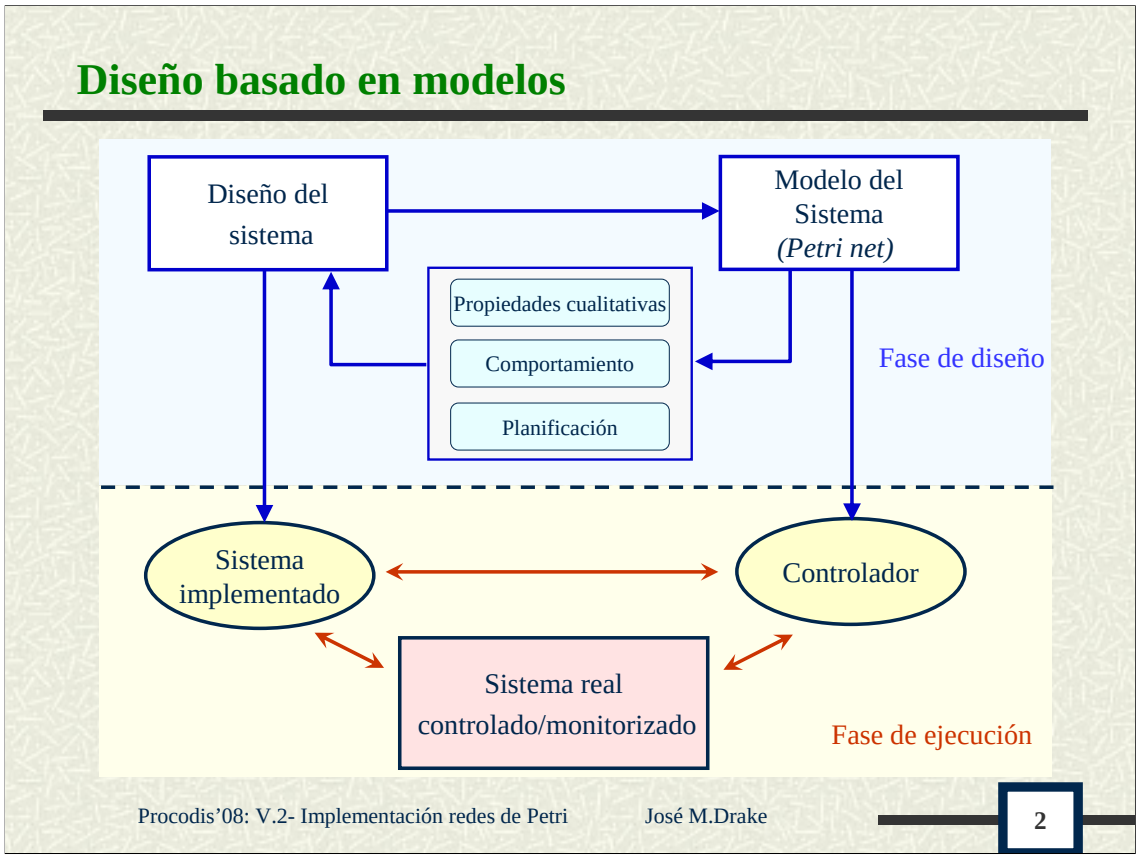
PROGRAMACION CONCURRENTE Y DISTRIBUIDA

V.2 Redes de Petri: Implementación.

J.M. Drake

Notas:


Diseño basado en modelos
Diseño del sistema
Modelo del Sistema (Petri net)
Propiedades cualitativas
Comportamiento
Planificación
Fase de diseño
Sistema implementado
Controlador
Sistema real controlado/monitorizado
Fase de ejecución
Procodis'08: V.2- Implementación redes de Petri
José M.Drake
2

Notas:

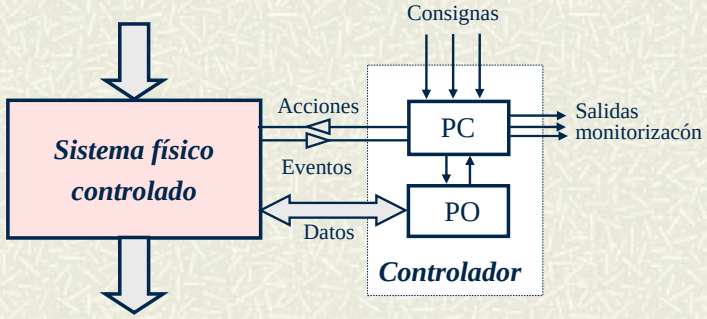
Ventajas de utilizar redes de Petri

- Facilidad para modelar características relevantes de sistemas concurrentes; Concurrencia, sincronización, conflictos, exclusión mutua, bloqueos, etc.
  - Buena visualización de las dependencias entre elementos del sistema
  - Capacidad de sucesivas vistas Top-Down (Refinamiento del modelo).
  - Capacidad de sucesivas vistas Bottom-Up (Composición modular).
- Capacidad de implementar el sistema a partir del modelo.
- Capacidad de validar el sistema contra patologías (Bloqueos, inestabilidad, ilimitación, etc.)
- Análisis de comportamiento (velocidad de respuesta, frecuencia de generación, utilización de recursos, etc.) sin necesidad de simulación o ejecución de pruebas.
- Posibilidad de análisis y animación mediante simuladores basados en eventos discretos.

Notas:

Parte de control y parte operativa

- Para facilitar el diseño y mantenimiento del proceso de control de un sistema físico, se descompone el controlador en dos subsistema:
  - **Parte operativa (PO):** Ejecuta los comandos de control y monitorización del sistema real.
  - **Parte de control o mando (PC):** Parte inteligente que dirige la estrategia de control y monitorización.



Notas:

Formas de implementar una red de Petri

- Implementación cableada (Lógica electrónica)
- Implementación con PLAs (Matrices lógicas programables)
- Implementación con computadores:
  - Flexibilidad frente a los cambios.
  - Potencia al poder integrar funciones complejas
  - Reducción del coste de desarrollo y fabricación.
  - Aumento de la fiabilidad de la operación:
    - Disminución del número de componentes
    - Tecnología mas fiable
    - Mayor capacidad de auto-diagnóstico.
  - Posibilidad de gestión distribuida

Notas:

Implementación de una RdeP mediante programa

- ***Implementación programada de una RdeP:*** *Es un programa que simula el disparo de transiciones de la RdeP, respetando la reglas de evolución del modelo teórico.*

- Diferencias entre el modelo teórico y la implementación:
  - Modelo teórico: El disparo de una transición es instantáneo e indivisible.
    - No hay código asociado a la transición
  - Modelo programado: El disparo de una transición no es ni instantáneo ni indivisible.
    - Existe un código asociado a la transición => Varias fases en el disparo:
      - Retirada de marcas de los lugares de entrada
      - Ejecución del código asociado
      - Depósito de marcas en los lugares de salida
    - ESTADO RdeP:= marcado + estado interno de las transiciones en disparo

Notas:

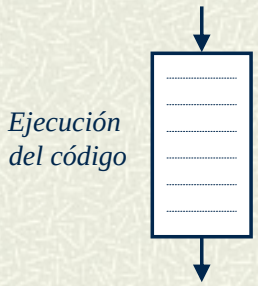
Modelo de representación

- **Entradas**: Estado del sistema real que se controla u ordenes del operador del sistema:
  - Eventos: Espera su llegada para que evolucione el sistema.
    - Polling: Periódicamente se exlora su estado.
    - Interrupción: Su llegada cambia el flujo del programa.
  - Entradas de nivel:
    - Lectura asíncrona: Las entradas se leen cuando se necesitan para tomar decisiones (produce aleatoriedad)
    - Lectura síncrona: Todas las entradas se leen simultáneamente, y en función de sus valores almacenados se toman las decisiones de control.
- **Salidas**: Variables de control del sistema real, mensajes de monitorización:
  - Acciones. Tiempo breve y semántica "se inician para ser terminadas"
    - Se asocian a la ocurrencia del disparo de una transición.
  - Actividades:: Tiempos prolongados y semántica "se ejecuta hasta que sea interrumpida"
    - Se asocian a la presencia de un testigo en una plaza.
  - Se pueden generar asíncronamente (cuando se calculan) o síncronamente (simultáneamente al final).

Notas:

Implementación de las actividades

Transición temporizada



- Transición no instantánea
- Requieren nuevas normas de gestión de los eventos

Plaza ocupada por testigo

- Transiciones instantáneas
- No se requieren nuevas normas de gestión de los eventos

Notas:

Tipos de implementaciones(1)

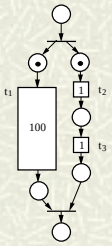
- Interpretadas:*La estructura de la RdeP y su marcado se codifican en estructuras de datos de las cuales uno o varios programas extraen la información para hacer evolucionar la RdeP.*
  - *La RdeP se codifica como una estructura de datos.*
  - *Requieren un interprete que es independiente de la topología de la RdeP*
- Compiladas: *Se generan uno o varios programas cuyos flujos de control reproducen el comportamiento de la RdeP.*

Notas:

Tipos de implementación (2)

- Secuencial: *Un único proceso ejecuta la RdeP*
  - Si hay código asociado a las transiciones => El único proceso secuencializa su ejecución.
  - El orden en el que el proceso dispara las transiciones es crítica.
    - Retrasos sistemáticos.
    - Resolución determinista de los conflictos.

t_2 y t_3 se retrasan sistemáticamente por t_1



  - Esta clase de implementación es adecuada para aplicaciones cuya parte operacional está compuesta por acciones cuyo tiempo de ejecución es irrelevante.
- Concurrente: *Varios procesos inter-actuantes ejecutan la RdeP*
  - En entorno monoprocesador => no hay concurrencia sino entrelazado

Notas:

Tipos de implementación (3)

- Centralizada: *La interpretación de la RdeP se implementa con un único proceso (coordinador) que actúa sobre la totalidad de la red.*

  La parte operacional de la implementación puede ser ejecutada secuencialmente o concurrentemente.

- Descentralizada: *La RdeP se descompone en diferentes subredes, y de la evolución de cada una de ellas se realiza por un interprete diferente que opera localmente.*

  Es muy adecuada para ser implementada en sistemas distribuidos en los que cada computador gestiona una subred local.

Notas:

- **Síncrona:** *La RdeP es ejecutada en pasos ("steps"). Se disparan todas las transiciones concurrentemente disparables para un marcado M y no se considera ningún otro disparo hasta haber alcanzado el sucesor de M.*
- **Asíncrona:** *Cuando el disparo de una transición ha concluido se actualiza el marcado lo que puede posibilitar nuevos disparos sin espera alguna.*

Notas:

Ejemplo implementación con coordinador centralizado

- El disparo de una transición habilitada tiene tres fases:
  1. Extraer los testigos de las plazas de entrada de la transición.
  2. Ejecuta el código asociado a la transición.
  3. Entregar los testigos en las plazas de salida de la transición.
- Cada transición tiene asociada una prioridad. Si dos transiciones están habilitadas para un determinado marcado, se dispara la transición de mayor prioridad.
- Se manejan dos listas:
  - treatment_list: Contiene las transiciones que deben ser chequeadas en la actual activación.
  - list_in_formation: Contiene las transiciones que como consecuencia de la actual activación deben ser chequeadas en la próxima activación.
- Se utilizan dos tipos de thread:
  - Una thread de la máxima prioridad interpreta la red de Petri y determina las transiciones que están activadas.
  - Un pool de threads ejecutan concurrentemente las tareas asociadas al disparo de cada transición.

Notas:

Seudocódigo del coordinador centralizado

```
while(true){
   //enabling analysis phase
   while elements in treatment_list {
         T = next_element(treatment_list);
         if enabled (T) {
            //start of transition firings
            Start_fire (T) ;
            Demark_input_places (T);
         }else{
             Update (list_in_formation) ;          // only in SRP
             Change_representing_place (T) ; // only in DRP
         }
   }
   //end of firings phase
   while(not end_fire pending ){
       wait for end_fire (T) ;
       Mark_output_places (T) and Update (list_in_formation) ;
    }
   // lists update
   treatment_list =list_in_formation;
   clear list_in_formation ;
}
```


Notas:

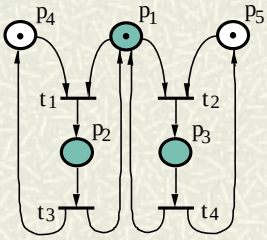
Estrategias de selección de transiciones analizables

- La eficiencia del algoritmo se basa en que en cada activación se pueda establecer un conjunto reducido de actividades de las que se deben analizar si están habilitadas o no.
- Estrategias:
  - Todas las transiciones (BF Brute Force): Se analizan todas la transiciones y se activan las que estan habilitadas por orden de prioridad
  - Static Representing Places (*SRP*). Se seleccionan las transiciones que son salida de las plazas con marcado, seleccionadas de entre un conjunto reducido de plazas (Representing Places) fijadas estáticamente.
  - Dynamic Representing Places (*DRP*). Igual que el caso anterior pero las Representing places se modifican dinámicamente.
  - Enabled Transitions (*ET*): Se seleccionan un conjunto reducido de transiciones generado en la activación previa.

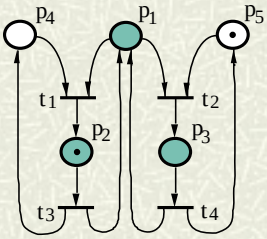
Notas:

Static Representing Places (*SRP*)

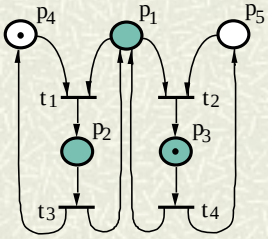
- A cada transición se le asocia una de sus plazas de entrada como su plaza representante. La selección se realiza estáticamente para que con el mínimo número de plazas se cubran el máximo numero de transiciones.
- Las restantes plazas se denominan plazas de sincronización.
- Solo se analizan las transiciones cuyas Representing Places están habilitadas.



Se analizan t_1 y t_2



Se analiza t_3

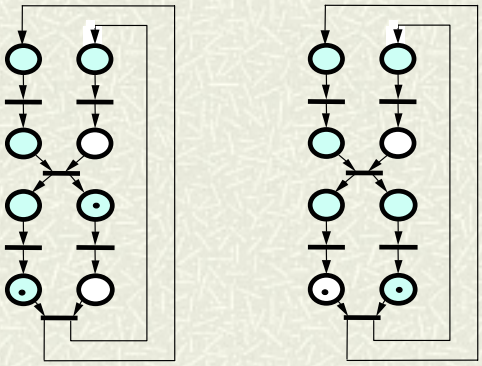


Se analiza t_4

Notas:

Dynamic Representing Places (*DRP*)

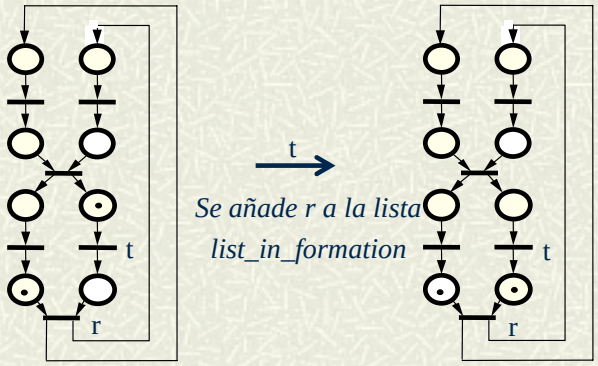
- Igualmente que antes sólo se analizan las transiciones cuyas Representing Places tienen testigos.
- Si estando la RepresentingPlace de una transición con testigo, no se dispara, se cambia la representingPlace a otra plaza de entrada que no tenga testigo.



Notas:

Enabled Transitions (*ET*)

- Cuando se finaliza una activación, se incluyen en la lista de transiciones que deben ser analizadas en la próxima activación, aquellas que son salida de plaza cuyo estado ha cambiado en la activación anterior.



Notas: