Informática Industrial
Presentación de la Asignatura

ESCUELA DE INGENIERÍAS INDUSTRIALES

Departamento de Ingeniería de Sistemas y Automática. EII.

Universidad de Valladolid

Presentación de la Asignatura.
Informática Industrial. Curso 2014/15

- Profesorado
- Horario
- Objetivos
- Programa
- Método y Criterios de Evaluación
- Exámenes
- Bibliografía
- Conocimientos Previos Recomendados

Profesorado

Dpto. Ingeniería de Sistemas y Automática

Eusebio de la Fuente e-mail: efuente@eii.uva.es

Rogelio Mazaeda Echevarría rogelio@cta.uva.es

Tutorías.

Eusebio de la Fuente López L 11:00-13:00 Sede Mendizábal M 10:00-12:00 Sede Mendizábal J 10:00-12:00 Sede Mendizábal	Rogelio Mazaeda Echevarría L 09:00-13:00 Sede Mergelina M 09:00-13:00 Sede Mergelina X 11:00-12:00 Sede Mendizábal V 09:00-12:30 Sede Mergelina

Horario

	LUNES	MARTES	MIÉRCOLES		JUEVES	VIERNES
8:00-9:00		Laboratorio Gpo 3 LAB-ISA 2				Laboratorio Gpo 2 (4sem: 1,2,3,4) LAB-ISA 1
9:00-10:00					Teoría. 7 sem. (9-15)	
10:00-11:00	Teoría					
11:00-12:00						
12:00-13:00		Laboratorio Gpo 2 LAB-ISA 1	Laboratorio Gpo 1 LAB-ISA 1	Lab. Gpo 3 (4sem: 1,2,3,4) LAB-ISA 2	Laboratorio Gpo 1 (4sem: 1,2,3,4) LAB-ISA 1	
13:00-14:00						

Objetivos

Conocer la programación orientada a objeto. Ser capaz de realizar programas en el lenguaje C++.

Conocer los principios y modelos fundamentales de la **programación concurrente**. Ser capaz de implementar programas concurrentes sencillos utilizando el lenguaje de programación C++.

Conocer los conceptos fundamentales de los sistemas de **tiempo real**. Entender cómo se aplican los conceptos de programación concurrente y tiempo real en controladores industriales.

Comprender las diferentes arquitecturas y formas de utilización del ordenador y la **informática las aplicaciones industriales**.

Conocer las implicaciones de la **seguridad y la tolerancia a fallos** aplicada a la informática industrial.

Programa Teoría

Bloque 1. Programación concurrente y Sistemas de Tiempo Real

1. Programación en C++ y estructuras de datos.
2. Introducción a los sistemas en tiempo real.
3. Programación concurrente.
4. Comunicación y Sincronización de Procesos.
5. Capacidades de tiempo real

Bloque 2. El computador en control y supervisión de la producción.

6. El computador en la automatización de la producción
7. Aplicaciones de control distribuido y supervisión.
8. La seguridad y la tolerancia a fallos en sist informáticos industriales.

Programa básico de prácticas

1. Programación en lenguaje C++ de estructuras de datos.

2. Programación Concurrente. Comunicación entre procesos (IPC): colas de mensajes, semáforos, memoria compartida...

Laboratorios. Distribución por grupos

Laboratorios:

Laboratorio ISA 1 (1ª planta): grupos 1L y 2L
Laboratorio ISA 2 (3ª planta): grupo 3L

Grupos:

Grupo 1L	De A a GAL…	laboratorio ISA 1 (1ª planta)
Grupo 2L	De GAM… a M	laboratorio ISA 1 (1ª planta)
Grupo 3L	De N a Z	laboratorio ISA 2 (3ª planta)

Método y Criterios de Evaluación

- **El 50 % de la nota en un examen final.**
- El **examen** constará de:

  - Cuestiones cortas a responder en no más de cinco líneas.
  - Problemas cortos.
  - Cuestiones sobre programación.

Método y Criterios de Evaluación

- **El 50% restante se otorgará en base a la evaluación de las prácticas.**
- **Prácticas.** Se entregará una memoria y el código de cada práctica antes de la fecha indicada. Los trabajos se revisarán personalmente con el profesor para la evaluación definitiva de la práctica.
- La evaluación de las prácticas se realizará teniendo en cuenta:
  - Informe de la práctica: 25% de la nota.
  - Funcionamiento y desempeño de la práctica: 50% de la nota.
  - Estructura y legibilidad del código: 25% de la nota.

Método y Criterios de Evaluación

- **50%** exámen + **50%** prácticas
- Para superar la asignatura**, se deberán aprobar ambas partes.**

Exámenes

- Ordinario

  viernes, 23 de enero de 2015 a las 9:00

- Extraordinario:

  martes, 30 de junio de 2015 a las 9:00

Conocimientos Previos

- **Fundamentos de informática**: principios de programación y de Sistemas Operativos
- **Fundamentos de Automática**: sistemas dinámicos, realimentación, estabilidad y diseño de controladores.
- **Sistemas de Producción y Fabricación:** conceptos de automatización industrial.

TEMA 1
Introducción. Hoja de ruta para el estudio de la informática en la industria

ESCUELA DE INGENIERÍAS INDUSTRIALES

Departamento de Ingeniería de Sistemas y Automática. EII.

Universidad de Valladolid

Indice

1. Importancia de la informática en la industria.
2. Retos que presenta el estudio de la Informática Industrial.
3. Diferentes dimensiones al enfrentar el estudio de la Informática Industrial.
    1. Implicaciones desde teoría de control automático.
    2. Necesidad de concurrencia y tiempo real.
    3. Sistemas industriales programables.
    4. Comunicaciones Industriales.
    5. Seguridad.

Importancia de la Informática Industrial

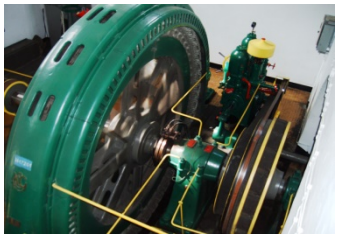
Integración mecánica

1770

Máquina de vapor, regulador de Watts

Integración eléctrica

1870



Invención de la dinamo. accionadores eléctricos, Irrupción de la electrónica

Integración informática

1970

Microprocesadores, Redes de ordenadores

Importancia de la Informática Industrial

Papel de la II:

Vinculado al concepto de automatización en su sentido más amplio:

Sustitución (total o parcial) de la labor humana en los procesos industriales involucrados.

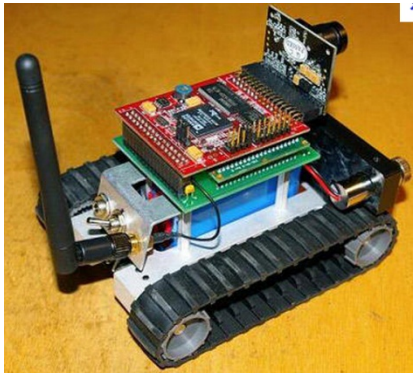
Informática Industrial. Retos.

Retos:

- Objeto de estudio amplio y dinámico
- Diferentes posibles campos de aplicación de la informática industrial.

Programas a diferentes niveles en el control de la industria

Programas empotrados conformando el producto



CAD/CAM/CAE (ciclo de vida del producto)

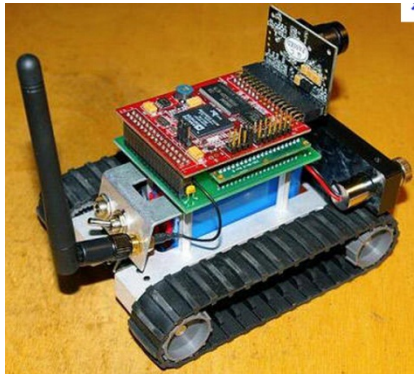
Informática Industrial. Retos.

Retos:

- Objeto de estudio amplio y dinámico
- Diferentes posibles campos de aplicación de la informática industrial.

Programas a diferentes niveles en el control de la industria

Programas empotrados conformando el producto



CAD/CAM/CAE (ciclo de vida del producto)

¿Cómo abordar ese estudio?

Informática Industrial. Retos.

¿Cómo abordar ese estudio?

CAD/CAM/CAE (ciclo de vida del producto)

Es un tema demasiado específico que debe ser abordado a profundidad en las respectivas disciplinas particulares:

- Ingeniería eléctrica.
- Ingeniería electrónica.
- Ingeniería mecánica.
- Ingeniería química.
- Ingeniería de control.

Informática Industrial. Retos.

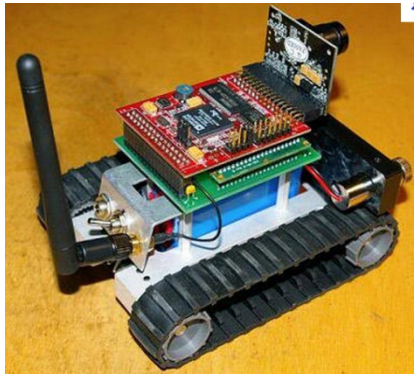
¿Cómo abordar ese estudio?

Apelar a elementos comunes básicos

Programas a diferentes niveles en el control de la industria

- Necesitan lidiar con el mundo físico.
- Presencia del control por realimentación.
- Aplicaciones distribuidas.

Programas empotrados conformando el producto

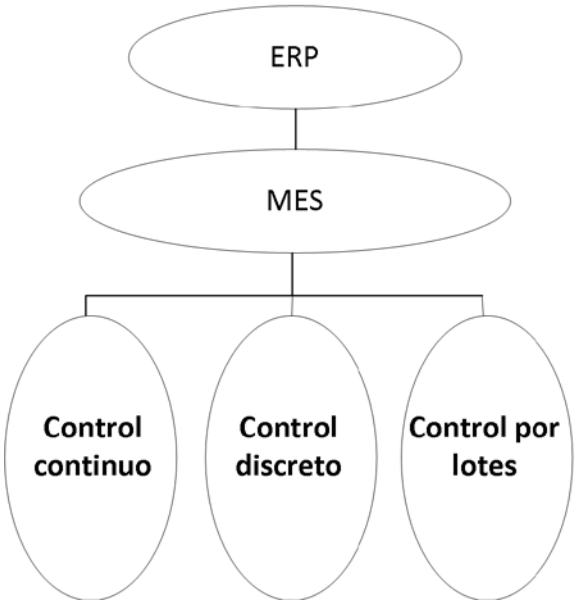


Informática Industrial. Pirámide industrial.

Retos:

¿Cómo encontrar relato coherente?

Diferentes necesidades a diferentes niveles.



Niveles:

- **Nivel ERP**: Gerencia y planificación económica a largo plazo.
- **Nivel MES**: Planificación operativa de la producción.
- **Nivel de Control y de campo**: dispositivo de medición, actuadores, controladores (CDD, PLC, DCS, SCADAS).

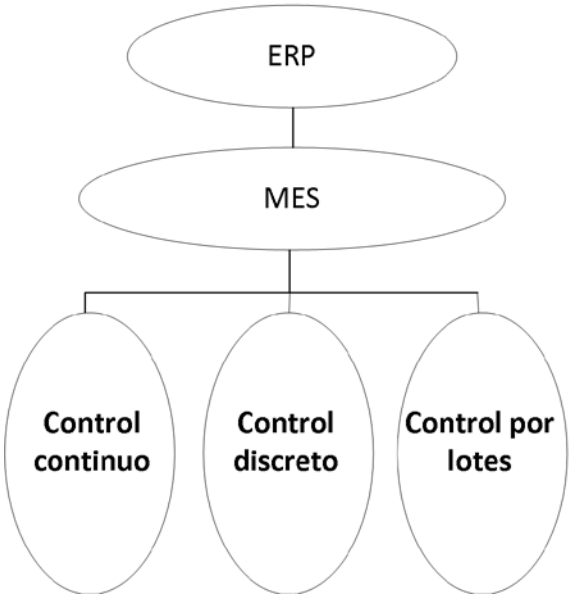
ISA-95

Informática Industrial. Pirámide industrial.

Retos:

¿Cómo encontrar relato coherente?

Diferentes necesidades a diferentes niveles.



- Las aplicaciones informáticas a cualquier nivel (sobre todo de MES hacia abajo) se benefician de programación concurrente y de sistemas de tipo real.
- Las periodicidades y los *deadlines* son más exigentes en los niveles inferiores.

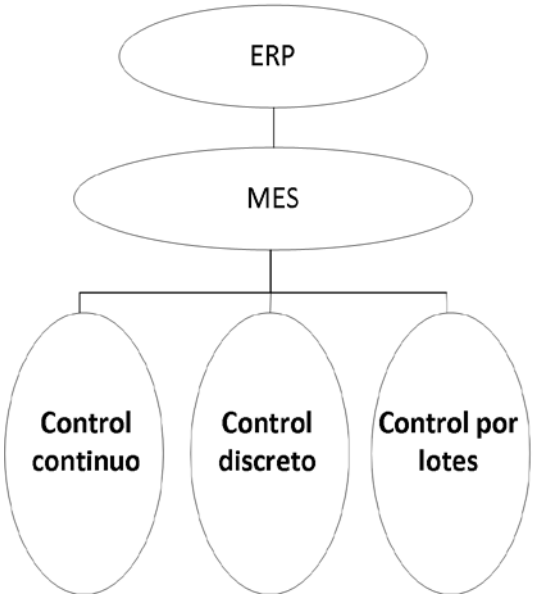
ISA-95

Informática Industrial. Pirámide industrial.

Retos:

¿Cómo encontrar relato coherente?

Diferentes necesidades a diferentes niveles.



ISA-95

ISA-95 caracteriza los diferentes tipos de industria:

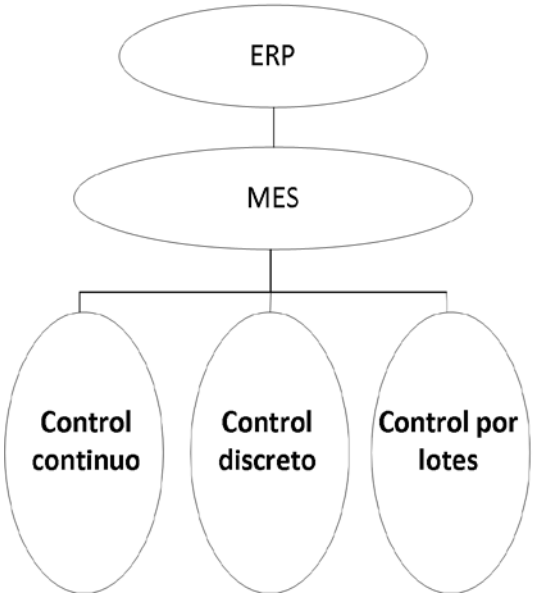
- **Control continuo**: típico de la industria de procesos. Ej: regulación de variables continuas como temperatura, presión, etc.
- **Control discreto**: típico de proceso de fabricación discreta de unidades contables. Ej: fabricación automóviles, semiconductores.
- **Control batch**: Se da en la industria de procesos. Combina elementos de producción continua con decisiones discretas.

Informática Industrial. Pirámide industrial.

Retos:

¿Cómo encontrar relato coherente?

Diferentes necesidades a diferentes niveles.



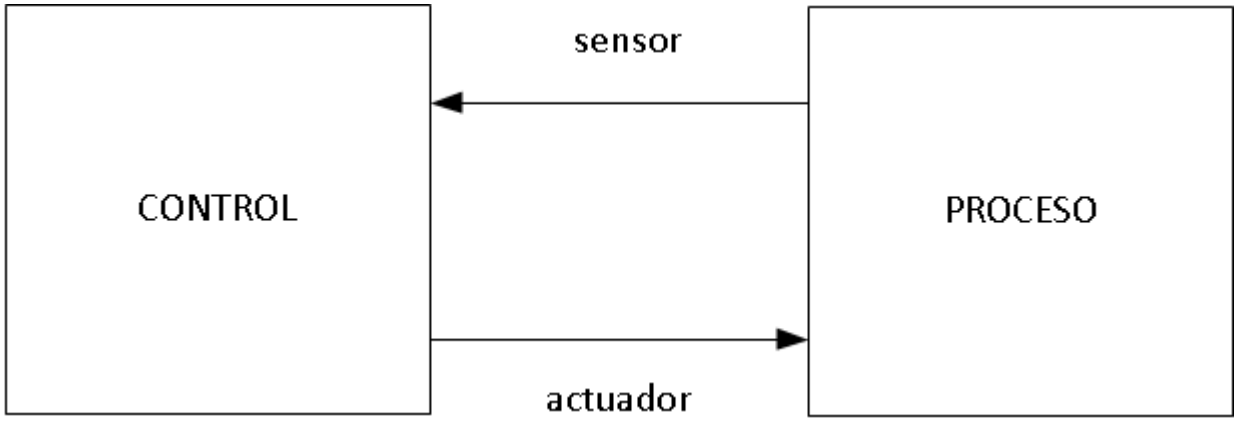
ISA-95

ISA-95 caracteriza los diferentes tipos de industria:

- **Control continuo**: típico de la industria de procesos. Ej: regulación de variables continuas como temperatura, presión, etc.
- **Control discreto**: típico de la fabricación discreta de unidades contables. Ej: fabricación automóviles, semiconductores.
- **Control batch**: En la industria de procesos. Combina elementos de producción continua con decisiones discretas.

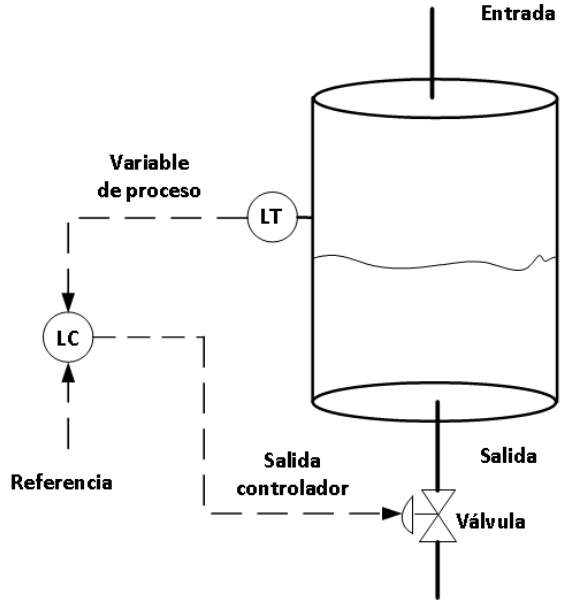
Informática Industrial. Sistemas reactivos.

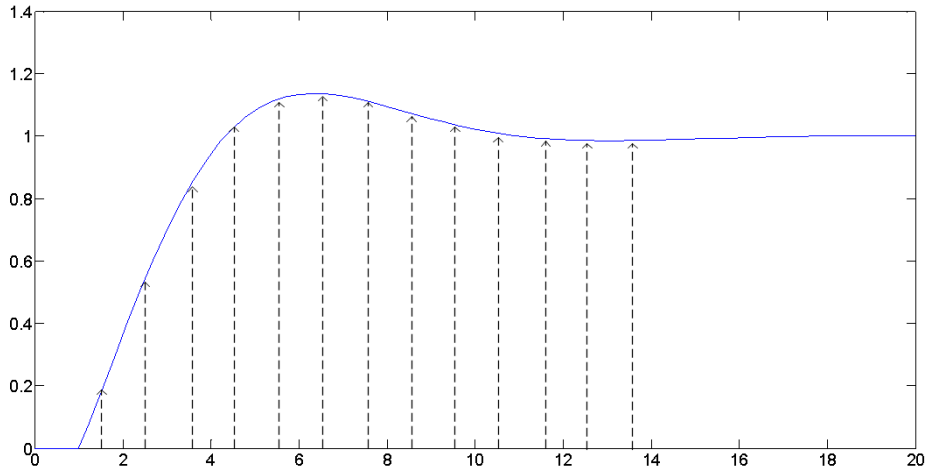
El **controlador industrial por computador** como **sistema reactivo**: lee desde sensores, aplica algoritmo de control y envía a actuadores.
La **planta física** y su **control por ordenado**r como sistemas reactivos colocados **frente a frente**.

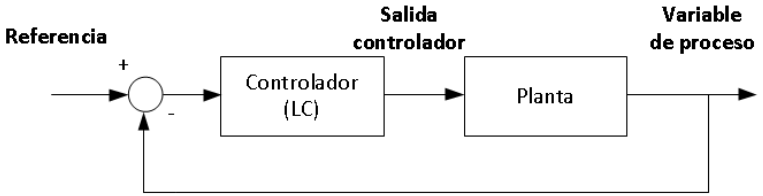


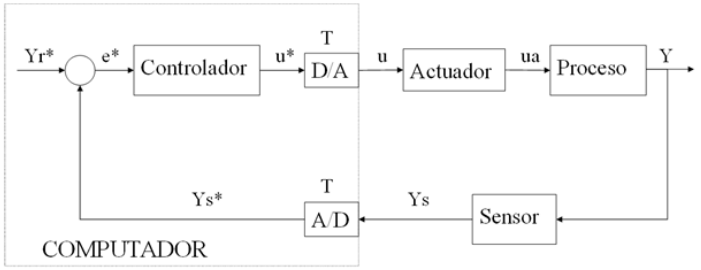
Informática Industrial. Sistemas reactivos.

El **control de variables continuas** por ordenador se realiza tradicionalmente **muestreando** la señales continuas a **intervalos regulares** y **cuantificando** la señal muestreada para codificar en la **palabra finita** del ordenador.



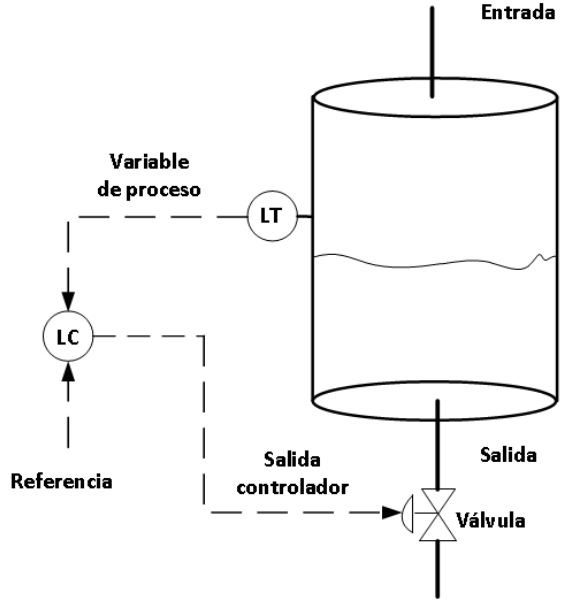


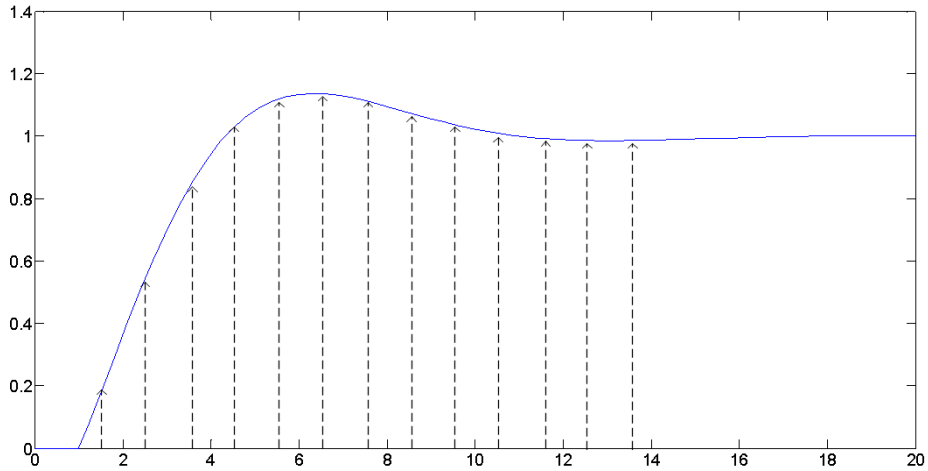


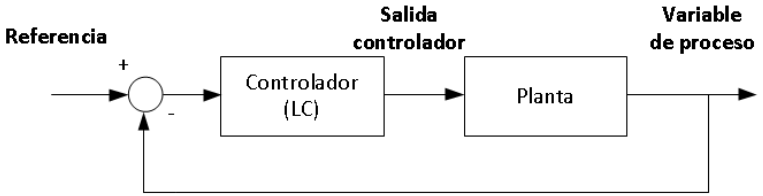


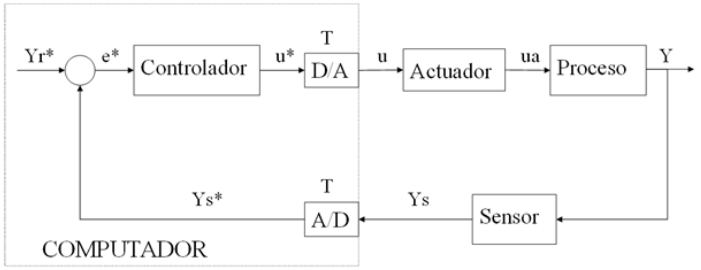
Informática Industrial. Sistemas reactivos.

- Teoría de sistemas muestreados bien asentada (transformada Z)
- Elección de **tiempo de muestreo**: **teorema de Shannon**
- **Posible problemas**: surgimiento de **alias** de las señal original, perturbaciones en la periodicidad del muestreo o ***jitter***…



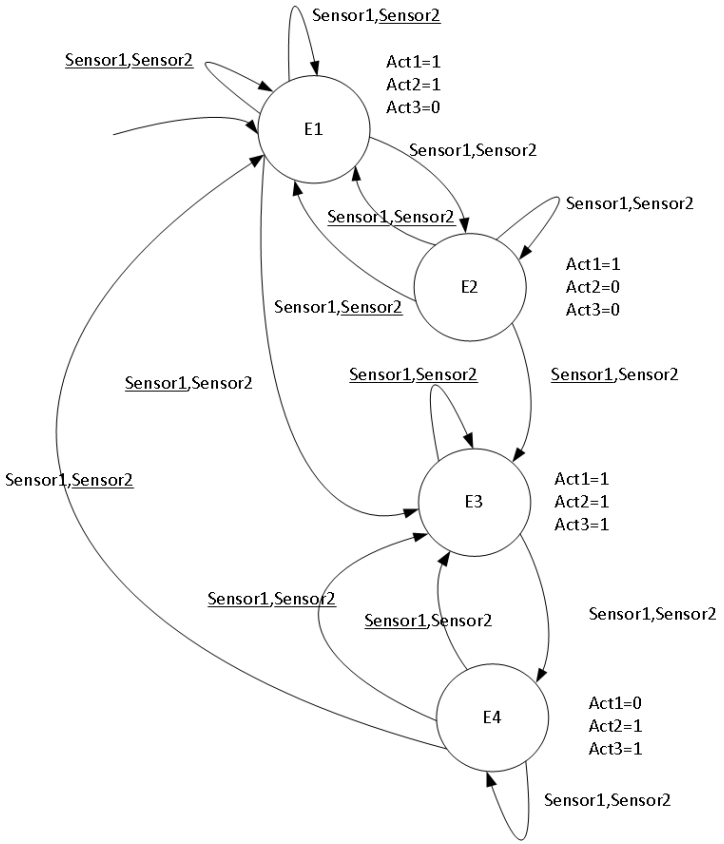


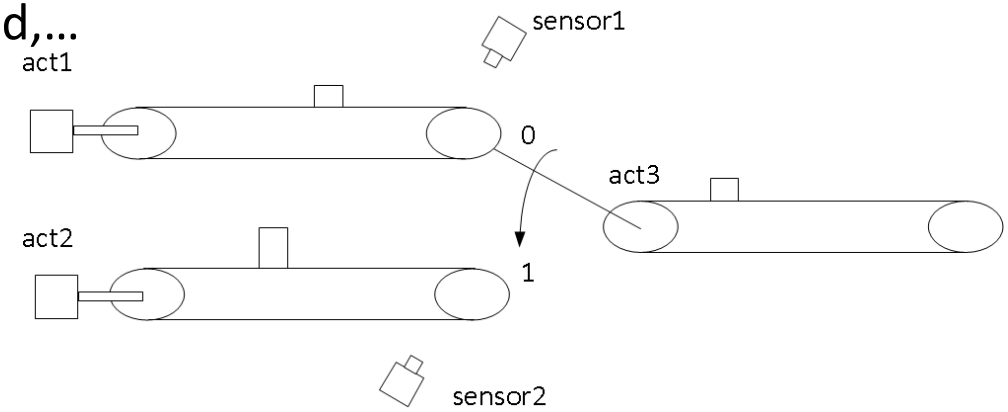




Informática Industrial. Sistemas reactivos.

- Control de sistemas discretos
- Se utilizan varios formalismos (máquinas de estado finitas, redes de Petri, *grafcets*)
- Evitar bloqueos, garantizar seguridad,…

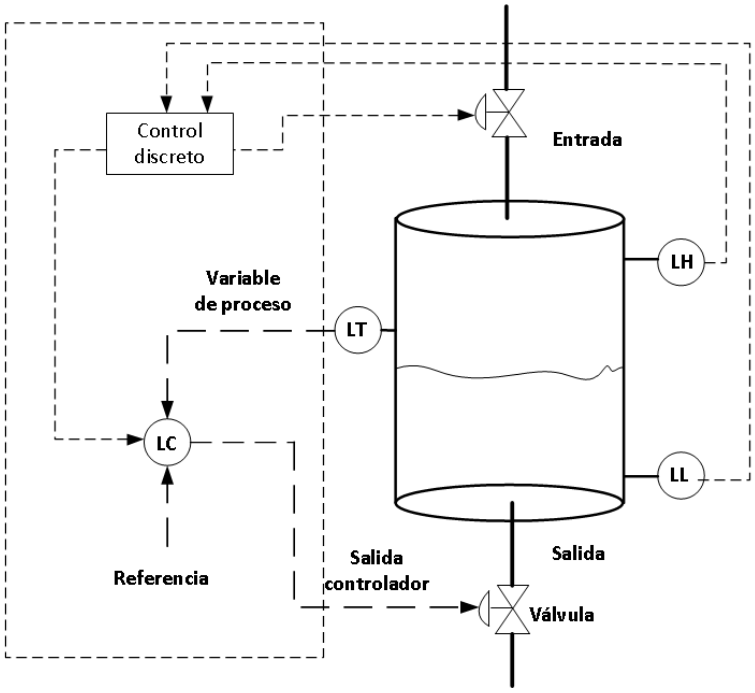


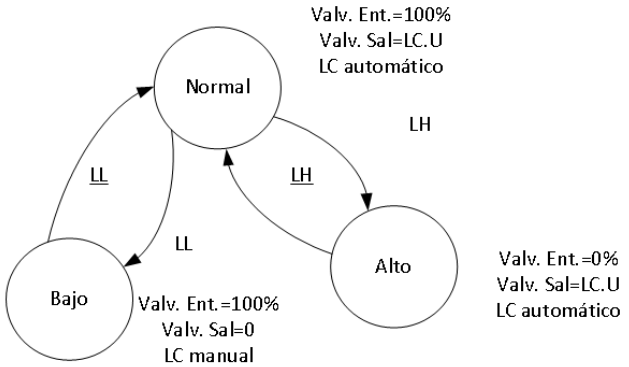


Ej: A partir de la lectura de los sensores (1y2) activar/detener cintas transportadoras (act1 y act2) y colocar la cinta común en el posición correspondiente (act3). Minimizar cambios de act3.

Informática Industrial. Sistemas reactivos.

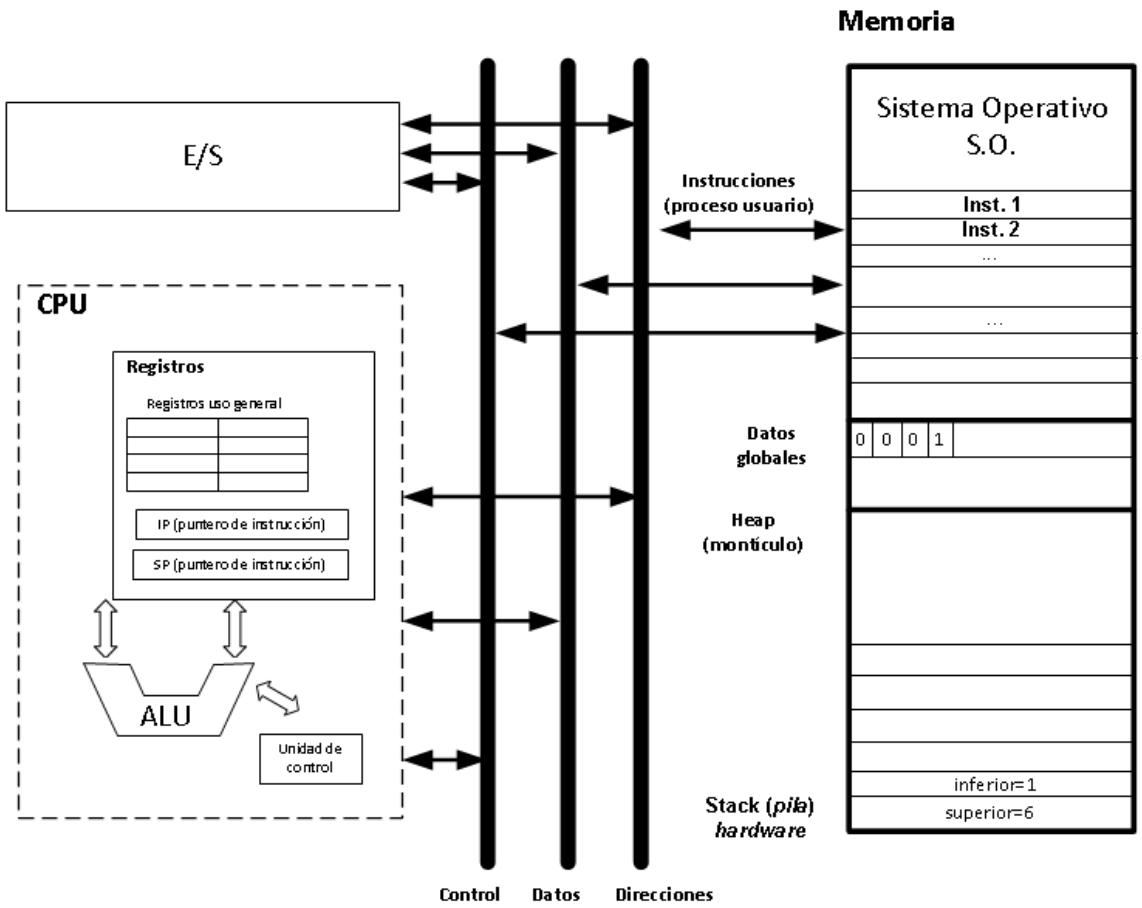
- En realidad se la situación híbrida (continua, discreta)
- La presencia del control por ordenador (elemento discreto) es en gran medida responsable que la planta en su conjunto tenga naturaleza híbrida.





Informática Industrial. Computador.

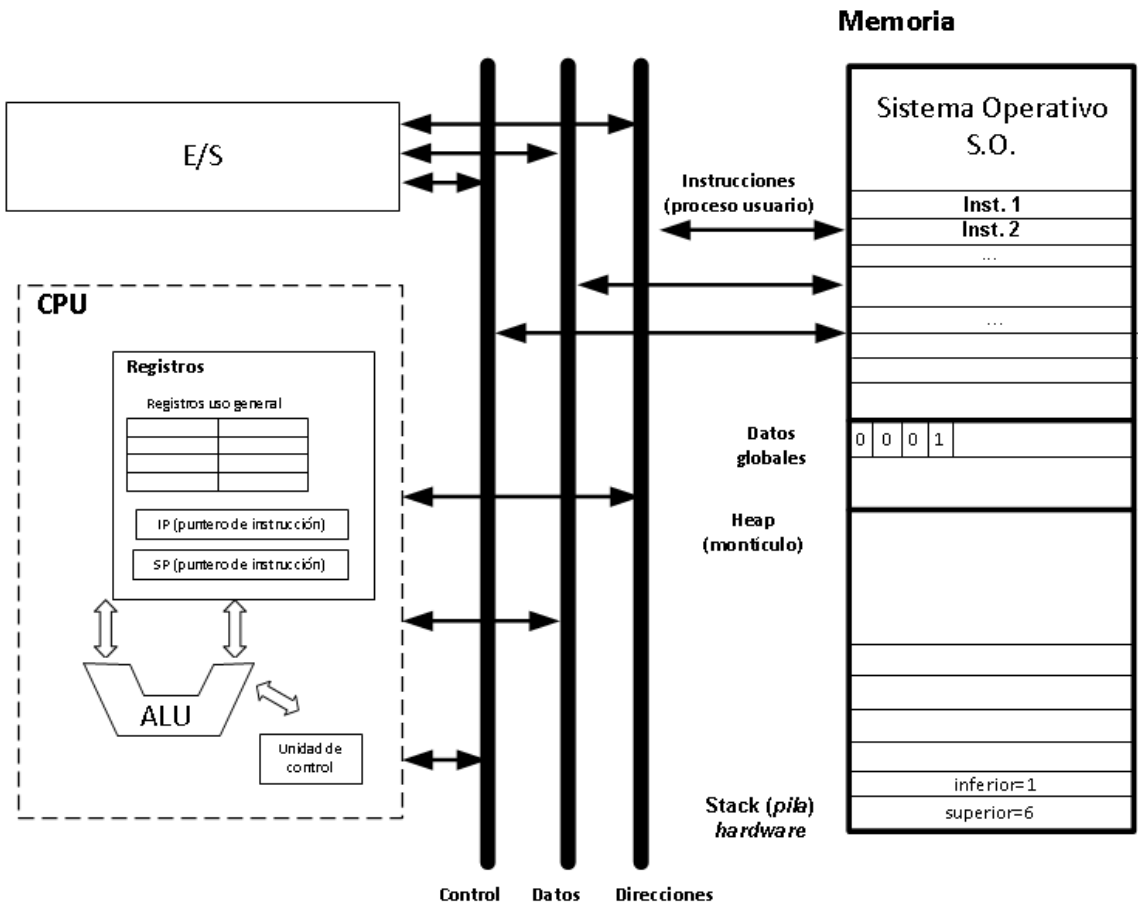
- El **computador digital** es **un sistema discreto**, que avanza al ritmo impuesto por reloj interno, con una memoria formada por palabras de longitud finita.
- La **arquitectura Von Neumann** es predominante: la **memoria** almacena **instrucciones de pro**grama y **datos**.



- La ejecución de un **proceso** en memoria se realiza de **forma secuencial**
- Sistemas Operativos Multitarea: varios procesos en memoria simultáneamente.
- Algoritmos de planificación: determina que proceso se ejecuta en la CPU.
- S.O. se ocupa del cambio de contexto de un proceso a otro.

Informática Industrial. Computador.

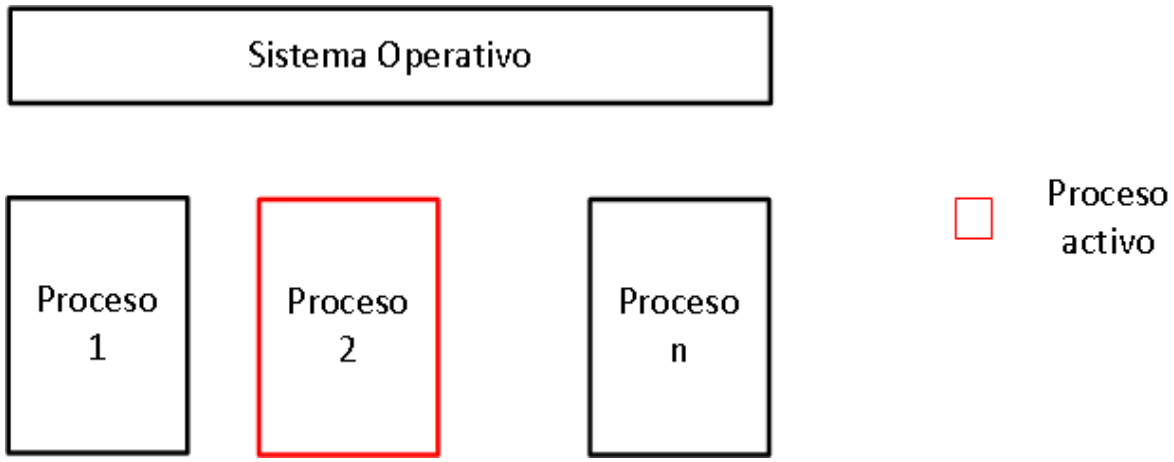
- El **computador digital** es **un sistema discreto**, que avanza al ritmo impuesto por un reloj interno, con una memoria formada por palabras de longitud finita.
- La **arquitectura Von Neumann** es predominante: la **memoria** almacena **instrucciones de programa** y **datos**.



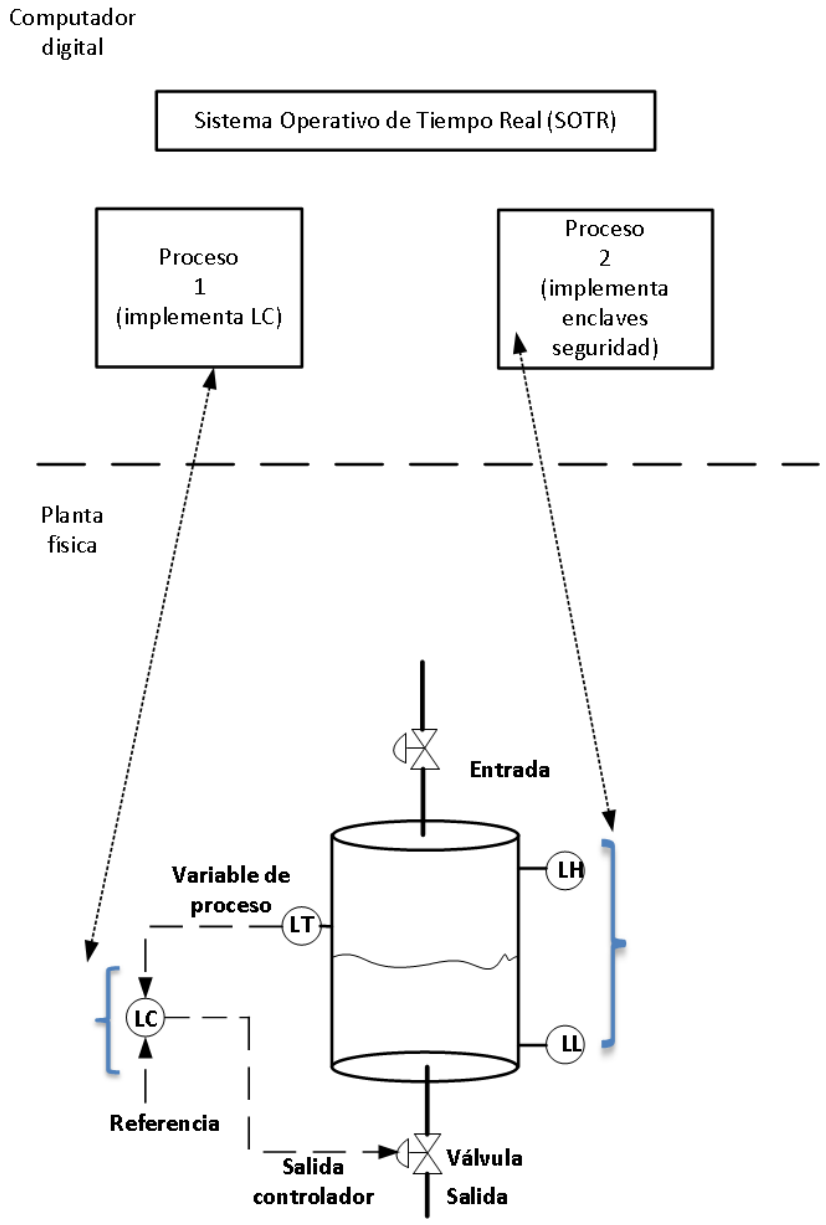
- La ejecución de un **proceso** en memoria se realiza de **forma secuencial**
- **Sistemas Operativos Multitarea**: varios procesos en memoria simultáneamente.
- **Algoritmos de planificación**: determina que proceso se ejecuta en la CPU.
- S.O. se ocupa del cambio de contexto de un proceso a otro.

Informática Industrial. Computador.

- Los **procesos** son, en general, independientes. Cada uno tiene acceso sólo a sus propios datos: espacio de direcciones (de memoria virtual) propio.
- Existen aplicaciones informáticas que se construyen a partir de varios procesos que colaboran.
- Estas **aplicaciones concurrentes** son más complejas de diseñar puesto que la colaboración implica la necesidad de **sincroniza**r el funcionamiento de los diferentes procesos y **comunicarlos** entre sí.
- Además de los **procesos**, existen los **hilos** de ejecución secuencial (***threads***). El cambio de contexto en los hilos es menos costoso. Los hilos tienen memoria compartida.



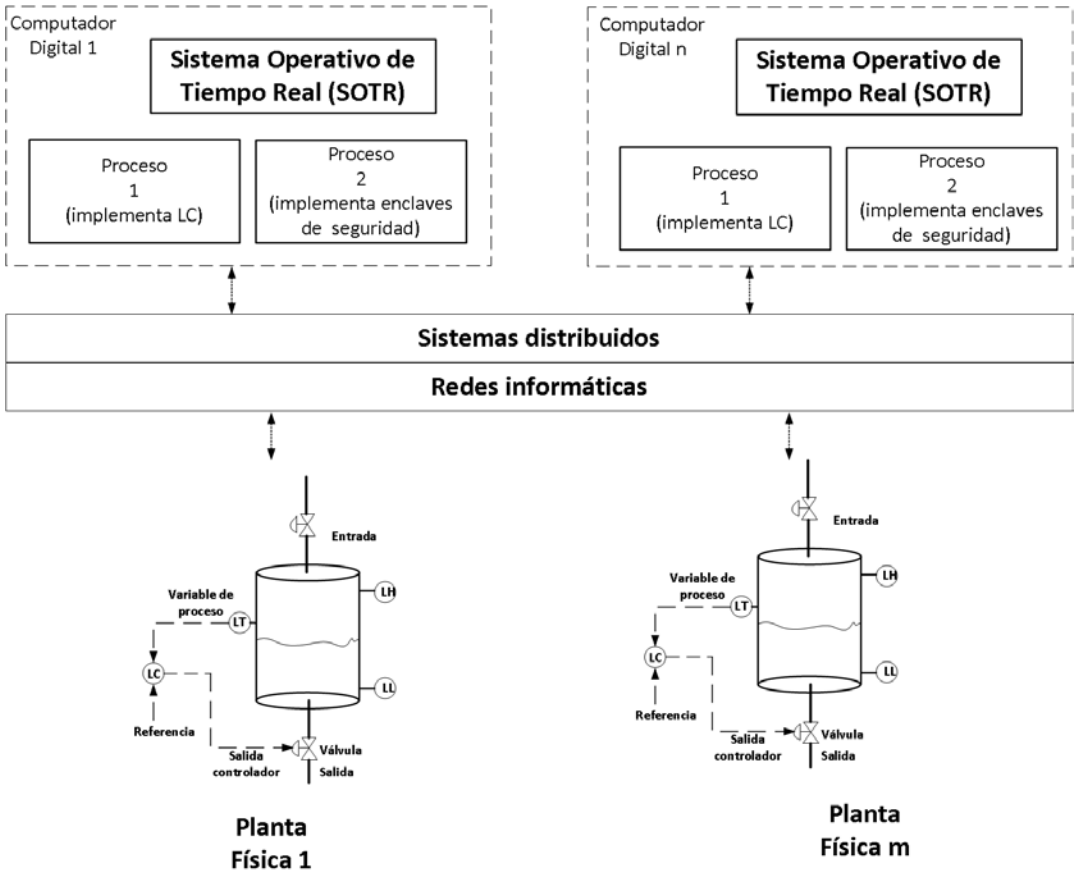
Informática Industrial. Computador.



- En la **realidad física** los diferentes **eventos** pueden ocurrir **simultáneamente**.
- Es natural el utilizar la **programación concurrente** para lidiar con plantas físicas.
- El control de plantas reales implica la necesidad de que el programa informático no sólo brinde las salidas adecuadas. También se exige que lo haga cumpliendo con **plazos** definidos en **tiempo físico real**.
- Los **sistemas operativos de tiempo real** planifican las **tareas concurrentes** buscando cumplir los plazos (*deadlines*) de cada una de ellas.

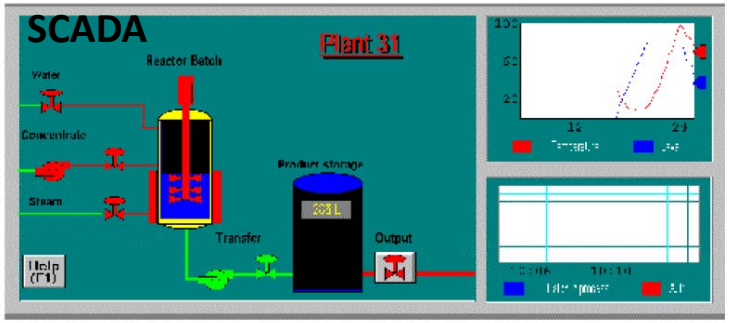
Informática Industrial. Computador.

- Tanto **las plantas** a controlar como los **computadores digitales** existen **distribuidas geográficamente**
- El uso de los **sistemas distribuidos** y las **redes informá**ticas es una necesidad
- El uso de **redes** en el control de plantas incorpora una latencia y un grado de incertidumbre añadidos que lo hacen más complejo.

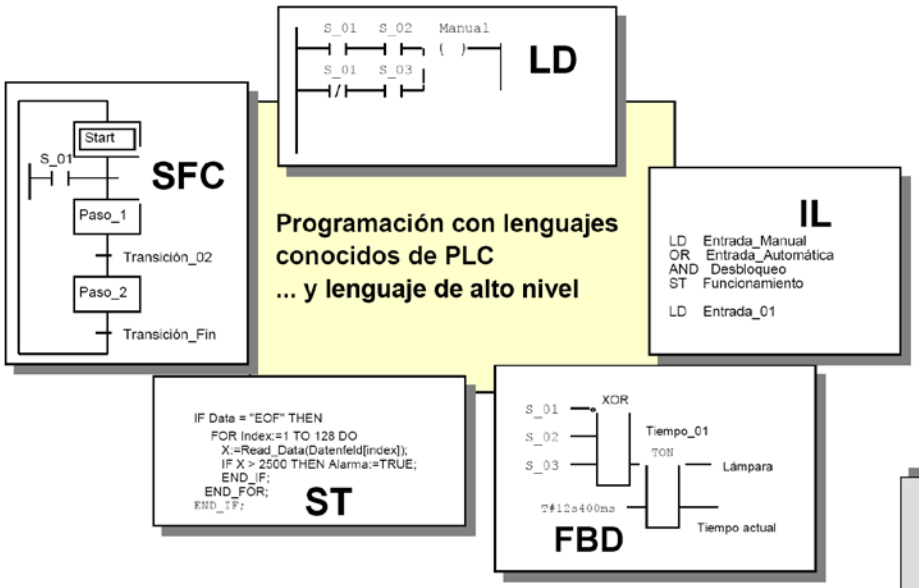


Sistemas programables

- Estudio de sistemas programables industriales.
  - Dispositivos de campo inteligentes.
  - **PLC** (autómatas programables).
  - Sistemas de Control distribuido (**DCS**).
  - Sistemas **SCADA** (Sistemas de supervisión y Control).

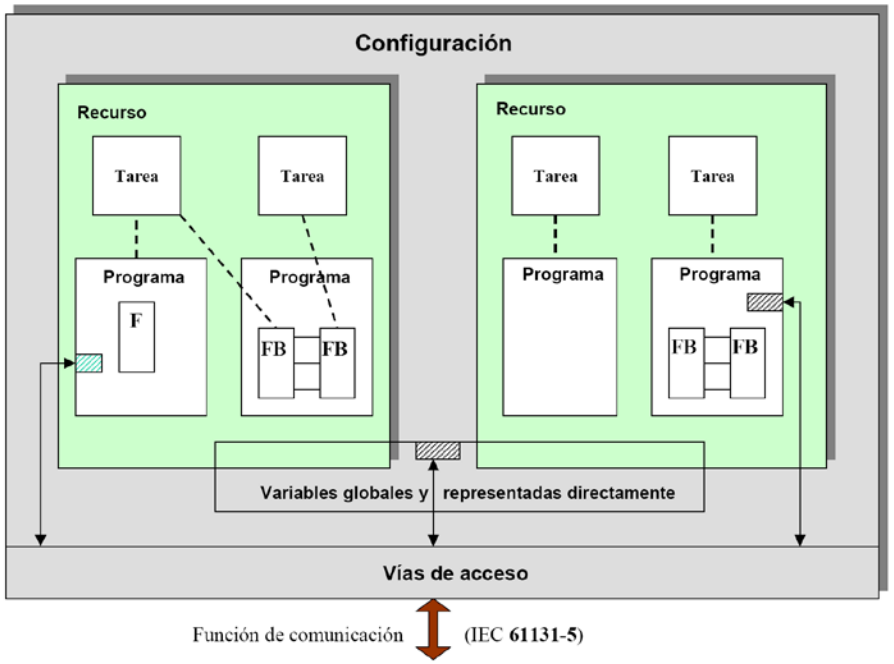


Sistemas programables



Estándares:

- IEC 61131-3
- IEC 61499, entre otros.



Informática Industrial. Estándares.

RETOS:

¿Cómo lidiar con la diversidad de productos comerciales en una disciplina que cambia tan rápidamente?

- Hacer uso de estándares.
- Escoger aquellos estándares *exitosos*.
- Preferir productos comerciales que soporten soluciones estándar.

- Mercado más abierto y competitivo (más para escoger).
- Reutilizas el conocimiento.

Informática Industrial. Fiabilidad y Seguridad.

Requerimientos:

- Elementos de Seguridad y tolerancia a fallos en sistemas programados es clave.

- Cada vez con mayor frecuencia los sistemas programados se utilizan en misiones críticas para la seguridad.
- Entender las implicaciones.

Seguridad funcional

- Necesidad de ***verificación*** de los programas.
- **Lenguajes de Variabilidad Total** (FVL) como el C/C++, son más flexibles, pero más proclives a los errores.
- Se tiende a utilizar un subconjunto seguro de lenguajes de **Bloques Funcionales (FB)**, que han sido probados: **lenguajes de variabilidad limitada** (LVL).
- Estándar de seguridad funcional **IEC 61508**.

Bibliografía

- González-Sánchez, J.L. Sistemas en Tiempo Real. UVA.
- Molina, et al, Using Industrial Standards in PLC programming Learning. (2007).
- Gambier, A. Real Time Control Systems: A Tutorial.
- Burns. A. y Wellings, A. Sistemas de Tiempo Real y Lenguajes de Programación. Addison Wesley, 2002.
- Stalling, W. Comunicaciones y redes de computadores. Prentice Hall, 2004.
- Tanenbaum, A. Sistemas distribuidos: principios y paradigmas. Pearson, 2008.
- Aström, H.J. y Witternmark, B. Sistemas controlados por ordenador. Paraninfo, 1988.
- Phillips, C.L. y Nagle, H. Sistemas de Control Digital: análisis y diseño. Gustavo Gili, México, 1993.
- Smith, J.B. Functional Safety: a straightforward guide to IEC 61508 and related standards. ISA, 2004.

TEMA 2
Introducción a los Sistemas operativos de tiempo real

ESCUELA DE INGENIERÍAS INDUSTRIALES

Departamento de Ingeniería de Sistemas y Automática. EII.

Universidad de Valladolid

Sistemas operativos de Tiempo Real

- Introducción Sistemas operativos de tiempo real (SOTR)
- Sistemas operativos
- Características de los SOTR

Introducción

Sistemas operativos de Tiempo Real

- Cualquier sistema de procesamiento que tenga que responder a estímulos de entrada generados externamente dentro de un periodo especificado y finito
- Un **sistema de tiempo real** es un sistema informático que
  - Interacciona repetidamente con su entorno físico
  - Responde a los estímulos que recibe del mismo dentro de un plazo de tiempo determinado
- Un sistema de tiempo real (STR) es aquel en el cual los resultados son correctos, no sólo si la computación es correcta, sino también **el tiempo en el cual se producen los resultados**

Sistemas operativos de Tiempo Real

- Si no se verifican las restricciones de tiempo, se dice que se ha producido un fallo en el sistema
- Así pues, es fundamental garantizar las restricciones de tiempo, lo cual implica que el sistema sea **predecible**.
- Ejemplo: robot que coge los productos que discurren por una cinta transportadora.
- Sistema de tiempo real no es sinónimo de sistema rápido. Ejecución a una velocidad que permita cumplir las especificaciones de tiempo.
- Aunque las velocidades de proceso aumenten los STR seguirán existiendo.

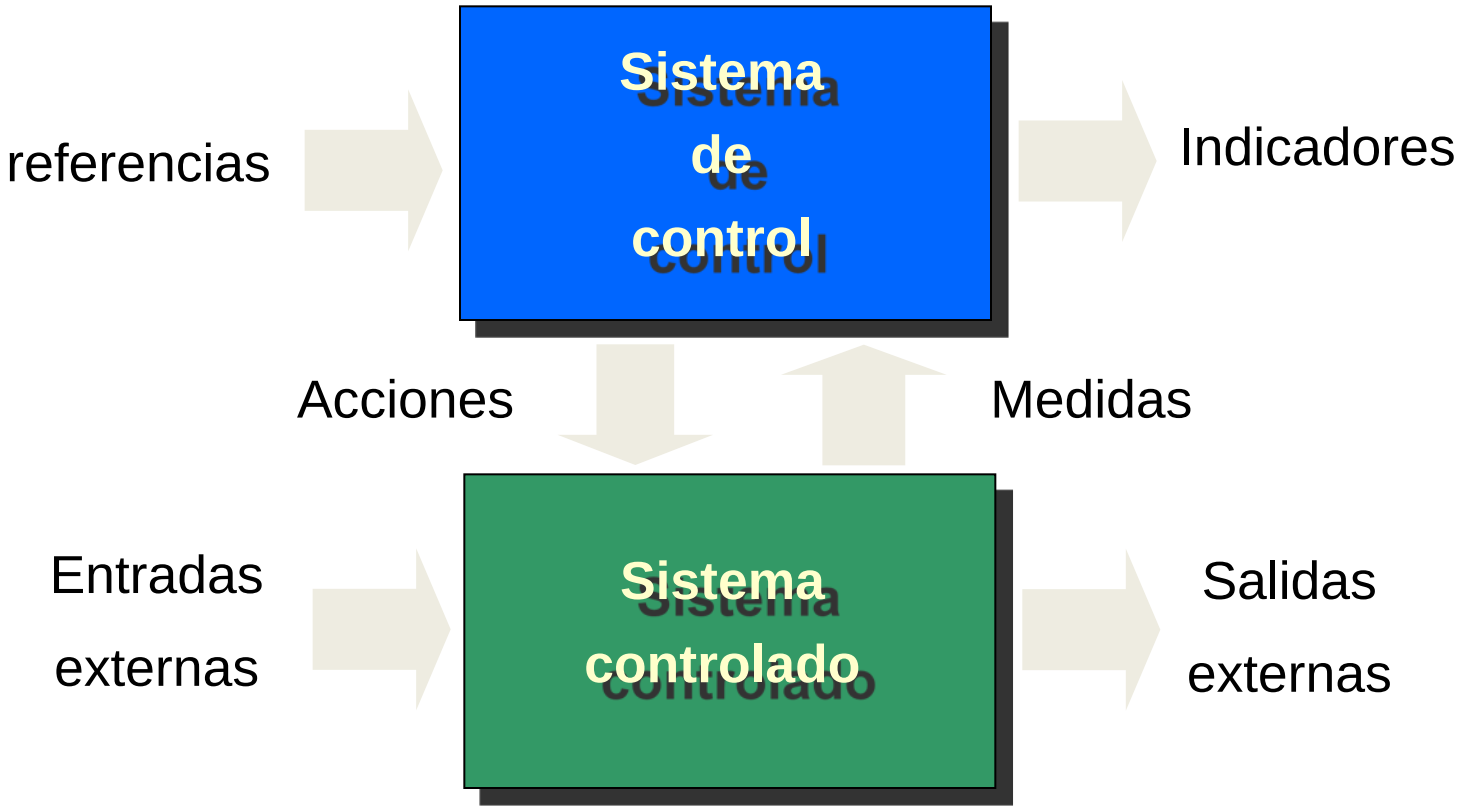
Dónde se utilizan los STR

- Sistemas de control de procesos
- Sistemas de control de vuelo
- Sistemas de control de automóviles
- Sistemas de defensa
- Sistemas de vigilancia intensiva
- Sistemas multimedia
- Electrónica de consumo
- Sistemas de telecomunicación, etc.

Clasificación de los STR

- Críticos (*hard real time systems*): los plazos de respuesta deben respetarse en todas las circunstancias, una sola respuesta tardía a un suceso puede tener consecuencias fatales (sistemas de control de centrales nucleares o de aeronaves)
- No críticos (*soft real time systems*): se puede tolerar retrasos ocasionales en la respuesta a un suceso (sistema multimedia)

Estructura de un STR



Estructura de un STR

- El sistema de control (STR) actúa sobre el sistema controlado para conseguir un comportamiento definido
- Es muy común emplear un esquema realimentado
  - En función de las medidas opera un algoritmo de control
  - Ese algoritmo de control determina las acciones impuestas al sistema controlado
  - Un aspecto de suma importancia son los tiempos en los que se lleva a cabo cada acción

Sistemas de tiempo real

- Dos tipos de sistemas de control de tiempo real:
  1. Sistemas empotrados (*embedded systems)*
     - Sistemas de control dedicados
     - El computador no es visible pues está integrado en algún elemento del equipo.
     - Empleo limitado de recursos
     - Los dispositivos de entrada y salida son especialespara cada sistema. No hay teclado ni pantalla normales
     - aeronaves, robots industriales, sistemas control automóviles, ...
  2. Sistemas de Control Industriales
     - Sistemas de control distribuido (DCS), controladores lógicos programables PLC
     - Organizados jerárquicamente, sistemas de control distribuidos
     - Industria de procesos, industria de fabricación

Fiabilidad y seguridad

- Fiabilidad
  - Es la probabilidad de proporcionar el servicio especificado
  - Medida de fiabilidad: la media de tiempo entre averías *(mean time to failure)* MTTF=1/ λ
  - Si MTTF > 10^9 hablamos de sistemas ultrafiables
- Seguridad
  - Los sistemas críticos deben ser ultrafiables

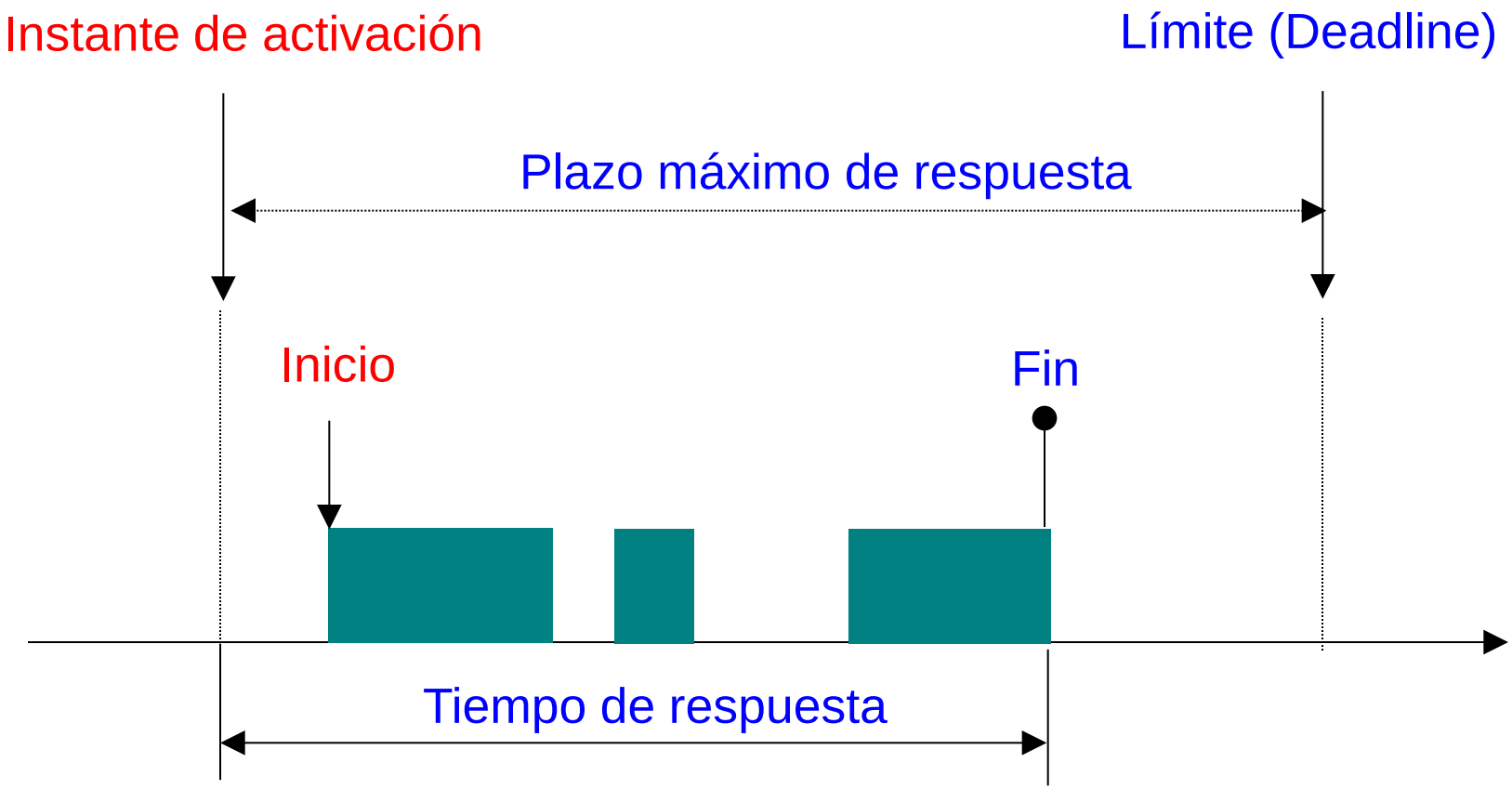
Fiabilidad y seguridad

- Los requisitos de fiabilidad y seguridad en los STR son mayores que en el resto
- Ejemplo: entre el 80 y el 90% de los sistemas de defensa, su control se realiza mediante software
- Según Hecht y Hecht (1986), los sistemas software complejos, por cada millón de líneas de código contienen una media de 20.000 errores
- El 90% de esos errores pueden ser detectados con sistemas de comprobación
- 200 errores de los restantes se detectan durante el primer año. Los 1800 restantes permanecen sin detectar

Dependencia del tiempo

- Tiempos en los cuales se deben llevar a cabo determinadas acciones
- Tiempos de terminación de cada acción
- Responder a situaciones transitorias en las que no se pueden garantizar todos los plazos
- El comportamiento del sistema debe ser DETERMINISTA

Aspectos temporales



El tiempo de respuesta puede variar

Sistemas operativos

¿Qué es un Sistema Operativo?

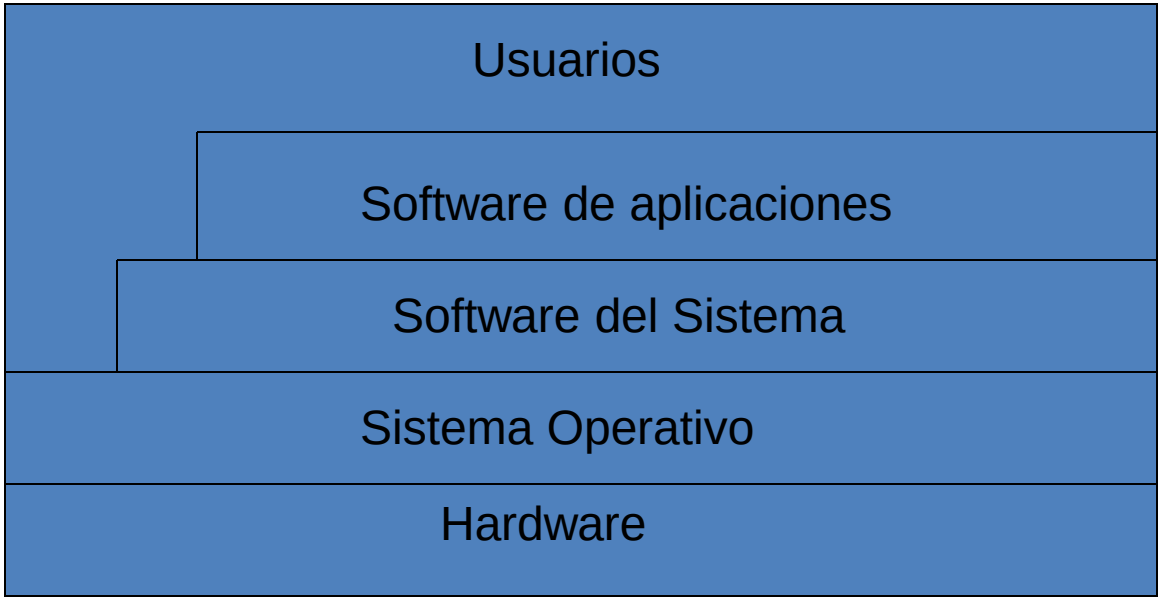
Un programa que actúa como intermediario entre el usuario y el hardware de la computadora.

Objetivos del Sistema Operativo:

- Ejecutar programas del usuario y resolver los problemas del usuario de manera fácil y sencilla.
- Hace que la computadora sea fácil de usar.
- Utiliza el hardware de la computadora de forma eficiente.

¿Qué es un Sistema Operativo?

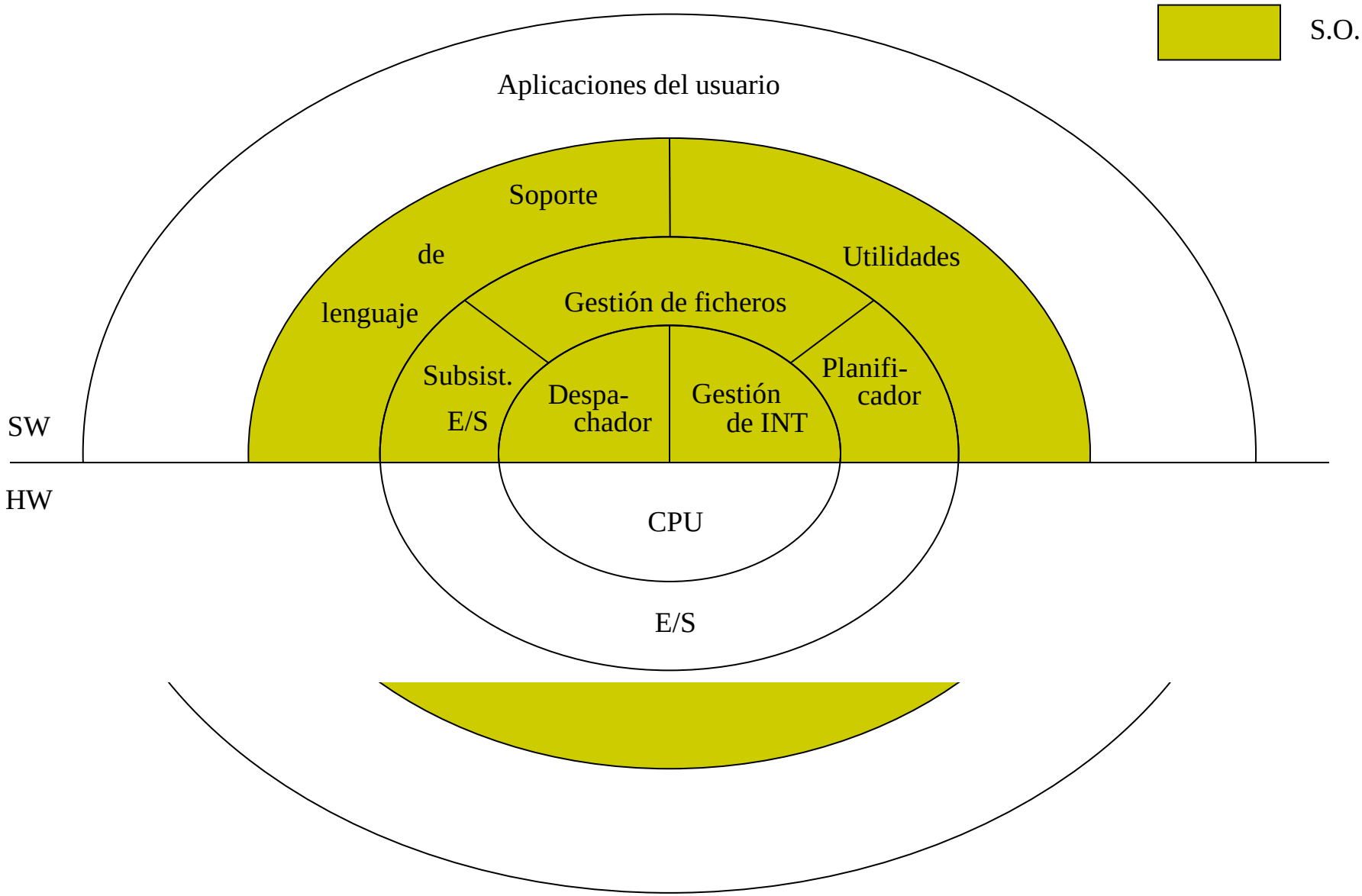
- Sistema de software que provee a los usuarios de un entorno eficiente par la ejecución de sus programas.



Componentes del sistema informático

- **Sistema Informático de propósito general**
  - **Hardware** - proveé los componentes básicos de cómputo (CPU, memoria, dispositivos de E/S).
  - **Sistema Operativo** - controla y coordina el uso del hardware entre los varios programas de aplicación para los diferentes usuarios.
  - **Programas de Aplicación** - define las formas en que los recursos del sistema son utilizados para resolver los problemas de cómputo de los usuarios (compiladores, bases de datos, juegos de video, programas de negocios).
  - **Usuarios** (gente, máquinas, otras computadoras).

Componentes del sistema informático

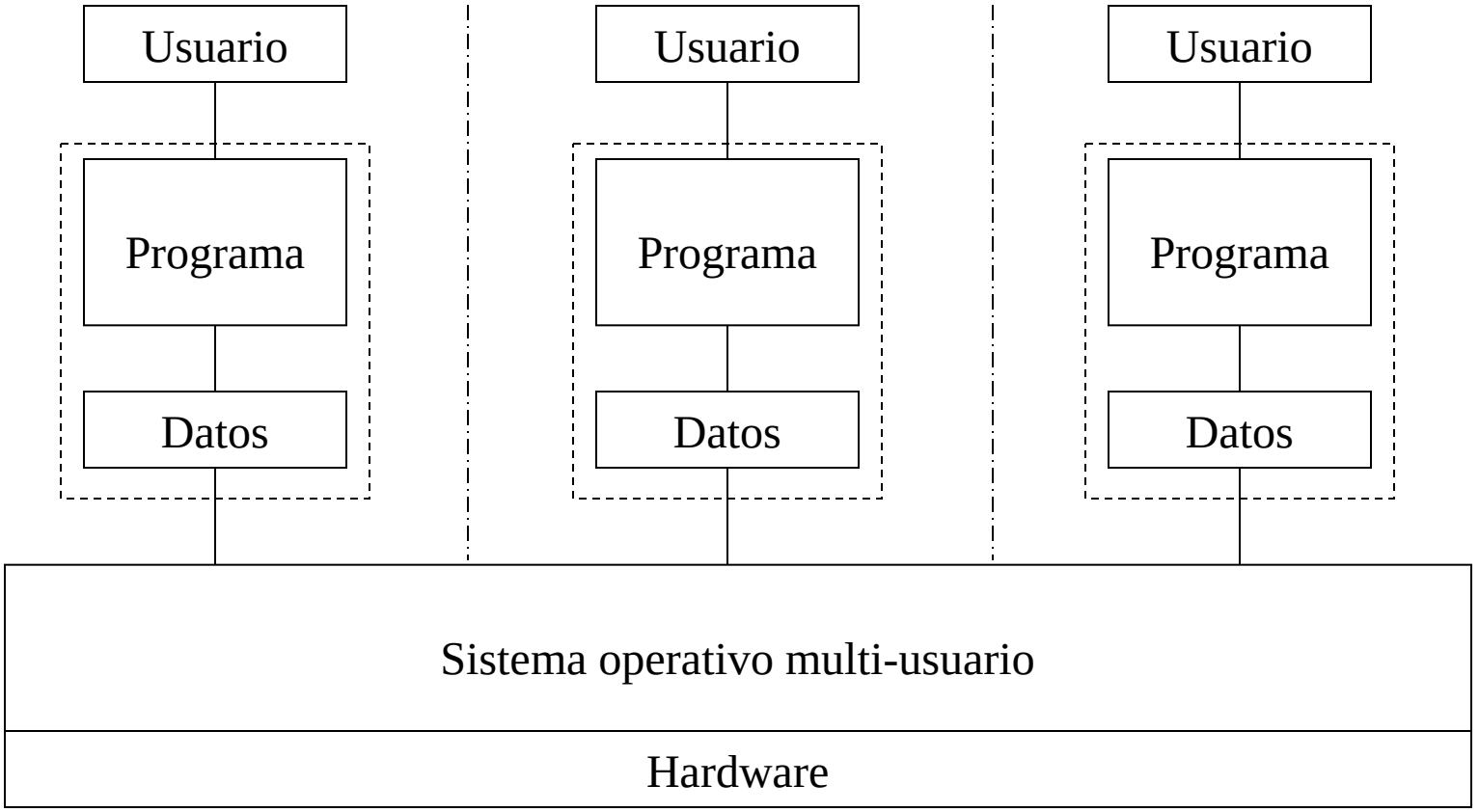


Tipos de sistemas operativos

- **Multiusuario.**
- **Multitarea**.
- **Multiusuario y multitarea**.

S.O. multiusuario

Permite el acceso o la operación de diversos usuarios, cada uno con su propio área de trabajo y sin que los procesos de cada uno interfieran entre si. El S.O. deberá asignar los recursos a todos los usuarios de forma que éstos consideren que disponen del sistema total.

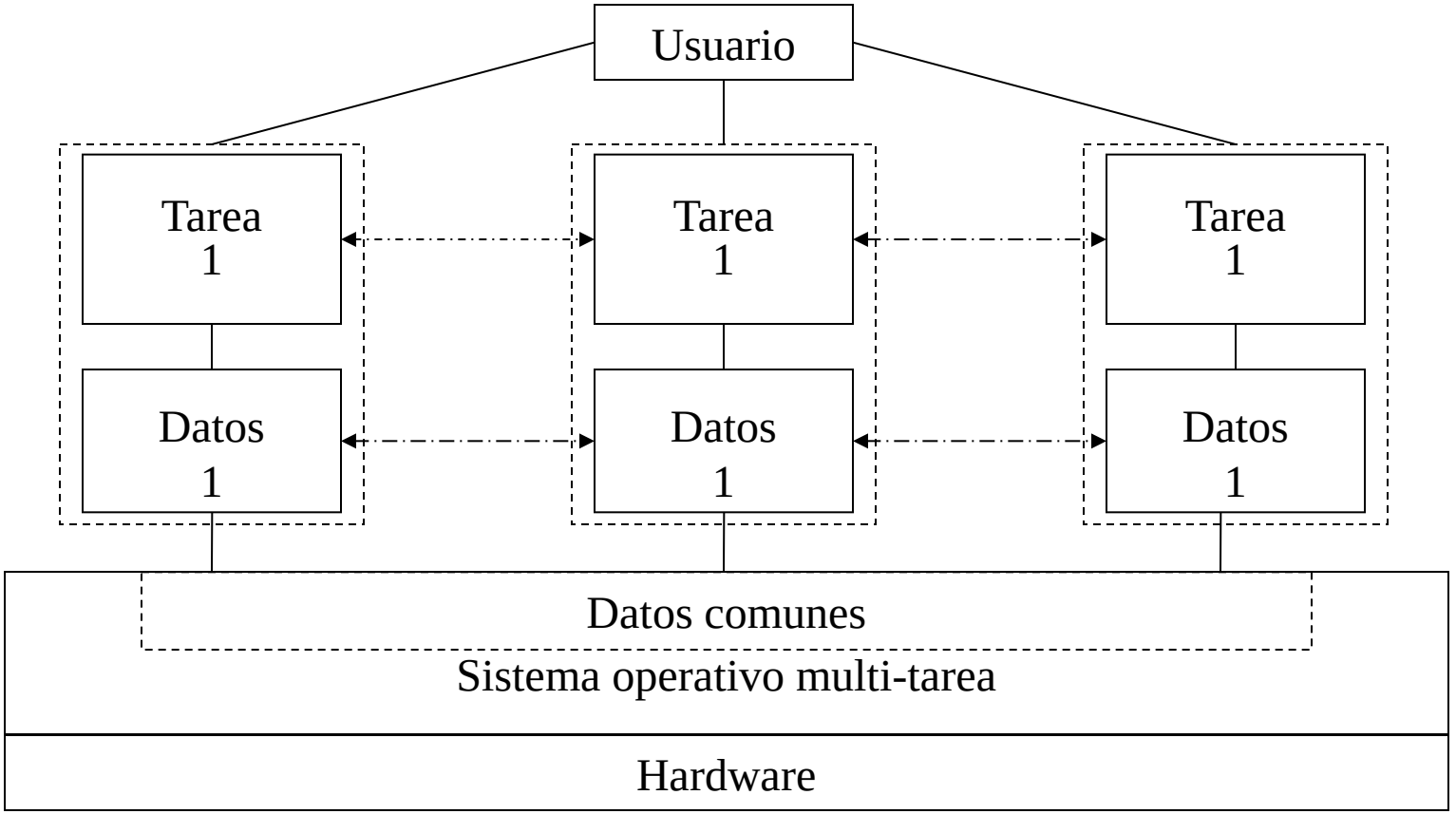


S.O. multitarea

El mismo usuario puede tener varias tareas o procesos que se comunican entre sí utilizando canales de comunicación gestionados por el S.O.:

intercambio de datos entre tareas (memoria compartida y cola de mensajes)

sincronización entre tareas (semáforos y señales).



S.O. multiusuario y multitarea

Tiene las características de ambos sistemas.

En el caso de STR, pueden ser sistemas multitarea, es decir, pueden disponer de canales de comunicación entre tareas pero de forma que se cumplan aquellas tareas de mayor prioridad para garantizar que se cumplen las restricciones temporales.

La CPU es uno de los recursos que hay que gestionar. Se gestiona de acuerdo a un esquema de prioridades. El planificador decide la asignación a tareas. Un STR multitarea tiene que soportar el reparto de recursos y además, el cumplimiento de las restricciones temporales.

Objetivos del S.O.

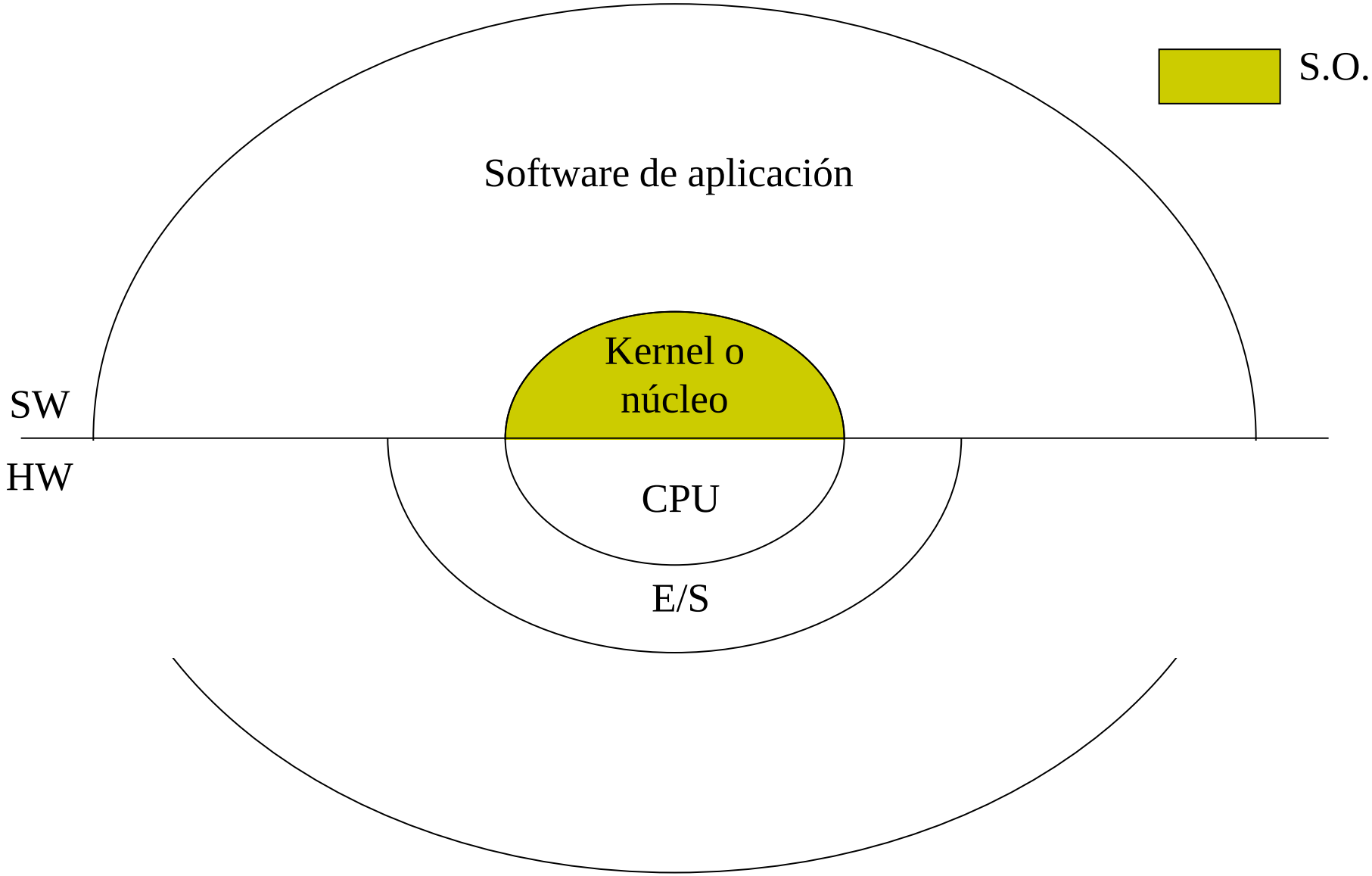
- **Gestión de tareas**: (*Scheduling*) asignación de tiempo de procesador a las tareas. Decide que tarea pasa a ejecutarse.
- **Gestión de memoria**: control de asignación de memoria.
- **Gestión de recursos**: Controla otros recursos compartidos diferentes a memoria y tiempo de procesador. Almacena la información asociada a los recursos compartidos por parte de las tareas.
- **Gestión y comunicación entre tareas**. Suministra los mecanismos que dan soporte a la comunicación segura entre tareas y a la sincronización de sus actividades. Creación de tareas y mantenimiento de la información asociada a tareas, así como la eliminación de esta información. Una tarea puede crearse por invocación de otra tarea

Características de los sistemas operativos de tiempo real

No se trata de que los SOTR sean sistemas rápidos sino de que sean fiables.

- debe satisfacer las restricciones temporales explícitas de modo que si no se cumpliesen, se darían en el sistema consecuencias de riesgo severo incluso el fallo total
- diseñar el SOTR pensando más en el peor caso que en optimizar el rendimiento medio.
- atender con una alta prioridad a las señales externas (interrupciones) provenientes del sistema , ya que pueden informarnos de un cambio de estado en el sistema.
- minimizar todo aquello que conlleve un alto precio en el tiempo de la CPU

Sistema operativo mínimo



Requisitos actuales de los SOTR

- Determinismo
  - tendencia que tiene el sistema a realizar una determinada acción en un tiempo predefinido. El minimizar el tiempo de respuesta a interrupciones garantiza un mayor nivel de determinismo.
- Capacidad para responder a eventos rápidamente
  - distinguiendo entre sucesos síncronos y asíncronos. Necesita una rutina de interrupciones para dar una respuesta “inmediata” a los sucesos asíncronos. Requiere que el desalojo y realojamiento de procesos de la CPU se haga con rapidez.
- Control del sistema por parte del usuario
  - de modo que los usuarios puedan conmutar entre distintos modos de ejecución. Por ejemplo, un operario que acciona un botón de emergencia de una máquina.

Requisitos actuales de los SOTR

- Gestión de prioridades
  - En los sistemas operativos tradicionales la *prioridad* es dinámica, el propio sistema operativo va incrementando o decrementando la prioridad conforme pasa el tiempo. En los SOTR la prioridad debe ser estática con prioridades determinadas, al menos para varios procesos especialmente críticos. Las interrupciones procedentes del exterior tienen una prioridad fija, que no depende de tiempos de espera o ejecución. Además, se ha de analizar si la tarea se realiza según los plazos (seguimiento preventivo y predictivo) y cómo interfiere con las demás
- Fiabilidad
  - en caso de fallo hardware ha de haber una solución de tipo software. Deben estar contempladas todas las respuestas que daríamos a cada uno de los posibles fallos que se pudiesen dar.

Requisitos actuales de los SOTR

- Gestión de Memoria
  - ha de ser más estricta que en los sistemas operativos tradicionales. Cuanta mayor facilidad se trata de dar al usuario mayor va a ser el kernel. En los SOTR el kernel debe ser lo menor posible.
- Comunicación entre tareas
  - debe ser muy rápida. Suele ser explícita, la tiene que hacer el usuario.
- Código Reentrante
  - se refiere a la posibilidad de que dos procesos puedan emplear un mismo código de programa, sin tener que tener una copia del mismo para cada uno y sin que haya problemas de interferencias entre ellos al manejar el mismo código.

Requisitos actuales de los SOTR

- Tamaño reducido de código
  - conviene que el *repertorio de rutinas* empleado sea lo menor posible, a costa de mayor coste de programación por parte del usuario, de modo que se minimice el número de primitivas y el kernel.
- SOTR distribuidos
  - Se trata de minimizar los tiempos de respuesta por parte de la red: buses de campo, redes industriales, han de minimizar el tiempo desde que se recoge algo en un sensor hasta que llega a un actuador para ejecutar la orden que corresponda, pasando por el gestor de la red. Otros problemas son considerar que puede haber sobrecarga en la red y problemas de sincronización de los relojes de los distintos elementos del sistema distribuido.

TEMA 3
Programación Concurrente

ESCUELA DE INGENIERÍAS INDUSTRIALES

Departamento de Ingeniería de Sistemas y Automática. EII.

Universidad de Valladolid

Capítulo 3: Programación Concurrente

- Concurrencia.
- Concepto de proceso. Estados. BCP.
- Cambio de contexto.
- Creación y terminación de procesos
- Relación entre procesos. Procesos cooperativos

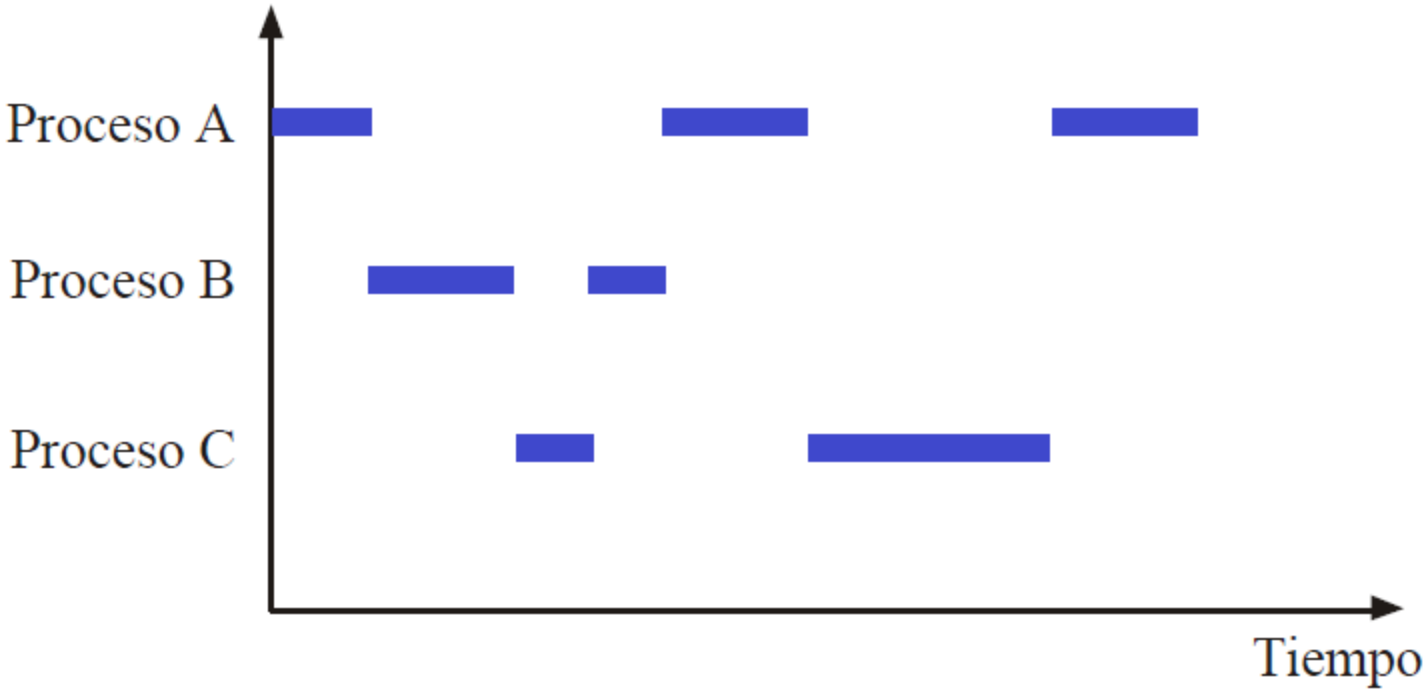
Concurrencia

- La **concurrencia** consiste en que haya más de un proceso ejecutándose simultáneamente. Es el concepto clave en la multiprogramación, en los sistemas multiprocesador y en los sistemas distribuidos.
- La concurrencia comprende un gran número de cuestiones de diseño como la comunicación entre procesos, compartimiento de recursos y sincronización de procesos.
- El acceso de forma concurrente a datos compartidos puede originar inconsistencia en los datos. La entrada de un proceso puede depender de la salida de otro, o el proceso puede necesitar el acceso a un mismo recurso y debe hacerlo de una manera controlada.

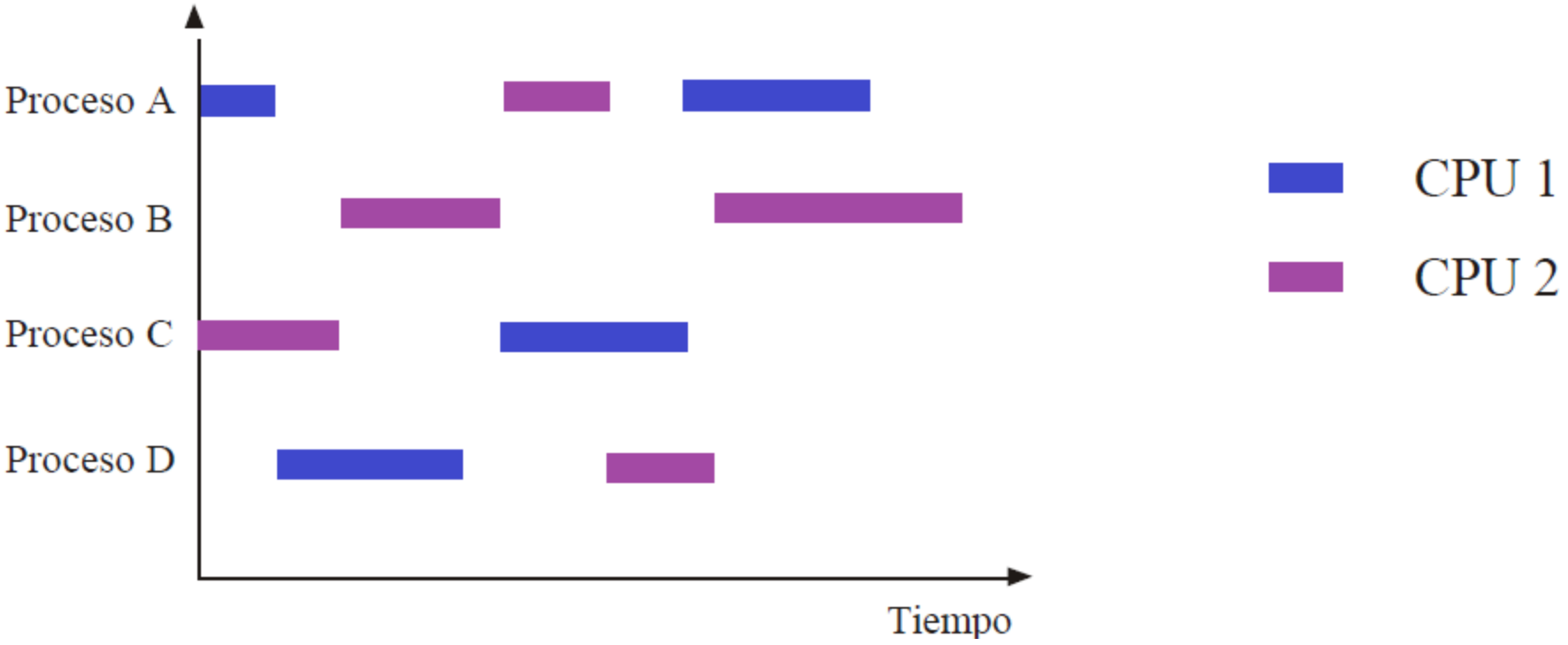
Concurrencia. Modelos.

- Multiprogramación en un único procesador
- Multiprocesador
- Sistemas distribuidos

Concurrencia. Una CPU



Concurrencia. Multiprocesador.

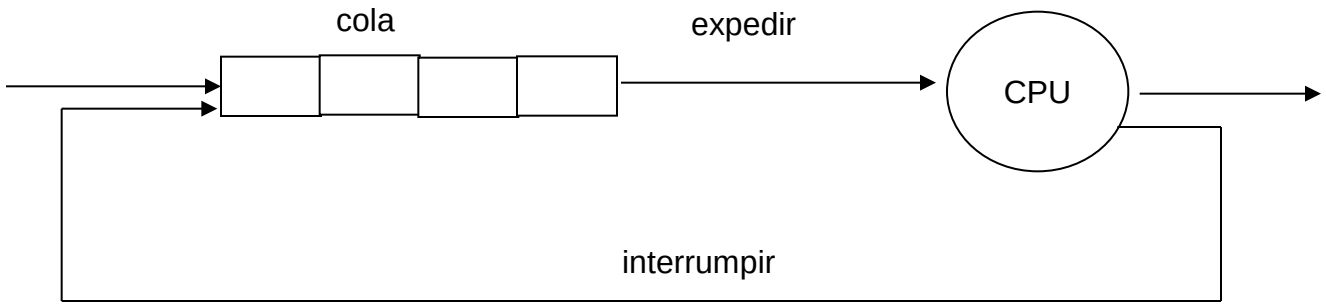


Concepto de proceso

- Un proceso es un programa en ejecución, es una entidad *activa*;
- Un proceso necesita recursos para poder ejecutarse: memoria, dispositivos de E/S, la CPU, etc.
- Un proceso comprende:
  - contador de programa,registros procesador
  - código
  - datos (variables globales, memoria dinámica)
  - pila (parámetros y variables locales de subrutinas)
- En relación con los procesos, un s.o. :
  - intercala la ejecución de los procesos según unos criterios
  - asigna recursos a los procesos (prioridad, evita interbloqueo ...)
  - crea procesos y soporta la comunicación

Estados de un proceso

- A medida que un proceso se ejecuta cambia de estados
- Modelo sencillo de dos estados
  - en ejecución : Las instrucciones están siendo ejecutadas.
  - en espera : el proceso está esperando a que ocurra algo para ejecutarse.
- Debe guardarse una información relativa a cada proceso que incluye el estado actual y posición en memoria.

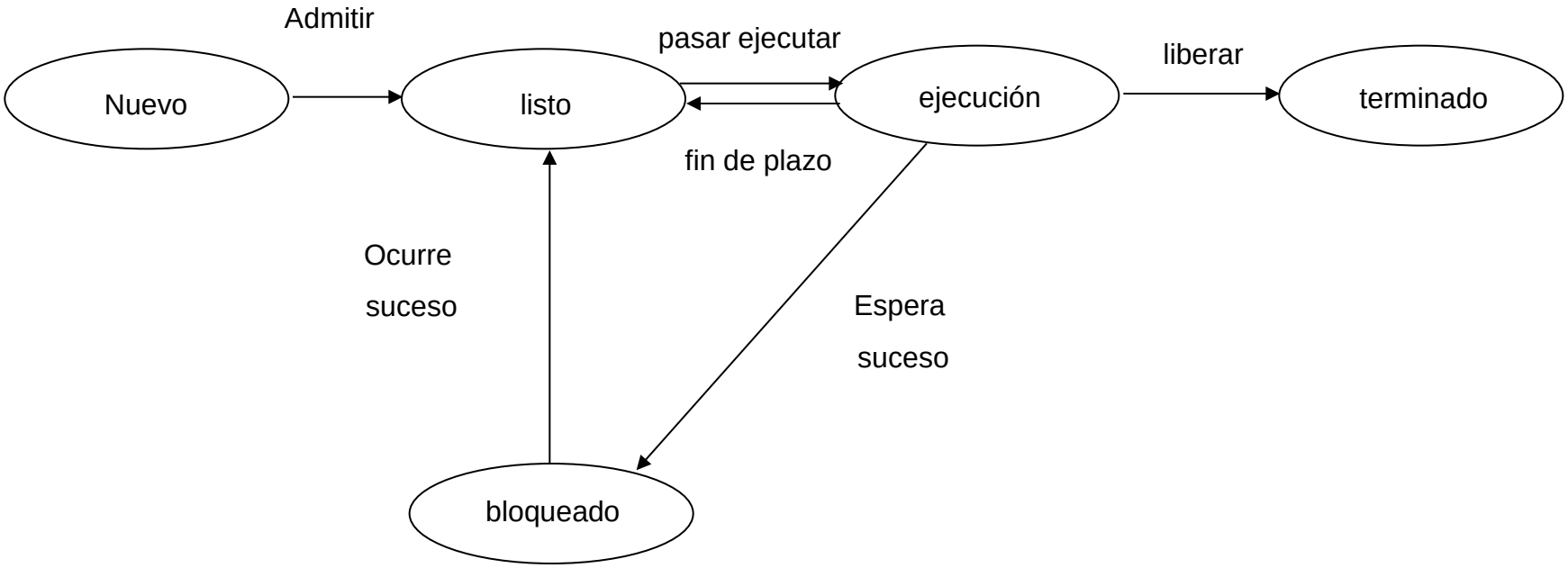


Estados de un proceso

- Modelo de cinco estados
  - **nuevo** *(new)*: El proceso está siendo creado. El S.O. crea las estructuras de datos para administrar el proceso y se le asigna un identificador y un espacio de direcciones.
  - **en ejecución** (*running)* : Las instrucciones están siendo ejecutadas.
  - **en espera** (*waiting)*: El proceso está esperando que ocurra algún suceso.
  - **listo** (*ready)*: El proceso está esperando que se le asigne a un procesador.
  - **terminado** (*terminated)*: El proceso ha terminado su ejecución.

Diagrama de estados de un proceso

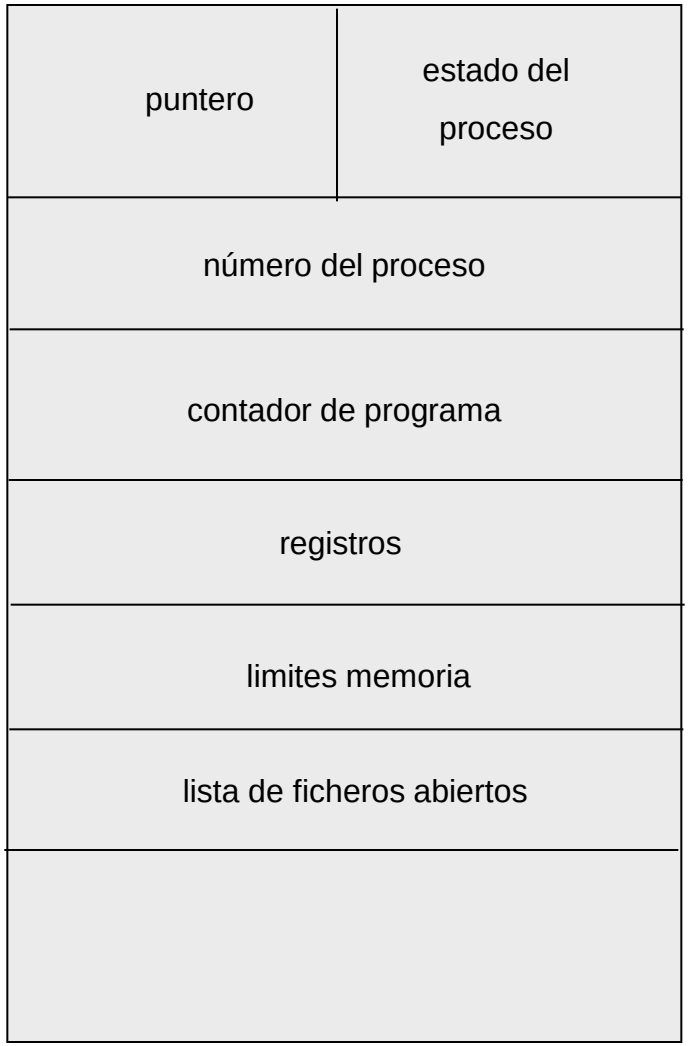
modelo procesos de cinco estados



Bloque de control de proceso (BCP)

- Para cada proceso, el S.O. debe guardar su estado y cualquier otra información que debe mantenerse mientras no está en ejecución.
- Bloque de control de proceso:
  - estado actual (preparado, en espera...)
  - registros de la CPU (contador de programa, otros registros)
  - información del planificador (id, prioridad, etc.)
  - apuntadores a las zonas de memoria del proceso
  - info. de contabilidad (tiempo CPU consumido, etc.)
  - info. de E/S (dispositivos por los que espera, ficheros abiertos,etc.)
  - etc.

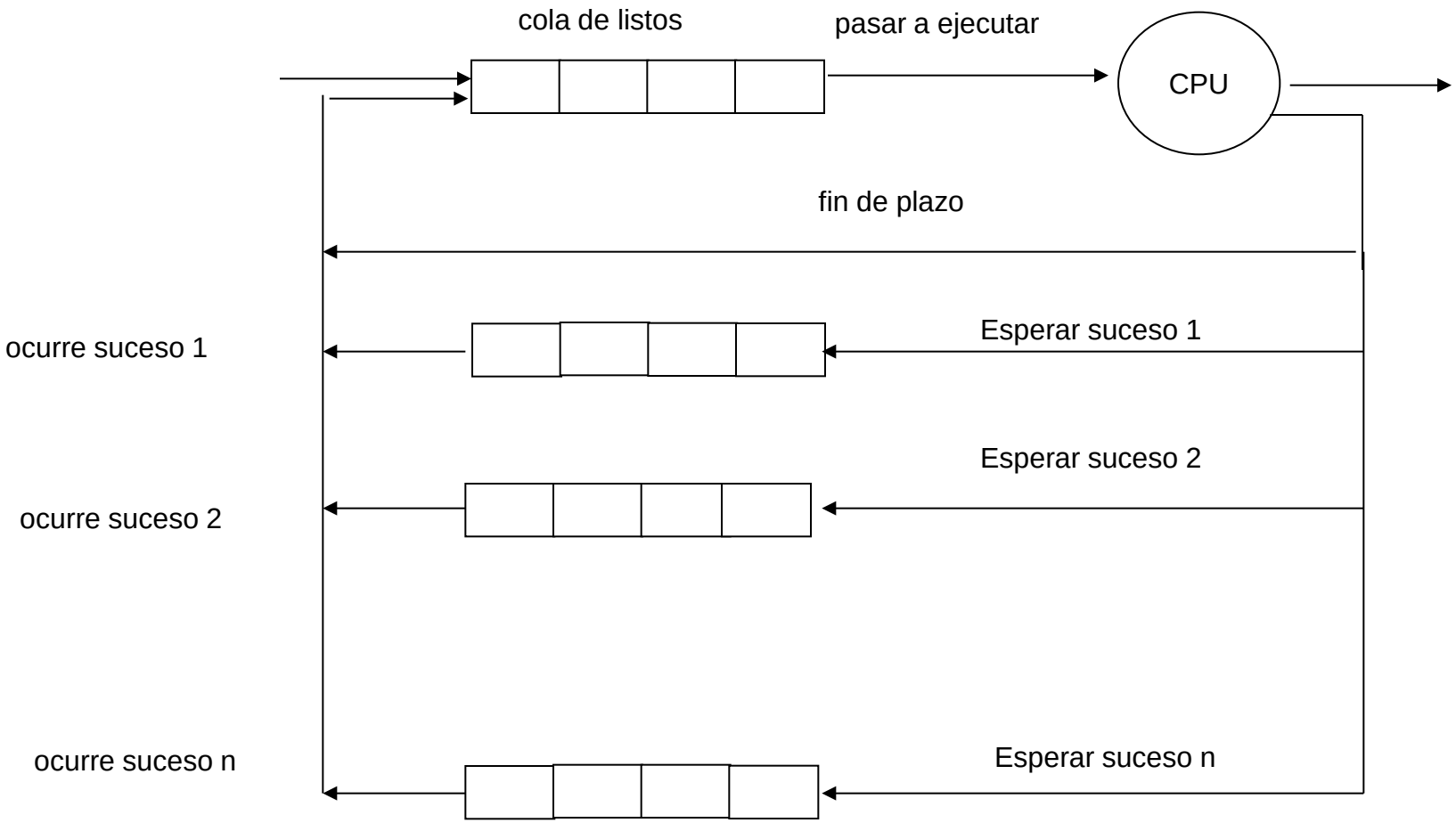
Bloque de control de proceso(BCP)



Colas de planificación

- El S.O. organiza los BCP en colas de espera por el procesador o por los dispositivos de E/S.
  - Cola de trabajos – Conjunto de todos los procesos del sistema.
  - Cola de proc. listos – Conjunto de procesos cargados en la memoria principal,listos para su ejecución.
  - Cola de dispositivo – conjunto de procesos esperando a un dispositivo de E/S.
- Los procesos pasan de unas colas a otras.

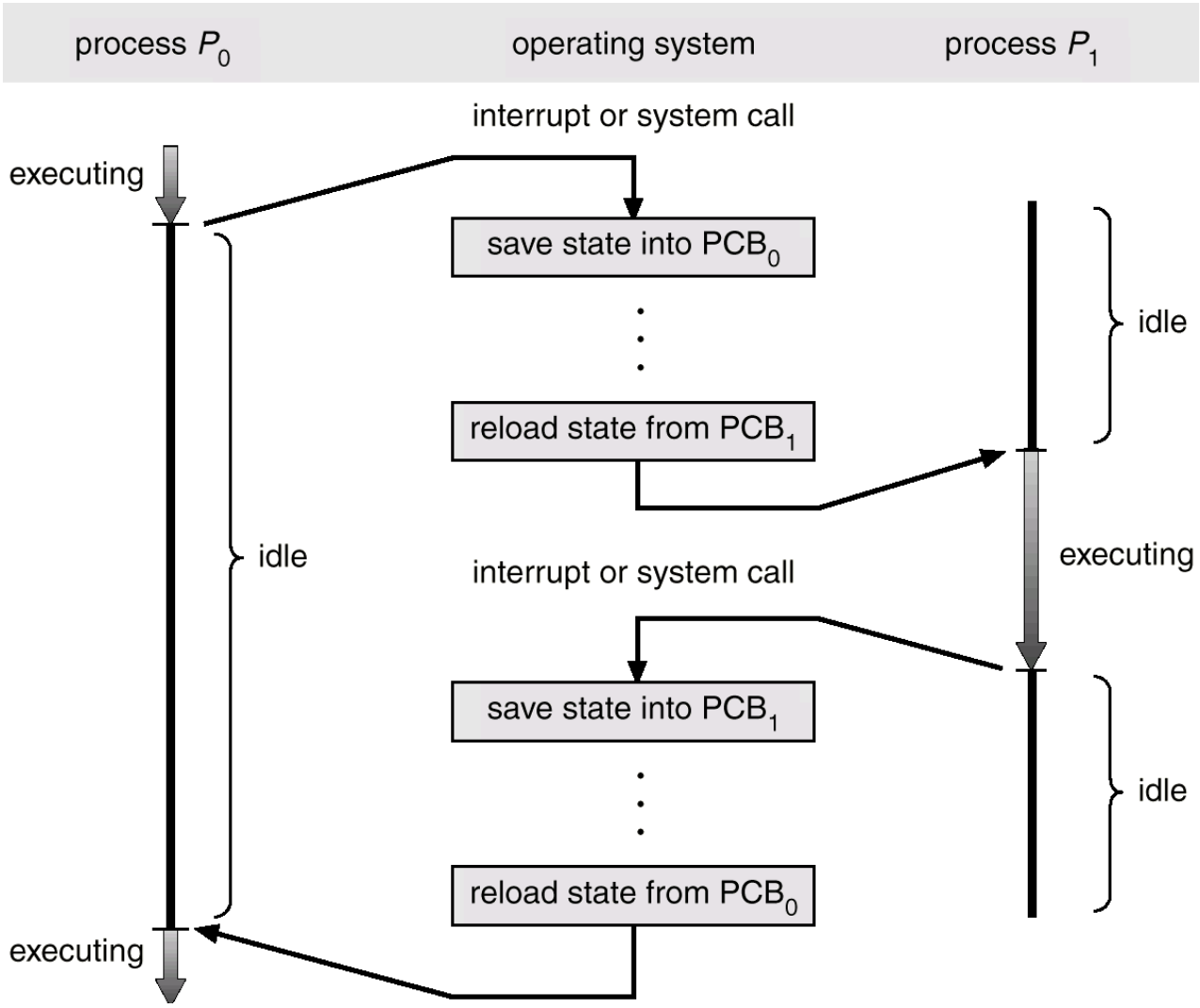
Cola de proc. listos y varias colas de disp. E/S



Cambio de contexto

- El cambio de contexto es la operación que consiste en desalojar a un proceso de la CPU y reanudar otro. El sistema debe guardar el estado del proceso expulsado y cargar el estado guardado del proceso que se admite.
- La conmutación de contexto es un gasto extra; el sistema no realiza ningún trabajo útil mientras conmuta.
- Los tiempos de conmutación dependen en buena medida del apoyo del hardware. El hardware a veces tiene instrucciones para facilitar el cambio de contexto (ej. PUSHA = guardar todos los registros)

Cambio de Contexto



Creación de procesos

- El s.o. cuando crea un proceso:
  1. asigna un identificador al proceso y lo refleja en la tabla de procesos
  2. asigna memoria al proceso y BCP
  3. inicializa el BCP: contador programa, punteros pilas sistema, estado proceso, prioridad, recursos.
  4. se coloca el proceso en la cola de listos o listos suspendidos.
  5. puede que haya que actualizar otras estructuras de datos p.e. archivos para contabilidad.

Creación de procesos

- Un proceso se crea mediante una llamada al sistema.
- El proceso creador se denomina padre, y el creado, hijo. Los hijos pueden a su vez crear otros procesos, formando un árbol de procesos
- Ejecución:
  - Padre e hijos se ejecutan concurrentemente.
  - El padre espera a que algunos de los hijos o todos terminen.
- Espacio de direcciones:
  - El proceso hijo es un duplicado del proceso padre.
  - Se carga un programa en el proceso hijo.

Terminación de procesos

- Un proceso termina cuando finaliza la ejecución de su último enunciado y le pide al sistema operativo que lo elimine invocando una llamada al sistema específica (ej:exit).
- También si se genera una excepción y el S.O. decide abortarlo
- Podría existir una llamada al sistema para abortar otro proceso

Relaciones de comunicación entre procesos

- Tipos de procesos
  - Independientes
  - Cooperantes
- Un proceso es **independiente** si no puede afectar ni ser afectado por la ejecución de otro proceso.
- Un proceso es **cooperante** si puede afectar o ser afectado por la ejecución de otro proceso
- Ventajas de la cooperación entre procesos
  - Compartir información
  - Aceleración de los cálculos
  - Modularidad
  - Comodidad

Problemas clásicos de comunicación y sincronización

- El problema de la sección crítica
- El problema del productor-consumidor
- El problema de los lectores-escritores
- Comunicación cliente-servidor

Problema del productor-consumidor

- Paradigma de procesos cooperativos, el proceso *productor* produce información que es consumida por un proceso *consumidor*. Ej.un programa de impresión produce caracteres que son consumidos por el controlador de la impresora.

  - *buffer ilimitado* no establece limitación en el tamaño del buffer.
  - *buffer limitado* supone que existe un tamaño determinado para el buffer.

Problema del productor-consumidor
Buffer limitado – Solución con Memoria Compartida

Productor

```
while(CIERTO)
{
        sig_prod = Produce( );
        while(cont == n)
        {
                /*No hace nada. Espera*/
        }
        buffer[ent] = sig_prod;
        ent ++;    /* ent = (ent+1) % n;  buffer circular*/
        cont ++;
}
```

Consumidor

```
while(CIERTO)
{
        while(cont==0)
        {
                /*No hace nada. Espera*/
        }
        sig_cons = buffer[sal];
        sal ++;    /* sal = (sal+1) % n;  buffer circular*/
        cont --;
        Consume(sig_cons);
}
```

Aunque ambas rutinas son correctas por separado puede haber problemas al ejecutarlas concurrentemente

Problema del productor-consumidor
Buffer limitado – Solución con Memoria Compartida

- Para mantener la consistencia de los datos se requieren mecanismos que aseguren una ejecución ordenada de los procesos cooperativos.
- Consideremos la solución con memoria compartida al problema del productor-consumidor. En el código la variable *cont,* es inicializada al comienzo a 0 y modificada cada vez que se añade o retira un elemento del buffer.
- El orden en que se ejecuten las lecturas y escrituras sobre la variable *cont* es crítico.
- El compartimiento de recursos globales puede traer muchos problemas.

Problema del productor-consumidor
Buffer limitado – Solución con Memoria Compartida

- La solución para evitar los problemas presentados es proporcionar algún mecanismo de exclusión mutua. Es decir, algún procedimiento que garantice que cuando un proceso esta accediendo a una variable compartida, el otro proceso estará excluido de hacer lo mismo.

TEMA 4
Comunicación y Sincronización de Procesos

ESCUELA DE INGENIERÍAS INDUSTRIALES

Departamento de Ingeniería de Sistemas y Automática. EII.

Universidad de Valladolid

Capítulo 4: Comunicación y Sincronización de Procesos

- Introducción
- El problema de la sección crítica
- Semáforos
- Problemas clásicos de sincronización

Introducción

- Hemos considerado la solución con memoria compartida al problema del productor-consumidor y hemos visto que el orden en que se ejecutaban las lecturas y escrituras sobre la variable compartida *cont* es crítico. Para mantener la consistencia de los datos se requieren mecanismos que aseguren una ejecución ordenada de los procesos cooperativos.

Introducción

- Estas situaciones donde dos o más procesos están leyendo o escribiendo algunos datos compartidos y el resultado final depende de la secuencia de ejecución se denominan **condiciones de competencia**.
- En este contexto hay que contemplar dos asuntos:
  - Cómo garantizar que dos procesos no interfieren cuando acceden a un recurso común
  - Cómo secuenciar los procesos cuando existe dependencia entre ellos.
- Veremos que la exigencia básica de la concurrencia es la posibilidad de hacer cumplir la **exclusión mutua**: excluir a los demás procesos de realizar ciertas acciones cuando es otro el que tiene permiso

Introducción

- La solución para evitar las condiciones de competencia es proporcionar algún mecanismo de exclusión mutua. Es decir, algún procedimiento que garantice que cuando un proceso esta accediendo a una variable compartida, el otro proceso estará excluido de hacer lo mismo.
- La parte del programa donde se accede a variables compartidas se denomina **sección crítica**.
- Se pueden evitar las condiciones de competencia si se controla la ejecución de forma que no puedan estar dos procesos en sus secciones críticas simultáneamente

Problemas clásicos de comunicación y sincronización

- El problema de la sección crítica
- El problema del productor-consumidor
- El problema de los lectores-escritores
- Comunicación cliente-servidor

El problema de la sección crítica

- Sistema compuesto por *n* procesos
- Cada uno tiene un fragmento de código: *sección crítica*
- Sólo uno de los procesos en cada instante puede ejecutar en la sección crítica
- Cuando un proceso está ejecutando en la sección crítica, ningún otro puede hacerlo

El problema de la sección crítica

- n procesos que compiten por el uso de datos compartidos
- Cada proceso tiene su segmento de código en el que se accede a los datos compartidos (sección crítica).
- Problema – asegurar que cuando un proceso está ejecutando su sección crítica no se permita que otro proceso pueda ejecutar su sección crítica.
- Estructura del proceso P_i

repeat

sección de entrada

sección crítica

sección de salida

sección restante

Solución al problema de la sección crítica

1. Dos procesos no pueden estar al mismo tiempo en sus regiones críticas
2. Ningún proceso deberá esperar indefinidamente para entrar en su región crítica.
3. Ningún proceso que no esté en su región crítica puede bloquear otros procesos
4 . No puede hacerse ninguna suposición sobre la velocidad relativa de los procesos.

Primeras tentativas para resolver el problema

- Consideremos 2 procesos, P_0 y P_1
- La estructura general del proceso P_i

repeat

sección de entrada

sección crítica

sección de salida

sección restante

- Los procesos pueden compartir variables comunes para sincronizar sus acciones.

Alternancia estricta

PROCESO 0

```
.
.
.

while (turno!=0)
{
/*No hace nada. Espera*/
}

SeccionCritica ();
turno=1;
SeccionNoCritica ();
.
.
.
```

PROCESO 1

```
.
.
.

while (turno!=1)
{
/*No hace nada. Espera*/
}

SeccionCritica ();
turno=0;
SeccionNoCritica ();
.
.
.
```

Alternancia estricta

- La variable *turno,* indica a que proceso le toca entrar en la región crítica. Al proceso que no le toca se mantiene en un ciclo para detectar cuando conmuta la variable *turno* y por tanto le corresponde entrar en la r.c. Este chequeo continuo de una variable se denomina *espera activa*.
- Inconvenientes de la alternancia estricta:
  - Los procesos deben alternarse de forma estricta en el uso de sus secciones críticas. No es un buen enfoque cuando un proceso es mucho más rápido que el otro. Se viola la condición 3 mencionada: un proceso está bloqueado por otro que no está en su región crítica.
  - Si un proceso falla el otro se queda bloqueado.

Segundo intento

```
# define FALSO 0
# define CIERTO 1
# define N 2                                  /* Número de procesos*/

int señalBloqueo[N];                          /* arreglo global compartido por los dos procesos*/

PROCESO 0                                     PROCESO 1
.                                             .
.                                             .
.                                             .


 while (señalBloqueo[1])                      while (señalBloqueo[0])
{                                             {

        /*No hace nada. Espera*/                      /*No hace nada. Espera*/


}                                             }
 señalBloqueo[0]=CIERTO;                      señalBloqueo[1]=CIERTO;
SeccionCritica ();                            SeccionCritica ();
 señalBloqueo[0]=FALSO;                       señalBloqueo[1]=FALSO;
 SeccionNoCritica ();                         SeccionNoCritica ();
.                                             .
.                                             .
.                                             .
```

Segundo intento

- No ralentiza el proceso rápido esperando al proceso lento.
- Si falla un proceso sólo se queda bloqueado el otro proceso si el fallo ha ocurrido en la sección crítica.
- Inconveniente:
  - P0 ejecuta el while y encuentra señal[1] a falso.
  - P1 ejecuta el while y encuentra señal[0] a falso.
  - P0 pone señal[0] a cierto y entra en su sección crítica.
  - P1 pone señal[1] a cierto y entra en su sección crítica.

  ¡¡¡ Ambos procesos se encuentran en su sección crítica !!!
- El problema aparece porque un proceso puede cambiar su señal después de que el otro proceso ha verificado esta.

Tercer intento

```
# define FALSO 0
# define CIERTO 1
# define N 2                          /* Número de procesos*/

int señalBloqueo[N];                  /* arreglo global compartido por los dos procesos*/

PROCESO 0                             PROCESO 1
.                                     .
.                                     .
.                                     .

 señalBloqueo[0]=CIERTO;               señalBloqueo[1]=CIERTO;
 while (señalBloqueo[1])               while (señalBloqueo[0])
{                                     {

        /*No hace nada. Espera*/              /*No hace nada. Espera*/

}                                     }

 SeccionCritica ();                    SeccionCritica ();
 señalBloqueo[0]=FALSO;                señalBloqueo[1]=FALSO;
 SeccionNoCritica ();                  SeccionNoCritica ();
.                                     .
.                                     .
.
```

Garantiza la exclusión mutua pero puede haber interbloqueo

Algoritmo de Peterson

```
 # define FALSO 0
 # define CIERTO 1
   # define N 2                          /* Número de        procesos*/
```

```
 PROCESO 0                                  PROCESO 1
.                                          .
.                                          .
.                                          .

  señal[0]=CIERTO;                          señal[1]=CIERTO;
 turno=1;                                  turno=0;
   while (señal[1] && turno==1)              while (señal[0] && turno==0)
{                                          {
  /*No hace nada.    Espera.*/              /*No hace nada.    Espera.*/
}                                          }

  SeccionCritica ();                        SeccionCritica ();
  señal[0]=FALSO;                           señal[1]=FALSO;
   SeccionNoCritica ();                      SeccionNoCritica ();
.                                      .   .
.                                          .
.                                          .
```

Algoritmo de Peterson

- La variable global *señal* indica el interés de cada proceso por acceder a su región crítica y la variable *turno* resuelve los conflictos de simultaneidad.
- El algoritmo de Peterson constituye una solución sencilla al problema de exclusión mutua.

Excl. Mutua. Soluciones por Hardware.

Una forma de lograr la exclusión mutua podría ser inhabilitar las interrupciones. Pero cuidado, las interrupciones son fundamentales, son el corazón del software y hardware.

- si son inhabilitadas durante demasiado tiempo, el sistema fallará en la respuesta a los sucesos de hardware y la conmutación entre tareas no funcionará
- cuando existe más de un procesador en el sistema no se garantiza la exclusión mutua puesto que el otro procesador no está interrumpido.
- al código de usuario no se le debe permitir desactivar las interrupciones.

El desactivar las interrupciones una solución razonable para la exclusión mutua sólo en las siguientes condiciones:

- para secciones de código cortas
- sobre sistemas monoprocesador
- para el mismo sistema operativo

Semáforos

- Un semáforo es una variable entera a la que se puede acceder sólo a través de dos operaciones atómicas: **semDown** y **semUp**.
- Para subir el semáforo en una unidad los procesos emplean **semUp**(s)
- Para bajar una unidad el semáforo emplean **semDown**(s)

```
semDown(s)
{
        s--;
        while(s<0)
        {
        /* No hace nada. Espera*/
        }
}
```

```
semUp(s)
{
        s++;
}
```

Semáforos

- Por tanto, los semáforos son variables enteras que permiten **sólo tres operaciones**:
  - Inicialización con un valor no negativo.
  - Decremento en una unidad llamando a **semDown**. Si el semáforo no es positivo al llamar a **semDown** se bloquea el proceso.
  - Incremento en una unidad llamando a **semUp**

Implementación de un semáforo

```
struct semaforo
 {
        int cont;
       struct proceso *cola;
 };
struct semaforo s;
semDown(s)
 {
       InhabilitarInterrupciones;
        s.cont--;
       if ( s.cont< 0 )
        {
       PoneProcesoEnCola( s.cola ) ;

   /* Bloquea  proceso que llamó a semDown */
       BloqueaProceso();

        }
       HabilitarInterrupciones;
 }
```

```
semUp(s)
 {
       InhabilitarInterrupciones;
        s.cont++;
        if ( s.cont<= 0 )
        {
              proc = SacaProcesoDeCola( s.cola ) ;

              /*pone al proceso extraído de la cola
              de bloqueados en la cola de listos*/
               DespertarProceso( proc );
        }
       HabilitarInterrupciones;
 }
```

Ejemplo: Sección crítica de *n* Procesos

- Variables semáforo mutex , inicialmente mutex=1;
- Proceso P_i

repeat

semDown(*mutex*);

sección critica

semUp(*mutex*);

sección no critica

until *false*;

Semáforo para sincronización

- Ejecutar el bloque de código *B* del proceso 2 después del bloque A del proceso 1.
- Usar semáforo *sinc* inicializado a 0
- Código

P_1	P_2
⋮	⋮
A	*semDown*(*sinc*)
semUp(*sinc*)	*B*

Interbloqueo e inanición

- **Interbloqueo** – dos o más procesos esperan indefinidamente un suceso que puede ser causado sólo por uno de los procesos que esperan.
- Sean *S* y *Q* dos semáforos inicializados a 1

P_0	P_1
semDown(*S*);	*semDown*(*Q*);
semDown(*Q*);	*semDown*(*S*);
⋮	⋮
semUp(*S*);	*semUp*(*Q*);
semUp(*Q*)	*semUp*(*S*);

- **Inanición** – Un proceso espera indefinidamente dentro de la cola de un semáforo.

Problemas clásicos de sincronización

- Productor consumidor con buffer acotado
- Problema de los lectores-escritores
- El problema cliente-servidor

Productor Consumidor buffer acotado

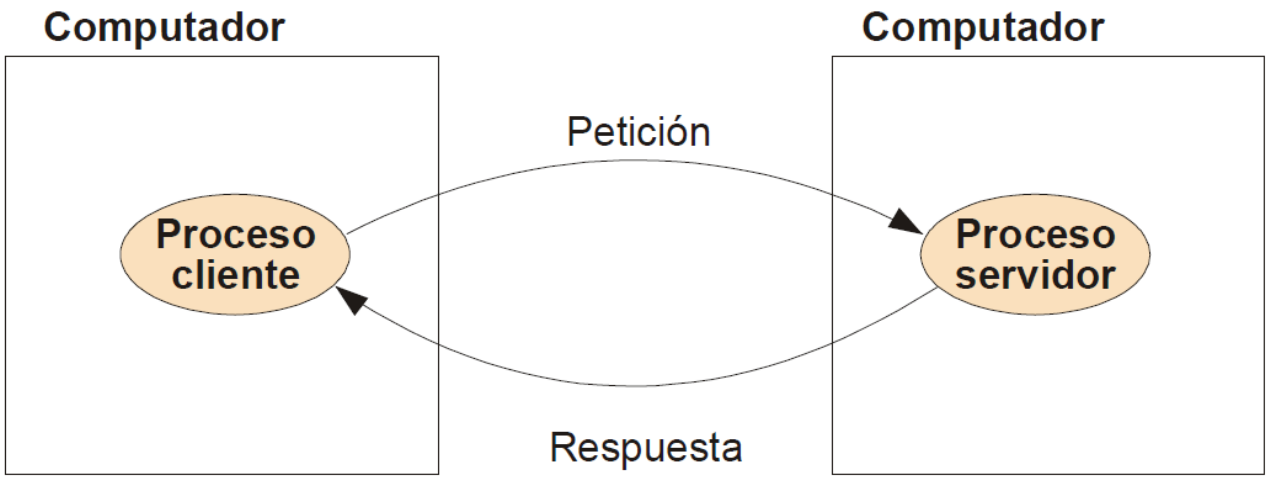
```
#define N 100                       /*capacidad del buffer*/
semaforo mutex=1;                   /*gestiona acceso a región critica*/
semaforo vacios=N;                  /*cuenta el numero de espacios vacíos en buffer*/
semaforo llenos=0;                  /*cuenta el numero de espacios llenos en el buffer*/

void Productor(void)
{
int elemento;
 while(TRUE)                        /*bucle infinito*/
        {
        ProducirElemento(elemento);/*generar algo para ponerlo en el buffer*/
        semDown(vacios);            /*disminuye el contador de espacios vacíos*/
        semDown(mutex);             /*entra en la región critica*/
        IntroducirElemento(elemento);/*coloca el nuevo elemento en el buffer*/
        semUp(mutex);               /*sale de la región critica*/
        semUp(llenos);              /*incrementa el contador de espacios llenos*/
        }
}
```

```
void Consumidor(void)
{
int elemento;
while(TRUE)                         /*bucle infinito*/
     {
     semDown(llenos);               /*disminuye el contador de espacios llenos*/
     semDown(mutex);                /*entra en la región critica*/
     ExtraerElemento(elemento);/*saca un elemento del buffer*/
     semUp(mutex);                  /*sale de la región critica*/
     semUp(vacios);         /*incrementa el contador de espacios llenos*/
     ConsumirElemento(elemento);  /*hacer algo con el elemento*/
     }
 }
```

Problema de los lectores escritores

```
/*************una solución al problema cuando los lectores tienen prioridad*************** */
semaforo mutex=1;       /*garantiza la mutua exclusión al actualizar num_lectores*/
semaforo escribe=1;     /* evita otro proceso (lector o escritor) cuando hay un escritor */
int num_lectores=0;     /*numero de procesos que están leyendo*/
void Escritor()
{
        CalculaNvosValores();       /*región no crítica*/
        semDown(escribe);           /*evita que entre otro escritor o lector en r.c.*/
        EscribeElemento();          /*actualiza los datos*/
        semUp(escribe);             /*da paso a otro escritor o lectores*/
}
void Lector()
{
        semDown(mutex);                         /*evita problemas con el contador num_lectores*/
        num_lectores ++;                        /*ahora hay un lector más*/
        if(num_lectores == 1) semDown(escribe); /*si este es el primer lector espera a que termine el escritor
                                                  si no hay escritor bloquea su entrada*/
        semUp(mutex);                           /*libera el acceso exclusivo a num_lectores*/
        LeeElemento();                          /*Accede a leer los datos*/
        semDown(mutex);                         /*entrar en la región critica*/
        num_lectores --;                        /*ya hay un lector menos*/
        if(num_lectores == 0) semUp(escribe);   /*si no hay ningún lector se da permiso a escritor*/
        semUp(mutex);                           /*salir de la región critica*/
        UtilizaElemento();                      /*región no crítica*/
}
```

Cliente - Servidor



- Ver programas C cliente-servidor con memoria compartida en prácticas

TEMA 5.

El computador en la automatización de la Producción

TEMA 5.
El computador en la automatización de la producción

ESCUELA DE INGENIERÍAS INDUSTRIALES

Departamento de Ingeniería de Sistemas y Automática. EII.

Universidad de Valladolid

El computador en la automatización de la producción

- Introducción a automatización industrial
- Diseño asistido por computador
- Ingeniería asistida por computador
- Fabricación asistida por computador
- Fabricación integrada por computador
- La pirámide de la automatización

Introducción Automatización Industrial

La automatización Industrial implica el uso de sistemas informáticos y electromecánicos para para llevar a cabo procesos que se autocomprueban y autocorrigen.

Abarca la **instrumentación industrial** (sensores, los buses de campo) los **sistemas de control y supervisión**, los **sistemas de transmisión y recolección de datos** y las aplicaciones de software en **tiempo real** para supervisar, controlar las operaciones de plantas o procesos industriales

Introducción Automatización Industrial

El concepto de automatización sufrió un gran empuje con la incorporación de las **computadoras digitales** a los procesos productivos, cuya flexibilidad permitió manejar cualquier clase de tarea.

A partir de la década de los 60, las computadoras digitales se incorporaron en el control de tareas simples, repetitivas, tareas semiespecializadas y especializadas, con algunas excepciones notables en la producción e inspección de alimentos.

Introducción Automatización Industrial

Algunas ventajas son repetitividad, control de calidad más fiable, mayor eficiencia, integración con otros sistemas de la empresa, aumento de la productividad .

Introducción Automatización Industrial

Ventajas competitivas de la automatización del diseño y la fabricación:

1. Aumentar la productividad del trabajo
2. Reducir costes laborales
3. Mitigar los efectos de la escasez de mano de obra
4. Reducir o eliminar tareas rutinarias manuales y tareas rutinarias de oficina
5. Mejorar la seguridad de los trabajadores
6. Mejorar la calidad del producto
7. Reducir el plazo de ejecución de la fabricación
8. Lograr lo que no se puede hacer manualmente
9. Evitar el alto coste de no automatizar , ya que conlleva pérdida de competitividad

Introducción Automatización Industrial

Los elementos más emblemáticos de la automatización actual puede ser los **PLCs**, máquinas herramienta con **control numérico y robots industriales**

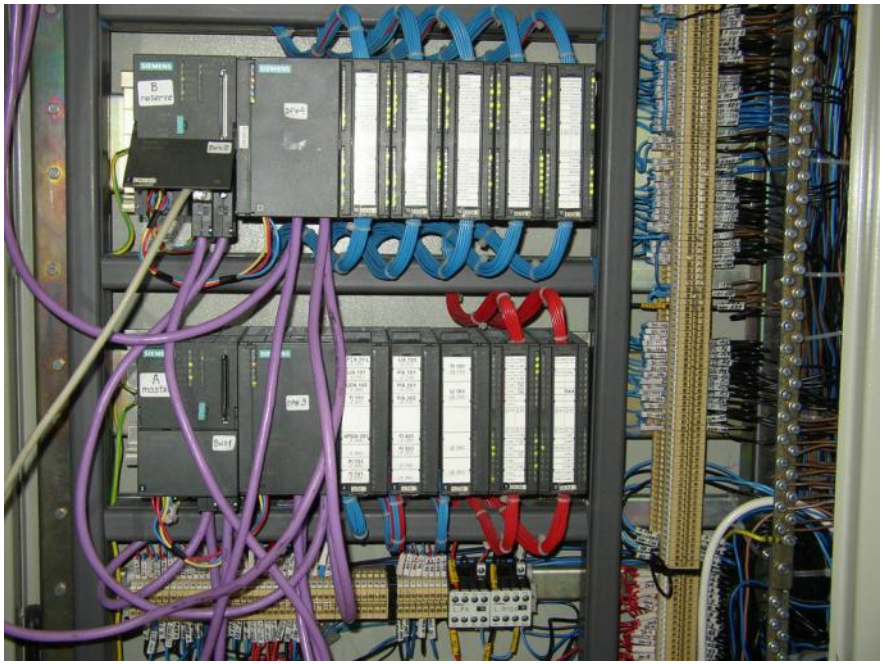
- Robótica Industrial – se controlan las articulaciones del manipulador para mover y orientar el extremo del brazo mediante una secuencia de posiciones en el ciclo de trabajo.

- Control numérico por computador (CNC) – el computador controla una herramienta en relación con el objeto que se procesa. Realiza los cálculos necesarios para determinar la trayectoria de la herramienta.

Controlador lógico programable (PLC)

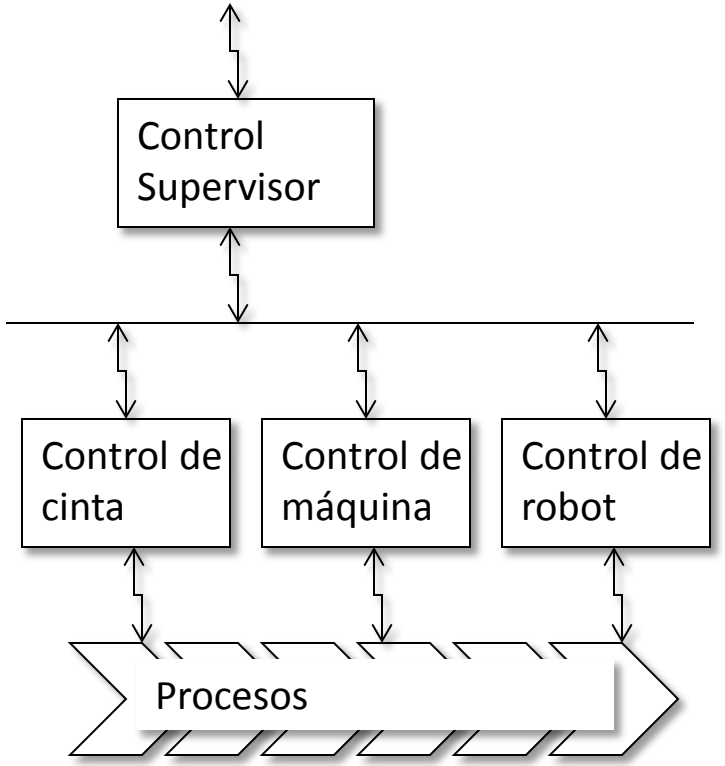
Autómata programable

- Implementa la lógica, secuenciación, cálculo y funciones aritméticas para controlar máquinas y procesos industriales.
- Funcionalidades de control discreto y continuo

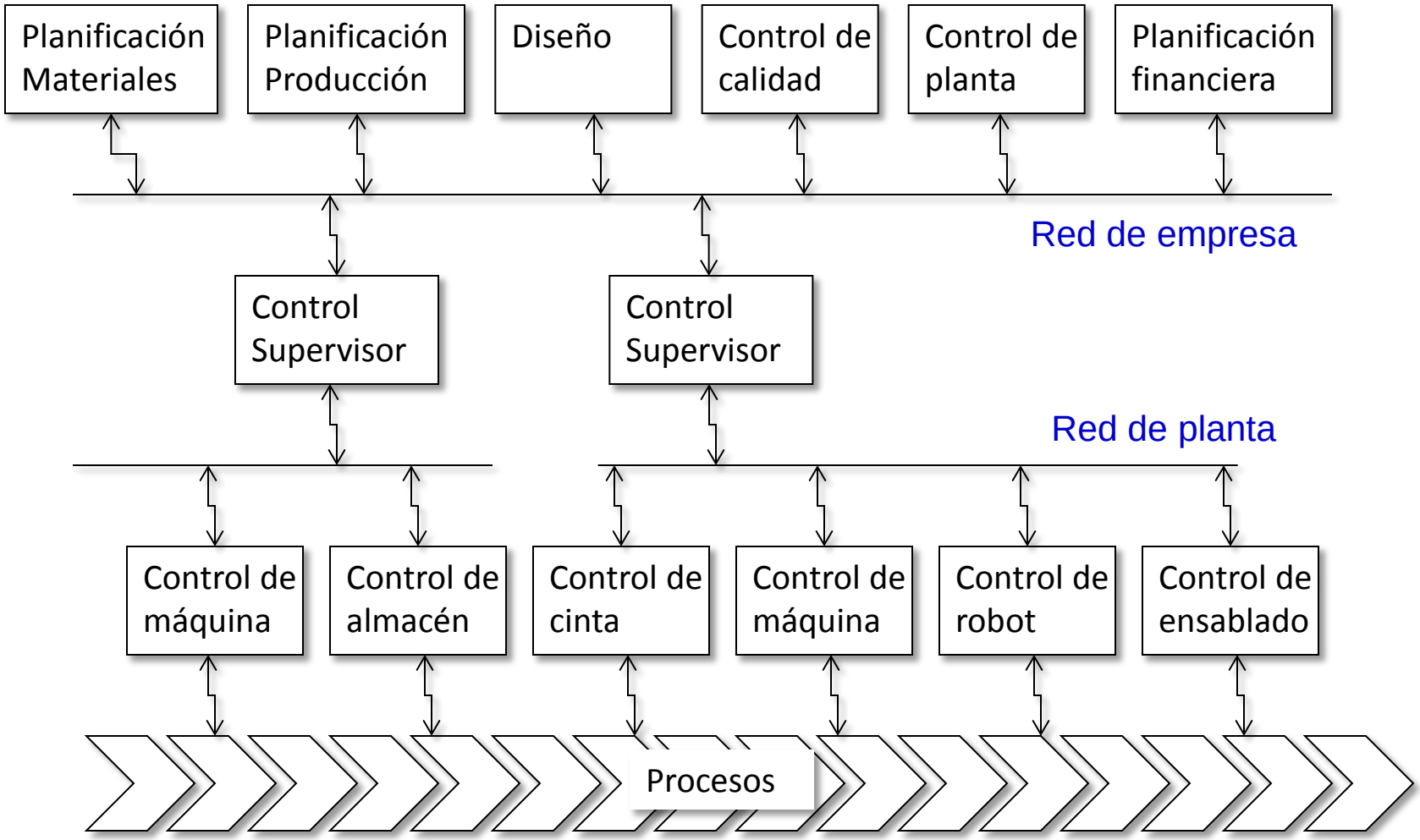


Control Supervisor

- Control de supervisión, controla las operaciones de una serie de unidades integradas en una célula:
  - Máquinas
  - Robots
  - Cintas transportadoras



Sistema de Control Distribuido



El ordenador y el ciclo de proceso de un producto

El incremento de la demanda de productos a escala mundial ha supuesto:

- Disminuir los lotes de fabricación y los plazos de industrialización.
- Incrementar la variedad y complejidad de los productos a desarrollar.

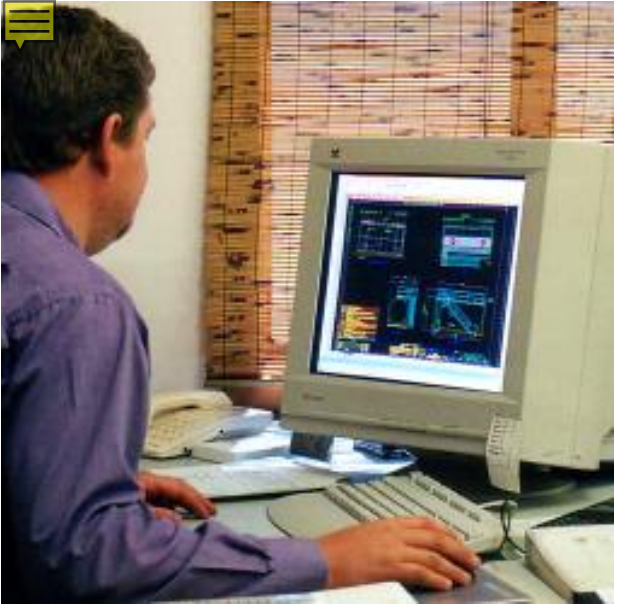
Surge el **ciclo del proceso de un producto**, en el que se utiliza el computador para **automatizar el diseño y la fabricación** y lograr dos grandes objetivos:

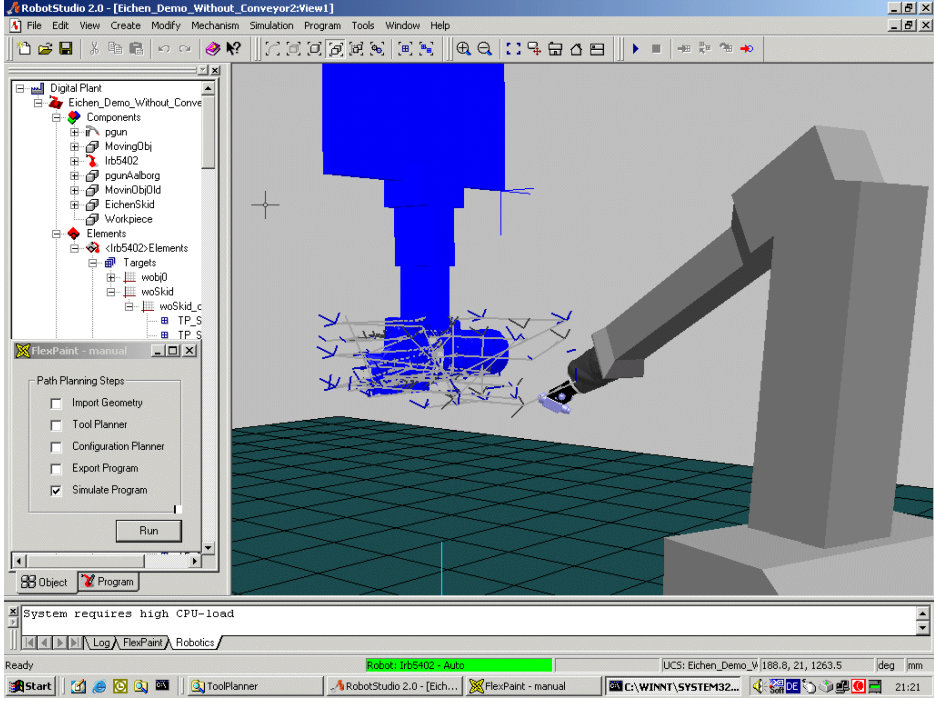
- Aplicar métodos de diseño de sistemas complejos que permitan validar prototipos y llevar a cabo su producción en serie.
- La ejecución de tareas de producción mediante máquinas con un alto nivel de automatización.

El ordenador y el ciclo de proceso de un producto

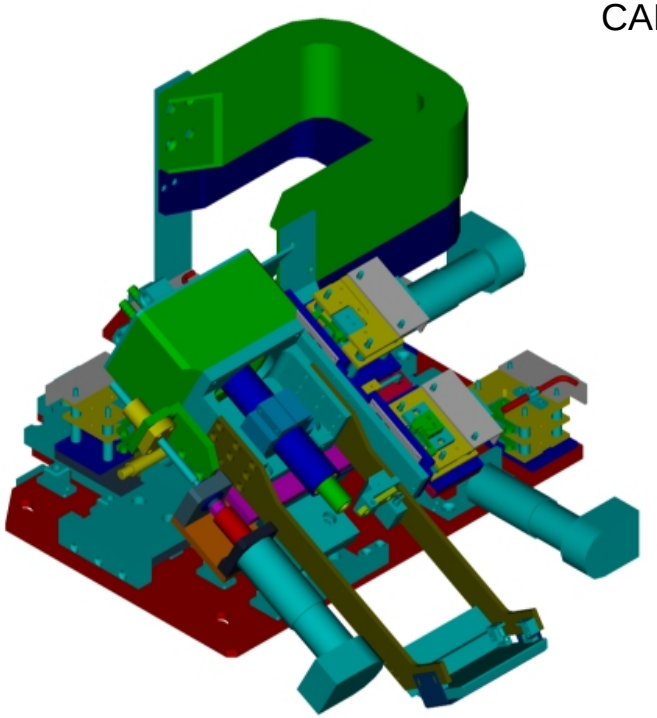
La automatización del diseño y la fabricación se llevan a cabo mediante el siguiente conjunto de técnicas:

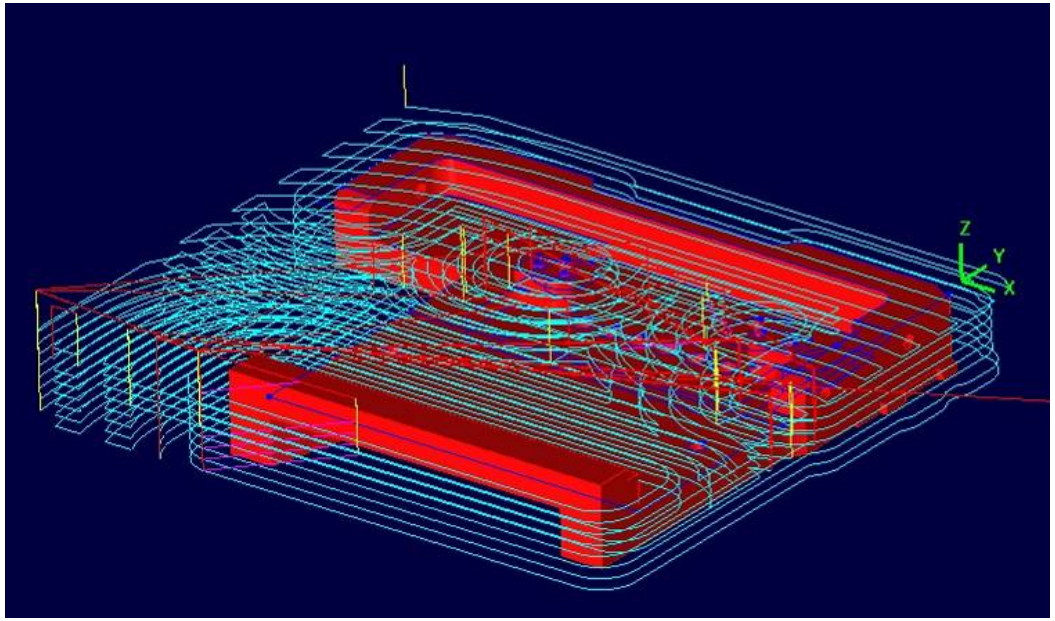
- El diseño asistido por computador, conocido como CAD (*Computer Aided Design*)
- La simulación e ingeniería asistida por computador, conocidas como CAE (*Computer Aided Engineering*)
- La fabricación asistida por computador, conocida como CAM (*Computer Aided Manufacturing*)




RobotStudio 2.0 - [Eichen_Demo_Without_Conveyor2:View1]
File Edit View Create Modify Mechanism Simulation Program Tools Window Help
Digital Plant
Eichen_Demo_Without_Conve
Components
pgun
MovingObj
Irb5402
pgunAalborg
MovinObjOld
EichenSkid
Workpiece
Elements
<Irb5402>Elements
Targets
wobj0
woSkid
FlexPaint - manual
Path Planning Steps
Import Geometry
Tool Planner
Configuration Planner
Export Program
Simulate Program
Run
Object
Program
System requires high CPU-load
Log
FlexPaint
Robotics
Ready
Robot: Irb5402 - Auto
UCS: Eichen_Demo_W
188.8, 21, 1263.5
deg
mm
Start
ToolPlanner
RobotStudio 2.0 - [Eich...
FlexPaint - manual
C:\WINNT\SYSTEM32...
21:21

CAD/CAM




Z
Y
X

Diseño asistido por computador (CAD)

Mediante el CAD se logra:

- **Mejorar la calidad** de los productos, porque permite desarrollar alternativas y solventar los problemas en las etapas iniciales del proceso de diseño
- **Reducir el tiempo de diseño**, lo cual disminuye el coste y el tiempo que se tarda en lanzar un producto
- **Reducir los costes** de fabricación, porque facilita los cambios
- Facilitar la capacidad de **reutilización de los diseños**

Simulación/Ingeniería asistidos por computador (CAE)

Utiliza la descripción del sistema como entrada de un programa de computador que hace que este último se comporte igual que él.

Permite llevar a cabo las **pruebas** necesarias para garantizar el correcto funcionamiento de un producto **sin necesidad de implementar un prototipo**.

CAE es el conjunto de técnicas que utilizan el computador para:

- analizar el resultado de un diseño,
- facilitar al máximo su fabricación y optimizar las prestaciones y costes totales del producto final.

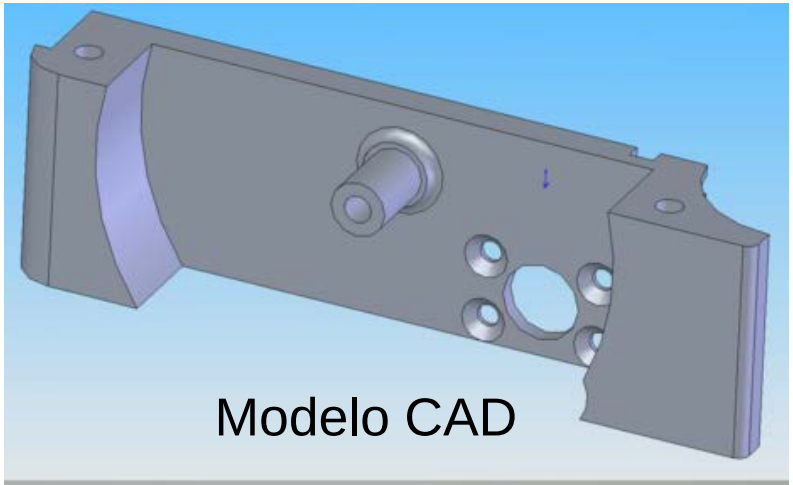
Fabricación asistida por computador (CAM)

CAM es el empleo de software para controlar máquinas herramientas en la fabricación de piezas.

CAM es un proceso a continuación del CAD y CAE, ya que el modelo generado en CAD y verificado en CAE se utiliza como entrada en el software CAM, el cual entonces controla la máquina herramienta.

Algunos ejemplos de CAM son: el fresado programado por control numérico, la realización de agujeros en circuitos automáticamente por un robot, y la soldadura automática de componentes electrónicos.

Fabricación asistida por computador (CAM)



Fabricación asistida por computador (CAM)

CAM puede también referirse al empleo del ordenador para mejorar las operaciones de una planta de fabricación: planificación, gestión, transporte y almacenamiento. En este caso el objetivo es crear un proceso de producción más rápido, más preciso, que minimice el desperdicio y reduzca el consumo de energía.

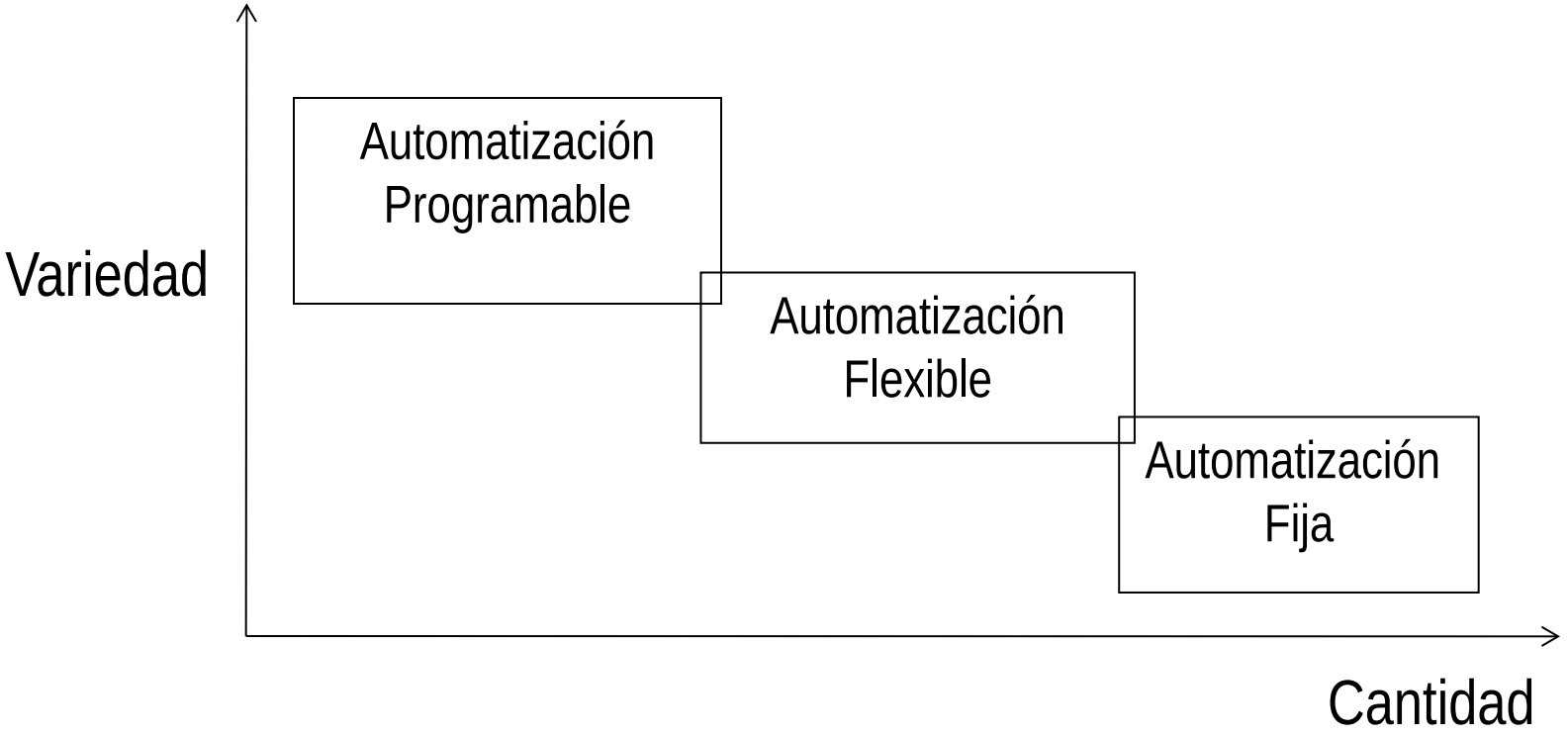
Fabricación asistida por computador (CAM) se caracteriza por:

- Necesita un conocimiento profundo del proceso productivo
- Puede ser la mejor solución para elevar la rentabilidad y garantizar la competitividad de una empresa industrial
- No siempre implica la mayor automatización posible, porque la automatización supone una inversión en activos fijos que, si es elevada, conduce a un considerable incremento de costes fijos y de mantenimiento.

Sistemas de Producción Automatizados

Cuatro tipos básicos:

- automatización fija
- automatización programable
- automatización flexible
- automatización integrada



Automatización fija

Sistema de fabricación en el que las operaciones de procesamiento (o montaje) vienen fijadas por la configuración de los equipos.

Características típicas:

- Adaptado a altas cantidades de producción
- Alta inversión inicial para equipos diseñados a medida
- Altas tasas de producción
- Relativamente inflexible en la fabricación de variedad de productos

Automatización programable

Sistema de fabricación diseñado con la capacidad de cambiar la secuencia de operaciones mediante reprogramación para adaptarse a diferentes configuraciones de productos.

Características típicas:

- Alta inversión en equipos programables
- Menores tasas de producción que la automatización fija
- Flexibilidad para hacer frente a las variaciones y los cambios en la configuración del producto
- El más conveniente para la producción de lotes
- La configuración física y el programa pieza debe ser cambiado entre órdenes de trabajo (lotes)

Automatización flexible

Extensión de la automatización programable en la que el sistema es capaz de pasar de un tipo de producto a otro con pocas pérdidas de tiempo entre ellos.

Características típicas:

- Alta inversión para el diseño personalizado del sistema
- Producción continua de mezcla variable de productos
- Tasas de producción medias
- Flexibilidad para hacer frente a variedad de productos parecidos

Automatización Integrada

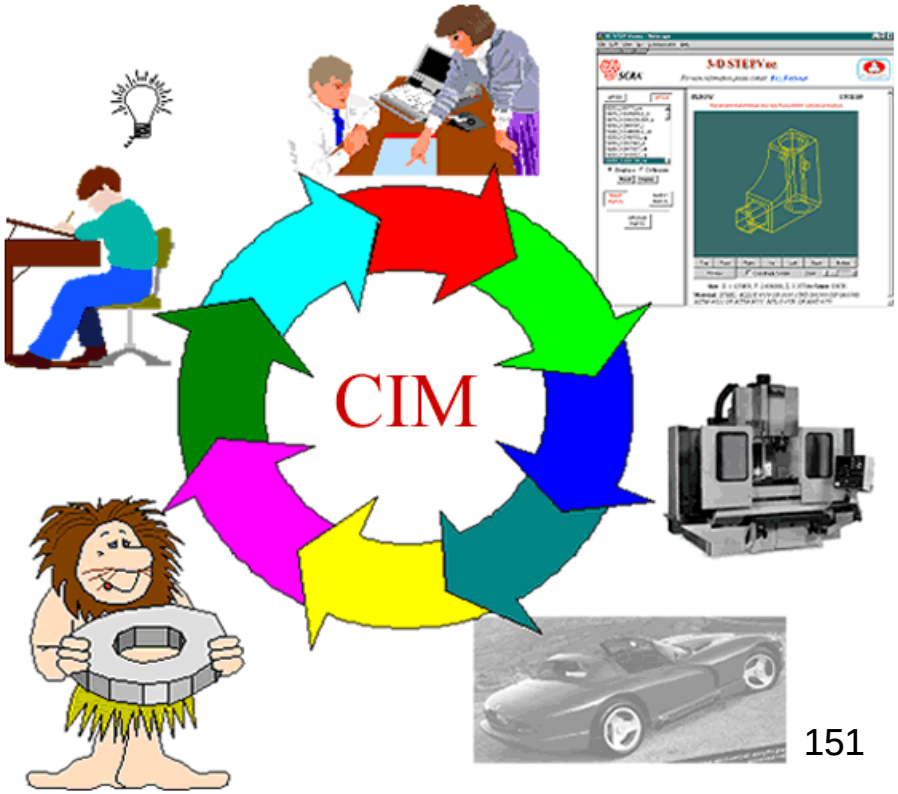
Es un sistema de fabricación que integra el diseño asistido por computador (CAD), la ingeniería asistida por computador (CAE) y la fabricación asistida por computador (CAM) con la verificación, la comercialización y la distribución.

La automatización integrada suele recibir el nombre de CIM (*Computer Integrated Manufacturing*). Se automatizan, de forma coordinada, todas las tareas que forman parte del ciclo completo de proceso del producto (*Totally Integrated Automation*)

Computer Integrated Manufacturing (CIM)

- El paradigma **CIM** (Computer Integrated Manufacturing), consiste en la integración de las actividades empresariales relacionadas con la producción, mediante el uso de tecnologías de la información, como bases de datos y redes, lo que permite el intercambio y la compartición de datos.

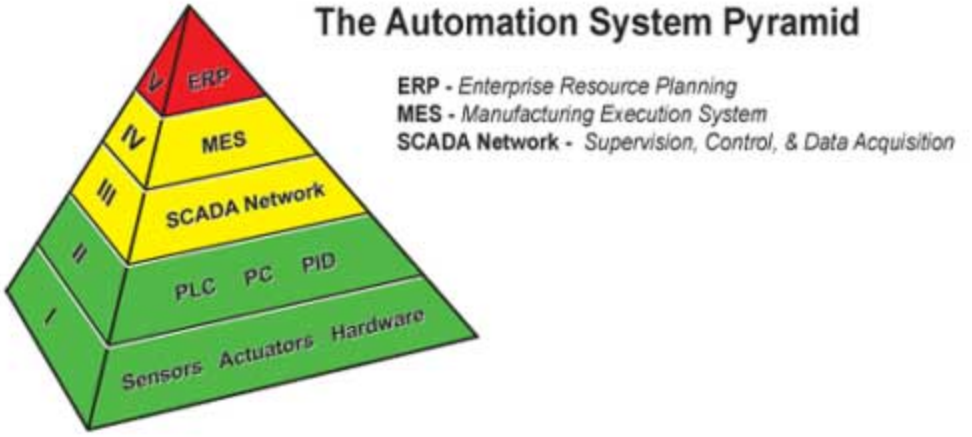
- El paradigma CIM tiene como objetivo la integración de varias tecnologías asistida por ordenador que dan soporte a los sistemas de producción, tales como Diseño Asistido por Ordenador (CAD), Ingeniería Asistida por Ordenador (CAE), Fabricación Asistida por Ordenador (CAM)



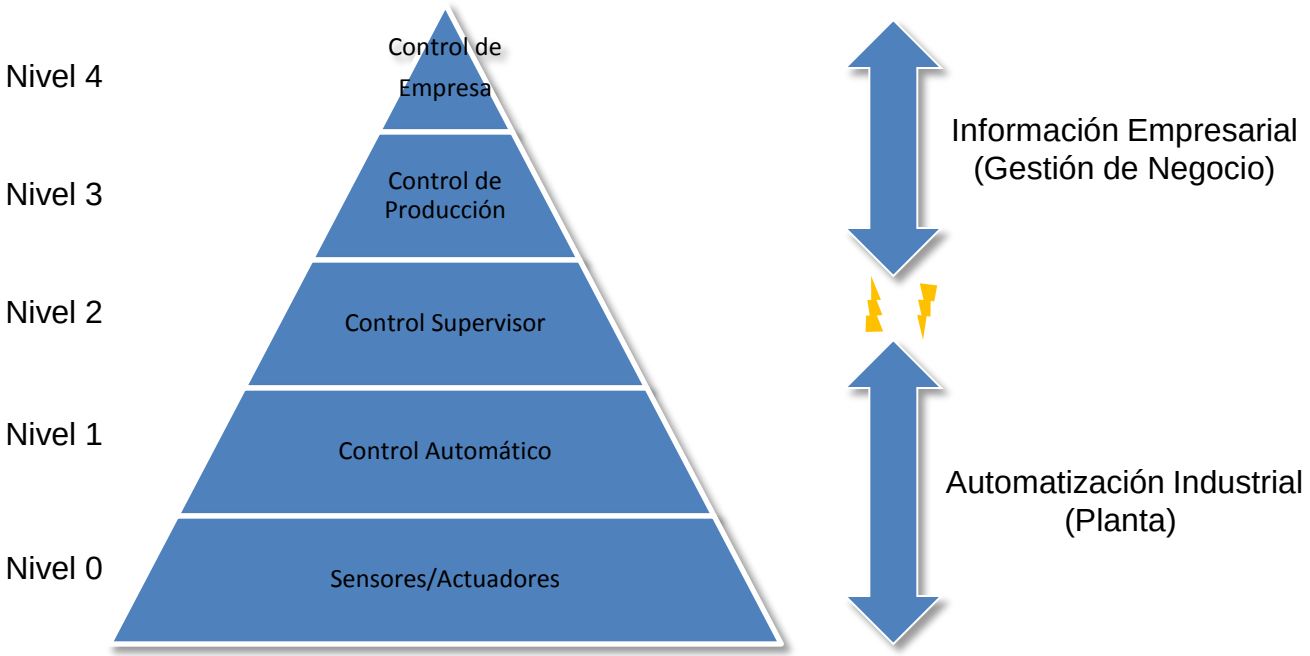
La pirámide de automatización

Los sistemas de automatización industrial son muy complejos y tienen un gran número de dispositivos con diferentes tecnologías que trabajan en sincronización. Los sistemas de automatización industrial están **organizados jerárquicamente** como muestra la **pirámide de la automatización**.

La anchura de cada una de las capas representa el número de dispositivos y la rapidez de los componentes en la escala de tiempo.



Arquitectura de Control



La pirámide de automatización

Capa 0. Sensores y actuadores. Esta capa es la más cercana a los procesos y las máquinas, empleada para captar y traducir las señales de los sensores y aplicar mediante actuadores las señales de control a los procesos.

Capa 1. Control automático. Esta capa consiste en los sistemas de control y monitorización, que gestionan los actuadores empleando la información del proceso dada por los sensores.

Capa 2. Control supervisado. Esta capa lleva el sistema de control automático estableciendo el objetivo al controlador.

Capa 3. Control de producción. Controla la producción, reserva de recursos, tareas en maquinaria, gestion del mantenimiento, etc

Capa 4. Control de empresa. Este nivel trata menos del aspecto técnico y está mas enfocado al aspecto comercial tal como abastecimiento, demanda, flujo de caja, marketing.

Capa 2. Control supervisado

El control supervisado se lleva a cabo a un nivel jerárquico superior sobre los controladores automáticos que controlan subsistemas más pequeños.

En sentido estricto, un control supervisado significa que uno o más operadores humanos están intermitentemente programando y continuamente recibiendo información de un computador encargado de cerrar un lazo autónomo de control mediante actuadores en proceso que se controla.

Un uso más específico del término es para un sistema SCADA (Supervisory Control and Data Acquisition), el cual se refiere a una clase específica de sistemas que pueden ser instalados generalmente para su empleo en plantas remotas tales como una red de tuberías, distribución de agua o una estación depuradora de aguas.

Capa 3. Control de la producción. MES

El control de la producción se lleva a cabo en un nivel jerárquico superior sobre el control supervisado.
Las funciones típicas que lleva a cabo son:

- **Planificación de los procesos**: dependiendo de la secuencia de operaciones que deben ser llevadas a cabo sobre los lotes de productos existentes realizar una planificación óptima de los recursos disponibles .
- **Gestión del mantenimiento**: detección e intervención para la operaciones de mantenimiento.
- **Gestión del inventario**: monitorización del estado del inventario de materias primas, bienes acabados, etc. y las operaciones relacionadas con su gestión.
- **Gestión de calidad**: Evaluación, documentación y gestión de la calidad.

Nivel 4. Control de Empresa. ERP

Los sistemas ERP *Enterprise Resource Planning* integran y sincronizan todas las operaciones de la compañía incluyendo: recursos humanos, finanzas, fabricación y distribución, así como también debe permitir conectar a la empresa con sus clientes y proveedores

- Control, gestión y planificación de los recursos financieros
- Planificación de Procesos
- Aprovisionamiento de Materiales
- Manejo de Inventarios
- Interacción con Proveedores
- Proveen Servicio a Clientes
- Seguimiento de órdenes
- Manejo de Recursos Humanos
- Gestión de costos de distribución y manufactura

Control de Empresa. ERP

Los ERP ofrecen una interfaz con el usuario para ejecutar las transacciones de la empresa y bases de datos centralizadas para almacenar toda la información. Los ERP integran los diferentes sistemas de información en todas las áreas de las empresas.

Los ERP hacen fluir la información a lo largo y ancho de la empresa proporcionando una sola versión de la información.

- Facilita el intercambio de datos entre las divisiones de la empresa
- Provee prácticas de negocio probadas en un solo sistema de información.
- Cada módulo del sistema trabaja individualmente, pero al mismo tiempo en coordinación e integración con el resto de módulos que conforman el ERP completo.

Control de Empresa. ERP

Beneficios

- Disminuye los costos de producción y de gestión de inventario
- Mejora el servicio a los clientes
- Planea y pronostica la demanda de producto
- Permite anticipar los costos.
- Asegura que los materiales requeridos estén disponibles cuando se necesitan.
- Mejorar el flujo de procesos
- Reducir Inventarios
- Datos ordenados, mejor análisis
- Información al cliente adecuada y validada

El computador y el ciclo de proceso de un producto
http://tv.uvigo.es/uploads/material/Video/1366/ISAD_Tema2.pdf

Enterprise Resource Planning (ERP)
http://www.acis.org.co/fileadmin/Conferencias/ConfMiguelBecerraSep6.pdf

TEMA 6
Sistemas SCADA y DCS

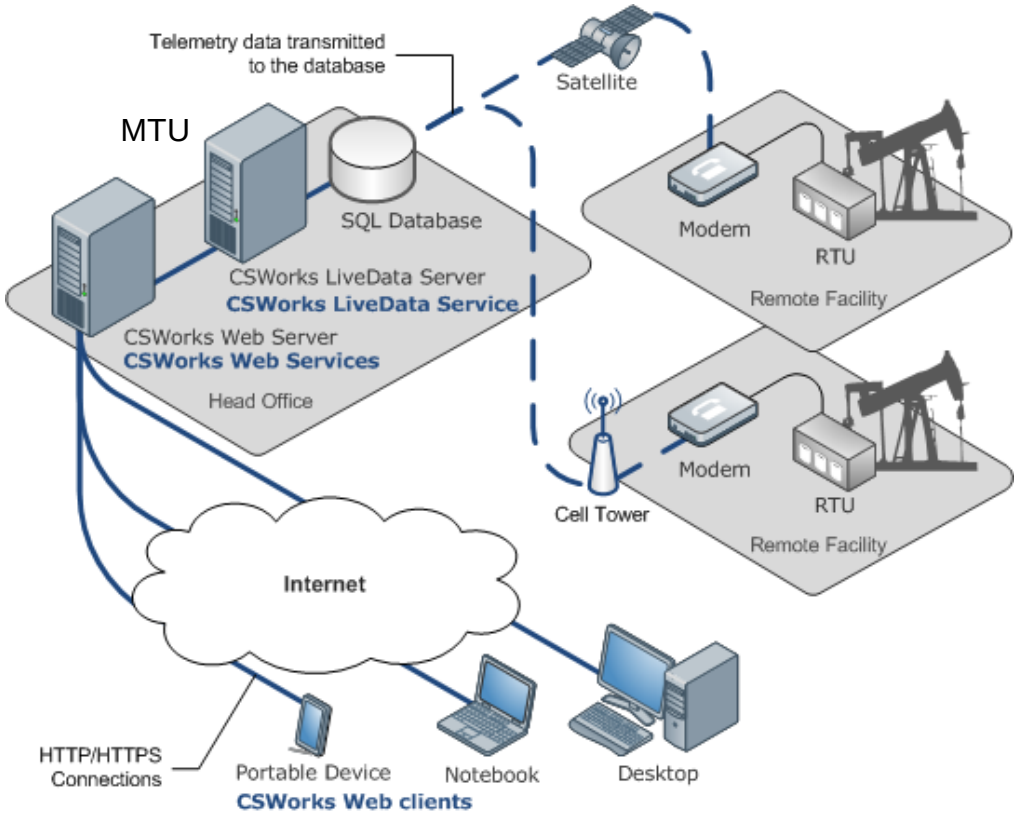
ESCUELA DE INGENIERÍAS INDUSTRIALES

Departamento de Ingeniería de Sistemas y Automática. EII.

Universidad de Valladolid

Introducción

- Tradicionalmente se define a un SCADA como un sistema que permite supervisar una planta o proceso por medio de una estación central (MTU unidad terminal maestra) y una o varias unidades remotas (RTUs) por medio de las cuales se hace el control y adquisición de datos.



Introducción

SCADA proviene de las siglas de **S**upervisory **C**ontrol **A**nd **D**ata **A**cquisition (Adquisición de datos y Control supervisado).

- Adquisición de datos
  - Acceso y adquisición de información o datos de los dispositivos de campo
  - Se envían a estación central a través de comunicaciones (telemetría)
  - Analógico / Digital

- Control supervisado
  - Proporciona información del proceso a diversos usuarios: operadores, supervisores de control de calidad, ingenieros mantenimiento, etc. para monitorización. Control Remoto/Local

Introducción

- El término **supervisado** significa que un operador humano es el que al final tiene la última decisión sobre el control del sistema o una planta industrial.

Introducción

- Los sistemas SCADA originalmente se destinaron a procesos o complejos industriales distribuidos sobre áreas geográficas muy extensas: gaseoductos, oleoductos, redes de distribución eléctrica, transporte público, tratamiento de aguas residuales, etc
- Actualmente, los sistemas SCADA no solo se emplean en procesos industriales ampliamente distribuidos geográficamente sino que también se implementan en procesos industriales ubicados en una misma planta.
- Aprovechando las redes de comunicación, industriales los dispositivos de campo y control se conectan una computadora central que efectúa tareas de supervisión, así como tratamiento de datos y generación de tendencias de un proceso.

Introducción

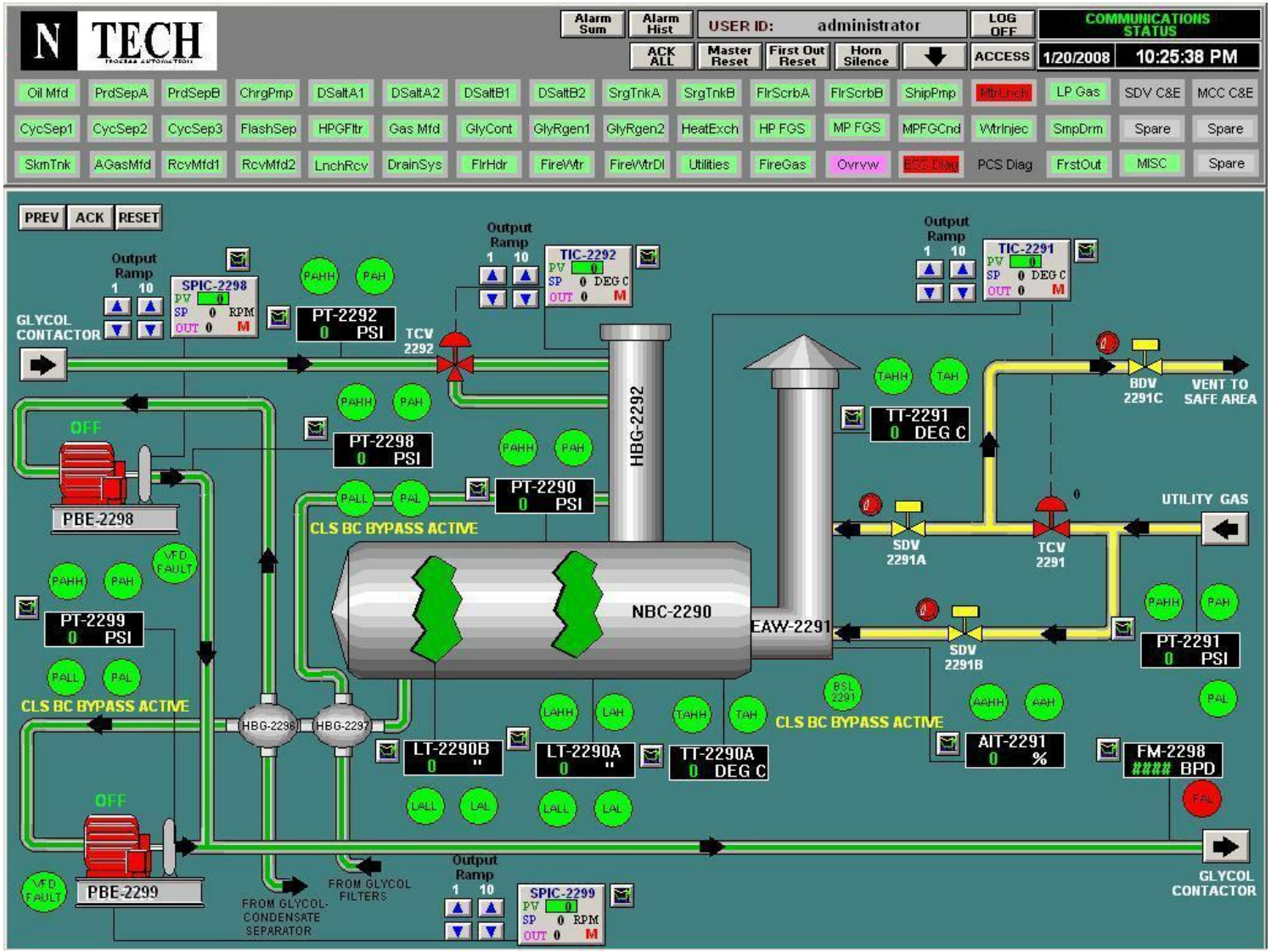
- Es importante señalar que el control directo lo realizan los controladores autónomos digitales y/o autómatas programables. Estos están conectados a la estación central que realiza las funciones de diálogo con el operador, tratamiento de la información y control de la producción.

Introducción

- Para implantar un sistema SCADA generalmente se recurre a un paquete de software especializado.
- Este software permite desarrollar los sinópticos que actúan como una interfaz gráfica entre el hombre y la máquina o el proceso.
- De esta forma es posible supervisar o cambiar puntos de consigna o reconfigurar dispositivos en el proceso supervisado por medio de acciones realizadas por el operador en la computadora.

Introducción

Introducción



Funciones principales

- **Adquisición de datos**, para recoger, procesar y almacenar la información recibida.
- **Supervisión**, para observar desde un monitor la evolución de las variables de control.
- **Control**, para modificar la evolución del proceso, actuando bien sobre los reguladores autónomos básicos (consignas, alarmas, menús, etc.) bien directamente sobre el proceso mediante las salidas conectadas.

Funciones más específicas

- **Transmisión.** De información con dispositivos de campo y otros PC.
- **Base de datos.** Gestión de datos con bajos tiempos de acceso.
- **Presentación.** Representación gráfica de los datos. Interfaz del Operador o HMI (Human Machine Interface).
- **Explotación.** De los datos adquiridos para gestión de la calidad, control estadístico, gestión de la producción y gestión administrativa y financiera.

Elementos de un sistema SCADA

- Sensores and actuadores
- RTUs/PLCs (remote terminal units)
- Comunicación
- MTU (master terminal unit)
  - Front End Processor
  - SCADA server
  - Historical/Redundant/Safety Server
  - HMI computer
  - HMI software

Sensores

Tipos de sensores

- Sensores de presión
- Sensores de temperatura
- Sensores de luz
- Sensores de humedad
- Sensores de velocidad de viento
- Sensores de nivel de líquido
- Sensores de distancia

Actuadores

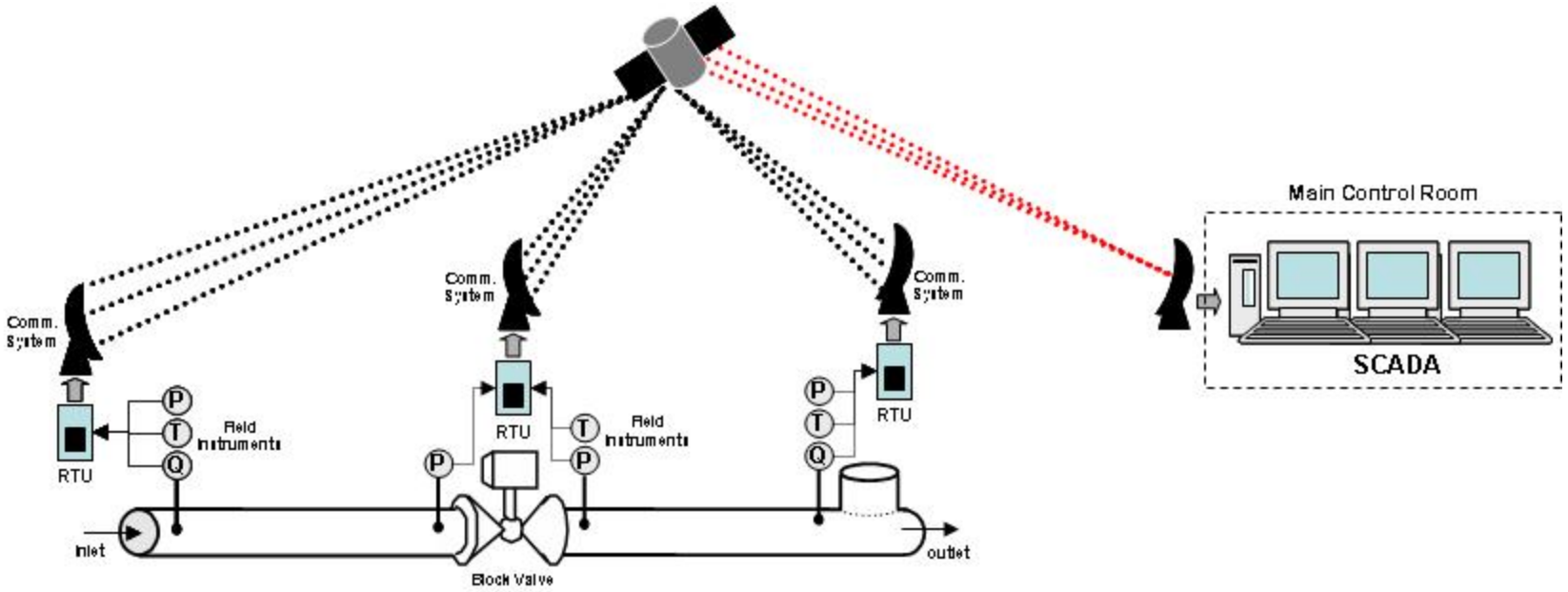
Actuadores:

- Válvulas
- Bombas
- Motores

RTUs *(Remote Terminal Unit)*

- Los sensores y relés de control no pueden generar o intepretar el protocolo de comunicación . Se necesita una RTU para proporcionar una interfaz entre los dispositivos de campo y la red del SCADA.
- Una RTU es un equipo instalado en una localidad remota que codifica los datos proporcionados por el sensor con el formato del protocolo de comunicación y lo envía a la estación central del SCADA (Master Terminal Unit, MTU)
- La RTU también recibe órdenes de control con el formato del protocolo de comunicación desde la estación central y transmite las señales a los relés de control correspondientes.
- Las RTUs pueden comunicarse con la estación central a través de líneas telefónicas, fibra óptica o transmisiones de radio o microondas.

RTUs


Main Control Room
SCADA
Comm. System
Comm. System
Comm. System
RTU
RTU
RTU
Field Instruments
Field Instruments
P
T
Q
Inlet
outlet
Block Valve

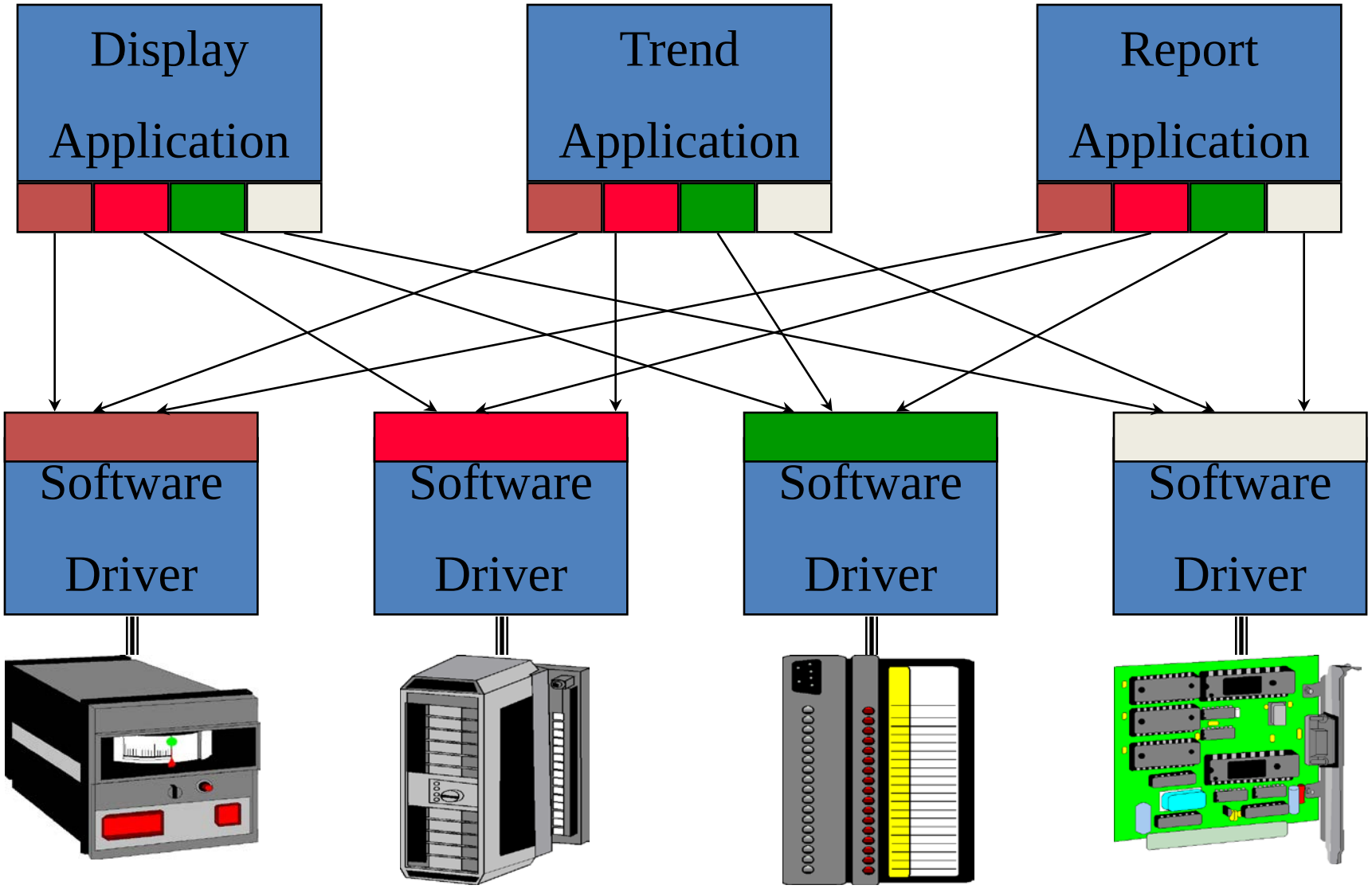
Sistemas de comunicación

- Red telefónica conmutada
- Líneas alquiladas
- Red privada(LAN/RS-485)
- Internet
- Comunicación inalámbrica
  - Wireless LAN
  - Red GSM. *Global System for Mobile Communication*
  - Radio modems

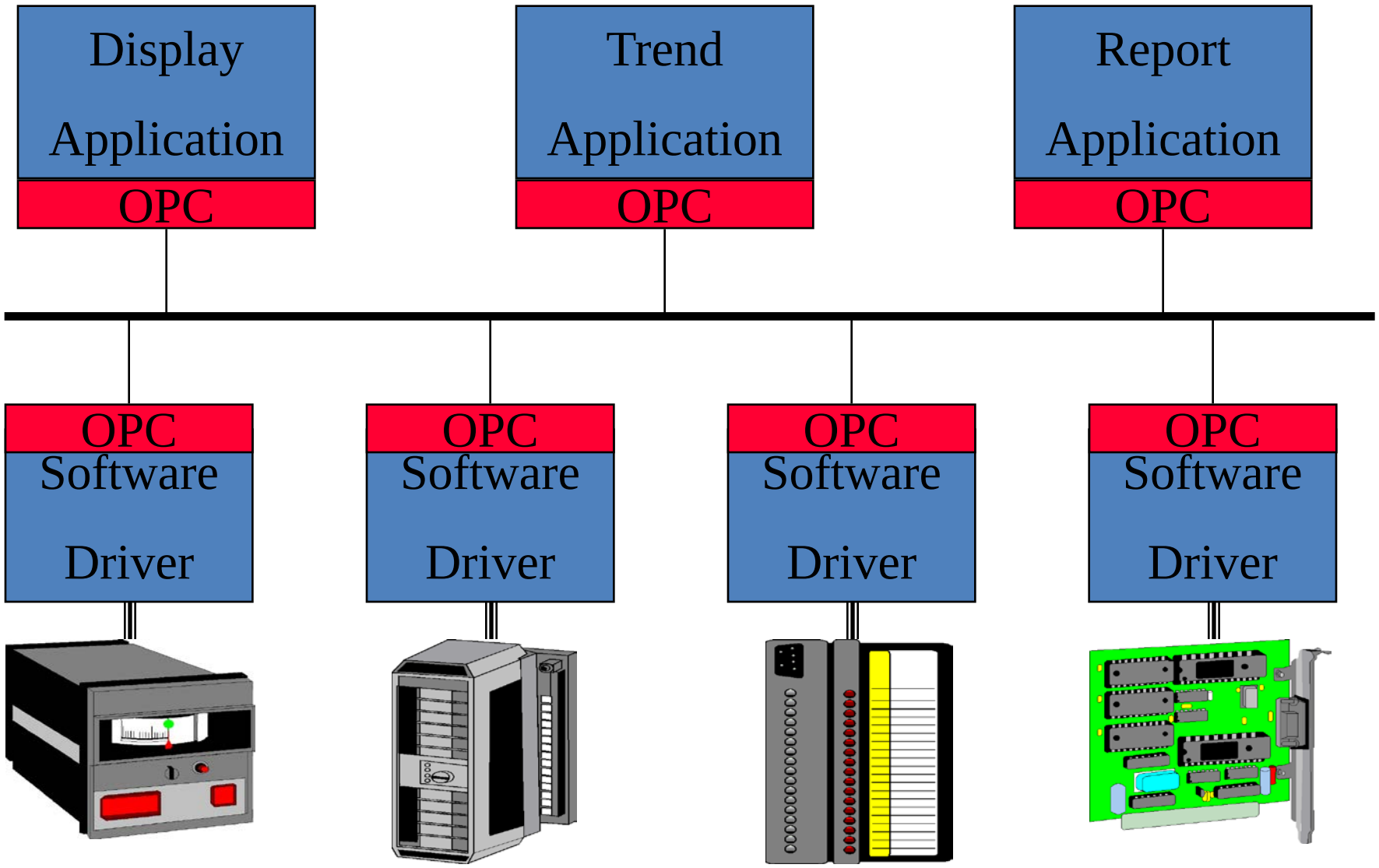
Interfase de comunicación

- Uno de los problemas mas difíciles de solucionar en el campo industrial es el de la integración de los sistemas. multitud de sistemas de control y monitorizacion, cada uno con sus propias ideas de comunicación, deben ponerse de acuerdo y trabajar en armonía para permitir la máxima eficiencia y proporcionar un acceso seguro a la información.
- **Drivers Específicos.** Utilizar el driver específico al bus de campo.
- **Drivers OPC.** Utilizar un driver genérico OPC que cada fabricante proporciona.

Interfase de comunicación



Interfase OPC



Interfase OPC

- El **OPC** (OLE (*Object Linking and Embedding*) for Process Control) es un estándar de comunicación en el campo del control y supervisión de procesos industriales, basado en una tecnología Microsoft, que ofrece **una interfaz común para comunicación** que permite que componentes software individuales interaccionen y compartan datos.
- La comunicación OPC se realiza a través de una arquitectura Cliente-servidor. El servidor OPC es la fuente de datos y cualquier aplicación basada en OPC puede acceder a dicho servidor para leer/escribir cualquier variable que ofrezca el servidor.
- Es una solución abierta y flexible al clásico problema de los drivers propietarios. Prácticamente todos los fabricantes de sistemas de control, instrumentación y de procesos han incluido OPC en sus productos.

Interfase OPC

- La idea básica del OPC está en normalizar el interfase entre el servidor OPC y el cliente OPC independientemente de cualquier fabricante particular.
- Los servicios prestados por los servidores OPC para clientes OPC por medio del interfase OPC típicamente implican la lectura, cambio y verificación de variables de proceso.

MTU

- Una vez que los datos llegan a la estación central, se los almacena para su análisis, generación de históricos y toma de decisiones. Por medio de una HMI se muestra la información al operador del sistema, para la supervisión.
- Basado en la información, el operador puede tomar decisiones que pueden modificar el trabajo del proceso supervisado. El operador a través de la interfaz gráfica puede operar sobre los actuadores del sistema.

MTU (Master Terminal Unit)

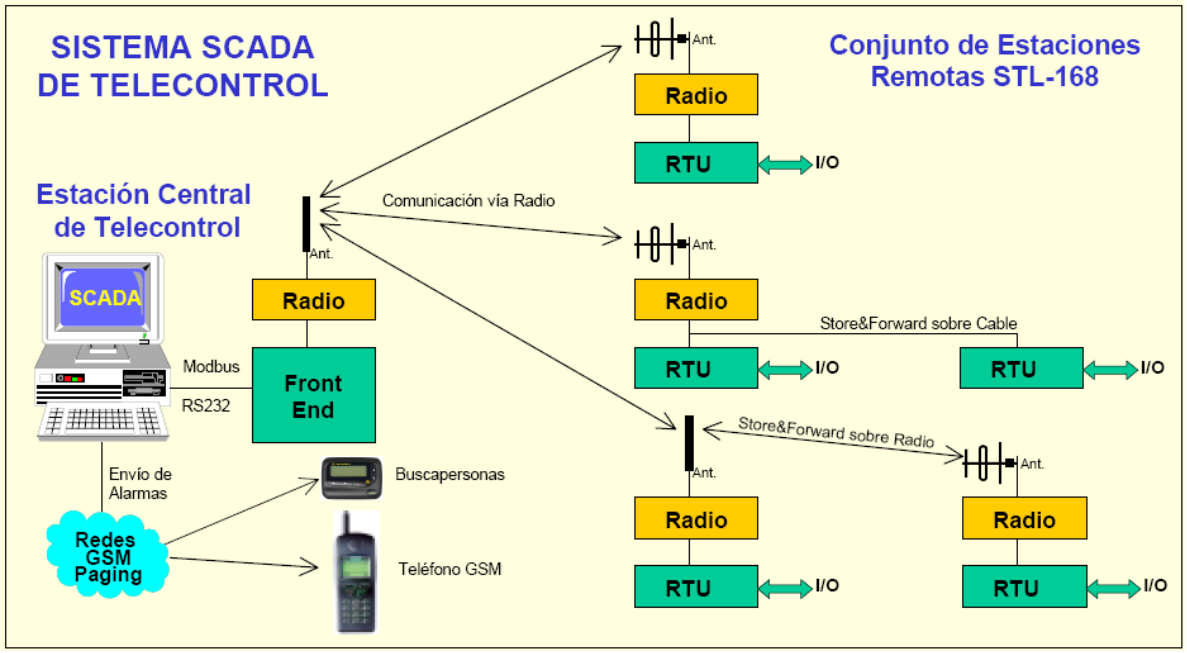
- La MTU visualiza la información que recoge de los dispositivos de campo de una forma que sea significativa para los operadores humanos (sinópticos, gráficos de tendencias, visualización de alarmas. ..)

- Características:

  - **Posibilidad de crear paneles de alarma**, que exigen la presencia del operador para reconocer una parada o situación de alarma, con registro de incidencias. Generación de informes con eventos y fechas/hora.
  - **Generación de históricos de señal de planta**, que pueden ser volcados para su proceso sobre una hoja de cálculo.
  - Capacidad para **mandar mensajes** directamente a los técnicos de mantenimiento proporcionando información detallada en lenguaje natural (castellano o inglés) con una descripción completa de qué ha ocurrido y cómo gestinarlo.
  - Soportar **sistemas de backup** en diferentes localizaciones. Si el sistema SCADA maestro falla, una segunda estación en red debe tomar el mando sin interrupción de las funciones de supervisión y control.
  - Soportar varios **protocolos abiertos** para salvaguardar el sistema SCADA frente a la obsolescencia

Front End Processor

- Interfaz de comunicación entre varios canales RTU y el ordenador maestro central



SCADA Server

- Acceso a datos
- Análisis de datos
- Sirve a los clientes a través de firewall
- Clientes en la empresa o fuera conectados a través de internet
- Pide información a las RTU

Historical/Safety/Redundant Server

- Accede a los datos en el servidor SCADA y los guarda como un backup en caso de un desastre.
- Básicamente es un servidor para seguridad

HMI (*Human Machine Interface*)

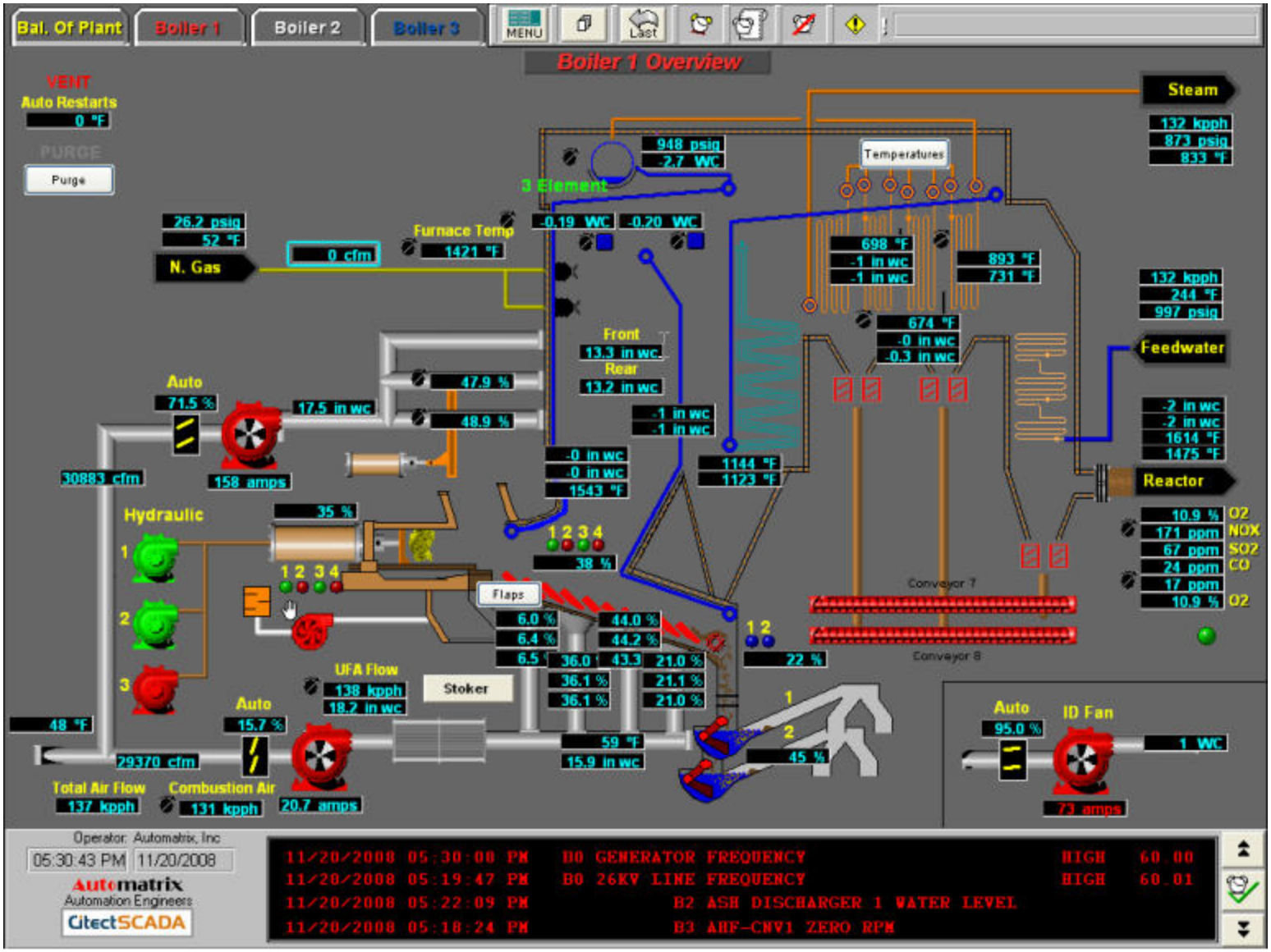
Interfaz Hombre Máquina

- Accede al servidor SCADA
- Controla el sistema
- Interfaz operador
- Software
  - Amigable
  - Programable

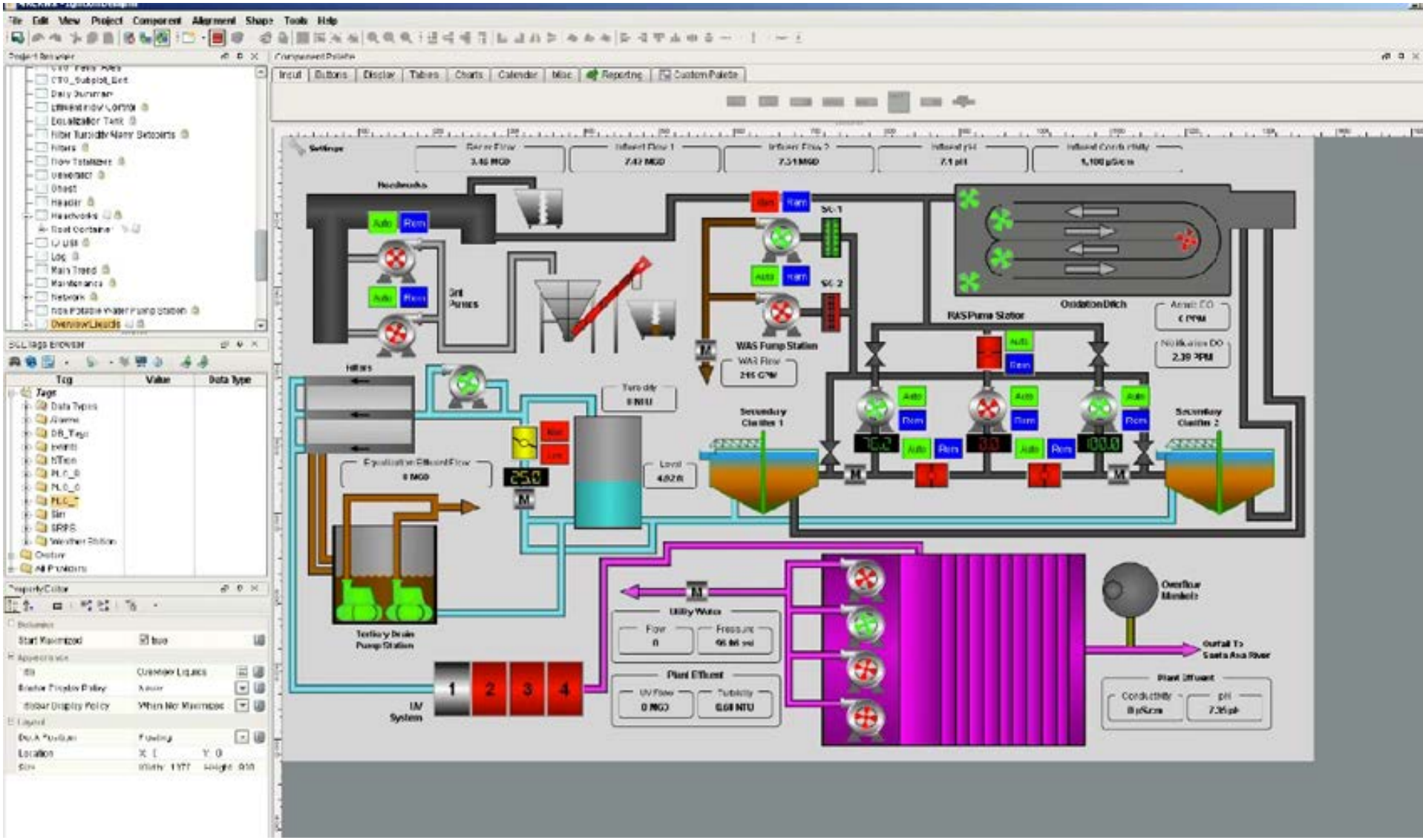
HMI

- Los sistemas de interfaz entre usuario y planta basados en paneles de control repletos de indicadores luminosos, instrumentos de medida y pulsadores, se sustituyen por sinópticos sobre la pantalla de un ordenador.

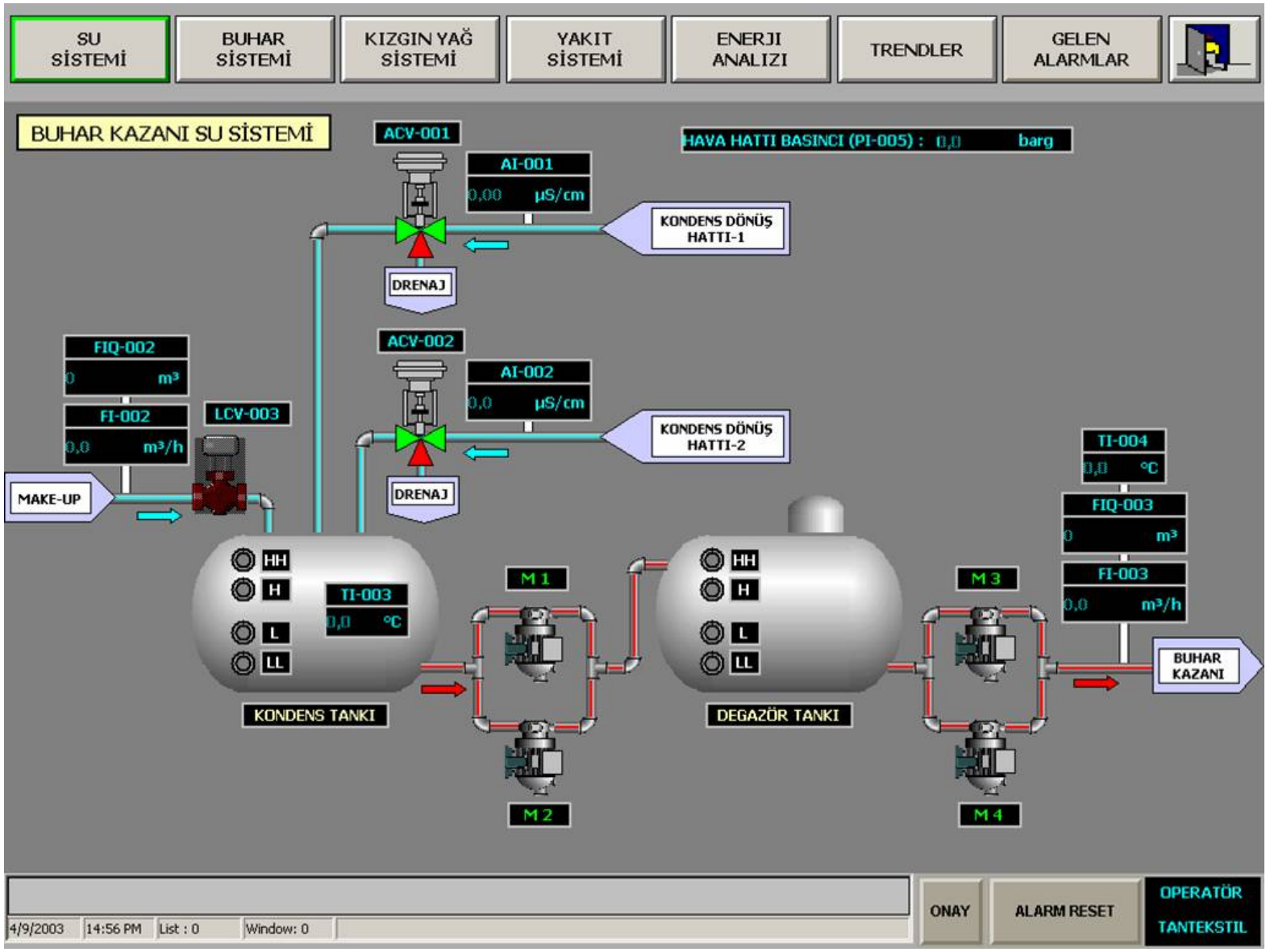
Ejemplo interfaz operario



Ejemplo interfaz operario



Ejemplo interfaz operario



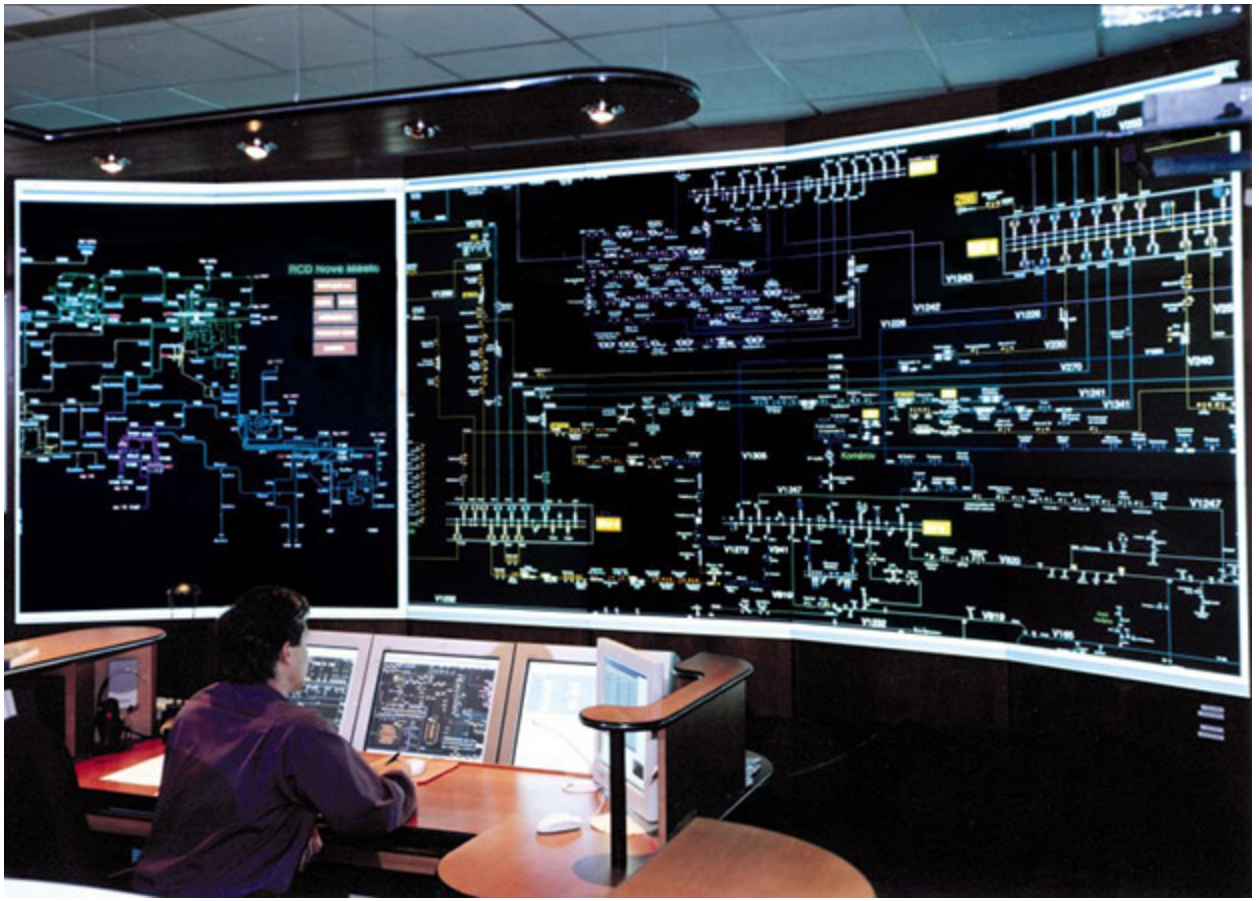
Ejemplo interfaz operario

Gráficos de tendencias

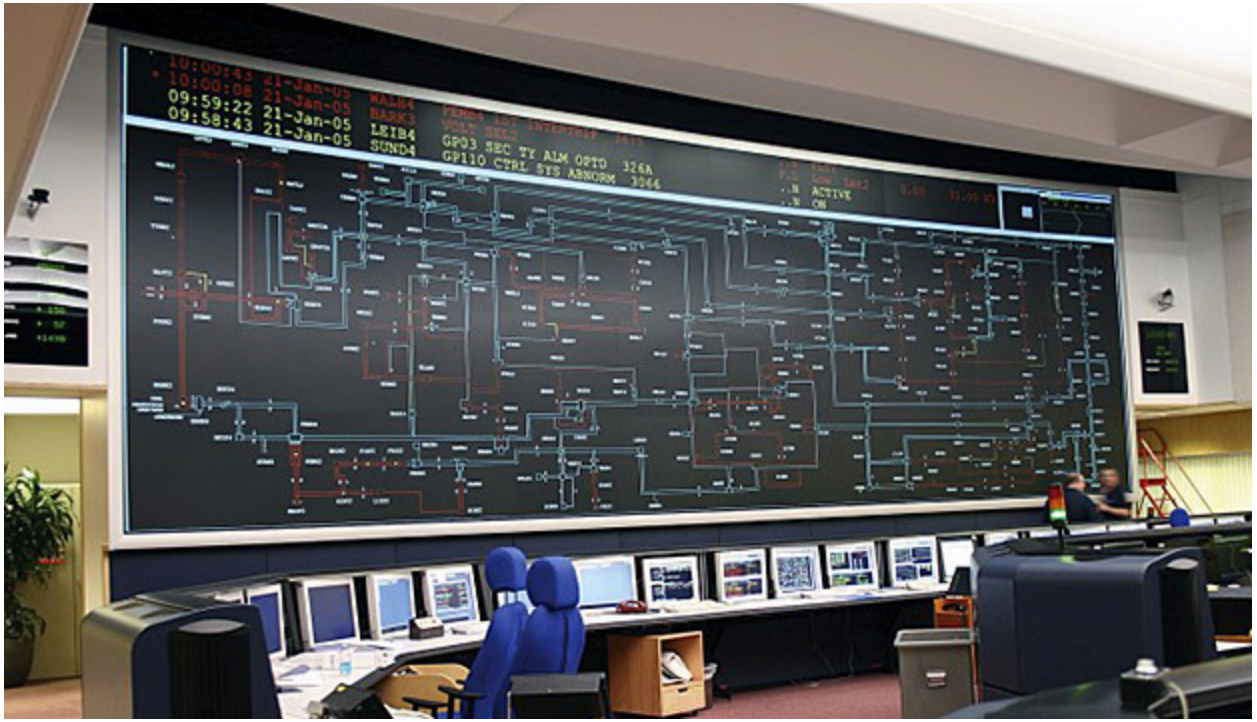
HMI

Los dispositivos HMI generalmente son computadores personales localizados en una habitación de control o oficinas de ingeniería. Las instalaciones sencillas pueden operar con una única estación de trabajo. Las instalaciones más complejas precisan de más estaciones de trabajo.

HMI



HMI


09:59:22 21-Jan-05 LEIB4
09:58:43 21-Jan-05 SUND4
GP03 SEC TY ALM OPTO 326A
GP110 CTRL SYS ABNORM 3066
ACTIVE
ON

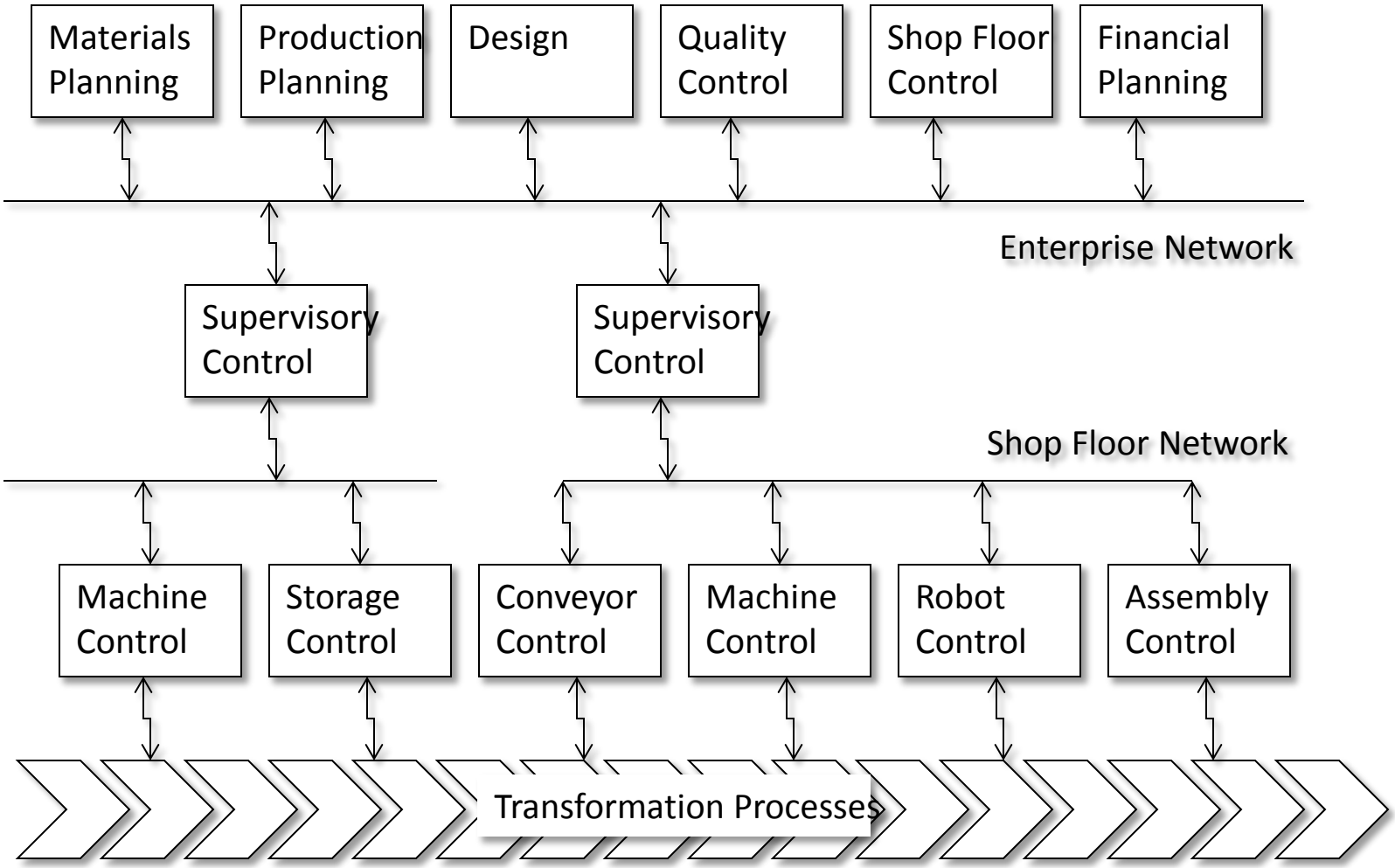
Tendencias

- Deben ser sistemas de arquitectura abierta, capaces de crecer o adaptarse según las necesidades cambiantes de la empresa.
- Deben comunicarse con facilidad y de forma transparente al usuario con el equipo de planta y con el resto de la empresa (redes locales y de gestión).
- Deben ser programas sin excesivas exigencias de hardware, y fáciles de utilizar, con interfaces amigables con el usuario.

Sistema de Control Distribuido

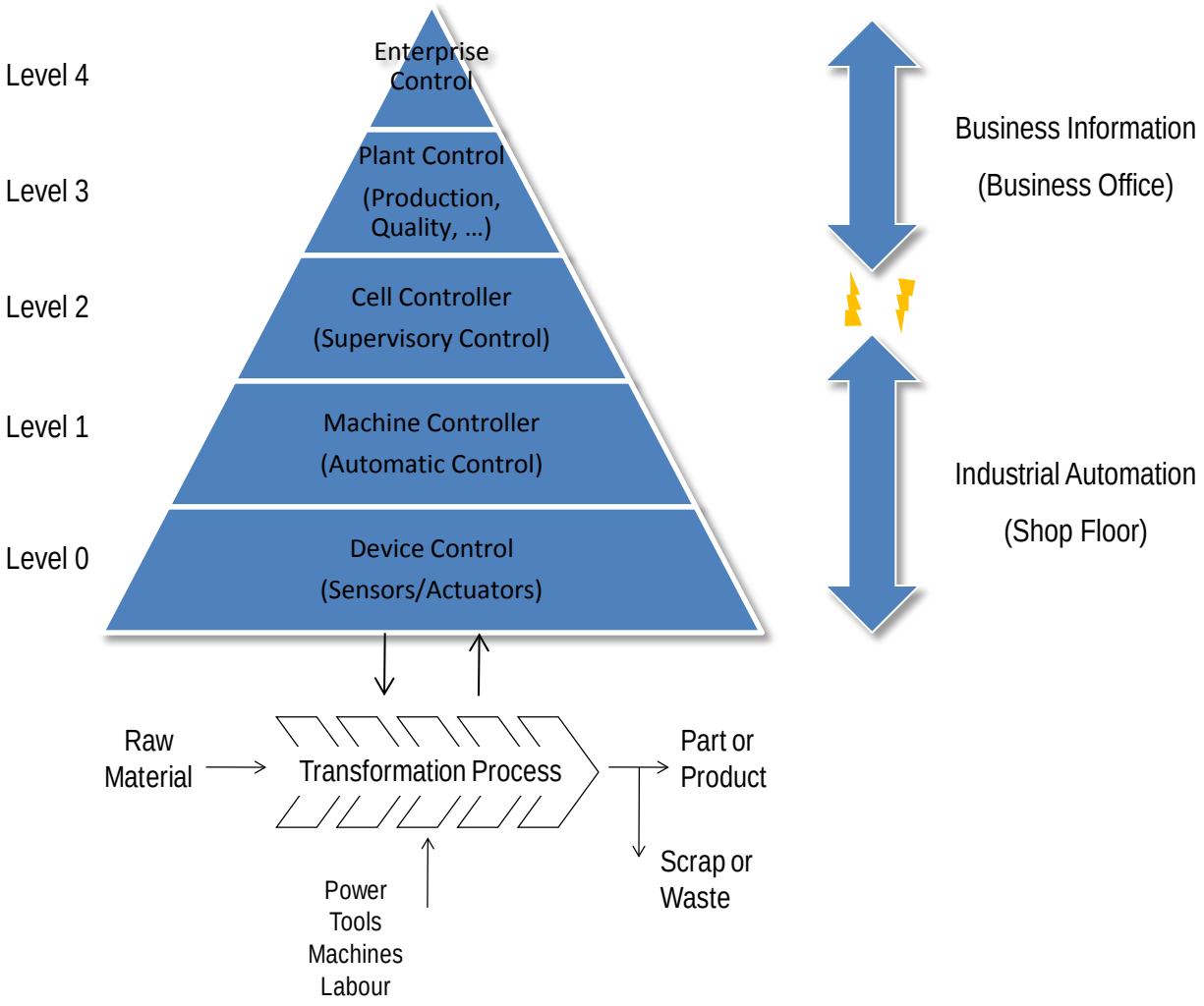
- Un sistema de control distribuido *(DCS distributed control system)* se refiere a un sistema de control en el cual los **controladores** no están en una localización central sino que están **distribuidos por toda la planta** estando cada subsistema gestionado por un controlador. El sistema de entero de controladores está conectado por **redes de comunicación**.

Sistema de Control Distribuido



Sistema de Control Distribuido

- Es típico que un operador humano trate de comunicarse con el DCS de alguna forma sobre todo para cambiar su programación o configuración. Esta tarea se puede hacer mediante consolas de mano, pero ahora hay la opción de hacer estas tareas desde el SCADA


Level 4
Level 3
Level 2
Level 1
Level 0
Enterprise Control
Plant Control (Production, Quality, ...)
Cell Controller (Supervisory Control)
Machine Controller (Automatic Control)
Device Control (Sensors/Actuators)
Business Information (Business Office)
Industrial Automation (Shop Floor)
Raw Material
Transformation Process
Part or Product
Scrap or Waste
Power
Tools
Machines
Labour

Otro ejemplo SDC: automóviles

VW Phaeton:

- 11.136 electrical parts in total

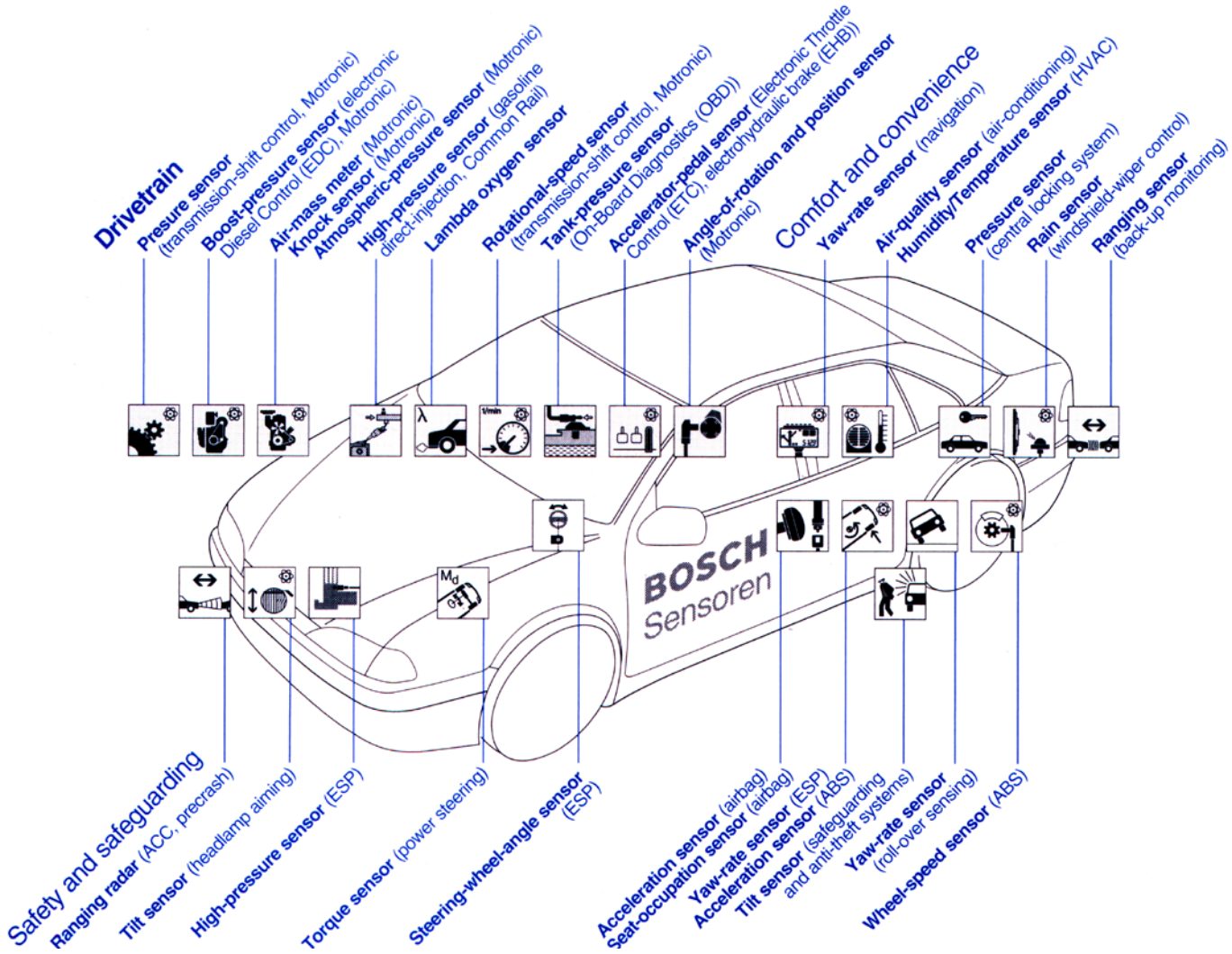
communication:

- **61 ECUs in total**
- external diagnosis for 31 ECUs via serial communication
- optical bus for high bandwidth Infotainment-data
- **sub-networks** based on proprietary serial bus
- **35 ECUs** connected by **3 CAN-busses**

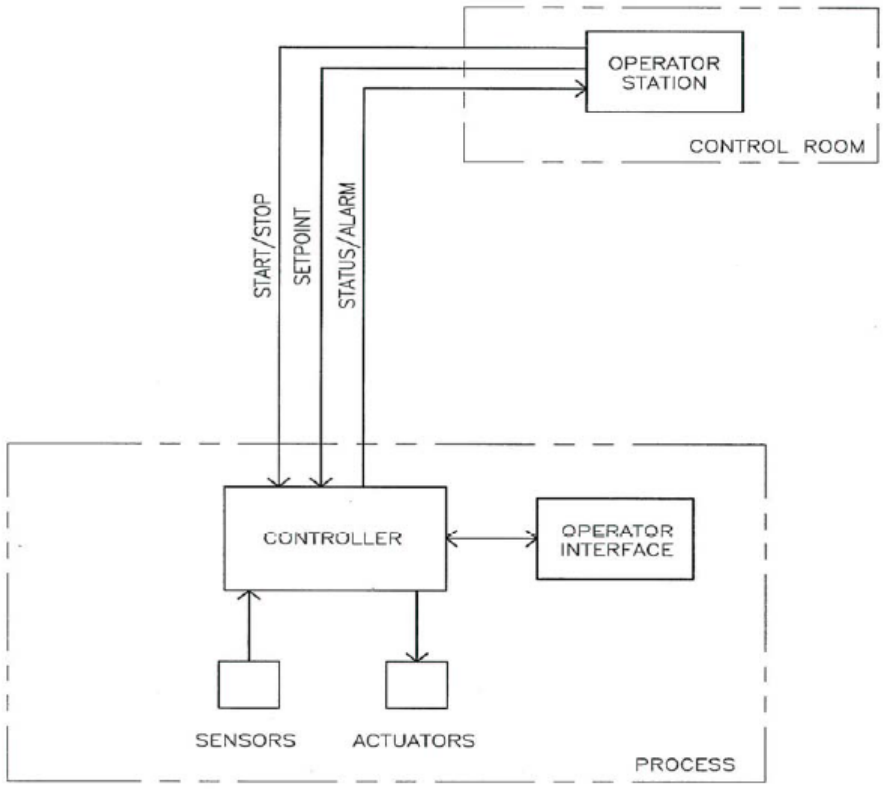
sharing

- appr. **2500 signals**
- in **250 CAN messages**

Otro ejemplo SDC: automóviles

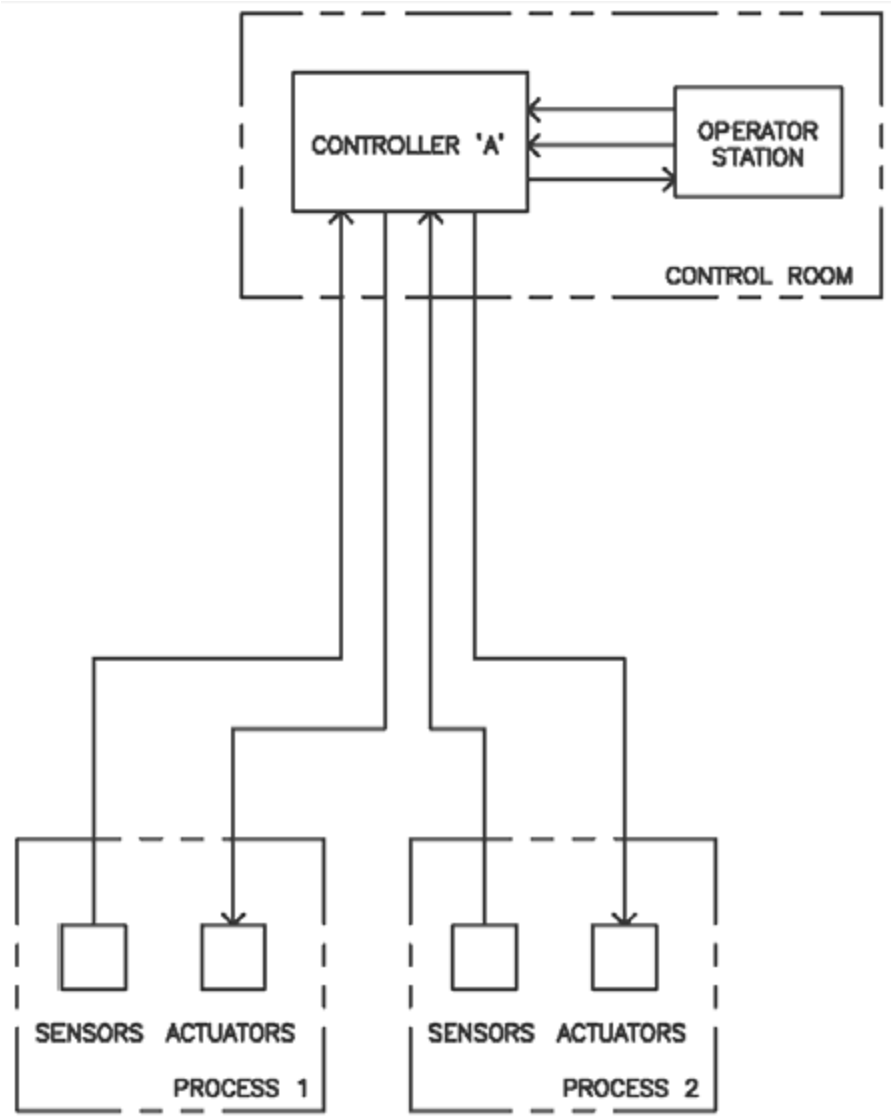


Control local



Es una arquitectura en la que los sensores, controlador y el equipo a controlar se encuentran próximos y el alcance de cada controlador se limita a un subsistema específico. Los controladores locales generalmente son capaces de aceptar comandos de un control supervisor para iniciar o terminar las secuencias localmente controladas o para ajustar los valores de referencia, pero la acción de control en sí misma se realiza en el controlador local.

Control centralizado



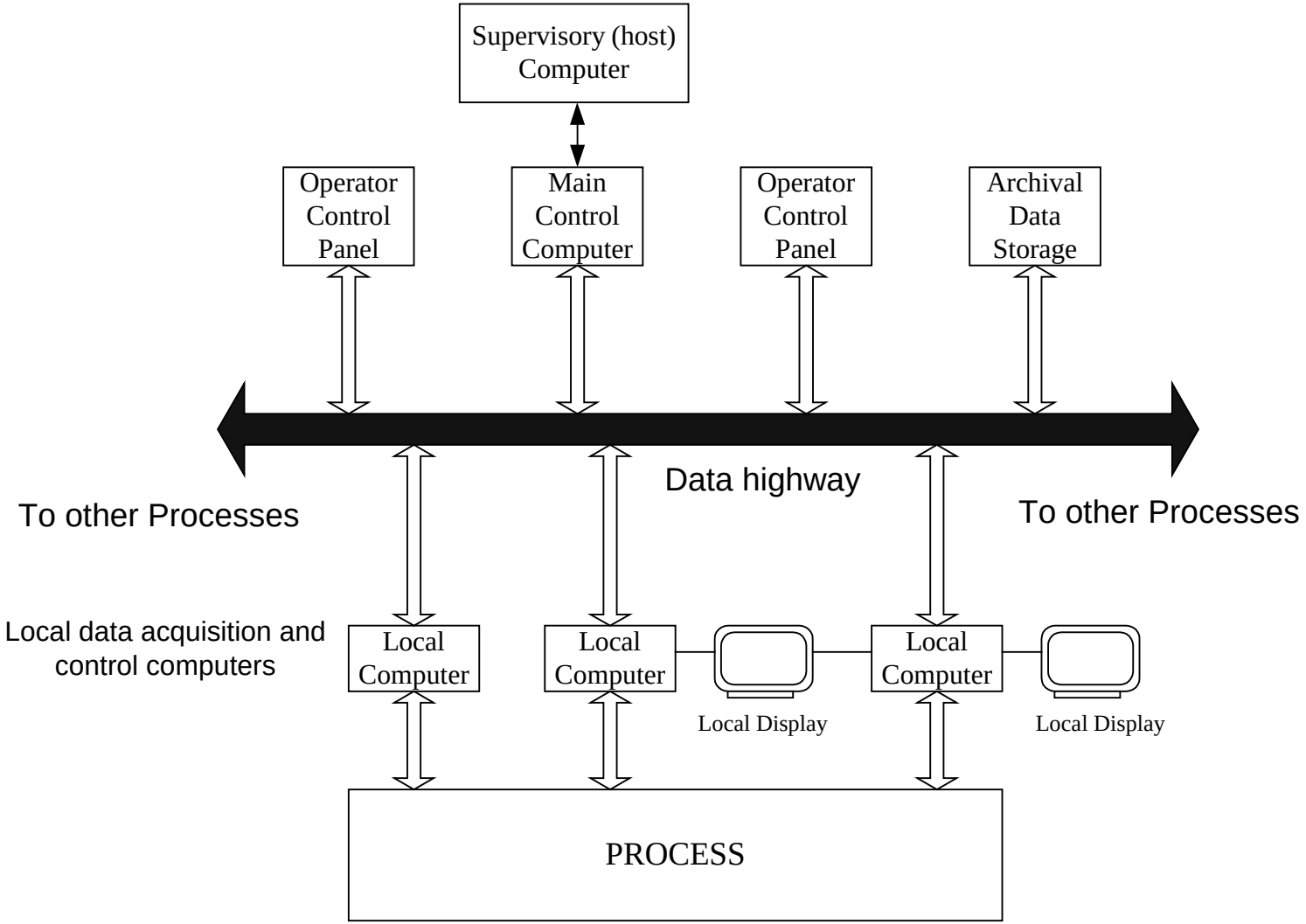
El control centralizado describe un sistema en el cual todos **los sensores y actuadores** de la planta **se conectan** a uno o varios **controladores localizados en una sala de control**. La localización de todos los controles, interfaces de operador y indicadores en una misma sala mejora el conocimiento por parte del usuario de las condiciones del sistema y la velocidad de respuesta ante contingencias. Este tipo de arquitectura era común antes pero actualmente ha sido desplazada por el control distribuido debido al alto coste para llevar todo el **cableado** a una estación central.

Control distribuido

La arquitectura de un sistema de control distribuido ofrece las **mejores características** tanto del control local como del centralizado.

En un sistema de control distribuido, los controladores están ubicados junto a los sistemas a controlar pero además están conectados a una o más estaciones centrales por medio de un circuito de comunicación digital. La acción de control para cada subsistema se lleva a cabo en el controlador local, pero en la estación central se dispone de una visión completa del estado de todos los sistemas

Control distribuido

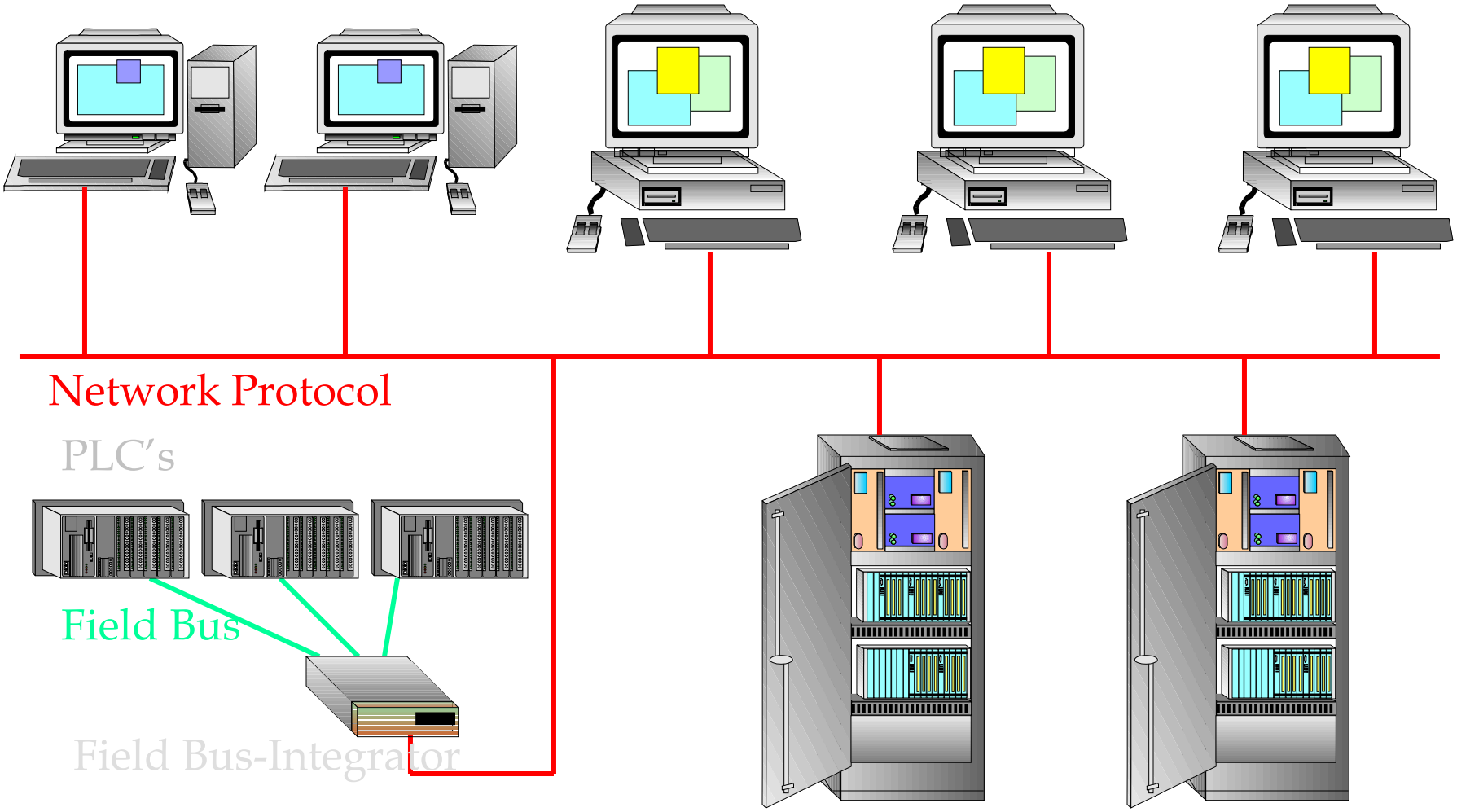


Fiabilidad DCS

La fiabilidad de la arquitectura distribuida mejora por los siguientes motivos:

- Los cableados de entrada y salida son más cortos y menos vulnerables a pertubaciones electromagnéticas.
- Un fallo por una catástrofe medioambiental en una zona de la planta no afectará a los controladores o cableado localizado en otro área.
- Cada controlador local puede funcionar por su cuenta ante la pérdida de comunicación con controlador central.

Control Distribuido



Conclusiones DCS

- Los sistema de control distribuido realizan sus funciones de control a través de una serie de módulos de control automáticos e independientes, distribuidos en una planta
- La filosofía de funcionamiento de esta arquitectura es evitar que el control de toda la planta esté centralizado en una sola unidad.
- De esta forma, si una unidad de control falla, el resto de unidades podría seguir funcionando.

TEMA 7
Sistemas instrumentados de seguridad (SIS)

ESCUELA DE INGENIERÍAS INDUSTRIALES

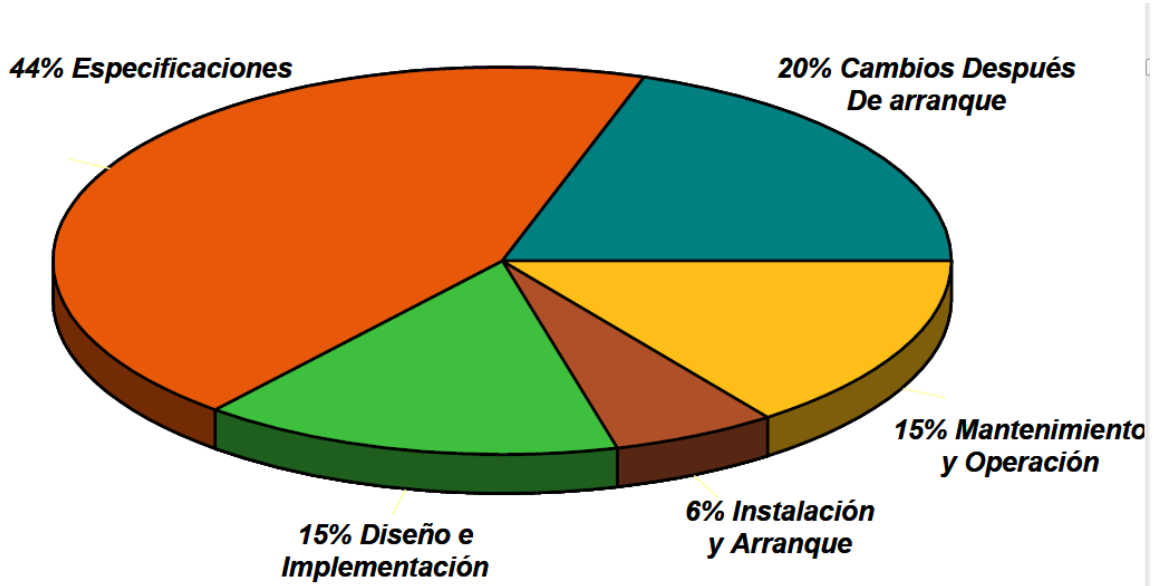
Departamento de Ingeniería de Sistemas y Automática. EII.

Universidad de Valladolid

Esta exposición se ha elaborado a partir del artículo de Luis García Pecsén
http://larevistadelgasnatural.osinerg.gob.pe/articulos_recientes/files/archivos/50.pdf

Introducción

En determinadas plantas industriales pueden presentarse situaciones de riesgo como consecuencia del almacenamiento, procesamiento y generación de sustancias peligrosas. Un suceso incontrolado en estas instalaciones podría ocasionar daños (térmicos, físicos y/o químicos) sobre las personas, bienes materiales y/o medio ambiente.



¿por qué fallan los sistemas de control?

Introducción

Existen una serie de normas, no solo en el diseño de las instalaciones y equipos, también en la adopción de medidas de seguridad.

La elaboración de un análisis de riesgos específico y aplicación de una o varias técnicas de identificación y evaluación de riesgos, derivarán en las medidas de seguridad o capas de protección a implementarse en las instalaciones industriales, entre ellas un **Sistema Instrumentado de Seguridad (SIS).**

Sistema Instrumentado de Seguridad (SIS)

SIS, Sistema Instrumentado de Seguridad (SIS, Safety Instrumented System)

Un SIS es una capa de protección de una planta o sistema industrial, el cual una vez que detecta que una variable del proceso ha alcanzado un valor peligroso predeterminado**, realizará las acciones correctivas para conducir el estado de las instalaciones a una condición segura**.

En caso que no se disponga de SIS en las instalaciones, o que se disponga de SIS pero éste no actúa o lo hace incorrectamente, entonces se generará en el proceso una situación de riesgo que finalmente puede provocar la ocurrencia de un accidente grave en las instalaciones.

Sistema Instrumentado de Seguridad (SIS)

- **SIS, Sistema Instrumentado de Seguridad (SIS, Safety Instrumented System)**
- Es un sistema compuesto por sensores, procesadores lógicos y elementos finales de control que tiene el propósito de **implementar las funciones de seguridad necesarias para llevar al proceso a un estado seguro**, cuando se han violado condiciones predeterminadas.

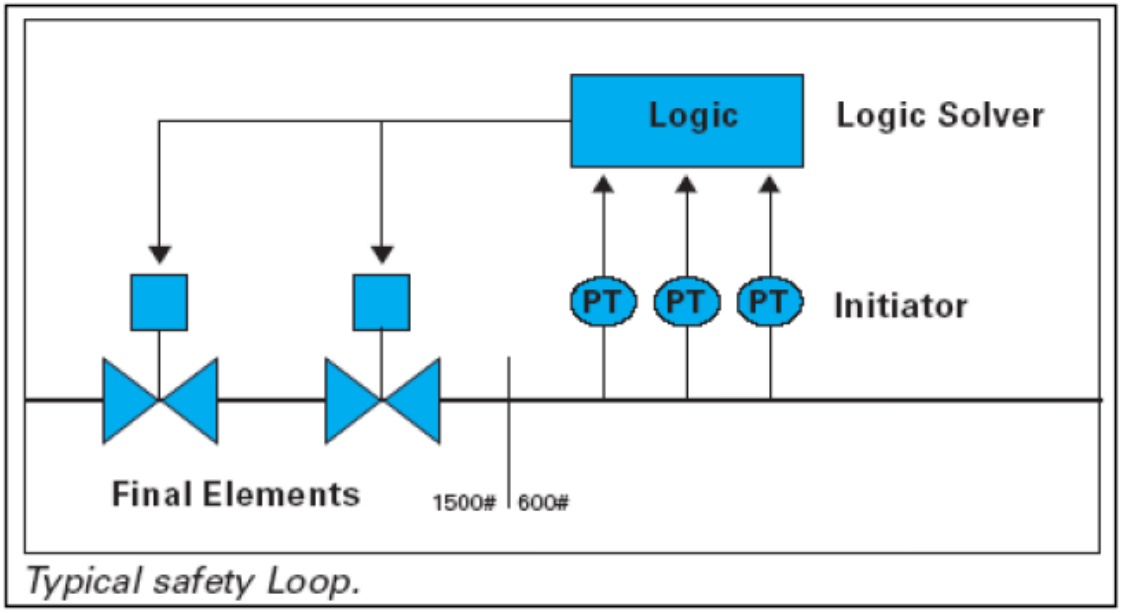
Sistema Instrumentado de Seguridad (SIS)

Un sistema instrumentado de seguridad y un sistema de control (*BPCS Basic Process Control System*) constan de los **mismos elementos**, **pero con objetivos distintos**: el sistema de control regula y controla el proceso, el sistema instrumentado brinda seguridad al proceso.

Por ejemplo en un BPCS, la señal actuará sobre la válvula para regular o controlar el proceso. En un **SIS**, la señal actuará sobre la válvula para brindar **seguridad** a las instalaciones, abriéndola o cerrándola totalmente de acuerdo al diseño del SIS.

La señal del **SIS tiene preferencia** sobre la señal del BPCS, es decir, una vez que actúa la señal del SIS la señal del BPCS queda anulada.

Ejemplo Sistema Instrumentado de Seguridad (SIS)



SIS con nivel de seguridad SIL 3, que se utiliza en un HIPPS5* instalado en una Planta de Gas Natural Licuado. Nótese la redundancia: tres transmisores de presión y dos válvulas de cierre rápido “shut off valves”.

HIPPS: High Integrity Pressure Protection System, es un sistema instrumentado de seguridad (SIS), diseñado para cumplir con el nivel de seguridad SIL 3, de acuerdo a la norma ANSI/ISA S84.00.01 y IEC 61511. El propósito de un sistema HIPPS es el de proteger equipos e instalaciones aguas abajo contra una sobrepresión de gas, cerrando las válvulas “shut off” que lo aíslan de la fuente de esa sobrepresión.

Función Instrumentada de Seguridad (SIF)

SIF, Función Instrumentada de Seguridad (SIF, Safety Instrumented Function)

- Es una capa de protección a ser implementada por un SIS con la finalidad de lograr o **mantener el proceso en un estado seguro frente a un evento peligroso específico**.
- Se define también como un **conjunto de acciones específicas y su equipo correspondiente**, necesario para identificar un peligro sencillo y actuar para llevar al proceso a un estado seguro.
- Una SIF es diferente a un **SIS**, el cual **puede abarcar múltiples funciones instrumentadas de seguridad** y actuar en múltiples formas para prevenir múltiples resultados peligrosos.

Ejemplos de SIF

- Cierre del suministro de combustible a un horno, en caso de pérdida de llama.
- Suministro de enfriamiento de emergencia para reducir una temperatura excesiva.
- Apertura de una válvula para reducir el exceso de presión.
- Dirigir un derrame hacia el sistema de manejo de residuales.
- Activación de la alarma de fuego luego de producirse un incendio.
- Activación de mensajes de emergencia pre-grabados para el equipo de respuesta ante emergencias.

Función Instrumentada de Seguridad (SIF)

- Un **SIS** puede tener **múltiples SIF**
- **Cada una** de estas SIF es un lazo de seguridad que cuenta con los mismos elementos de un BPCS, y **tiene un SIL** (Nivel de Integridad de Seguridad) que puede ser diferente.
- Resulta incorrecto y ambiguo definir un único SIL para todo un SIS.

Nivel de Integridad de Seguridad (SIL)

- **SIL, Nivel de Integridad de Seguridad (SIL, Safety Integrity Level).**
- Es un nivel discreto para la especificación de los requerimientos de integridad de las funciones de seguridad a ser asignadas a los sistemas instrumentados de seguridad.
- Cada nivel discreto se refiere a la probabilidad de que un sistema referido a la seguridad realice satisfactoriamente las funciones de seguridad requeridas bajo todas las condiciones establecidas en un periodo de tiempo dado.

Nivel de Integridad de Seguridad (SIL)

- SIL
  - NO es una MEDIDA DEL RIESGO
  - Si no mas bien la FIABILIDAD esperada o prevista de un sistema o función

Nivel de Integridad de Seguridad (SIL)

Safety Integrity Level	Probability of failure on demand, average (Low demand mode of operation)	Risk Reduction Factor
SIL 4	$>=10^{-5}$ to $<10^{-4}$	100000 to 10000
SIL 3	$>=10^{-4}$ to $<10^{-3}$	10000 to 1000
SIL 2	$>=10^{-3}$ to $<10^{-2}$	1000 to 100
SIL 1	$>=10^{-2}$ to $<10^{-1}$	100 to 10

Nivel de Integridad de Seguridad (SIL)

SIL	PFD	Fallo máximo aceptado del SIS
SIL 4	$>=10^{-5}$ to $<10^{-4}$	Un fallo peligroso en 10000 años
SIL 3	$>=10^{-4}$ to $<10^{-3}$	Un fallo peligroso en 1000 años
SIL 2	$>=10^{-3}$ to $<10^{-2}$	Un fallo peligroso en 100 años
SIL 1	$>=10^{-2}$ to $<10^{-1}$	Un fallo peligroso en 10 años

SIL 4 es utilizado en plantas nucleares

Nivel de Integridad de Seguridad (SIL)

"SIL 4"	Protection of environment & community Nuclear	PFD = 10^{-4} PFH = 10^{-8}
"SIL 3"	Human protection	PFD = 10^{-3} PFH = 10^{-7}
"SIL 2"	Protection of ownership and manufacturing. Human protection	PFD = 10^{-2} PFH = 10^{-6}
"SIL 1"	Protection of plants	PFD = 10^{-1} PFH = 10^{-5}

Nivel de Integridad de Seguridad (SIL)

Relación de índices SIL con probabilidad de fallo en demanda			
SIL	Consecuencias	Disponibilidad requerida (%)	PFD media[2]
4'	Daños catastróficos en el exterior	> 99,99	10^{-5} -10^{-4}
3	Daños humanos en el interior y daños materiales en el exterior	99,90-99,99	10^{-4}-10^{-3}
2	Daños materiales y posibles daños humanos en el interior	99,00-99,90	10^{-3}-10^{-2}
1	Pequeños daños materiales en el interior	90,00 - 99,00	10^{-2} -10^{-1}

SIL 4 es utilizado en plantas nucleares

PFD, Probabilidad de Falla en Demanda

- **PFD, Probabilidad de Falla en Demanda (PFD, Probability of Failure on Demand)**
- Es un valor que indica la probabilidad de que un SIS falle al responder a una demanda, ésta es una condición o evento que requiere que el SIS lleve a cabo una acción apropiada para prevenir un evento peligroso. Se define también como la inversa del Factor de Reducción del Riesgo

Ciclo de Vida de Seguridad (SLC)

SLC, Ciclo de vida de seguridad (SLC, Safety Life Cycle)

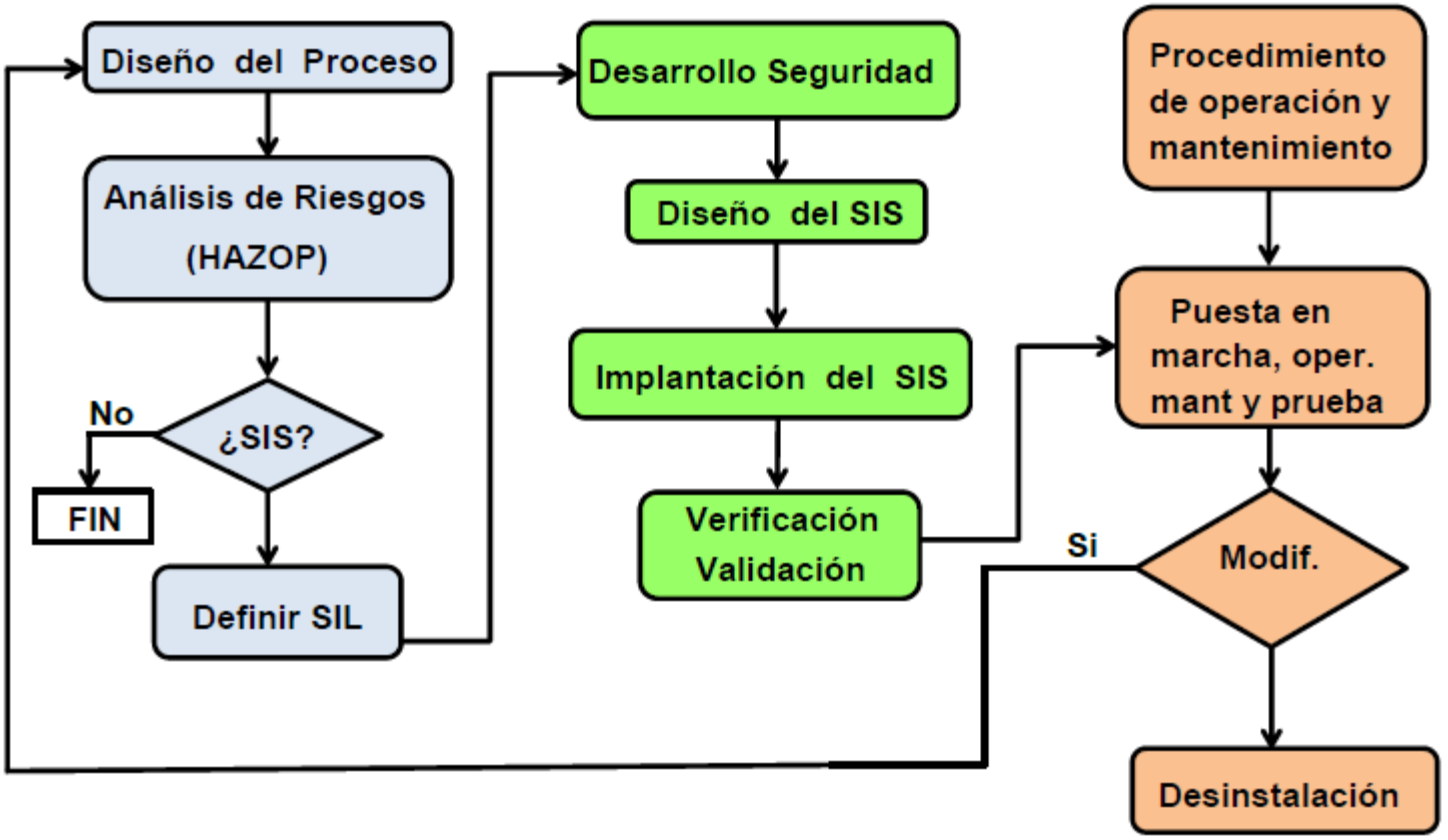
Es **una metodología práctica que establece los pasos necesarios para alcanzar la seguridad integral de una planta industrial**, definiendo la secuencia y documentación desde la etapa de concepción y diseño hasta la etapa de abandono de la misma.

Ciclo de Vida de Seguridad (SLC)

Puede dividirse en tres fases:

1. **La fase de análisis** (azul en fig. siguiente) Orientada a resolver y evitar el 44%(fig.inicial) de los accidentes debido a **especificaciones inadecuadas**.
2. **La fase de ejecución o realización** (verde) Orientada a disminuir el 21% de los accidentes causados por errores y/o omisiones durante el **diseño, la implementación, la instalación** y la puesta en servicio
3. **La fase de operación** (rojo), Orientada a disminuir o eliminar el 35% de los accidentes que son causados por incorrecta operación o mal mantenimiento y cambios realizados después que el sistema ha sido puesto en servicio.

Ciclo de Vida de Seguridad (SLC)

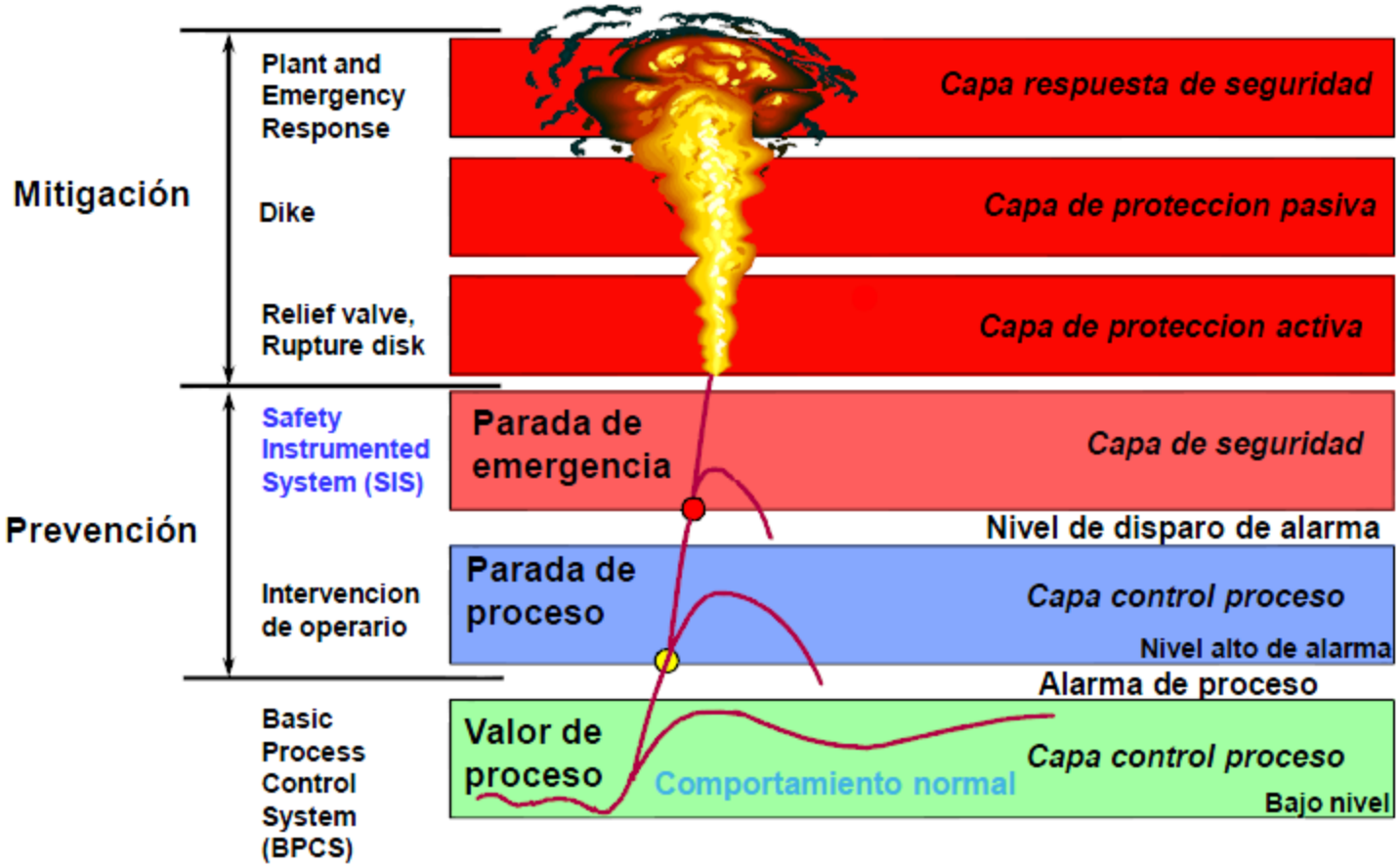


LOPA, Análisis de Capas de Protección

LOPA, Análisis de Capas de Protección (LOPA, Layers of Protection Analysis)

- Las capas de protección son sistemas de protección independientes que reducen el riesgo mediante el control, la prevención o la mitigación.
- Las capas de protección más adecuadas a adoptar en las instalaciones se derivarán de la elaboración de un análisis de riesgos específico en las mismas, mediante la aplicación de una o varias técnicas de identificación de riesgos.

LOPA, Análisis de Capas de Protección



LOPA, Análisis de Capas de Protección

Estándares Seguridad

Estándares internacionales ISA y IEC que nos permiten exigir criterios estrictos en el diseño de las instalaciones y equipos así como en la adopción de medidas de seguridad :

- **ANSI/ISA-84.01**, Application of Safety Instrumented Systems for the Process Industries, The Instrumentation, Systems, and Automation Society, Research Triangle Park, NC, 1996
- **IEC 61508**, Functional Safety of Electrical, Electronic and Programmable Electronic Safety-related Systems, Part 1-7, Geneva: International Electrotechnical Commission, 1998
- **IEC 61511**, Functional Safety: Safety Instrumented Systems for the Process Industry Sector, Parts 1-3, Geneva: International Electrotechnical Commission, 2003