Modelling Workflow with Petri Nets

Workflow Management Issues
Georgakopoulos, Hornick, Sheth

Workflows & Petri Nets (PNs)
- WFMS give an explicit representation of the BP logic thus allowing for computerized support
- PNs are an established tool for modelling & analyzing business processes:
  - Can be used as a design language for the specification of complex WPs
  - PN theory provides for powerful analysis techniques for verifying the correctness of W/F procedures.
- PN primarily used to study dynamic concurrent behaviour of n/w-based systems with a discrete flow.
Workflows & Petri Nets (cont’d)

- Workflows are case-based, i.e., every piece of work is executed for a specific case.
  - Case: the subject of operation in a business process execution, e.g., mortgage application, hospital admission, insurance claim, tax declaration, order, request for information...
- A workflow process is designed to handle similar cases. Cases are handled by executing tasks in a specific order.

A three-dimensional view of a WF

Basics of Petri Nets

- Petri nets comprise two types of nodes: places and transitions. An arc exists only from a place to a transition or from a transition to a place.
- A place may have zero or more tokens.
- Graphically, places, transitions, arcs, and tokens are represented respectively by: circles, bars, arrows, and dots.
Dynamic modelling with Petri nets

- **Transitions** are the active components.
  - Often represent an event, an operation, a transformation or a transportation.

- **Places** are passive.
  - Usually represents a medium, a buffer, a geographical location, a state, a phase or a condition.
  - Depends on how the token is placed is interpreted.

- **Tokens** often indicate objects.
  - Can play a role as physical object, e.g. a product/person;
  - As an info object, e.g. a message;
  - An indicator of state a process is in or state of an object;
  - An indicator of a condition, i.e. the presence of a token indicates whether a certain condition is fulfilled.

Object Life Cycle (OLC) with Petri Nets

- A Petri net attaches to a life cycle of objects of a class.
- States correspond to places.
  - Initial state: state with token, there is only one initial state in an OLC.
- Transitions correspond to events, conditions (verify a condition) or processes (or atomic process: method) that changes object state.
- Tokens represent objects in this class.

Basics of Petri Nets (cont'd)

- Place
- Transition
- Arc
- Token
Example – claims process

- Transition node is ready to fire if & only if there is at least one token at each of its input places.

Basics of Petri Nets (cont’d)

Formal Notation of Petri Nets

- A bipartite graph, PN=(P, T, I, O)
  - P: finite set of places
  - T: finite set of transitions
    - I: (P*T) → N, I(p,t)=n, if n>0, p ∈ P, t ∈ T, then p is an input place of t: n is an input multiplicity (weight) for each input arc (p,t)
    - O: (T*P) → N, O(t,p)=m, if m>0, p ∈ P, t ∈ T, then p is an output place of t: m is an output multiplicity(weight) for each output arc (t,p)

By default, the weight of an arc is equal 1, otherwise it will be noted.
The input multiplicity of an arc between an input place and a transition determines how many tokens have to be present in the place so that the transition is enabled.
Formal Notation of Petri Nets (cont'd)

- A state of a Petri net is a function \( s: P \rightarrow N \), assigning to each place \( p \in P \) a number of tokens at this place. A state space of a Petri net is a set of all \( s(p) \), \( p \in P \). (E.g. state space is \( \{2, 1, 0, 0, 0\} \))

- A transition \( t \) is enabled, \( t \in T \) in state \( s: P \rightarrow N \), if there are enough tokens present in each of the input places of \( t \), i.e. if and only if \( \forall p \in P, s(p) \geq I(p, t) \)

- A transition \( t \) can fire in a state \( s \) whenever it is enabled in this state. When it fires, it consumes \( I(p, t) \) tokens from each input place \( p \) and produces \( O(t, q) \) tokens in each output place \( q \).

  If \( t \) fires in state \( s \), this leads to a new state \( s' \) where \( \forall p \in P, s'(p) = s(p) - I(p, t) + O(t, p) \)

Properties of Petri Nets

- **Sequential Execution**
  Transition \( t_2 \) can fire only after the firing of \( t_1 \). This imposes the precedence of constraints "\( t_2 \) after \( t_1 \)."

- **Synchronization**
  Transition \( t_1 \) will be enabled only when a token is present at each of its input places.

- **Merging**
  Happens when tokens from several places arrive for service at the same transition.

Properties of Petri Nets (contd)

- **Concurrency**
  \( t_1 \) and \( t_2 \) are concurrent. With this property, Petri nets can model systems of distributed control with multiple processes executing concurrently in time.
Properties of Petri Nets (contd)

- **Conflict**
  \[ t_1 \text{ and } t_2 \text{ are both ready to fire but the firing of one leads to the disabling of the other transitions.} \]

- **Conflict** - (contd)
  - The resulting conflict may be resolved in a purely non-deterministic way or in a probabilistic way, by assigning appropriate probabilities to the conflicting transitions, e.g:

Example: Patients & a Specialist

Tokens: Specialist

Tokens: Patient

(W.M.P. van der Aalst)
Example: Patients & a Specialist (cont'd)

The process of a specialist treating patients:

If a specialist always treats two patients at the same time?
Example: In a Restaurant (cont’d)
Two Scenarios

- **Scenario 1:**
  - Waiter takes order from customer 1; serves customer 1; takes order from customer 2; serves customer 2.

- **Scenario 2:**
  - Waiter takes order from customer 1; takes order from customer 2; serves customer 2; serves customer 1.

Example: In a Restaurant (Scenario 1)

Example: In a Restaurant (Scenario 2)
Example: Vending Machine

- Scenario 1:
  - Deposit 5c, deposit 5c, deposit 5c, take 20c snack bar.

- Scenario 2:
  - Deposit 10c, deposit 5c, take 15c snack bar.

- Scenario 3:
  - Deposit 5c, deposit 10c, deposit 5c, take 20c snack bar.

Example: Vending Machine (Token Games)
Example: Insurance complaint process

To manage different cases, two solutions:
1. Token is added a value (case identifier or colour) for distinguish different cases
2. Each case corresponds to a unique instance of the Petri nets

Petri Nets over Time

- 1962 - Carl Petri originally proposed Petri Nets without any notion of time. Concept of time was intentionally avoided because addition of time restricts the behavior of the net.
- 1970s ~ - Addition of time has been discussed in order to analyze the performance of modelled system.
- Many properties are still undecided for Petri nets extended with data and time.

References

- http://www.wfmc.org/standards/model.htm
- “Coupling Object-Oriented and Workflow Modelling in Business and Information Process Reengineering”, Gregory N. Nentzas. IOS Press. 1999