

Diseño de sistemas de tiempo real

Juan Antonio de la Puente
DIT/UPM

Objetivos

- ◆ Repasaremos algunos conceptos de ingeniería de software y su aplicación a sistemas de tiempo real
- ◆ Introduciremos brevemente el método HRT-HOOD
- ◆ Analizaremos los lenguajes de programación y sistemas operativos más adecuados para realizar sistemas de tiempo real

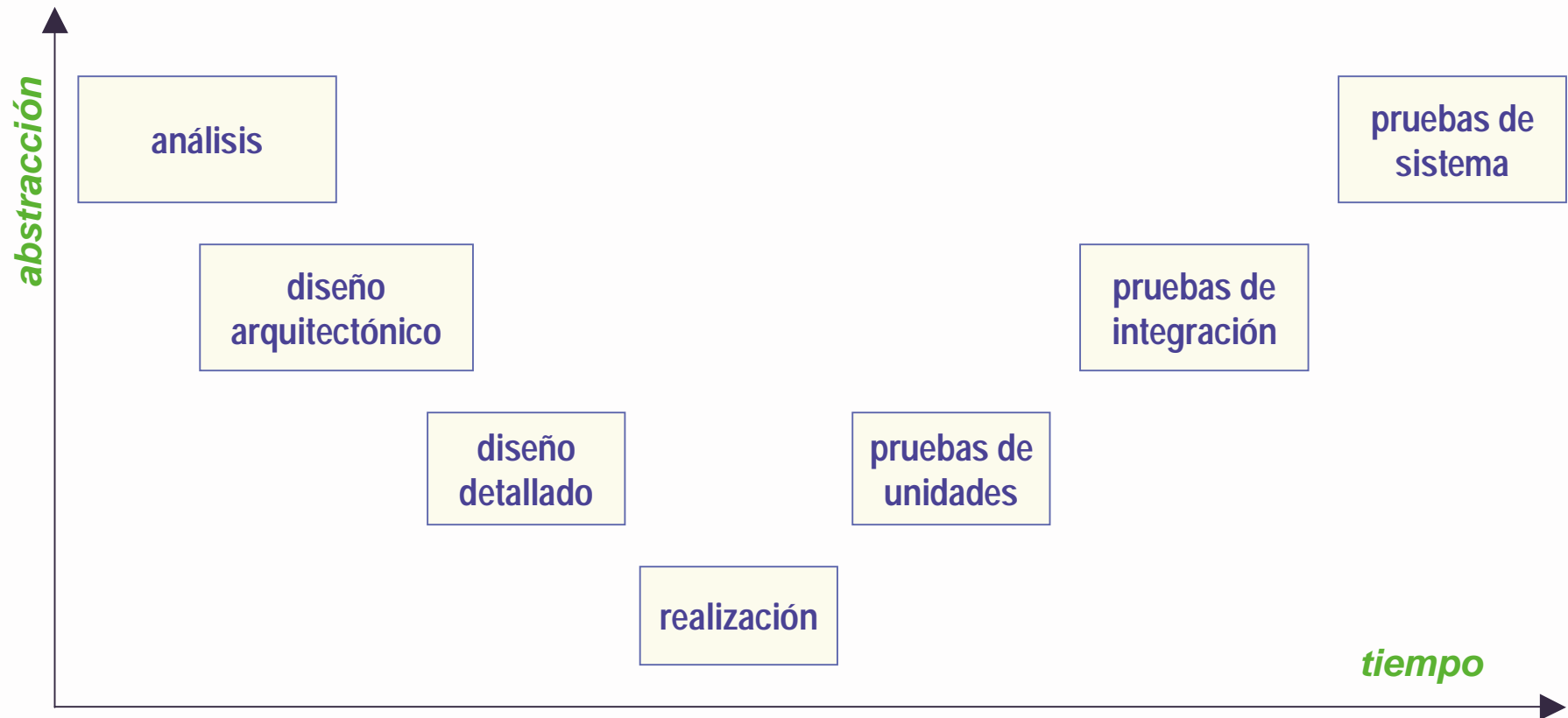
Motivación

- ◆ Los métodos y herramientas que se usan para construir otros tipos de sistemas no sirven para el software de tiempo real
 - no son suficientemente fiables
 - sólo contemplan el tiempo de respuesta medio, no el peor
 - no se garantizan los requisitos temporales
- ◆ Las plataformas de desarrollo y ejecución suelen ser diferentes
 - es difícil hacer pruebas en la plataforma de ejecución
 - es difícil medir los tiempos con precisión

Niveles de abstracción

- ◆ Los métodos de diseño de software comprenden una serie de transformaciones desde los requisitos iniciales hasta el código ejecutable
- ◆ Normalmente se consideran distintos niveles de abstracción en la descripción de un sistema:
 - Especificación de requisitos
 - Diseño arquitectónico
 - Diseño detallado
 - Codificación
 - Pruebas
- ◆ Cuanto más precisa sea la notación empleada en cada nivel mejor será la calidad del sistema final

Ciclo de desarrollo secuencial



Características

- ◆ Se descompone en una secuencia de *etapas*
 - Hay que completar cada etapa antes de empezar la siguiente
- ◆ Las *pruebas* se llevan a cabo después de la realización
 - Muchos errores se encuentran sólo al final
 - Volver atrás es muy costoso
 - A veces se hace sin documentar y de forma poco rigurosa
- ◆ Es mejor utilizar un proceso iterativo
 - Veremos uno centrado en la etapa de diseño
 - Se trata de *validar* todos los aspectos que se pueda en la etapa de diseño del sistema
 - Prestaremos especial atención a la validación del comportamiento temporal

HRT-HOOD

◆ HRT-HOOD

- *Hard Real-Time Hierarchical Object-Oriented Design*
- desarrollado por Burns & Wellings en 1994

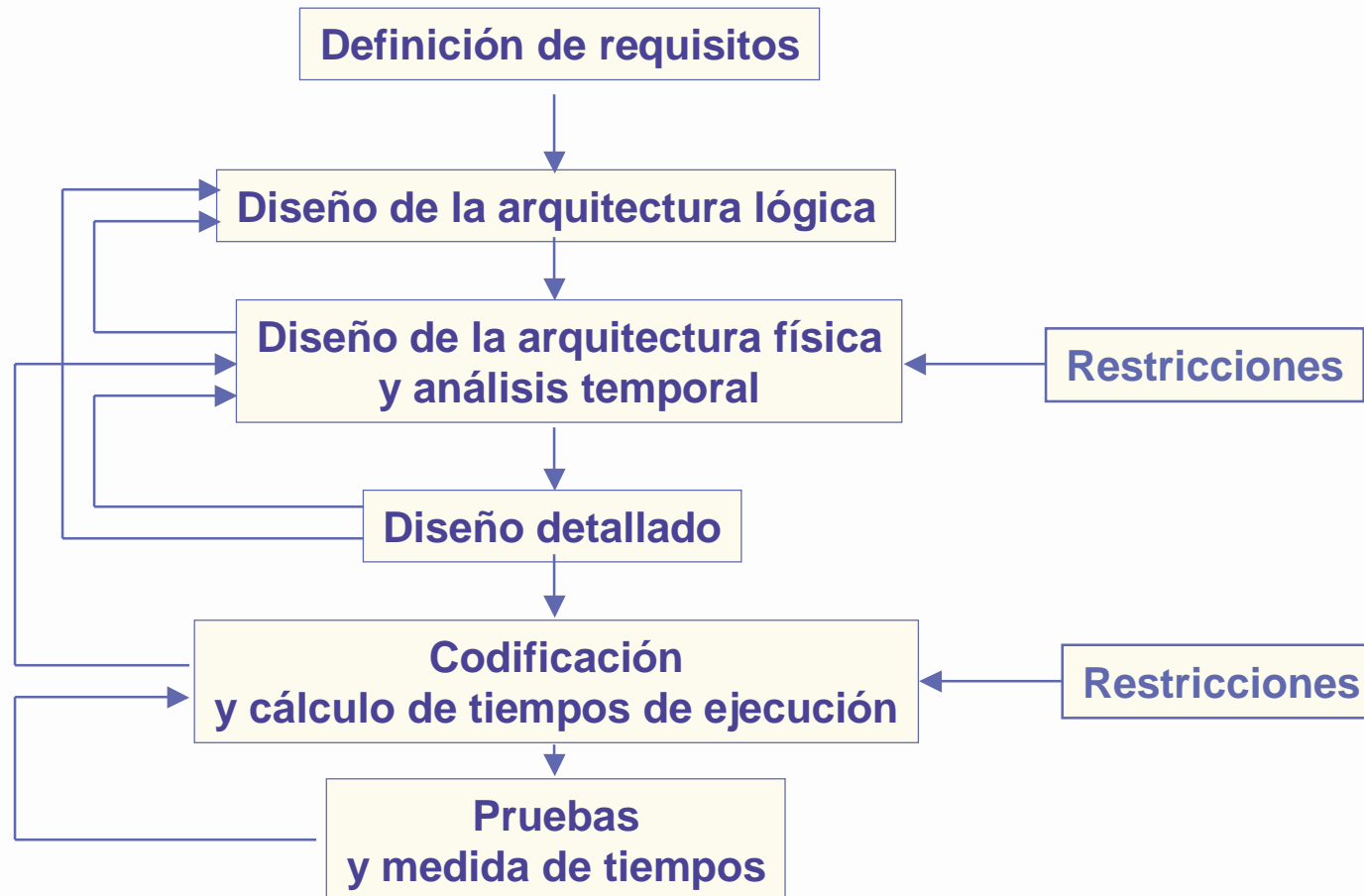
◆ Es un método de diseño estructurado, basado en objetos, para sistemas de tiempo real estricto

- derivado de HOOD (*Hierarchical Object-Oriented Design*)
- estándar en la Agencia Europea del Espacio (ESA)

◆ Principios

- abstracción
- descomposición jerárquica
- ocultamiento de información
- análisis temporal

Proceso de desarrollo iterativo



Características

- ◆ Elementos en cada nivel de abstracción
 - **Compromisos**: propiedades que ya no se cambiarán
 - **Obligaciones** que se dejan para niveles inferiores
- ◆ En el diseño se van transformando las obligaciones en compromisos
 - Este proceso está sujeto a **restricciones** impuestas por el entorno de ejecución
- ◆ Dos niveles de diseño arquitectónico
 - **Modelo lógico** - compromisos independientes del entorno
 - **Modelo físico** - compromisos dependientes del entorno

UML

◆ UML

- *Unified Modeling Language*
- desarrollado por Booch, Rumbaugh y Jacobson

◆ Notación basada en objetos con diversos aspectos

- clases y objetos
- casos de uso
- comportamiento
- paquetes
- estructura del software
- estructura física

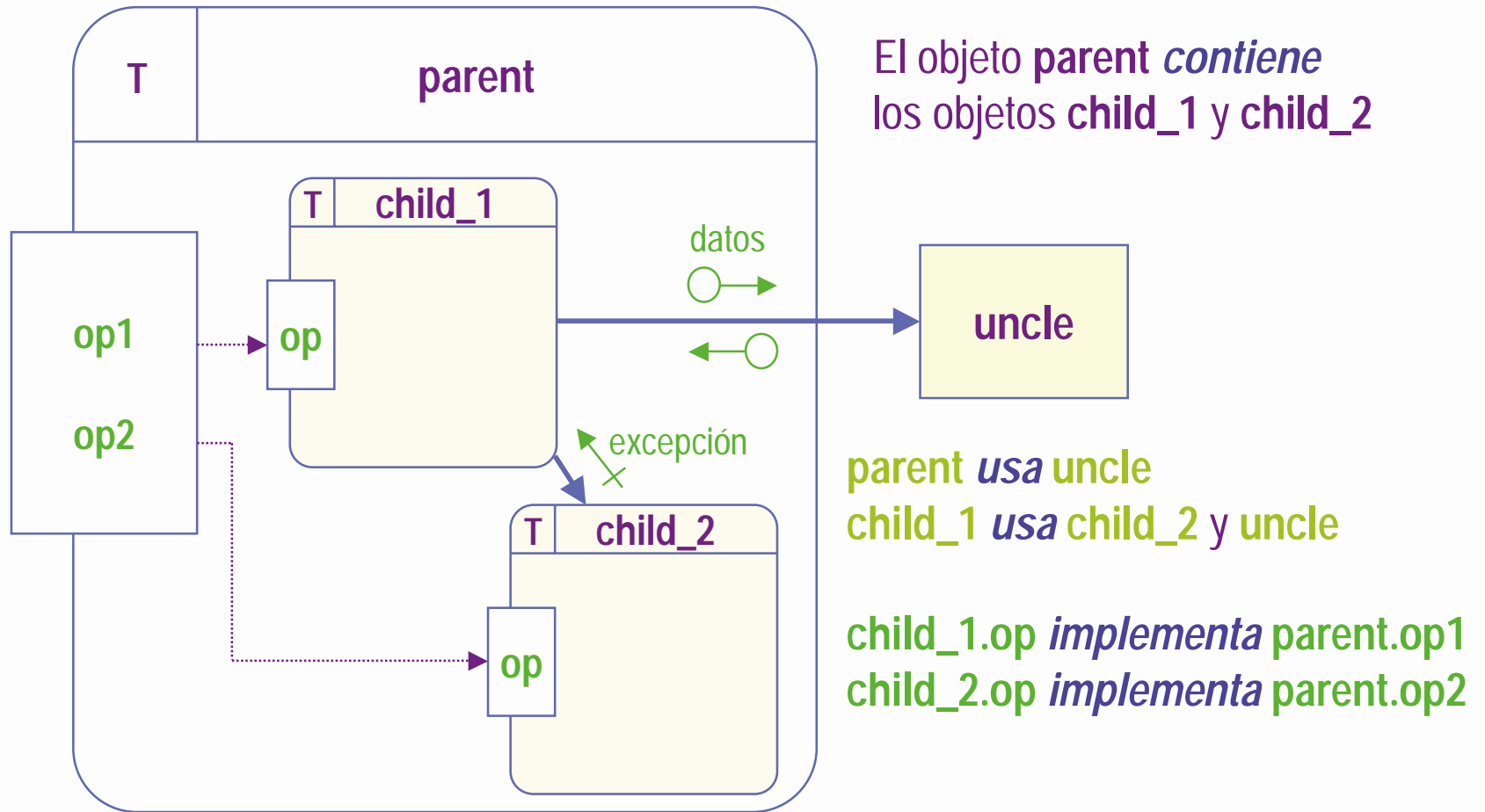
◆ UML es un lenguaje, no un método

◆ Se están desarrollando variantes para STR

Elementos de HRT-HOOD

- ◆ Un sistema se diseña como una jerarquía de **objetos abstractos**
 - un objeto se caracteriza por sus operaciones y su comportamiento (abstracción y ocultamiento de información)
 - cada objeto se puede descomponer en otros de más bajo nivel
 - modelo de objetos simple, sin herencia
 - » apropiado para sistemas de tiempo real
- ◆ Se puede **analizar el comportamiento temporal** si el entorno de ejecución es conocido y predecible
 - los objetos tienen atributos temporales
 - las relaciones entre objetos están restringidas para asegurar que el diseño se puede analizar

Objetos y relaciones



Tipos de objetos

◆ Pasivos

- » no controlan cuándo se ejecutan sus operaciones
- » no invocan operaciones de otros objetos espontáneamente

◆ Protegidos

- » pueden controlar cuándo se ejecutan sus operaciones
- » no invocan operaciones de otros objetos espontáneamente

◆ Activos

- » pueden controlar cuándo se ejecutan sus operaciones
- » pueden invocar operaciones de otros objetos espontáneamente

– Cíclicos

- » sus operaciones se ejecutan a intervalos regulares

– Esporádicos

- » sus operaciones se ejecutan cuando ocurre un suceso externo o interno

Diseño de sistemas y lenguajes de programación

- ◆ El diseño debe tener en cuenta lo que se puede hacer en los niveles de abstracción inferiores
- ◆ Los lenguajes de programación proporcionan la notación básica para la realización de los diseños
- ◆ La elección de un lenguaje es importante desde el punto de vista de la eficiencia y la fiabilidad del sistema
 - también tiene que ver con la productividad de la programación
 - un programa se escribe una vez, pero se lee y se modifica muchas durante las etapas de mantenimiento
 - la vida útil de los sistemas empotrados puede ser muy larga

Lenguajes de programación

- ◆ Un lenguaje de programación de sistemas de tiempo real debe facilitar la realización de sistemas
 - concurrentes,
 - fiables,
 - con un comportamiento temporal analizable
- ◆ Hay varias clases de lenguajes de interés para STR:
 - Lenguajes ensambladores
 - » flexibles y eficientes, pero costosos y poco fiables
 - Lenguajes secuenciales (Fortran, Pascal, C, C++)
 - » necesitan un SO para concurrencia y tiempo real
 - Lenguajes concurrentes (Modula, Ada, Java, ...)
 - » concurrencia y tiempo real incluidos en el lenguaje

C

- ◆ Es un lenguaje muy utilizado para programación de sistemas
- ◆ Es un lenguaje
 - estructurado, con bloques
 - sin tipado fuerte
 - muy flexible (pero a veces poco seguro)
- ◆ No tiene integrada la concurrencia ni el tiempo real
 - se consigue invocando servicios del sistema operativo de forma explícita
- ◆ No facilita la descomposición en módulos ni la programación con objetos
 - se puede hacer con C++
 - » extensión de C para programar con objetos
 - » no se suele usar en STR por problemas de fiabilidad

Ada

- ◆ Es un lenguaje diseñado específicamente para sistemas de tiempo real empotrados
 - concurrencia
 - tiempo real
 - acceso al hardware e interrupciones
- ◆ Es un lenguaje descendiente de Pascal
 - estructura en bloques
 - fuertemente tipado
- ◆ Está pensado para construir sistemas grandes y cambiantes
 - paquetes (módulos) y esquemas genéricos
 - extensión de tipos con herencia
 - biblioteca jerárquica
 - interfaces normalizadas con otros lenguajes (C, Fortran)

Ada 95

- ◆ Es la versión actual normalizada de Ada
- ◆ La norma define
 - un núcleo común para todas las implementaciones (*core language*)
 - unos *anexos* especializados para
 - » programación de sistemas
 - » sistemas de tiempo real
 - » sistemas distribuidos
 - » sistemas de información
 - » cálculo numérico
 - » fiabilidad y seguridad
 - Los anexos definen
 - » paquetes de biblioteca
 - » mecanismos de implementación

No añaden sintaxis ni vocabulario al lenguaje

Perfiles para sistemas críticos

- ◆ El documento *Guide for the use of the Ada programming language in high-integrity systems* define **subconjuntos seguros** de Ada para aplicaciones críticas
- ◆ **SPARK** es un subconjunto / extensión de Ada que permite el uso de técnicas de análisis estático
- ◆ El **perfil de Ravenscar** define un subconjunto seguro de la parte concurrente de Ada, y los correspondientes servicios de sistema operativo

Java

- ◆ Es un lenguaje pensado para construir sistemas distribuidos
 - basado en objetos dinámicos
 - con concurrencia integrada en el lenguaje
 - bibliotecas de clases (APIs) muy útiles
 - pensado para que el código objeto sea portátil
 - » interpretado por una *máquina virtual (JVM)*
 - » “*write once, run everywhere*”
- ◆ La definición original no es adecuada para tiempo real
 - la planificación de actividades concurrentes no está bien definida
 - los mecanismos de sincronización son inadecuados
 - la gestión dinámica de memoria introduce indeterminismo
 - la medida del tiempo no es suficientemente precisa
 - otros problemas con excepciones y concurrencia

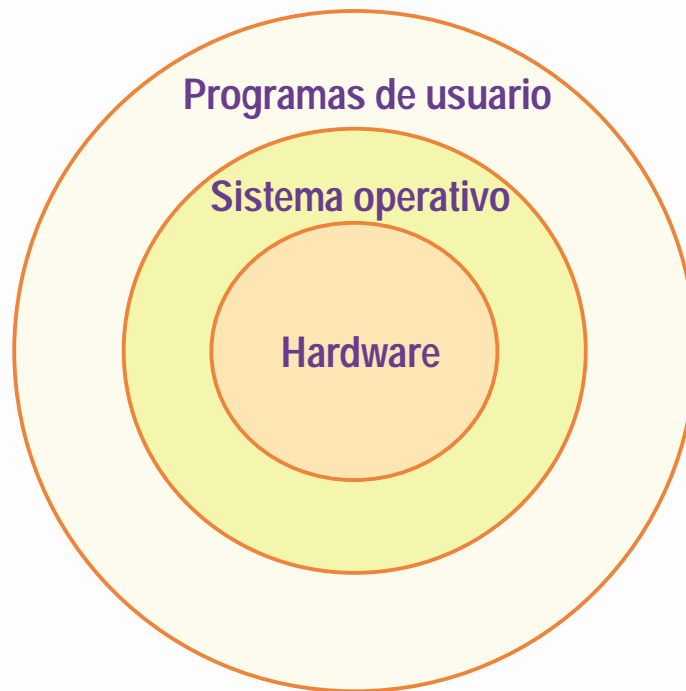
Java para tiempo real

- ◆ Hay varias propuestas de modificaciones para usar Java en sistemas de tiempo real
 - NIST *Requirements for Real-Time Extensions to Java* (1999)
 - » no modificar la sintaxis, coexistencia con aplicaciones convencionales
 - Java Real-time Experts Group (Sun & otros)
 - » *Real-Time Specification for Java* (2000)
 - » basada en un máquina virtual extendida para STR
 - Real-Time Java Working Group (J-Consortium)
 - » *Real-Time Core Specification*
 - » basada en una máquina virtual separada para STR
- ◆ Todavía no hay compiladores ni máquinas virtuales para Java de tiempo real

Sistemas operativos

- ◆ Los sistemas operativos convencionales no son adecuados para realizar sistemas de tiempo real
 - no tienen un comportamiento determinista
 - no permiten garantizar los tiempos de respuesta
 - algunos de ellos son poco fiables
- ◆ Un sistema operativo de tiempo real (SOTR) debe soportar
 - concurrencia: procesos ligeros (threads) con memoria común
 - temporización: medida de tiempos y ejecución periódica
 - planificación determinista
 - » ej.: prioridades fijas con desalojo
 - dispositivos de E/S: acceso a recursos de hardware e interrupciones

Arquitectura de software y SO (1)

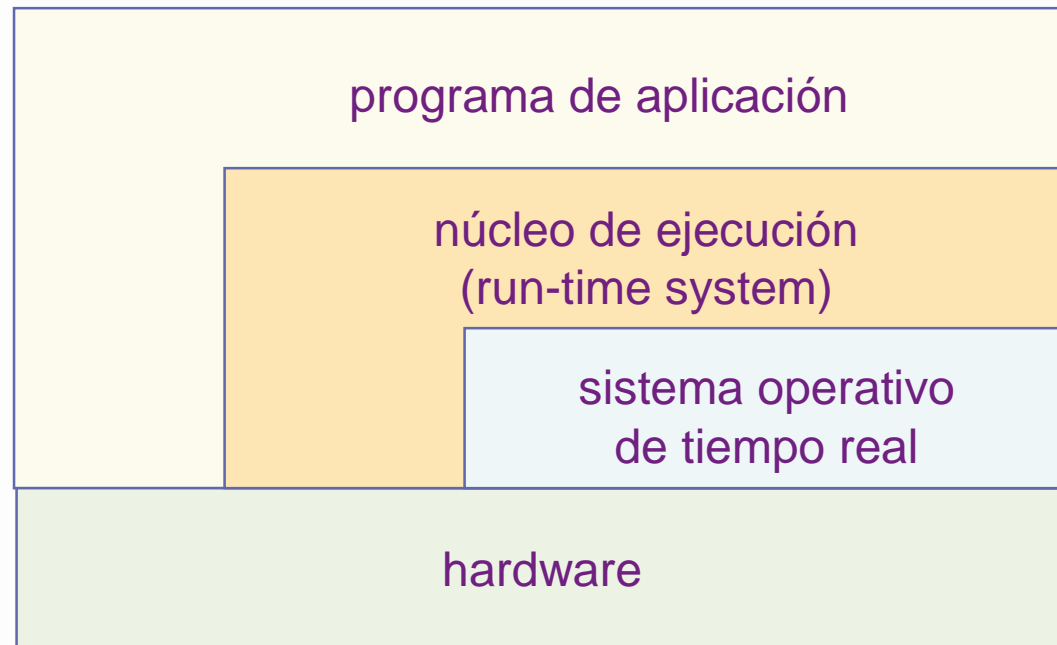


Aplicación convencional



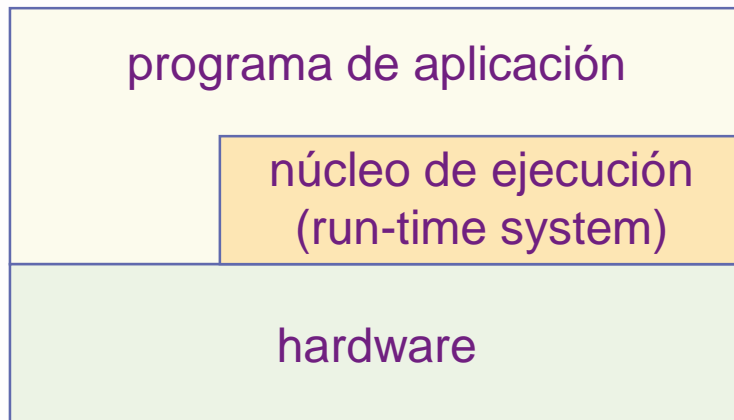
Sistema empujado

Arquitectura de software (2)

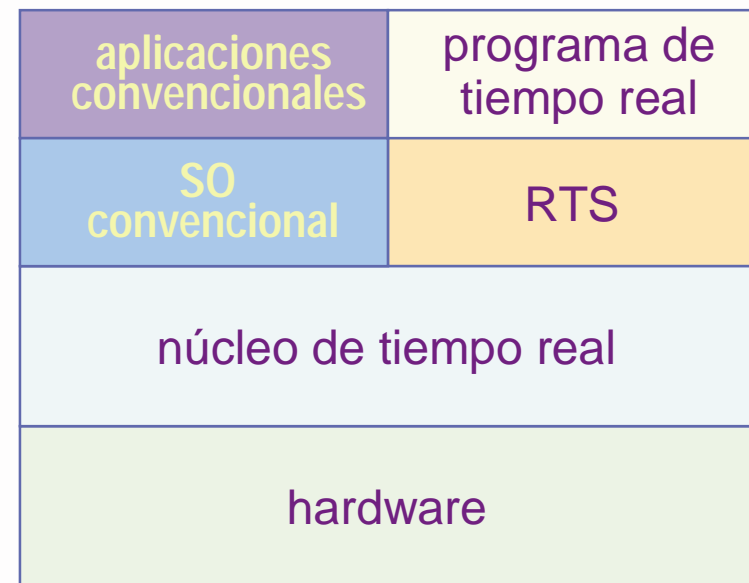


Arquitectura con sistema operativo de tiempo real

Arquitectura de software (3)



Arquitectura con máquina desnuda



Arquitectura con máquina virtual

POSIX

- ◆ Es un conjunto de normas IEEE/ISO que definen interfaces de sistemas operativos
- ◆ Permiten desarrollar software portátil y reutilizable (*Portable Operating System Interface*)
- ◆ Las normas definen servicios que se pueden incluir o no en un sistema operativo particular
- ◆ Además se definen *perfiles de aplicación* con conjuntos de servicios estándar
- ◆ Hay interfaces para C, Ada, y otros lenguajes

Normas POSIX

POSIX 1, 1a	Interfaz básica similar a UNIX™
POSIX 1b,1d,1j	Extensiones de tiempo real
POSIX 1c	Procesos ligeros (<i>threads</i>)
POSIX 1e	Seguridad
POSIX 1f	NFS
POSIX 1g	Servicios de red (sockets, XTI)
POSIX 1h	Tolerancia de fallos
POSIX 5,5a,5b	Interfaces para Ada
POSIX 13	Perfiles para sistemas de tiempo real
POSIX 21	Comunicaciones de tiempo real

Perfiles de aplicación

- ◆ Definen subconjuntos de servicios para distintos tipos de aplicaciones

- ◆ **POSIX 13** : Perfiles para sistemas de tiempo real
 - **PSE50 : sistema de tiempo real mínimo**
 - » sin gestión de memoria, ficheros ni terminal
 - » sólo *threads* (no procesos pesados)
 - **PSE51 : controlador de tiempo real**
 - » tiene sistema de ficheros y terminal
 - **PSE52 : sistema de tiempo real dedicado**
 - » tiene gestión de memoria y procesos pesados
 - **PSE53 : sistema de tiempo real generalizado**
 - » sistema completo con todo tipo de servicios

Ejemplos de SOTR

- ◆ LynxOS
- ◆ pSOS
- ◆ QNX
- ◆ VRTX
- ◆ VxWorks
- ◆ RTEMS
- ◆ RT-Linux
- ◆ **Marte** — Universidad de Cantabria
 - perfil POSIX PSE50
 - para sistemas empotrados en PCx86
- ◆ **Open Ravenscar Kernel (ORK)** — DIT/UPM
 - núcleo de SOTR para Ada / Ravenscar

Resumen

- ◆ Los métodos y herramientas convencionales no son adecuados para desarrollar sistemas de tiempo real
- ◆ HRT-HOOD es un método de diseño bien adaptado a este tipo de sistemas
- ◆ En el curso usaremos algunos lenguajes de programación para ilustrar los conceptos más importantes de los STR
 - C / POSIX
 - Ada 95
 - RT Java