Diseño de Interfaces de Usuario

Introducción

Interfaces Persoa Máquina

Marcos Ortega (m.ortega@udc.es)

Interfaz de Usuario

- Definición básica: Es el mecanismo mediante el cual un usuario de un sistema **interactúa** con el mismo
- Dicho mecanismo debe permitir la **entrada** de información por parte del usuario y abastecerle con la **salida** pertinente
- No se ciñe a la informática: **cualquier** dispositivo que requiere entrada y/o salida para el usuario necesita un interfaz!

Ejemplos clásicos (Hall of Shame)

Source: Interface Hall of Shame

Ejemplos clásicos (Hall of Shame)

Source: Interface Hall of Shame

- Usar una ventana bonita/amigable no confiere usabilidad automáticamente
- Interfaz gráfico
- No es necesario memorizar/teclear
- ¿Es usable?

Ejemplos clásicos (Hall of Shame)

Source: Interface Hall of Shame

- ¡Mensaje de ayuda!
- ¿Por qué es necesario tanto texto de ayuda?
  - El interfaz no es cómodo
- ¡Barra de scroll!
  - No tiene marcas
  - Cuántos modelos hay
  - ¿Cómo están ordenados?
  - ¿Cuánto hay que mover la barra para pasar al siguiente?

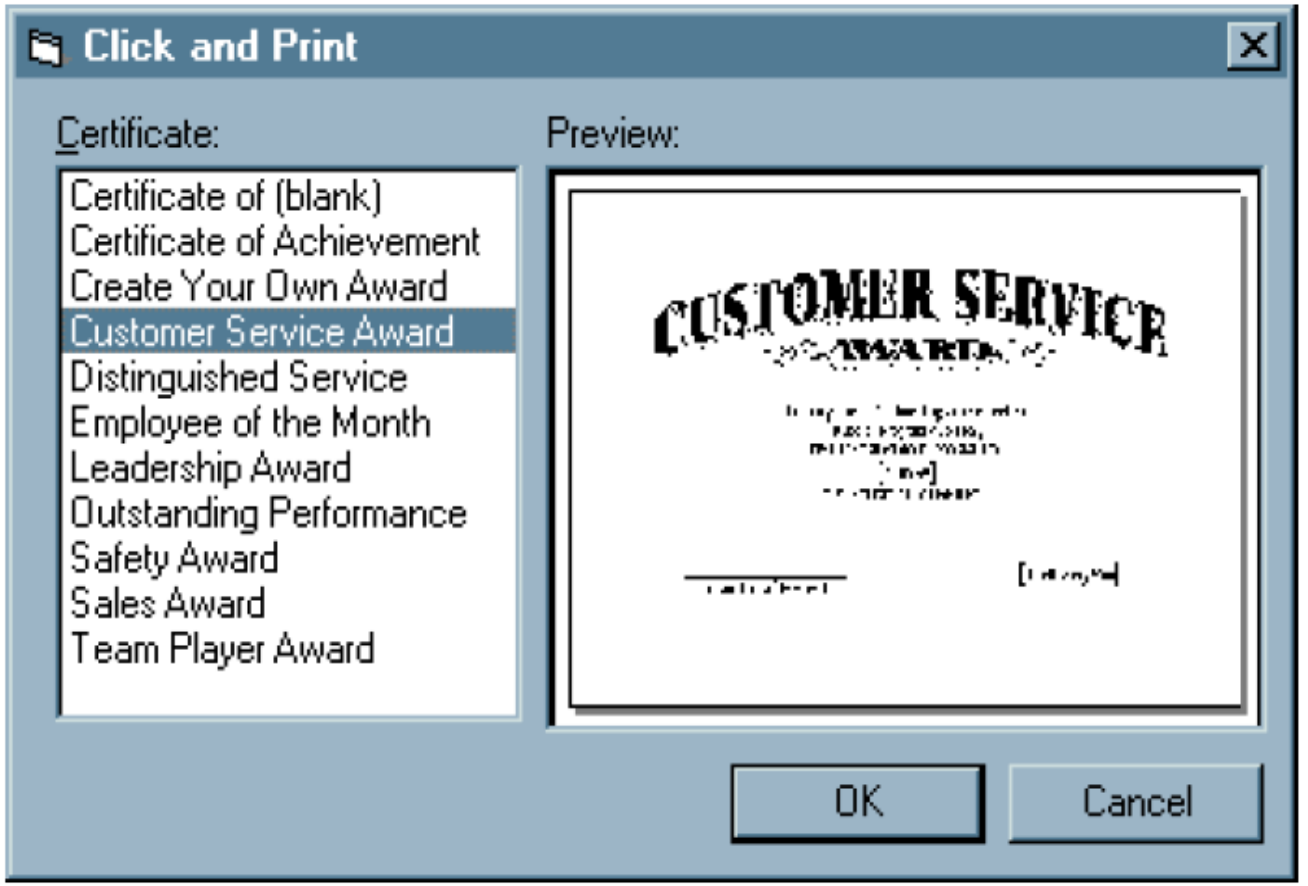
Ejemplos clásicos (Hall of Shame)

Source: Interface Hall of Shame

- Normalmente se asume que la barra de scroll sirve para desplazarse en el sentido indicado a través de un documento
- Este uso es confuso e **inconsistente** para un usuario poco frecuente
- Para usuarios más frecuentes: ¿cómo encontrar una plantilla usada previamente?

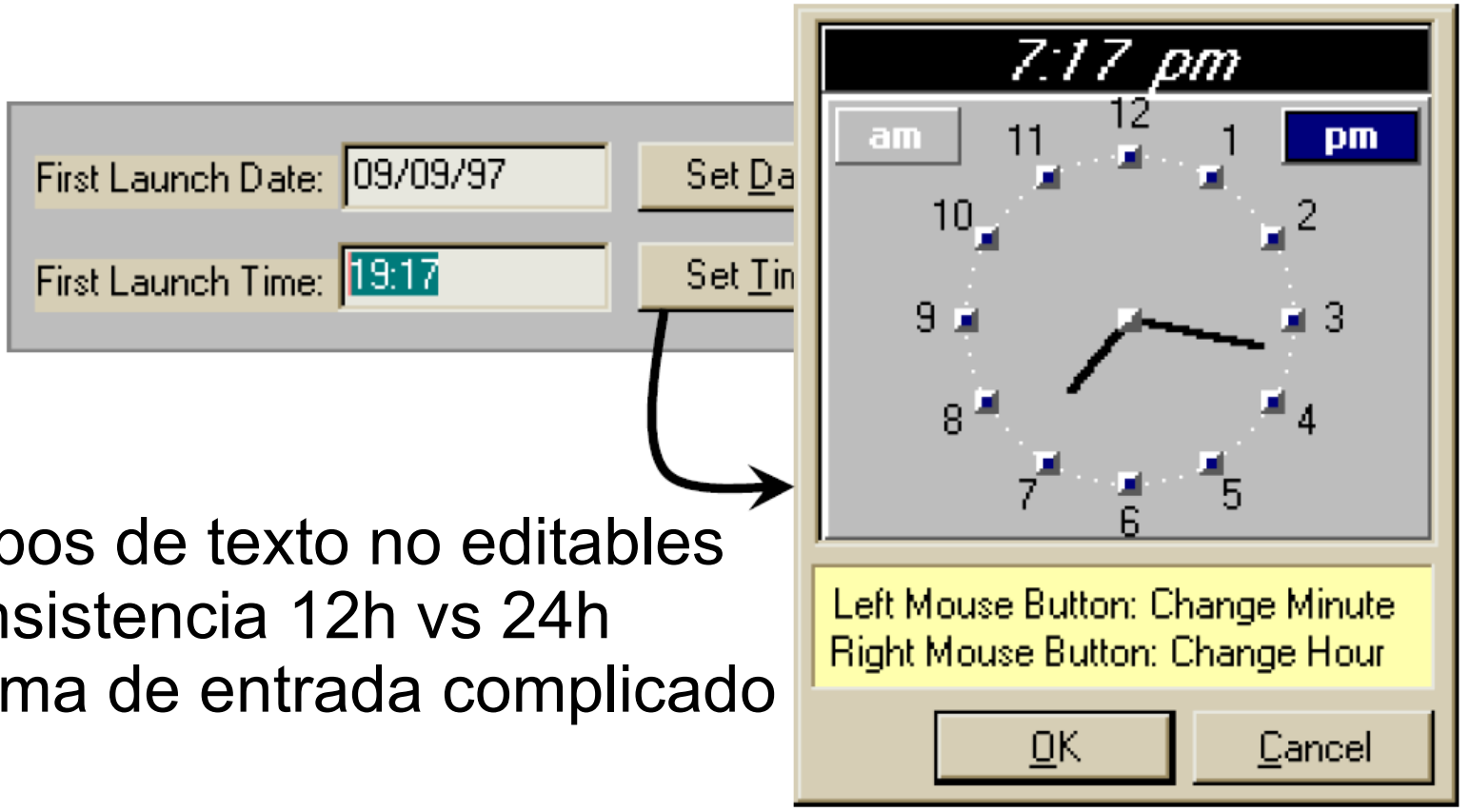
Ejemplos clásicos (Hall of Shame)

- Un diseño más apropiado



Source: Interface Hall of Shame

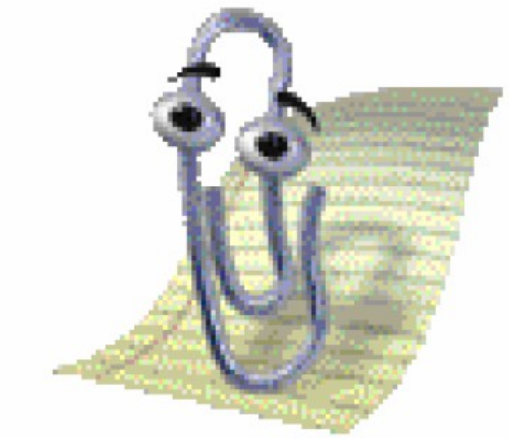
Ejemplos clásicos (Hall of Shame)



Source: Interface Hall of Shame

- Campos de texto no editables
- Inconsistencia 12h vs 24h
- Sistema de entrada complicado

Ejemplos clásicos (Hall of Shame)



- Intento de sobreponerse al usuario:
  - No leen manuales
  - No usan ayuda online
- Trata de sugerir soluciones a los problemas que “detecta”
- Desafortunadamente, la mayor parte del tiempo está equivocado, es molesto e intrusivo
- La percepción subjetiva es importante también

Importancia de los IU

- El interfaz de usuario es muy importante
- Afecta de manera decisiva en la percepción de un software
  - Software usable se vende mejor
  - Software/Webs no usables se abandonan rápidamente
- A veces la percepción es bastante superficial y depende de preconcepiones
  - Un usuario puede echarse la culpa de errores debidos a un mal diseño si le parece suficientemente atractivo de inicio.

Problemas de un mal IU

- El tiempo de los usuarios no se vuelve más barato con el tiempo
  - Los humanos no nos regimos por la ley de Moore
- Errores de diseño costosos: mantenimiento, atención al cliente...
- Sistemas críticos : apoyo al diagnóstico clínico, sistemas de control de maquinaria, etc...

Los IU son difíciles de diseñar

- Tú no eres el usuario normalmente
  - En Ingeniería del Software normalmente se comunican unos procesos con otros
  - En el desarrollo de un interfaz te comunicas con un usuario
- El usuario ¿siempre? tiene la razón
  - Si los errores son frecuentes, probablemente es un error de diseño
  - Los usuarios a veces no saben exactamente o no son capaces de expresar lo que quieren

Los IU son difíciles de construir

- Interfaces de usuario requieren una gran parte del esfuerzo de desarrollo de un proyecto software
- Normalmente están en el entorno del 50% en:
  - Tiempo de diseño
  - Tiempo de implementación
  - Tiempo de mantenimiento
  - Tamaño del código

Usabilidad

- Apartado clave en el diseño de interfaces: **Usabilidad**
- Lo bien (o mal!) que los usuarios son capaces de usar la funcionalidad del sistema.
- Varias dimensiones:
  - Aprendizaje: ¿Es fácil de aprender a usar?
  - Eficiencia: Una vez aprendido, ¿es rápido de usar?
  - Recuerdo: ¿Es fácil de retener lo aprendido?
  - Errores: ¿Son escasos los errores y el sistema fácilmente 'recuperable'?
  - Satisfacción: ¿Es 'disfrutable' su uso?

Usabilidad

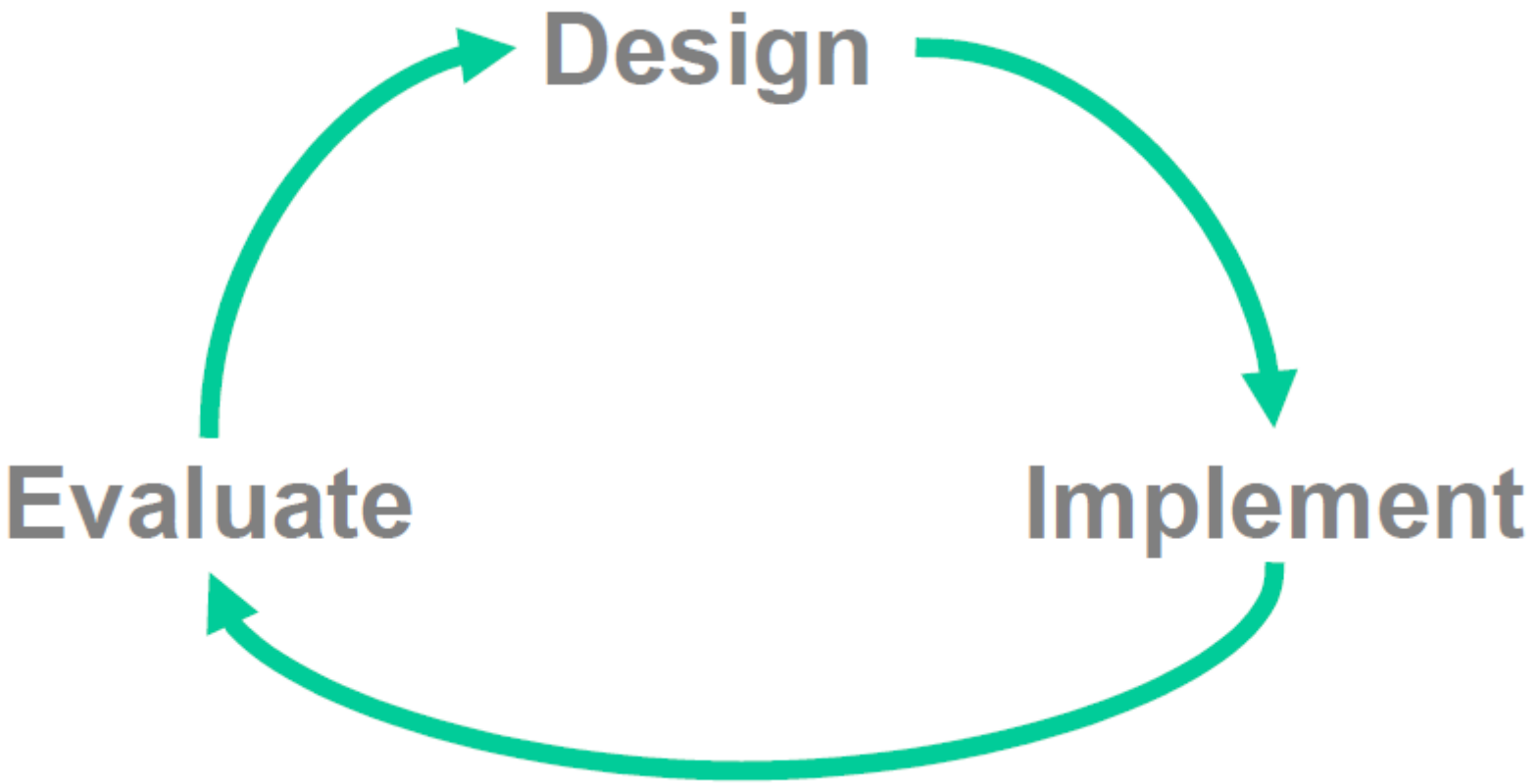
- Las dimensiones de la usabilidad varían en importancia
- Dependen del usuario, lógicamente:
  - Un usuario novato necesita un IU con buena tasa de *aprendizaje*
  - Un usuario poco frecuente necesita una buena tasa de *recuerdo*
  - Un usuario experto necesita *eficiencia*
- Pero ningún usuario es absolutamente novato o experto:
  - Experiencia en el dominio
  - Experiencia con la aplicación
  - Experiencia con funcionalidades

Usabilidad

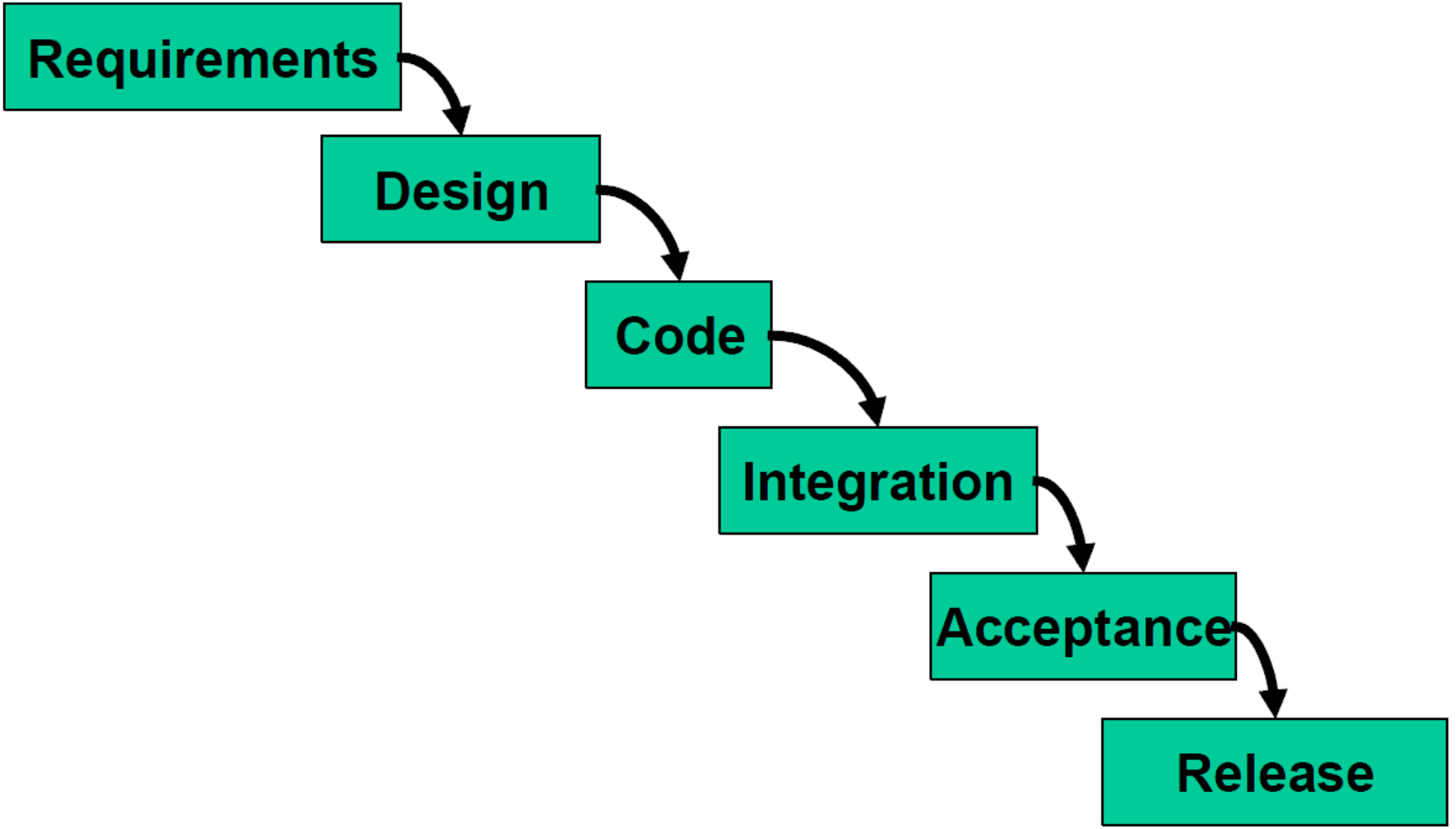
- La Usabilidad, a su vez, es una dimensión en el desarrollo de software:
  - Funcionalidad, Rendimiento, Coste, Seguridad, Usabilidad, Dimensiones, Fiabilidad,...
- Muchas (todas?) decisiones de diseño involucran un compromiso entre varias de dichas dimensiones
  - No siempre es posible maximizarlas todas

Desarrollo de Interfaces Usables

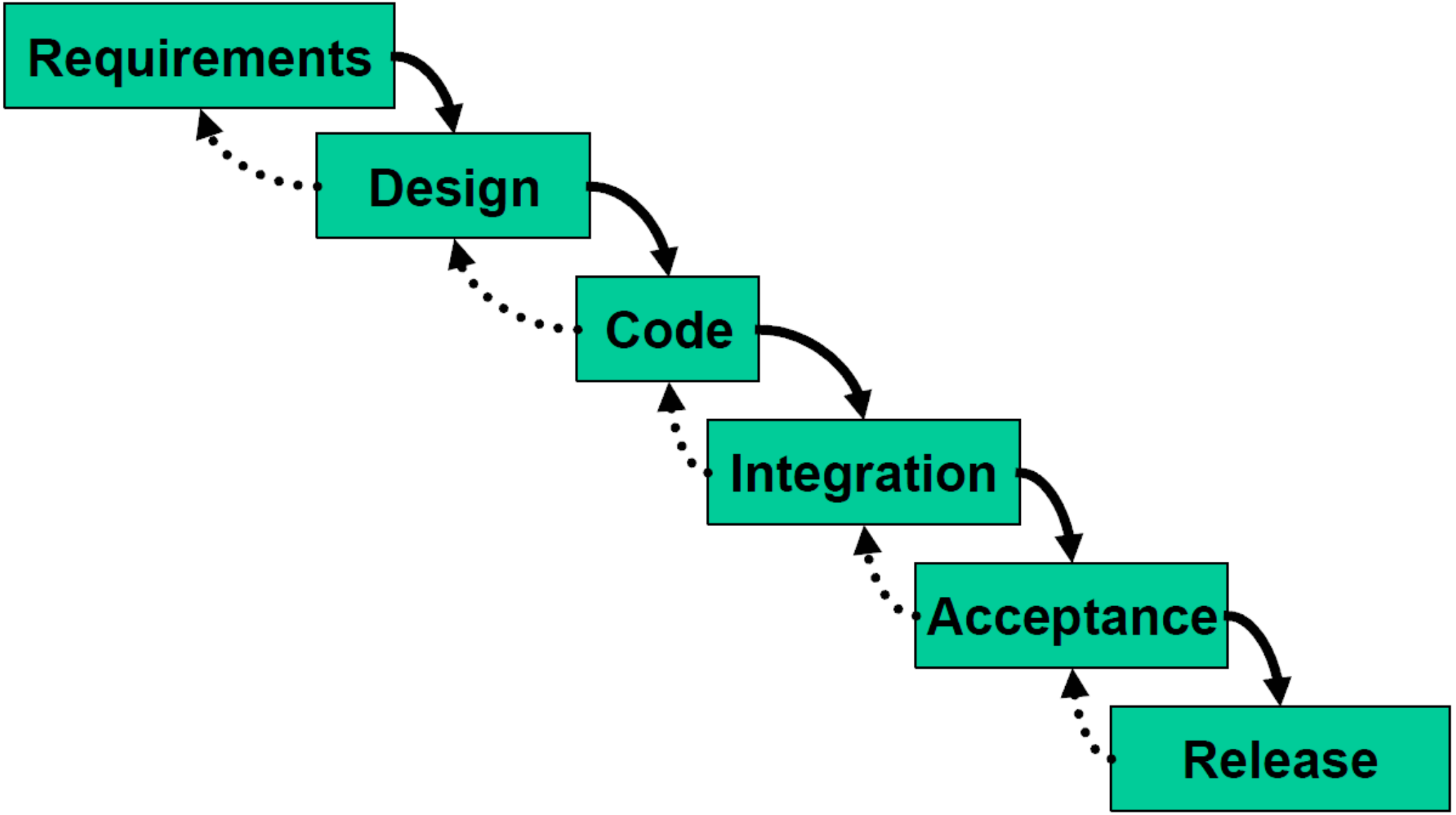
- Es un proceso continuo e iterativo



Metodologías tradicionales: Cascada



Cascada con feedback



Modelo en Cascada para IUs

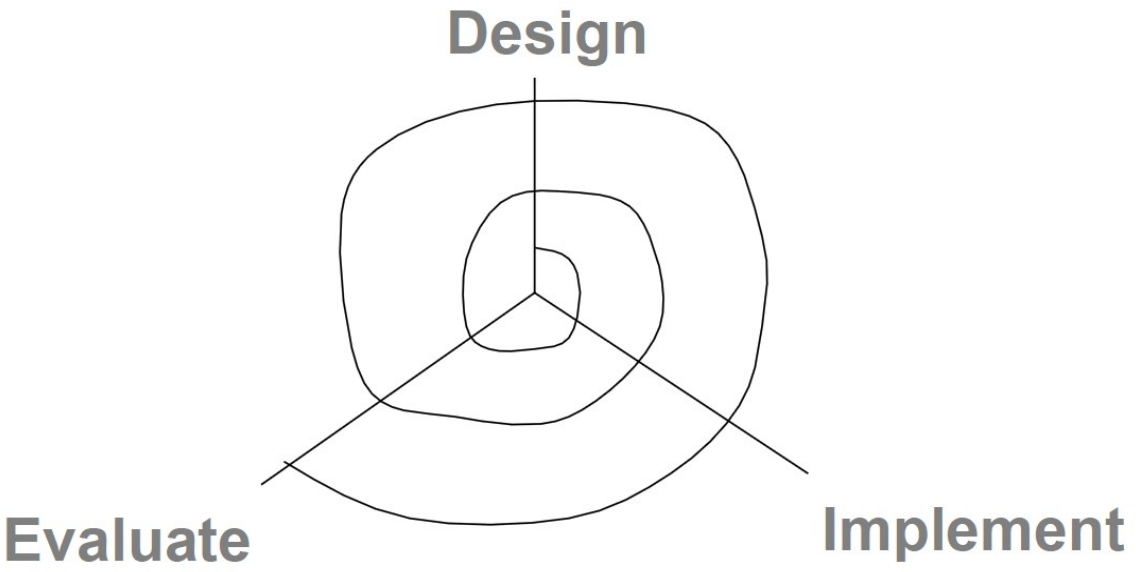
- El modelo clásico en cascada (y derivados) no es apropiado per se para IU:
  - Es un proceso más arriesgado y menos predecible de lo normal
  - El usuario no interviene hasta la fase de aceptación (desde la de requisitos)
  - Flaquezas en el proceso pueden provocar cambios sustanciales

Desarrollo de Interfaces Usables

- El modelo iterativo no es más que un modelo en cascada dividido en bloques
- El usuario interviene en la fase final de cada iteración
- Si las iteraciones separan distintas *releases*, los problemas son basicamente iguales
- ¿Cómo solucionar esto?

Desarrollo de Interfaces Usables

- Desarrollo en espiral



- La dimensión radial corresponde al coste de cada iteración
- Hay un mayor control del proceso

Modelo en Espiral

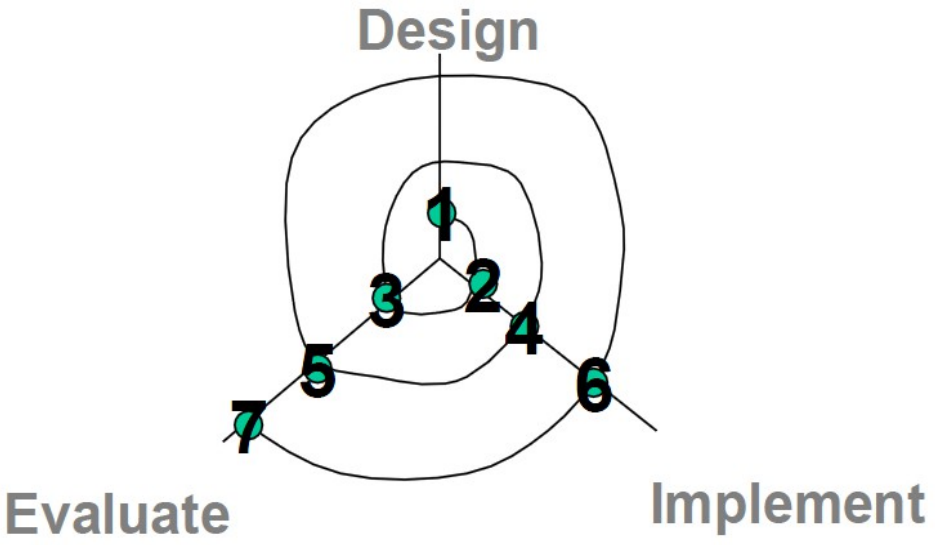
Source: Interface Hall of Shame

- Los errores serían más *baratos* de detectar en una etapa inicial (prototipado)
- Ahorro de costes de programación y tiempo de desarrollo
- Mayor satisfacción del usuario

Modelo en Espiral

- Iteraciones en el modelo en espiral
- Iteraciones iniciales usan prototipos más *baratos*
  - Habilita el diseño en paralelo de varios modelos para explorar diferentes vías
- Iteraciones más avanzadas poseen mayor detalle y riqueza de implementación
  - Los mayores riesgos ya han sido mitigados previamente
- Cada prototipo es evaluado por el usuario
- Solo iteraciones maduras son *sacadas a la luz*

Modelo en Espiral: ejemplo



1. Análisis de tareas/usuarios
2. Prototipo en papel
3. Testeo del prototipo con el usuario
4. Prototipo en PC
5. Evaluación heurística
6. Implementación
7. Evaluación del usuario

Desarrollo de Interfaces Usables

- A veces la terminología es confusa o solapada
- Según diferentes fuentes de la literatura a veces el diseño se divide en "investigación/análisis" y diseño
- Otras veces se incluye implementación y evaluación en la misma categoría
- En todo caso, las etapas (o subetapas) típicas del proceso son siempre las mismas independientemente de la etiqueta que se les ponga

Desarrollo de Interfaces Usables

- Análisis de usuarios/tareas
- Diseño de la solución propuesta
- Implementación de la solución
- Testeo de la solución (verificación y validación)

Análisis de usuarios/tareas

- Requiere la captura de información sobre el dominio, el entorno, los usuarios...
- Técnicas para recopilar información
  - Grupos focales de usuarios
  - Entrevistas individuales
  - Observación en el entorno de trabajo
  - Revisión de los procesos de workflow, tareas...
  - Documentar cómo los usuarios solucionan problemas particulares

Análisis de usuarios/tareas

- Una vez capturada la información hay que definir el problema y todos sus factores:
  - Actores
  - Roles
  - Tareas
  - Objetivos
  - Actividades
    ...

Diseño de la solución propuesta

- Creación de la solución:
  - Cómo debería de parecer el software para dar una solución al problema en cuestión
- Patrones de diseño a usar:
  - IU: Comando, Portal, Wizards,...
  - No-IU: MVC, Singleton, Fachadas,...
- Elección de la plataforma adecuada
- Estructura y *layout*
- Mockups, prototipos...

Diseño de solución propuesta

- Arquitectura
  - Lógica de presentación
    - Interacciones, tecnología, patrones de usabilidad
  - Lógica de negocio
    - Requisitos del dominio, casos de uso,...
  - Lógica de servicio
    - Gestión de datos, persistencia...
- Cada aspecto del diseño mencionado hasta ahora puede (y debe!) estar en iteraciones diferentes según el modelo en espiral

Diseño de la solución propuesta

- Técnicas
  - Desarrollo conjunto / sesiones de diseño colaborativo (CDS)
    - El equipo de trabajo colabora con los usuarios en diseñar bocetos
  - Arquitectura de la información
    - Taxonomía / Jerarquía de los objetos
    - Priorización de tareas y funcionalidades
  - Bocetos / Prototipos
    - Baja fidelidad: papel
    - Fidelidad media: Aplicación wireframe
    - Alta fidelidad: Aplicación de ejemplo
  - A más fidelidad, más coste de desarrollo

Implementación de la solución

- Se considera a construir un producto “entregable”
- Codificar funcionalidad
  - Escribir el código
  - Crear los gráficos
  - Crear las diferentes pantallas
  - Crear la navegación
  - ...

Prueba de la solución implementada

- Funcionalidad
  - ¿Hace lo que tiene que hacer?
- Usabilidad
  - Extraer feedback de los usuarios
- Pruebas de stress
  - Simulaciones de carga “reales”
- Recuperación ante errores
- Seguridad
- Alpha testing
  - Etapas iniciales de desarrollo con la funcionalidad apenas realizada
- Beta testing
  - Etapas más avanzadas del desarrollo pero todavía no en producción. Paso previo a la release.