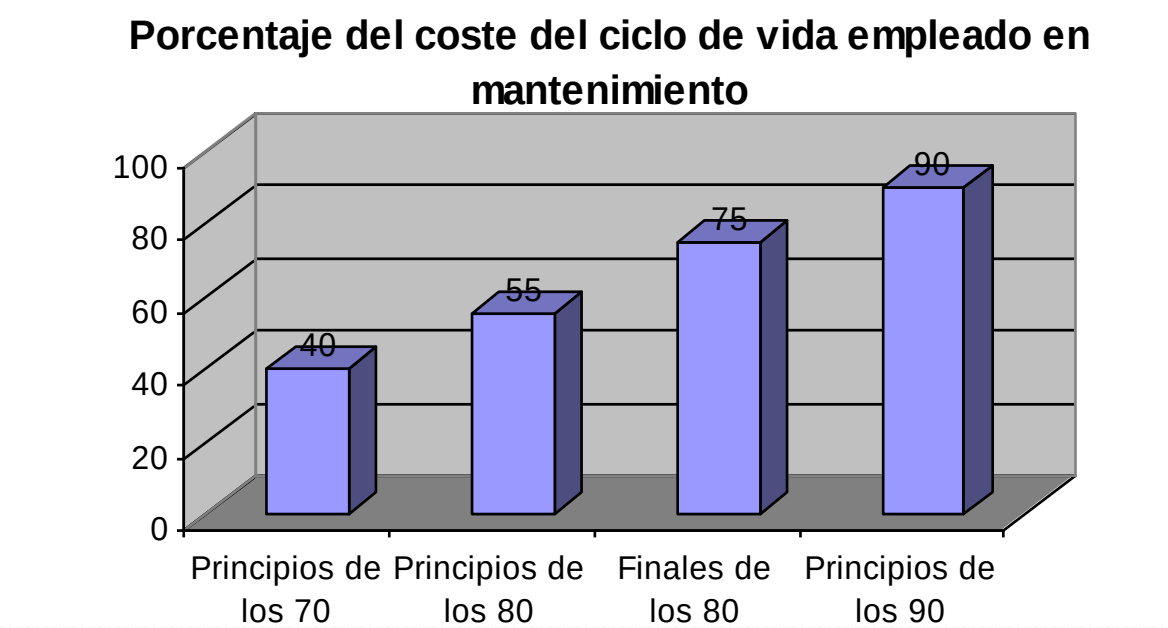
- Los **subsistemas no** son nunca completamente **independientes**, los cambios en uno, pueden afectar de forma adversa al funcionamiento o comportamiento de otros. Por lo tanto también hay que cambiar éstos.

- A menudo **no se registran las decisiones de diseño original**. Al no estar registradas dichas decisiones, los responsables de la evolución tienen que resolver por qué se tomaron esas decisiones particulares de diseño con el fin de conservarlas (si es que eso es lo mejor).

- **La estructura original se va corrompiendo por el cambio**, de tal forma que se incrementan los costos de cambio adicionales.

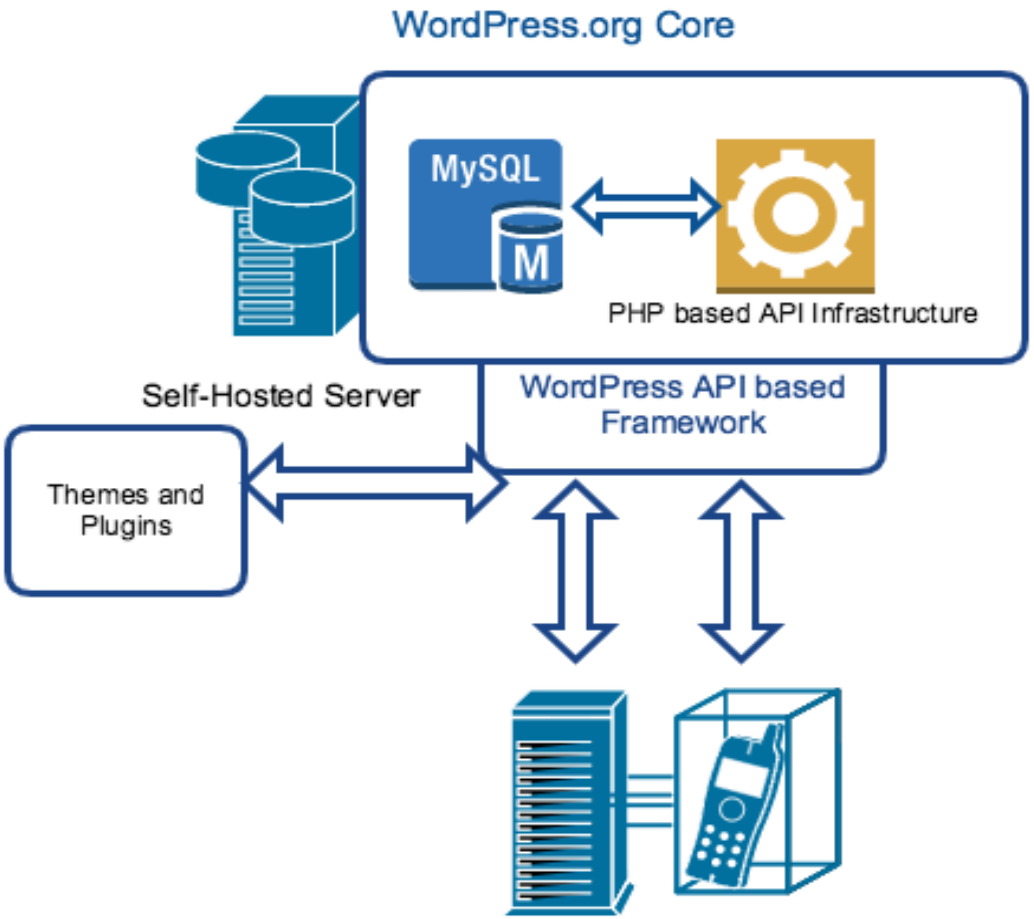
Coste del mantenimiento

- Entre el 80% y el 90% del presupuesto de muchas empresas de informática se gasta en mantenimiento.
- El mantenimiento de una aplicación grande consume de 2 a 4 veces más recursos que el desarrollo de la misma.
- Los costes de mantenimiento son difíciles de estimar.



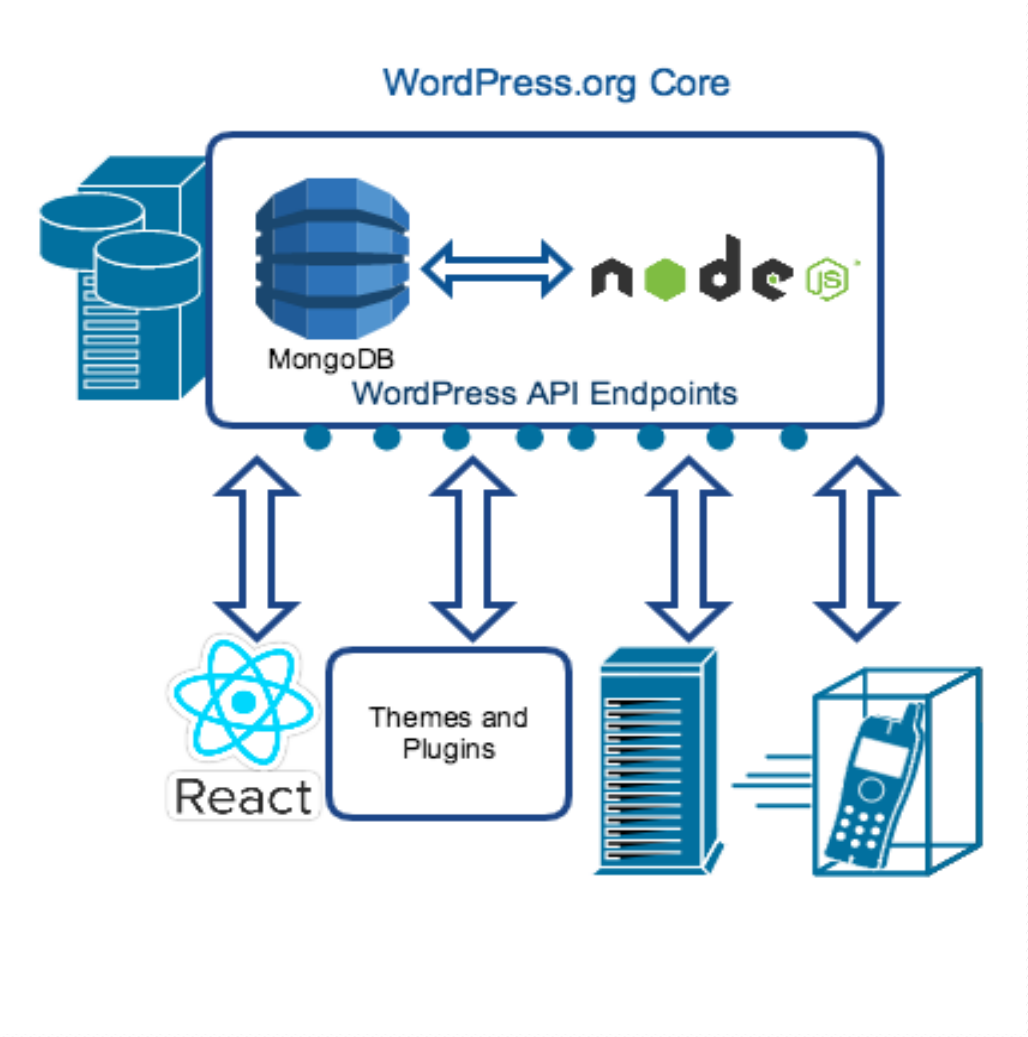
Wordpress.org – Evolución del Framework (I)

Versión original (en realidad, versión 2)



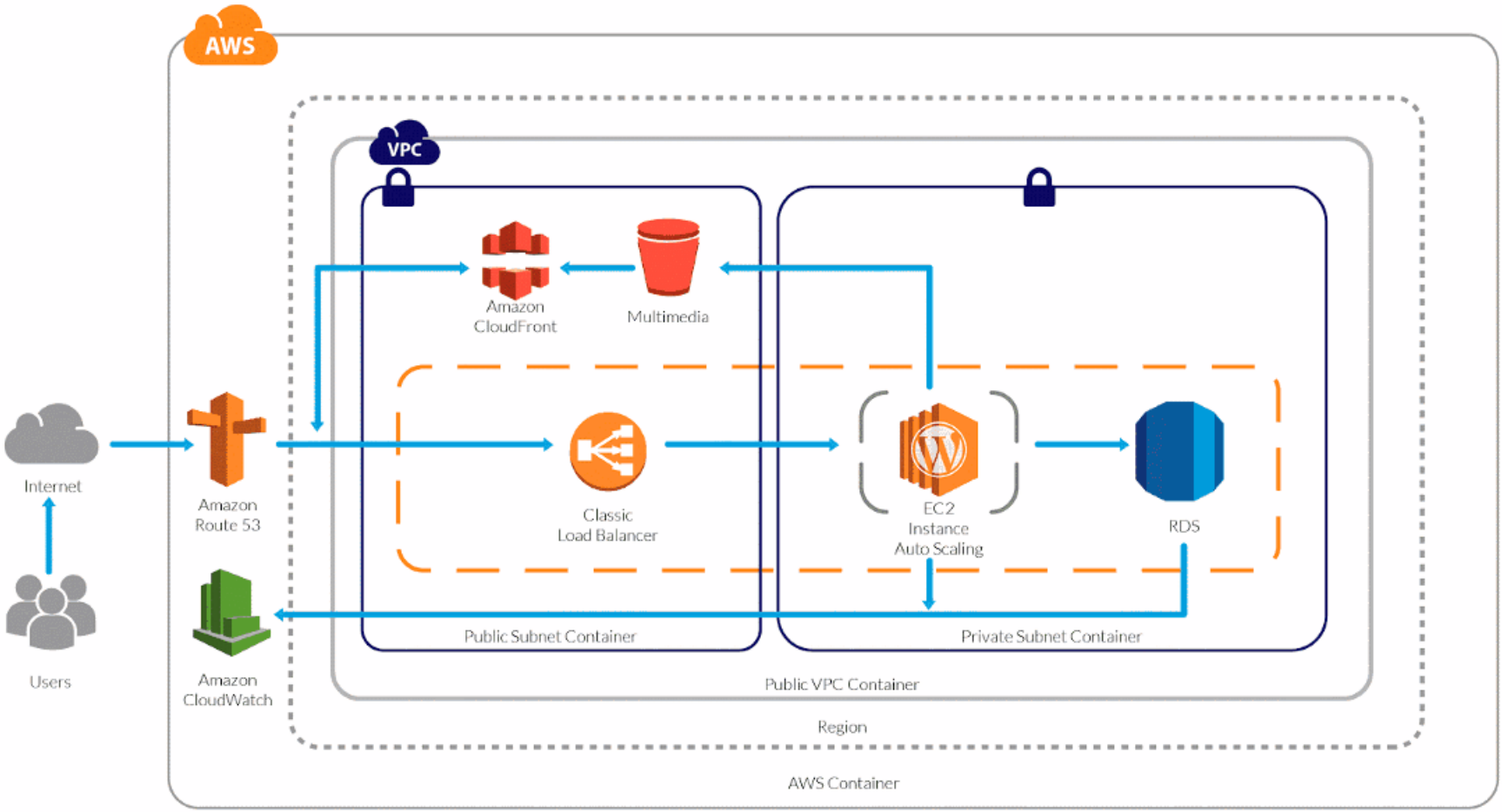
Wordpress.org – Evolución del Framework (II)

"Proyecto Calypso"



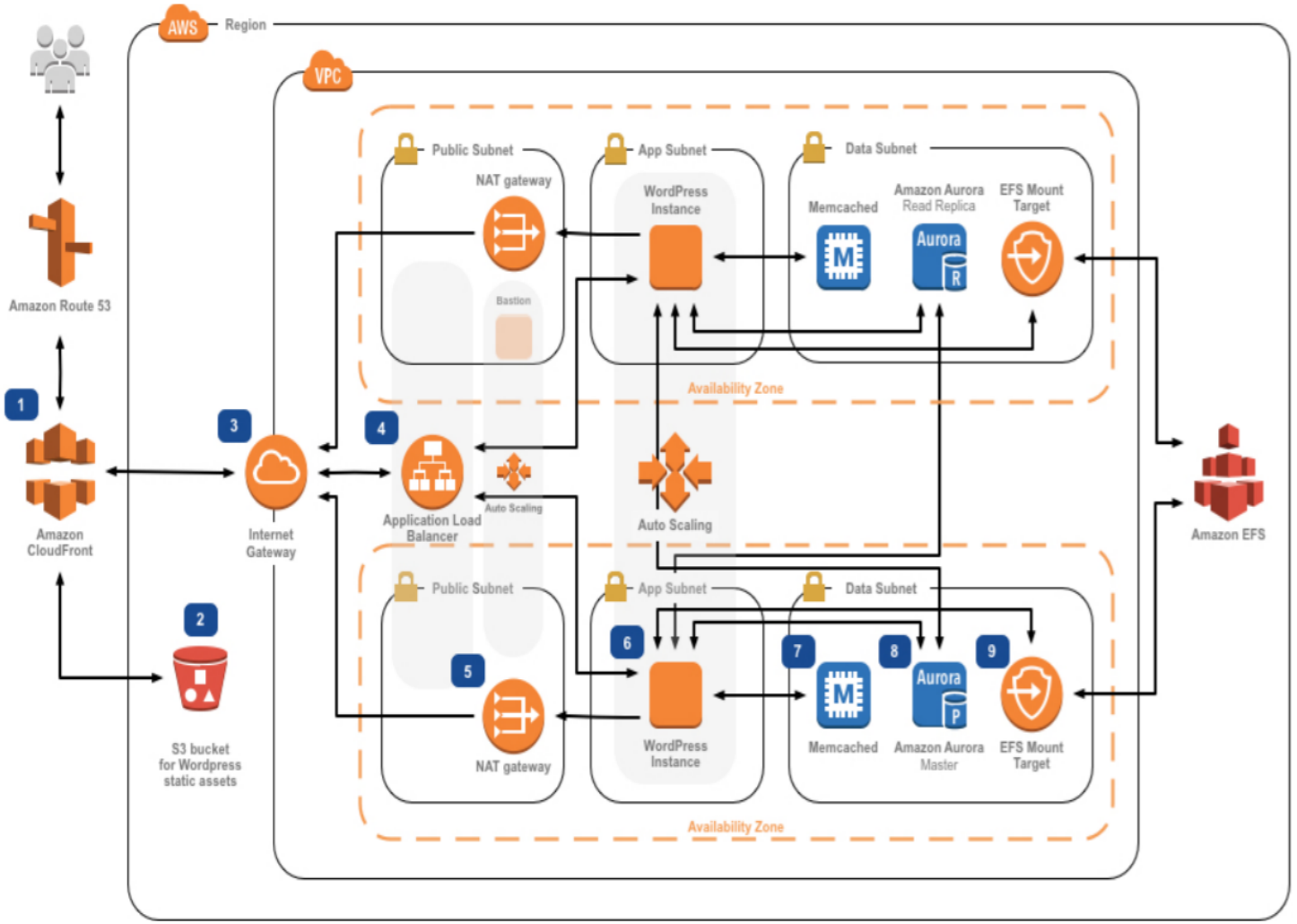
2. Importancia del mantenimiento (II)

Cambio de contexto: Wordpress en AWS



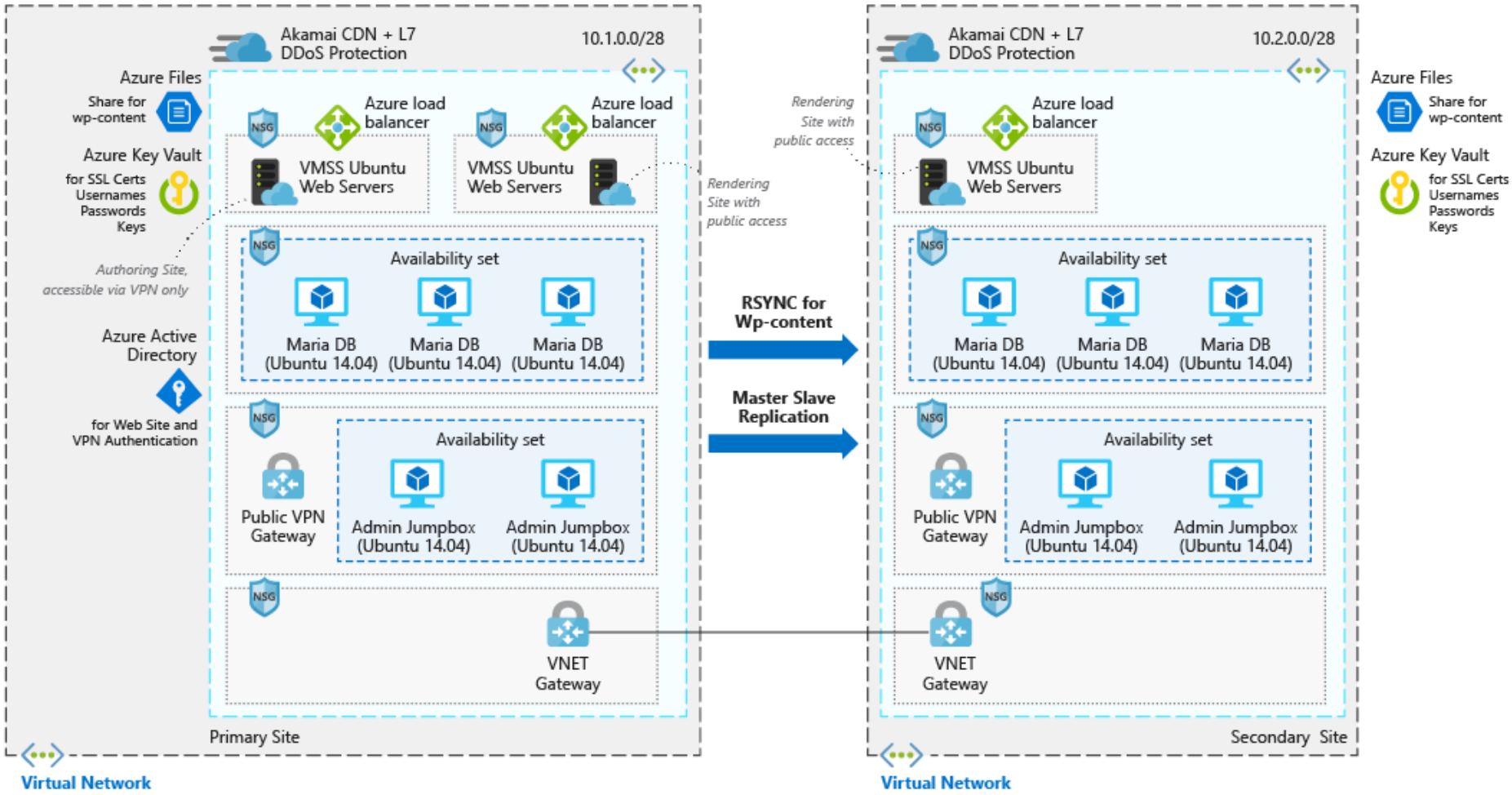
2. Importancia del mantenimiento (II)

Cambio de contexto: Wordpress AWS Reference Architecture



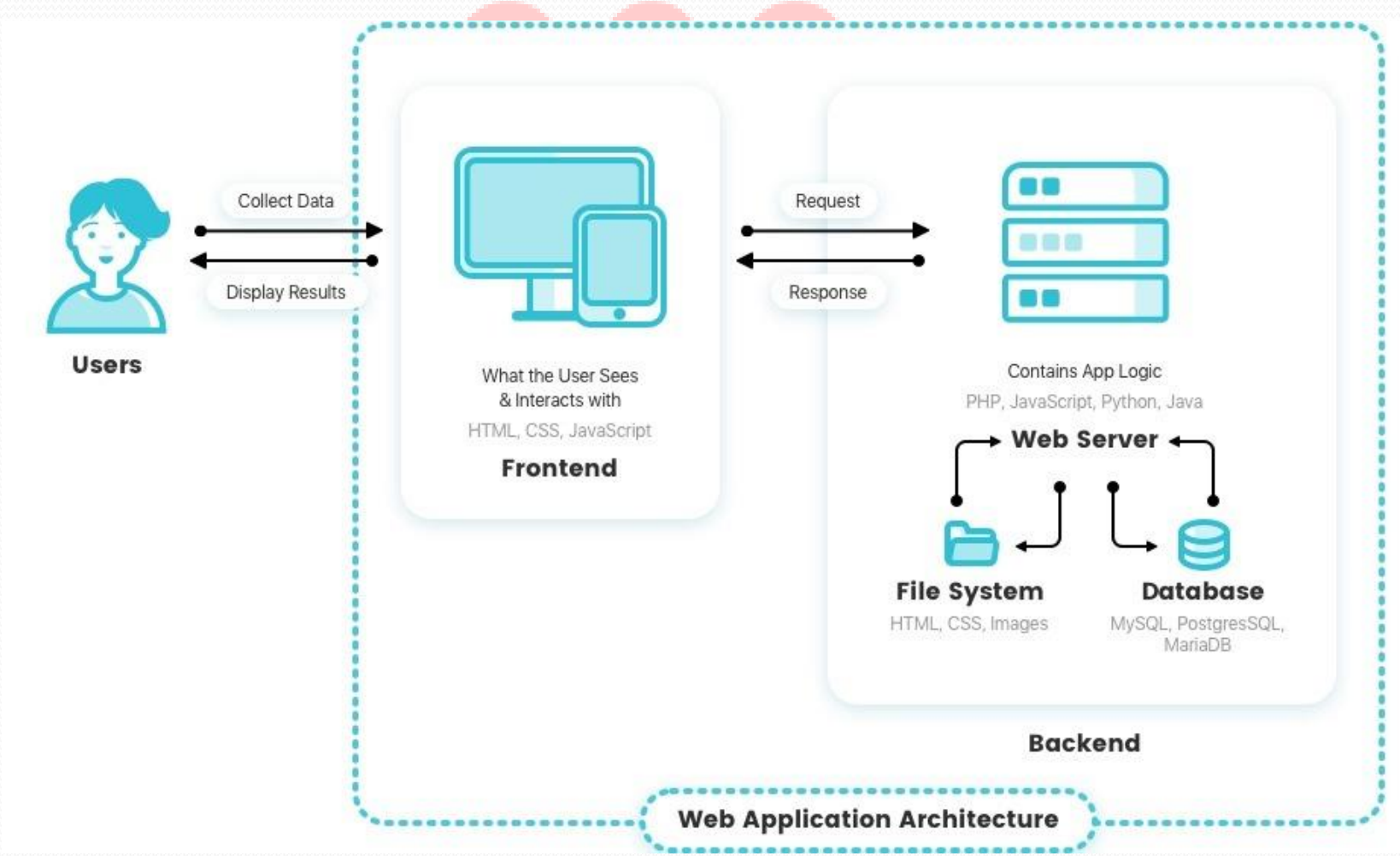
2. Importancia del mantenimiento (II)

Cambio de contexto: Wordpress Azure Reference Architecture



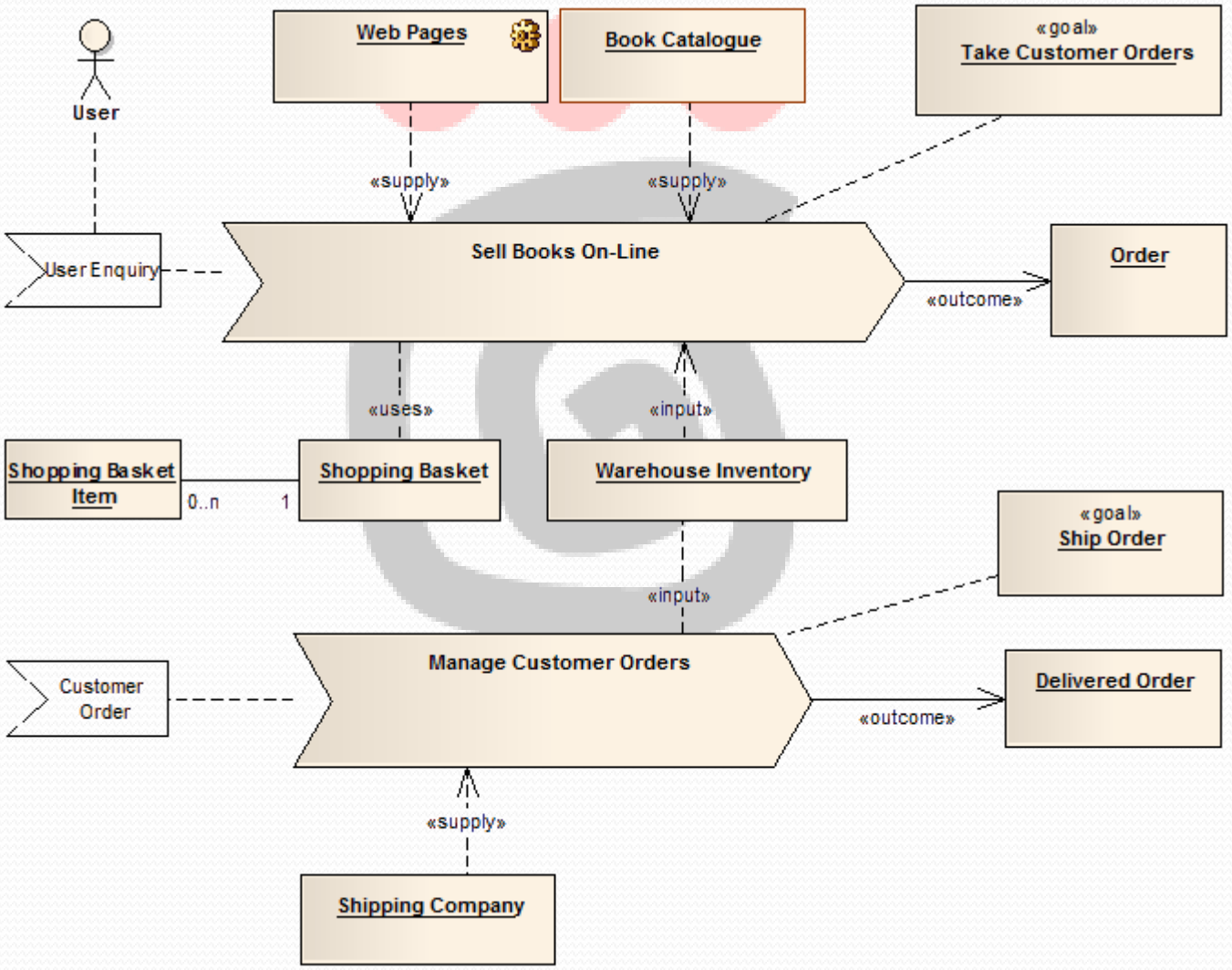
2. Importancia del mantenimiento (II)

Subsistemas y los cambios que les afectan



2. Importancia del mantenimiento (II)

Subsistemas y los cambios que les afectan



2. Importancia del mantenimiento (III)

Predicción del mantenimiento: Introducción

- La aceptación o no de un cambio en un software/sistema depende de la mantenibilidad de los componentes afectados por dicho cambio.

- La implementación de los cambios del sistema tiende a degradar la estructura de dicho sistema y, por lo tanto, reduce su mantenibilidad.

- Los costes de mantenimiento dependen del número de cambios, y los costes de la implementación de los cambios dependen de la mantenibilidad de los componentes del sistema.

2. Importancia del mantenimiento (IV)

Predicción del mantenimiento: Nº peticiones cambio

La **predicción del número de peticiones de cambios** para un sistema requiere entender la **relación entre el sistema y su entorno**. Algunos sistemas tienen una relación muy compleja con su entorno y los cambios en ese entorno inevitablemente provocan cambios en el sistema. Hay que tener en cuenta:

- El **número y la complejidad de las interfaces del sistema**. Cuanto mayor sea el número de interfaces y más complejas sean éstas, es más probable que se hagan más peticiones de cambio.

- El número de requisitos del sistema intrínsecamente volátiles. Los requisitos que reflejan políticas y procedimientos organizacionales son probablemente más volátiles que los requisitos que se basan en características estables del dominio.

- Los **procesos de negocios** en los que se utiliza el sistema. Puesto que los procesos de negocios evolucionan, generan peticiones de cambio en el sistema. Cuanto más procesos de negocios utilice el sistema, más peticiones de cambio habrá.

2. Importancia del mantenimiento (IV)

Predicción del mantenimiento: mantenibilidad

La **predicción de la mantenibilidad** de un sistema tiene que ver con el número y los tipos de relaciones entre los componentes del mismo. Cuanto más complejo es un sistema o componente, más caro es de mantener. Hay estudios sobre los diferentes tipos de complejidad (McCabe, 1976), y sobre la relación entre complejidad y mantenibilidad (Kafura y Reddy, 1987).

Reducir los costes de mantenimiento pasa por reemplazar los componentes complejos.

Métricas para evaluar la mantenibilidad una vez que el sistema se ha puesto en funcionamiento: (ver **métricas**).

3. Procesos Diferenciados *versus* Único Proceso

Históricamente, siempre ha existido una diferenciación entre el proceso de desarrollo y el proceso de evolución (o mantenimiento) del software. Siempre se ha considerado que el proceso de desarrollo es interesante por ser creativo y porque se crea algo desde el inicio. Por contra, el mantenimiento siempre se ha considerado como algo aburrido.

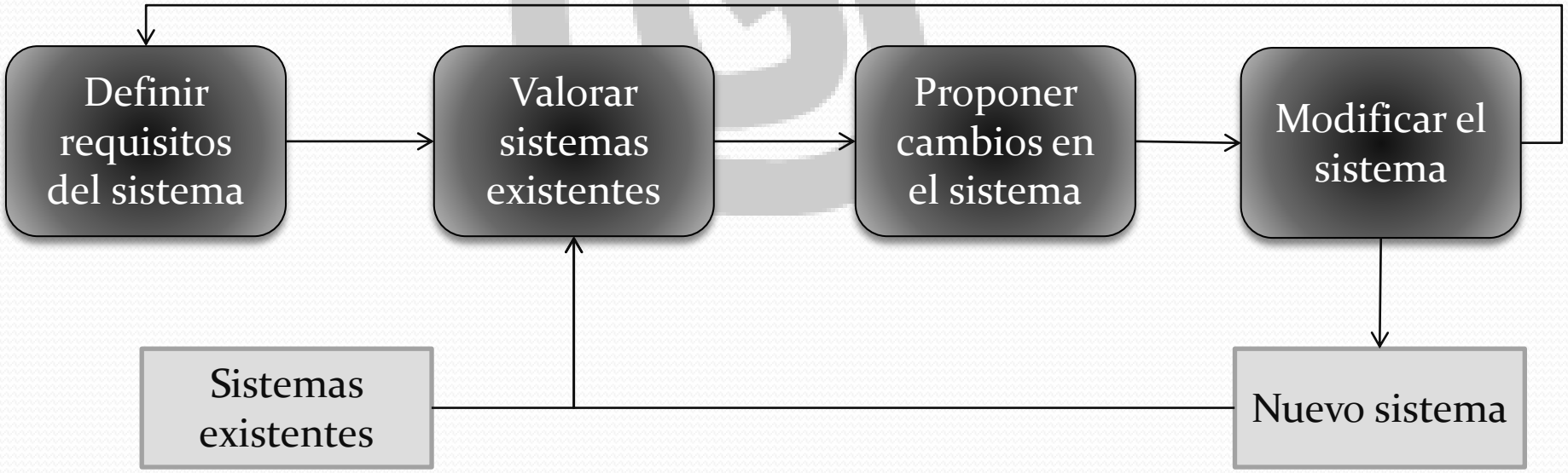
En la actualidad, la diferencia entre desarrollo y mantenimiento tiende a ser irrelevante, puesto que gran parte de lo que se desarrolla es para adaptar el sw ya existente o para que un sistema evolucione. Esto es debido a que se hicieron grandes inversiones, hace años, en desarrollos desde el principio y vale más la pena modificar y evolucionar que crear de nuevo.

El costo del mantenimiento es a veces mayor que el del desarrollo, pero da muchos más problemas el desarrollo desde el principio que el mantenimiento.

4. Proceso Evolutivo

La tendencia actual es ver el desarrollo y el mantenimiento dentro de un proceso único continuo y evolutivo, que integra tanto actividades de desarrollo como de mantenimiento.

De esta forma, se da soporte al **ciclo de vida del software** de forma completa y continua, según un proceso como el que se aprecia en la siguiente figura.



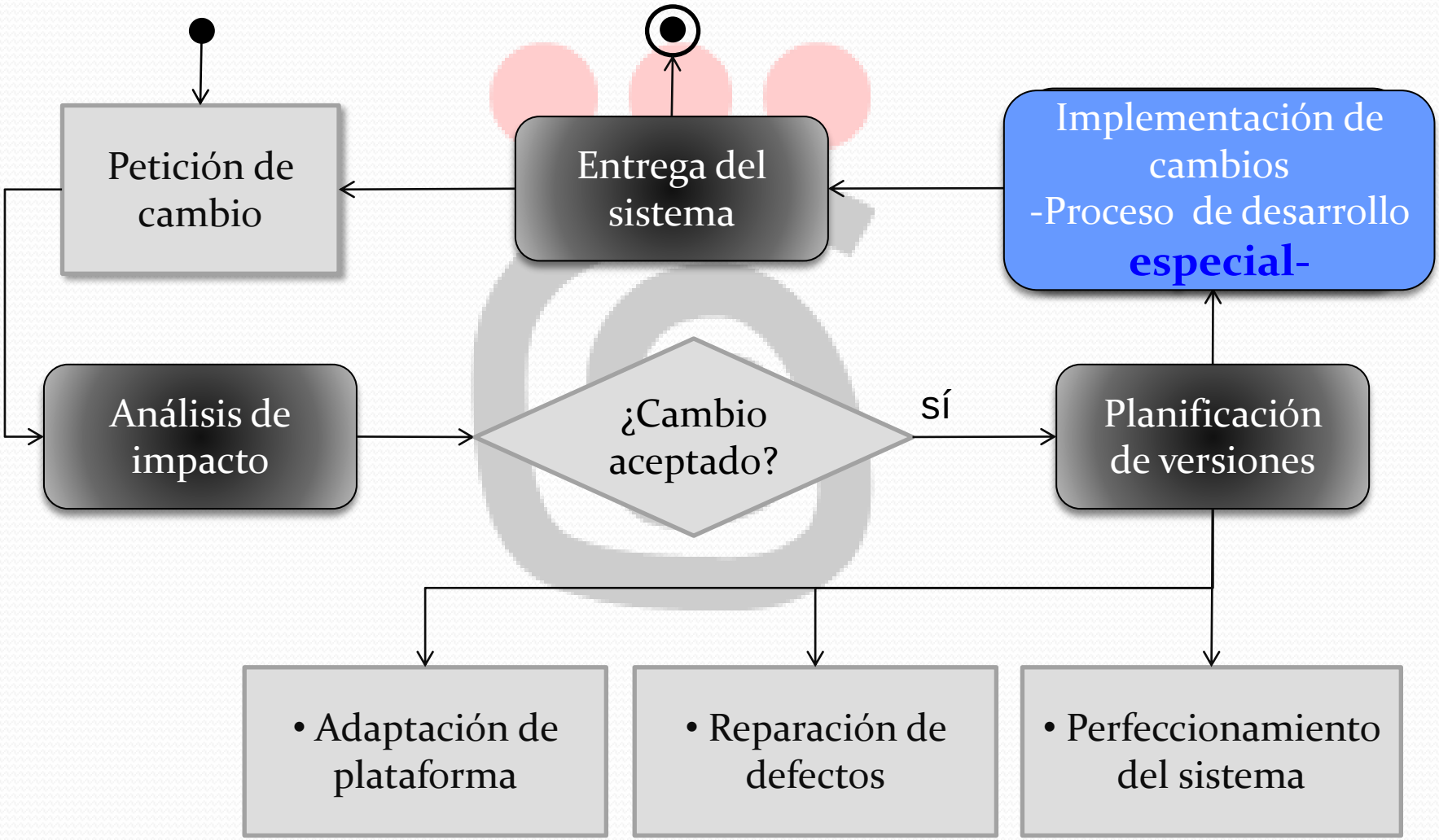
5. Objetivos docentes en la evolución del software

- Comprender que los cambios y modificaciones del sw son inevitables para que los sistemas sigan siendo útiles y que el desarrollo y la evolución pueden integrarse en un único modelo (modelo en espiral).
- Conocer los diferentes tipos de mantenimiento del sw y los factores que tienen que ver con el costo del mismo.
- Ser consciente de los procesos implicados en la evolución del software.
- Comprender cómo los sistemas heredados pueden ser evaluados para decidir si desecharlos, mantenerlos, re-desarrollarlos o reemplazarlos

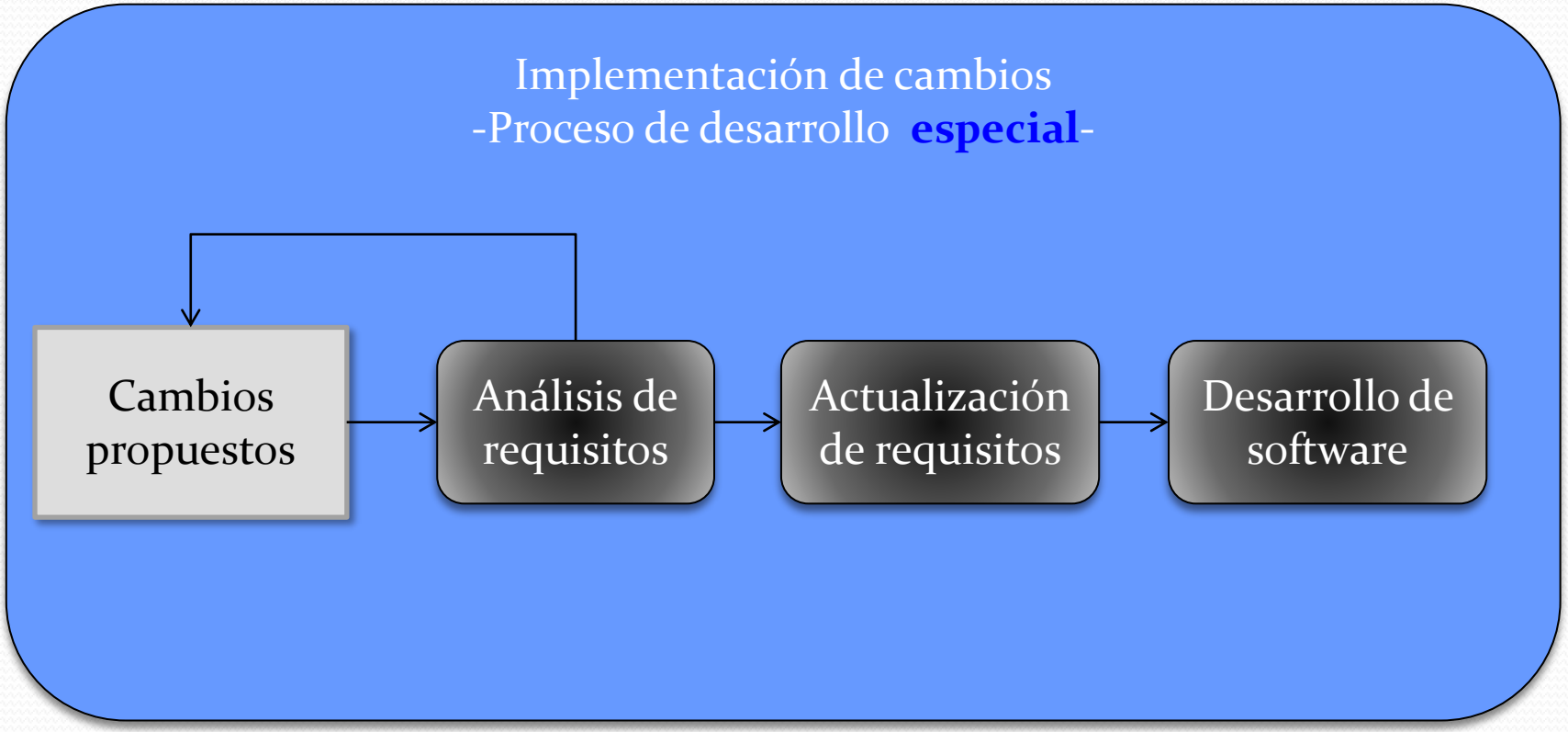
6. Procesos de evolución

- Los procesos de evolución dependen de cada organización (sw a mantener, proceso de desarrollo utilizados en la misma, personal implicado): formal e informal.
- Peticiones de cambio: conductores de los procesos de evolución del software en todas las organizaciones.
- Tipos de propuestas de cambio:
  - Requisitos ya existentes no integrados aún
  - Peticiones de nuevos requisitos
  - Reparaciones de errores
  - Ideas nuevas y mejoras indicadas por el equipo de desarrollo
- Los procesos de identificación de cambios y evolución del sistema son cíclicos y continúan durante toda la vida del sistema.

6. Procesos de evolución



6. Procesos de evolución



- Cambio del software ➡ Entregas sucesivas del sistema
- Entrega ➡ Versiones de los componentes del sistema

➡ Seguimiento de versiones para aseguramiento de versiones correctas en cada entrega: **Gestión de Configuraciones**

7. Mantenimiento y su clasificación

Actuaciones comunes para mantener la operatividad del software

- Corrección de defectos en el software
- Creación de nuevas funcionalidades en el software por nuevos requisitos de usuario
- Mejora de la funcionalidad y del rendimiento

7. Mantenimiento y su clasificación

Tipos de mantenimiento

Mantenimiento perfectivo

Conjunto de actividades para mejorar o añadir **nuevas funcionalidades** requeridas por el usuario

Mantenimiento adaptativo

Conjunto de actividades para **adaptar** el sistema a los cambios (hardware o software) en su entorno tecnológico

Mantenimiento correctivo

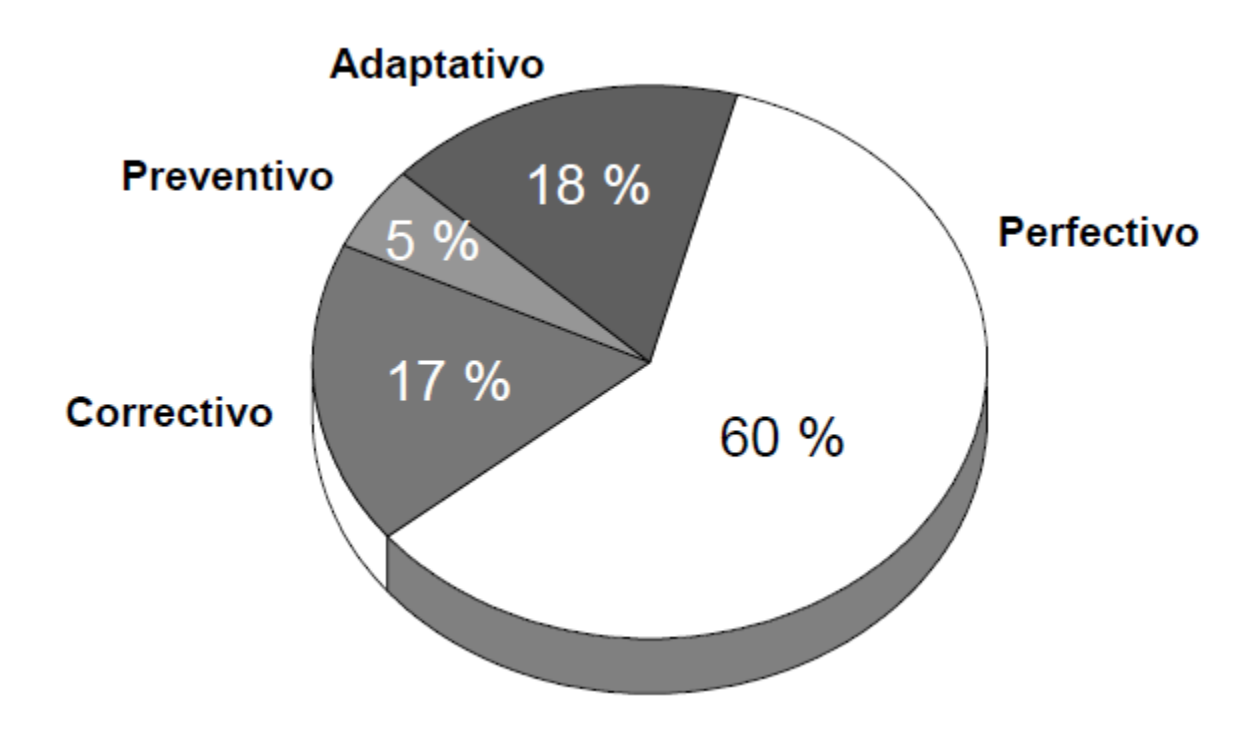
Conjunto de actividades dedicadas a corregir **defectos** en el hardware o en el software detectados por los usuarios durante la explotación del sistema

Mantenimiento preventivo

Conjunto de actividades para **facilitar** el mantenimiento futuro del sistema

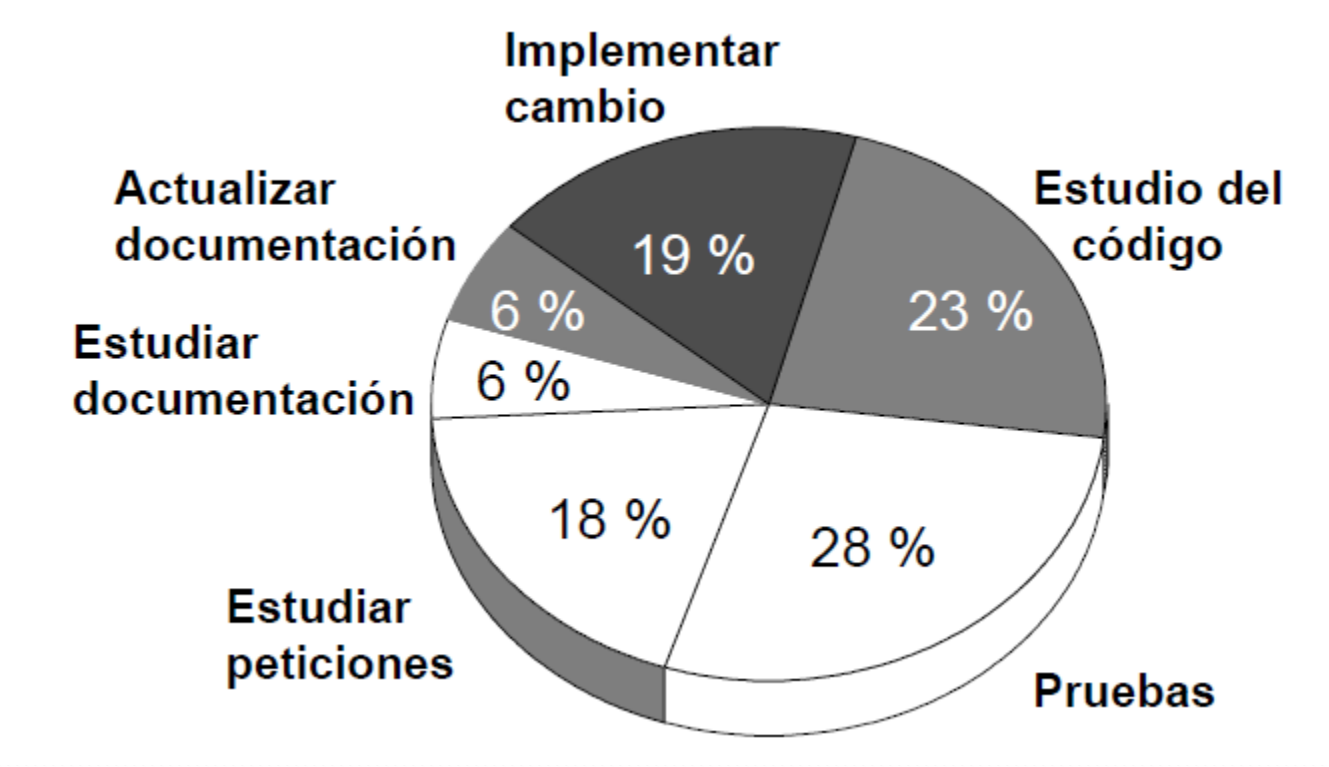
7. Mantenimiento y su clasificación

Tipos de mantenimiento y sus %



7. Mantenimiento y su clasificación

Distribución en tareas de mantenimiento



7. Mantenimiento y su clasificación

Métricas de predicción del mantenimiento

- ***El número de peticiones de mantenimiento correctivo.*** Un crecimiento en el número de informes de fallos de ejecución puede indicar que se han introducido más errores en el programa de los que se han corregido durante el proceso de mantenimiento. Esto puede suponer una disminución de la mantenibilidad.

- ***Tiempo medio requerido para el análisis de impacto.*** Este tiempo refleja el número de componentes del sistema que se ven implicados en una petición de cambio. Si este tiempo se incrementa implica que más y más componentes se ven implicados y que la mantenibilidad disminuye.

- ***Tiempo medio empleado en implementar una petición de cambio.*** No es el mismo tiempo que el del análisis de impacto aunque está relacionado con él. Se trata de la cantidad de tiempo que se necesita realmente para modificar el sistema y su documentación. Un incremento en el tiempo necesario para implementar un cambio puede indicar una disminución en la mantenibilidad.

- ***El número de peticiones de cambio pendientes.*** Un incremento en este número a lo largo del tiempo puede implicar una disminución en la mantenibilidad.

8. Legacy systems

Sistemas socio-técnicos

Un **sistema** es una colección de componentes interrelacionados que trabajan conjuntamente para cumplir un objetivo (lavadora, sistema de control del tráfico aéreo, semáforo de circulación, bicicleta…)

Los sistemas que incluyen **software** son de dos tipos:

- **Sistemas técnico-informáticos** son sistemas que incluyen hardware y software pero no procesos ni procedimientos –móviles, procesador de textos, lavadora-. Los individuos usan estos sistemas para algún fin, pero ese fin no es parte del sistema.

- **Sistemas socio-técnicos** son sistemas que comprenden uno o más sistemas técnicos pero que además incluyen conocimiento de cómo debe usarse el sistema para alcanzar algún objetivo más amplio. Han definido procesos operativos, incluyen personas como parte inherente del sistema, están gobernados por políticas y reglas organizacionales y pueden verse aceptados por restricciones externas como leyes nacionales y políticas reguladoras.

Sistemas socio-técnicos (II)

Características de los sistemas socio-técnicos:

- **Propiedades emergentes**. Propiedades de los componentes individuales más las relaciones entre ellos. Por ello, no pueden probarse hasta que no esté el sistema al completo.

- Generalmente, **no** d**eterministas**. Cuando se presenta una entrada determinada al sistema no siempre se obtiene la misma salida. El comportamiento depende de operadores humanos y no siempre se comportan de la misma manera.

- **Grado de apoyo a los objetivos organizacionales**. El grado de apoyo a los objetivos no sólo depende del sistema sino de la propia estabilidad de los objetivos, de las relaciones y conflictos y de cómo las personan interpretan esos objetivos. Una nueva dirección, en una organización, puede reinterpretar los objetivos y entonces un sistema que era todo un éxito, puede llegar a convertirse en un verdadero fracaso.

8. Legacy systems

Evolución en los sistemas heredados

Un **sistema heredado es** un sistema socio-técnico que es útil o fundamental para una organización, pero que ha sido desarrollado utilizando una tecnología o métodos obsoletos. Debido a que estos sistemas llevan a cabo funciones de negocio críticas, tienen que ser mantenidos [Sommerville, 2005]

Un sistema heredado es un sistema informático (o equipos informáticos o conjunto de aplicaciones) que ha quedado anticuado pero que se sigue utilizando, y no se quiere o no se puede reemplazar o actualizar de forma sencilla [Wikipedia].

En las grandes organizaciones existen sistemas heredados que son sistemas de negocio críticos, que han de extenderse para adaptarse a las prácticas de comercio/negocio electrónico (sistema de una entidad bancaria, control aéreo, sistema de distribución de productos alimenticios, sistema de un gran almacén “Corte Inglés”).

Generalmente las organizaciones cuentan con presupuestos limitados para mantener y actualizar estos sistemas heredados por lo que deben decidir cómo obtener los mayores beneficios . Por lo tanto, es necesario que se realicen **evaluaciones realistas** de sus sistemas heredados y decidir cuál es la **estrategia de evolución** más adecuada para estos sistemas.

8. Legacy systems

Estrategias de evolución (I)

- **Desechar completamente el sistema.** Esta opción debería elegirse cuando el sistema no constituye una contribución efectiva para los procesos de negocio. Esto ocurre cuando los procesos de negocio han cambiado desde que se instaló el sistema y ya no son completamente dependientes de éste. Por ejemplo, sustituir en las aulas de prácticas, los terminales y su mainframe, por PCs en red y un modelo cliente-servidor.

- **Dejar el sistema sin cambios y continuar con un mantenimiento regular.** Se debería optar por esto cuando el sistema aún es necesario y muy estable, y los usuarios solicitan un número relativamente pequeño de cambios. Por ejemplo, ¿qué queréis cambiar en la matrícula? ¿qué queréis cambiar en tutorías integrales a través del portal de servicios?

8. Legacy systems

Estrategias de evolución (II)

- **Reemplazar todo o parte del sistema con un nuevo sistema**. Esta opción debería elegirse cuando otros factores (por ejemplo, nuevo hardware) hacen que el sistema antiguo no pueda continuar. Puede optarse por una estrategia de reemplazo evolutivo en la que los mayores componentes del sistema son reemplazados por sistemas comerciales con otros componentes reutilizados siempre que sea posible. Sustituir WebCT por Moodle.

- **Hacer reingeniería del sistema para mejorar su mantenibilidad.** Cuando un sistema se ha ido modificando continuamente puede suceder que la calidad del sistema se ha degradado y los cambios son aún necesarios. Este proceso puede incluir desarrollo de nuevos componentes de interfaz para que el sistema original pueda trabajar con otros sistemas más nuevos.

9. Evaluación de los sistemas heredados

- **Perspectiva técnica y de negocio**.
  - Baja calidad y bajo valor de negocio: Caro mantenimiento y beneficios pequeños. Sistemas a desechar.
  - Baja calidad y alto valor de negocio: Caro mantenimiento y grandes beneficios, no puede desecharse. Sistemas a mejorar o reemplazar.
  - Alta calidad y bajo valor de negocio: Barato mantenimiento y beneficios pequeños. Si los cambios fueran caros, desecharlo.
  - Alta calidad y alto valor de negocio: Barato mantenimiento y grandes beneficios. El mantenimiento debería continuar.

9. Evaluación de los sistemas heredados

- Evaluar el valor de **negocio**.
  - Identificar a los stakeholders –usuarios y gestores-
  - Preguntar acerca del sistema a evaluar sobre:
    - **Uso del sistema**. Si es bajo, poco valor de negocio.
    - **Procesos de negocio soportados**. Si no se pueden introducir nuevos procesos de negocio, bajo valor de negocio.
    - **Confiabilidad del sistema**. Si un sistema no es confiable y puede afectar a los clientes del negocio, afectará al valor del negocio.
    - **Salidas del sistema**. Si las salidas del sistema son importantes, suponen un éxito para el negocio y un alto valor del mismo. Si esas salidas pueden realizarse de alguna otra forma, o si las salidas no se usan habitualmente, poco valor de negocio.

9. Evaluación de los sistemas heredados

- Evaluar la perspectiva **técnica** (I)
  - La **aplicación** en sí misma (software de aplicación) del sistema y el **entorno** del mismo (hardware y software de soporte).
  - **Factores utilizados en la evaluación de la *APLICACIÓN*:**
    - Comprensión del código fuente: variables, estructuras de control, etc.
    - Documentación: ¿existe? ¿es completa?
    - Datos: ¿existe un modelo de datos del sistema? ¿están los datos duplicados en el sistema? ¿son los datos consistentes? ¿son actuales?
    - Rendimiento: ¿es adecuado? ¿repercute en los usuarios del sistema?
    - Lenguajes de programación: ¿se utiliza en la actualidad en lenguaje de programación? ¿son modernos los compiladores?
    - Gestión de configuraciones: ¿están gestionadas todas las versiones? ¿existe una descripción de las versiones de los componentes del sistema actual?
    - Datos de prueba: ¿existen conjuntos de datos de prueba para el sistema? ¿y registro de las pruebas de regresión?
    - Habilidades del personal: ¿tienen capacidades para el mantenimiento? ¿cuántas personas comprenden el sistema?

9. Evaluación de los sistemas heredados

- Evaluar la perspectiva **técnica** (II)
  - La aplicación en sí misma (software de aplicación) del sistema y el **entorno** del mismo (hardware y software de soporte).
  - **Factores utilizados en la evaluación del *ENTORNO*:**
    - Estabilidad del proveedor: ¿existe? ¿garantías de continuidad? ¿proveedores alternativos?
    - Tasas de fallos de ejecución: ¿falla el hw? ¿falla el sw de soporte?
    - Edad: ¿años del hw y sw? ¿obsoleto? ¿beneficios?
    - Rendimiento del sistema: ¿es adecuado? ¿tiene efecto sobre los usuarios?
    - Costes de mantenimiento: ¿del hw? ¿licencias sw de soporte? El hw antiguo puede tener alto coste de mantenimiento, más que los hw modernos. Lo mismo para el sw de soporte.
    - Interoperabilidad: ¿Problemas de interfaz del sistema con otros sistemas? ¿o del sw de aplicación con el de soporte?

10. Bibliografía

- Sommerville, Ian (2005). *Ingeniería del Software*. Ed. Pearson – Addison Wesley, 7ª edición.
- Pressman, Roger S. (2002). *Ingeniería del Software: Un enfoque práctico*. Ed. McGraw Hill. 5ª edición.
- Madhavji, Nazim H. (2005). *Software evolution and feedback theory and practice*. Ed. Wiley.
- Mens, T., Demeyer, S. Eds. (2003). *Software Evolution*. Springer.
- Chapin, N. et al. (2001). *Types of software evolution and software maintenance*. JOURNAL OF SOFTWARE MAINTENANCE AND EVOLUTION: RESEARCH AND PRACTICE. J. Softw. Maint. Evol.: Res. Pract. 2001; 13:3–30
- Rodríguez, A. et al. (2003). *Gestión de la evolución del software. El eterno problema de los legacy systems.*