UD 2: PROGRAMACIÓN CONCURRENTE Y SISTEMAS DE TIEMPO REAL

TEMA 4: INTRODUCCIÓN A LOS SISTEMAS DE TIEMPO REAL

« Me lo explicaron y lo olvidé, lo ví y lo aprendí, lo hice y lo entendí.»

- Confucio -



industriales

etsii UPCT



Universidad Politécnica de Cartagena

INTRODUCCIÓN

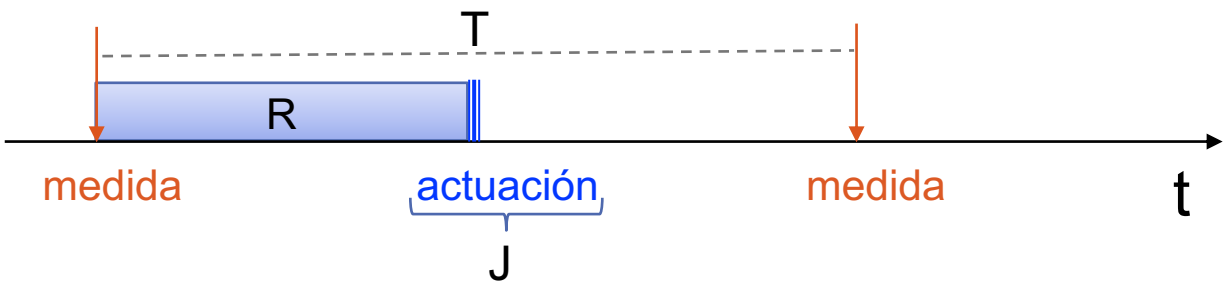
- Un sistema de tiempo real (STR) es un sistema de procesamiento de información que debe responder a estímulos de entrada generados externamente en un periodo de tiempo finito y especificado
  - La corrección depende no solo del resultado lógico, sino también del tiempo en que fue obtenido
  - El fallo en responder a tiempo es tan malo como una respuesta errónea

ORIGEN DE LOS REQUISITOS TEMPORALES

- *Teoría de control discreto*: el periodo de muestreo (T) aparece tanto en la señal de entrada como en la acción de control

$$s = \frac{z - 1}{T}$$

- Aunque el control sea digital (e.g., garaje) o no se actúe sobre el entorno físico (e.g., televisor), es necesario tener cierto control sobre el tiempo en que se realizan las acciones



J es la variación (*jitter*) en el tiempo de respuesta:
$J = R_{MAX} - R_{MIN}$

CLASIFICACIÓN DE LOS SISTEMAS TR

- Requisitos temporales y de fiabilidad: críticos o acríticos
- Comportamiento en caso de avería: parada segura o degradación aceptable
- Grado de determinismo temporal: respuesta garantizada o máximo esfuerzo
- Ejecución de actividades: dirigido por tiempo o por eventos

Una aplicación puede tener subsistemas de varios tipos mezclados

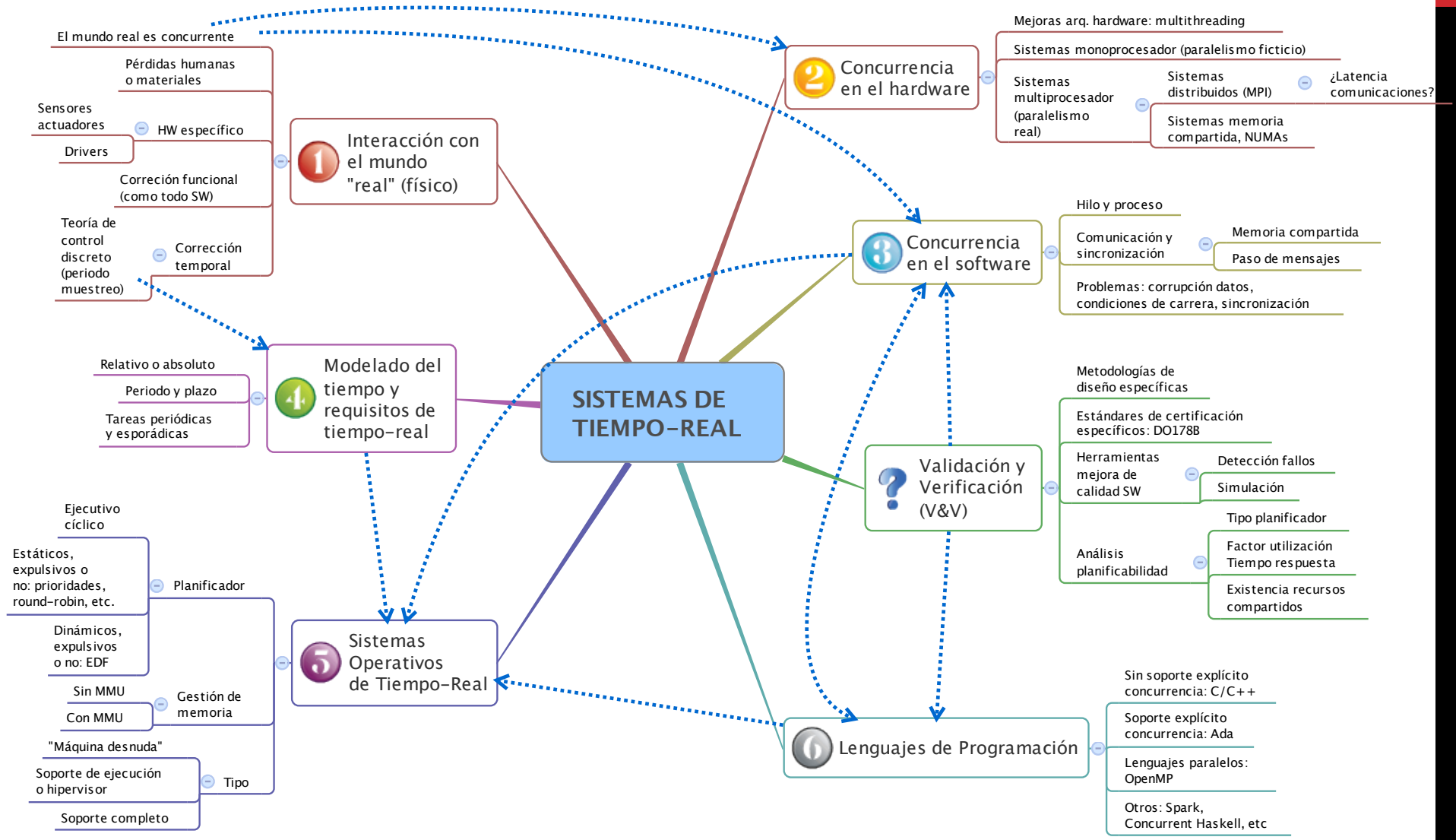
- **TR estricto** *(hard)*: es absolutamente imperativo que no se pierda ningún plazo (e.g., control de vuelo)
- **TR suave** *(soft)*: es tolerable recibir datos fuera de plazo eventualmente (e.g., adquisición de datos)
- **TR firme** *(firm)*: como el TR suave, pero sin beneficio en la recepción tardía de datos (e.g., un televisor)

- Sistemas con **parada segura** (*fail-safe*): detención en un estado seguro
- Sistemas con **degradación aceptable** (*fail-soft*): funcionamiento con pérdida parcial de funcionalidad o prestaciones

EJECUCIÓN DE ACTIVIDADES

- Actividad: unidad básica de computación con requisitos temporales
- Ejecución por **paso tiempo** (*time-triggered*)
  - Lanzar actividad a las 9:00
  - Lazar actividad cada 25ms (actividad periódica)
- Ejecución por **evento** (*event-triggered*)
  - Eventos internos (realización de un cómputo) o externos (interrupciones)
  - Actividad **esporádica**: el intervalo (tiempo) entre dos eventos sucesivos está acotado
  - Actividad **aperiódica**: no existe dicho intervalo

MAPA MENTAL


SISTEMAS DE TIEMPO-REAL
1 Interacción con el mundo "real" (físico)
El mundo real es concurrente
Pérdidas humanas o materiales
Sensores actuadores
HW específico
Drivers
Correción funcional (como todo SW)
Teoría de control discreto (periodo muestreo)
Corrección temporal
2 Concurrencia en el hardware
Mejoras arq. hardware: multithreading
Sistemas monoprocesador (paralelismo ficticio)
Sistemas multiprocesador (paralelismo real)
Sistemas distribuidos (MPI)
¿Latencia comunicaciones?
Sistemas memoria compartida, NUMAs
3 Concurrencia en el software
Hilo y proceso
Comunicación y sincronización
Memoria compartida
Paso de mensajes
Problemas: corrupción datos, condiciones de carrera, sincronización
4 Modelado del tiempo y requisitos de tiempo-real
Relativo o absoluto
Periodo y plazo
Tareas periódicas y esporádicas
Validación y Verificación (V&V)
Metodologías de diseño específicas
Estándares de certificación específicos: DO178B
Herramientas mejora de calidad SW
Detección fallos
Simulación
Análisis planificabilidad
Tipo planificador
Factor utilización Tiempo respuesta
Existencia recursos compartidos
5 Sistemas Operativos de Tiempo-Real
Ejecutivo cíclico
Estáticos, expulsivos o no: prioridades, round-robin, etc.
Planificador
Dinámicos, expulsivos o no: EDF
Sin MMU
Con MMU
Gestión de memoria
"Máquina desnuda"
Soporte de ejecución o hipervisor
Tipo
Soporte completo
6 Lenguajes de Programación
Sin soporte explícito concurrencia: C/C++
Soporte explícito concurrencia: Ada
Lenguajes paralelos: OpenMP
Otros: Spark, Concurrent Haskell, etc

ARQUITECTURA Y DISEÑO SOFTWARE







- ¿En qué casos haríais planos? ¿Cuántos? ¿Medidas de seguridad? ¿Cálculos estructurales? ¿Personal involucrado? ¿Herramientas?
- En el desarrollo de software también es necesario realizarse estas preguntas, aunque adaptado a sus características

FASES DEL PROCESO DESARROLLO

"Software Engineering is the study and application of engineering to the design, development, and maintenance of software."

IEEE

1. Definición de requisitos del SW
2. Diseño arquitectónico (alto nivel)
3. Implementación del SW (nivel detalle)
4. Transferencia del producto al cliente
5. Evolución del producto, incluyendo mantenimiento y retirada

LENGUAJES DE PROGRAMACIÓN PARA STR

- Lenguajes ensamblador
  - Flexibles y eficientes, pero costosos y poco fiables
- Lenguajes secuenciales (Fortran, C, etc.)
  - Concurrencia y tiempo real proporcionado por SO
  - Extensiones específicas para computación paralela (OpenMP, MPI)
- Lenguajes concurrentes (Ada, Java, C++11, etc.)
  - Concurrencia y tiempo real incluidos en el lenguaje
  - Necesitan soporte por parte del SO

HERRAMIENTAS DE DESARROLLO

- Depurador y pruebas unitarias, de regresión, de integración (*googletest*)
- Control de versiones (*git*)
- Análisis del código fuente (*cppcheck*) y en tiempo de ejecución (*profilers, gprof*, *valgrind*)
- Construcción (*Makefile*, *cmake*) y gestión de paquetes (*Conan.io*)



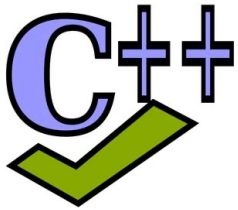
googletest
Google C++ Testing Framework





CONAN.io
C/C++ Package Manager







SISTEMAS OPERATIVOS

- Los SOs convencionales no son adecuados para TR
  - no tienen un comportamiento determinista
  - no permiten garantizar los tiempos de respuesta
  - algunos de ellos son poco fiables
- Un SO de tiempo real debe soportar
  - concurrencia: procesos ligeros (*threads*) con memoria compartida
  - temporización: medida de tiempos y ejecución periódica
  - planificación determinista: prioridades fijas con desalojo
  - acceso a recursos hardware e interrupciones

FREERTOS EN TELEVISOR 4K

