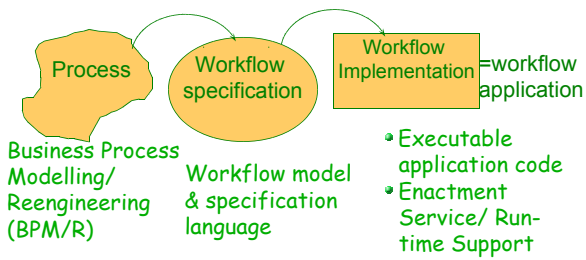


Modelling Workflow with Petri Nets

Workflow Management Issues

Georgakopoulos, Hornick, Sheth



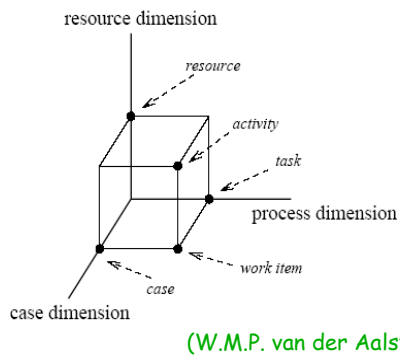
Workflows & Petri Nets (PNs)

- 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 WFs
 - PN theory provides for powerful analysis techniques for verifying the correctness of WF procedures.
- PN primarily used to study dynamic concurrent behaviour of n/w-based systems with a discrete flow.

Workflows & Petri Nets (cont'd)

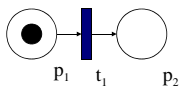
- Workflows are *case-based*, i.e., every piece of work is executed for a specific *case*.
 - Case: the subject of operation in a business process execution. E.g. mortgage application, hospital admission, insurance claim, tax declaration, order, request for information...
- A workflow process is designed 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

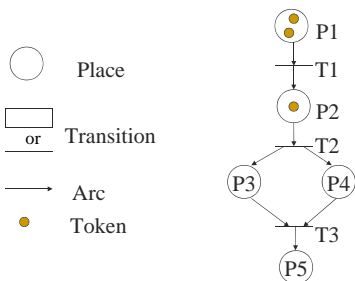
- **Transitions are the active components.**
 - often represent an event, an operation, a transformation or a transportation.
- **Places are passive.**
 - usually represents a medium, a buffer, a geographical location, a state a phase or a condition.
 - depends on how the token in place is interpreted
- **Tokens often indicate objects.**
 - can play a role as physical object, e.g. a product/person;
 - 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

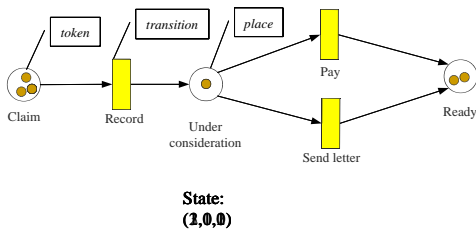
8

Basics of Petri Nets (cont'd)



9

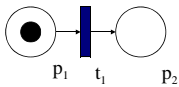
Example – claims process



10

Basics of Petri Nets (cont'd)

- Below is an example Petri net with two places and one transaction.
- Transition node is ready to *fire* if & only if there is at least one token at each of its input places



state transition of form $(1, 0) \rightarrow (0, 1)$
 p_1 : input place p_2 : output place

CA4 BPM PetriNets

11

Formal Notation of Petri Nets

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

12

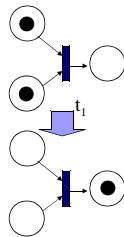
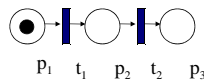
Formal Notation of Petri Nets (cont'd)

- A **state** of a Petri net is a function $s: P \rightarrow \mathbb{N}$, 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 $s: P \rightarrow \mathbb{N}$, 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 $\forall p \in P, s'(p) = s(p) - I(p,t) + O(t,p)$

13

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 there are at least one token at each of its input places.
- **Merging**
Happens when tokens from several places arrive for service at the same transition.



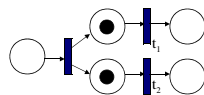
CA4 BPM PetriNets

14

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.



CA4 BPM PetriNets

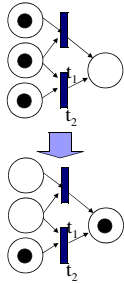
15

Properties of Petri Nets

(contd)

- **Conflict**

t_1 and t_2 are both ready to fire but the firing of one leads to the disabling of the other transitions.



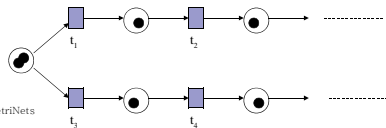
CA4 BPM PetriNets

16

Properties of Petri Nets (contd)

- **Conflict - (contd)**

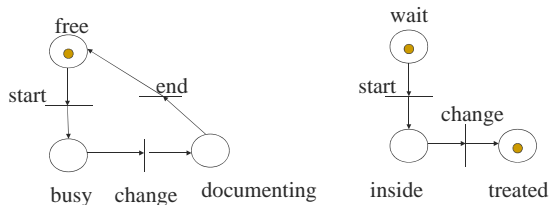
the resulting conflict may be resolved in a purely non-deterministic way or in a probabilistic way, by assigning appropriate probabilities to the conflicting transitions. e.g:



CA4 BPM PetriNets

17

Example: Patients & a Specialist



Tokens : Specialist

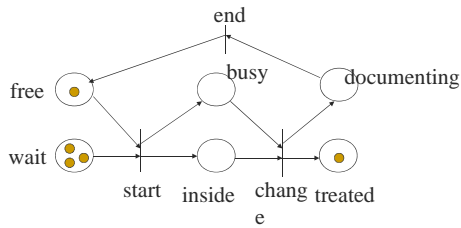
Tokens : Patient

(W.M.P. van der Aalst)

18

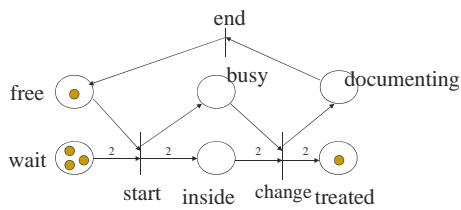
Example: Patients & a Specialist (cont'd)

The process of a specialist treating patients:



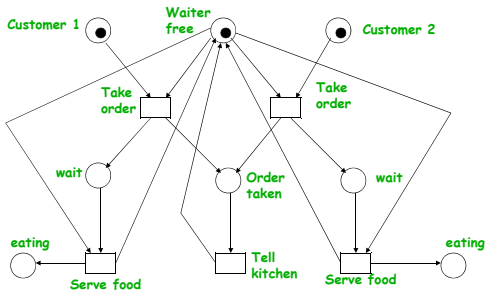
If a specialist always treats two patients at the same time? ¹⁹

Example: Patients & a Specialist (cont'd)



20

Example: In a Restaurant



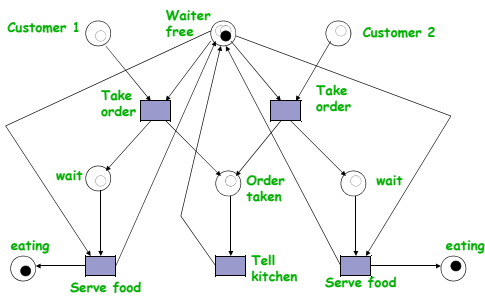
CA4 BPM PetriNets

21

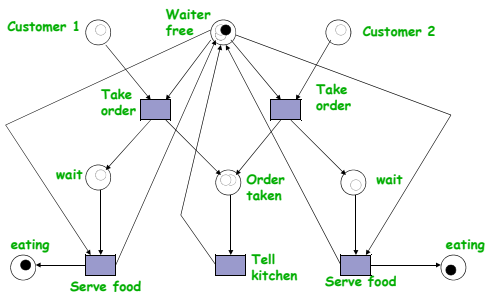
Example: In a Restaurant (cont'd) Two Scenarios

- **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.

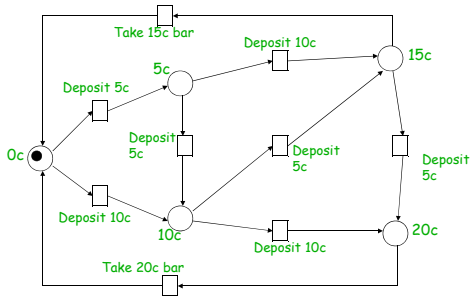
Example: In a Restaurant (Scenario 1)



Example: In a Restaurant (Scenario 2)



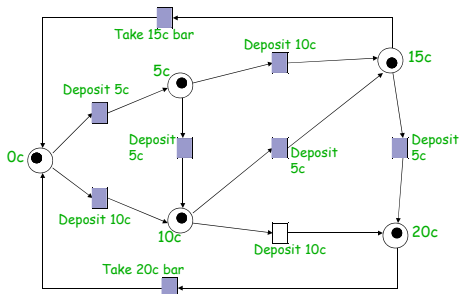
Example: Vending Machine



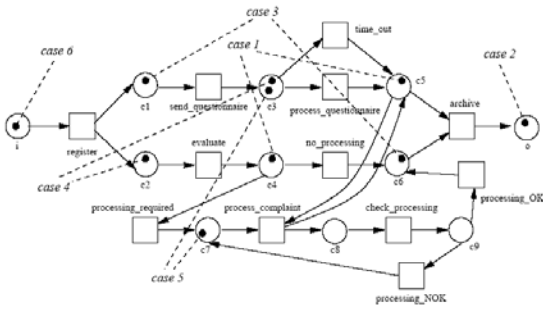
Example: Vending Machine (3 Scenarios)

- **Scenario 1:**
 - Deposit 5c, 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>
- "The Application of Petri Nets to Workflow Management", W. van der Aalst, J. Circuits, Systems & Computers, Vol. 8(1) (1998), pp. 21-66
- "Design and control of workflow process", Hajo A.Reijers, LNCS Springer, 2003
- "Coupling Object-Oriented and Workflow Modelling in Business and Information Process Reengineering", Gregory N. Mentzas, IOS Press, 1999
- "Workflow Management Coalition Terminology and Glossary" (WFMC-TC-1011). Technical report, Workflow Management Coalition, Brussels, 1996.
