

Fundamentos de Ingeniería del Software



Capítulo 3. Análisis de Requisitos Prototipado


Cap 3. Análisis de Requisitos Estructura



1. Actividades iniciales.
2. Técnicas de recogida de la información.
3. Requisitos y análisis de requisitos.
4. Actividades generales de análisis de requisitos.
5. Documentos de especificación de requisitos.
6. Análisis estructurado.
7. Introducción a los casos de uso.
8. Prototipado.

8. Prototipado.

Estructura



- 8.1. Concepto de prototipo
- 8.2. Uso de los prototipos
- 8.3. Tipos de prototipos
- 8.4. Herramientas para el prototipado
- 8.5. Prototipos de la interfaz de usuario

Prototipado.

Bibliografía



- (Piattini et al. 1996) (Piattini et al. 2004)
 - Apartado 6.3.3.
- (Pressman 2001) Apartados 11.4 y 2.5.
- (Pressman 2006) Apartado 3.4.1.
- (Sommerville 2002) Capítulo 8.

8.1. Concepto de prototipo




- Primera versión de un nuevo tipo de producto, en el que se han incorporado sólo algunas características del sistema final, o no se han realizado completamente.
- Modelo o maqueta del sistema que se construye para comprender mejor el problema y sus posibles soluciones:
 - Evaluar mejor los requisitos
 - Probar opciones de diseño

Concepto de prototipo (II)




- Características de los prototipos:
 - Funcionalidad limitada
 - Poca fiabilidad
 - Características de operación pobres
- Prototipo \approx 10% presupuesto del proyecto
 - Normalmente pocos días de desarrollo

8.2. Uso de los prototipos



- Se presenta al cliente un prototipo para su experimentación
 - Ayuda al cliente a establecer claramente los requisitos
- Ayuda a los desarrolladores a:
 - Validar corrección de la especificación.
 - Aprender sobre problemas que se presentarán durante el diseño e implementación del sistema.
 - Mejorar el producto.
 - Examinar viabilidad y utilidad de la aplicación.

¿Cuándo son interesantes los prototipos?



- Brooks considera el prototipado una de las aproximaciones realmente prometedoras, por dirigirse a la complejidad “esencial” del software “No silver bullet” (Brooks 87).
- **Siempre**, pero especialmente cuando...
 - Área de aplicación no bien definida (bien por su dificultad o por falta de tradición en su aplicación).
 - El coste de rechazo por parte de los usuarios, por no cumplir sus expectativas, es muy alto.
 - Es necesario evaluar previamente el impacto del sistema en los usuarios y en la organización.
 - Se usan nuevos métodos, técnicas, tecnología.

8.3. Tipos de prototipos



Taxonomía 1. Objetivo.

■ Prototipado de interfaz de usuario

- modelos de pantallas/ventanas.

■ Prototipado funcional (operacional)

- implementa algunas funciones.

■ Modelos de rendimiento

- evalúan el rendimiento de una aplicación crítica (no sirven al análisis de requisitos).

Tipos de prototipos (II)

Taxonomía 2. Desarrollo.

■ **Rápido o desechable:**

- Sirve al análisis y validación de los requisitos
- Después se redacta la especificación del sistema y se desecha el prototipo
- La aplicación se desarrolla siguiendo un paradigma diferente
- Problema: cuando el prototipo no se desecha, y termina convirtiéndose en el sistema final.
 - Seguramente el prototipo se habrá construido rápidamente, sin seguir un método de ingeniería del software, y utilizando un lenguaje que probablemente no será muy eficiente
 - Seguramente el mantenimiento será difícil.

■ **Evolutivos:**

- Comienza con un sistema relativamente simple que implementa los requisitos más importantes o mejor conocidos
- El prototipo se aumenta o cambia en cuanto se descubren nuevos requisitos
- Finalmente, se convierte en el sistema requerido
- También es difícil el mantenimiento de las aplicaciones desarrolladas como prototipos evolutivos, ya que muchas veces la documentación se reduce al mínimo o no se escribe, y no se aplican técnicas de ingeniería del software.
 - Estas aplicaciones suelen tener menos tiempo de vida que las aplicaciones tradicionales.
- Actualmente se usa en el desarrollo de sitios web y en aplicaciones de comercio electrónico

Tipos de prototipos (III)



Taxonomía 3. Alcance.

■ Vertical

- Desarrolla completamente alguna de las funciones.

■ Horizontal

- Desarrolla parcialmente todas las funciones.

8.4. Herramientas para el prototipado (Sommerville 2002) p.180



- Lenguajes dinámicos de alto nivel
- Lenguajes de cuarta generación (4GLs) (programación de BD)
- Ensamblaje de componentes y aplicaciones

Lenguajes dinámicos de alto nivel



■ Muy usados:

Smalltalk (basado en objetos, sistemas interactivos)

Java (basado en objetos, sistemas interactivos)

Prolog (lógico, procesamiento simbólico)

LISP (basado en listas, procesamiento simbólico)

■ Elección del lenguaje:

¿Cuál es el dominio de aplicación?

¿Cuál es la interacción de usuario requerida?

(Java, Smalltalk se integran bien con las interfaces Web.)

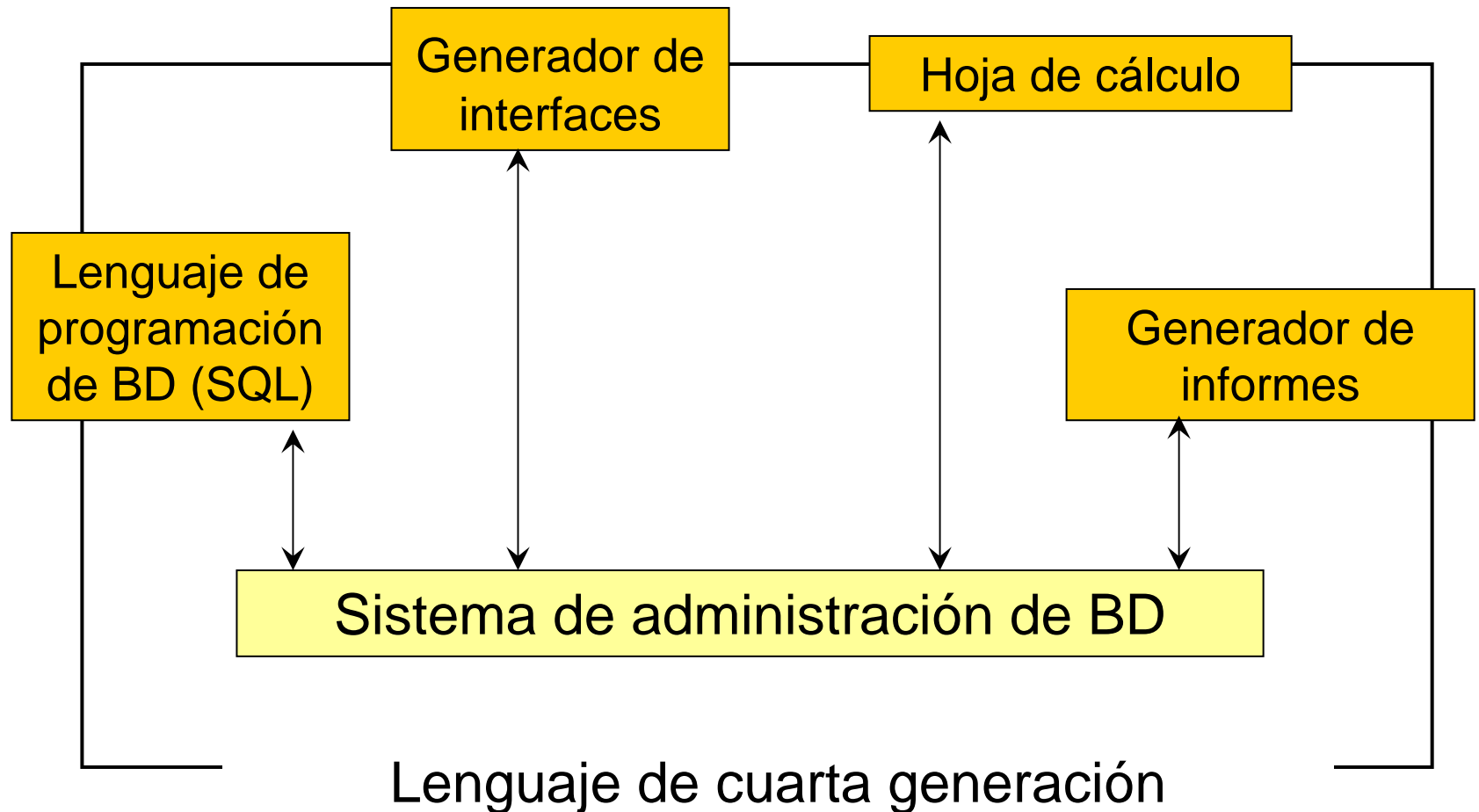
¿Cuál es el entorno proporcionado para el lenguaje?

Lenguajes de cuarta generación (4GLs)




- La mayoría de aplicaciones de gestión son interactivas e implican la manipulación de una BD y la producción de salidas que involucren organizar y dar formato a esos datos
- 4GL: lenguaje de programación de BD (y su entorno de desarrollo), que contiene conocimiento de la BD y operaciones para manipulación de la misma
- 4GL: lenguaje no procedimental

Lenguajes de cuarta generación (4GLs) (II)




Lenguajes de cuarta generación (4GLs) (III)



- Reducen claramente los costos del desarrollo
- Muy usados en prototipado evolutivo
- Muchos 4GLs permiten el desarrollo de interfaces de BD basadas en navegadores Web.
- Generan SQL o código en lenguaje “de bajo nivel” como COBOL

Lenguajes de cuarta generación (4GLs) (IV)



- Menos eficientes que los lenguajes de programación convencionales

p.ej. programa en 4GL reescrito en C++ (Sommerville 2002)

-50% requisitos de memoria


10 veces más rápido

- Reducen claramente los costos del desarrollo ...
¿y el mantenimiento?

- Programas no estructurados difíciles de mantener

- No están estandarizados ni son uniformes \Rightarrow los usuarios pueden tener que reescribir totalmente los programas debido a que el lenguaje ha quedado obsoleto

Entornos de cuarta generación



■ Ejemplos:


- Informix 4GL
- ORACLE 8i, ORACLE 10g
- ORACLE Developer 6i (Reports y Forms)
- CASE Studio 2
- Access
- ...

Ensamblaje de componentes y aplicaciones



- El desarrollo de prototipos con reutilización comprende dos niveles:
 - (a) El nivel de aplicación, en el que una aplicación completa se integra con el prototipo
 - P.ej., si el prototipo requiere procesamiento de textos, se puede integrar un sistema estándar de procesamiento de textos (MS Office u Open Office)

Ensamblaje de componentes y aplicaciones (II)



(b) El nivel de componente, en el que los componentes se integran en un marco de trabajo estándar

- Visual Basic, TCL/TK, Python, Perl...

- Lenguajes de alto nivel sin tipos, con kits de herramientas gráficas
- Desarrollo rápido de aplicaciones pequeñas y relativamente sencillas, construidas por una persona o conjunto de personas
- No existe una arquitectura explícita del sistema

- CORBA, DCOM, JavaBeans

- Junto con un marco arquitectónico, es más apropiado para sistemas grandes

8.5. Prototipos de la interfaz de usuario



- Las descripciones textuales y los diagramas no son suficientemente buenos para expresar los requisitos de la interfaz
- La construcción de prototipos evolutivos con la participación del usuario final es la forma más sensata de desarrollar una interfaz
- Los usuarios deben estar implicados en la evaluación y evolución del prototipo

Prototipos de la interfaz de usuario. Herramientas.



- Generadores de interfaz (4GLs, Visual Basic, etc.)
- Editores de páginas Web
- Herramientas CASE
 - Formularios, pantallas, generación de código...
- Bocetos en papel
- Aplicaciones de dibujo
 - Harward Graphics, etc.
- MS PowerPoint / Open Office Impress
- Etc.