Modeling the resource perspective

prof.dr.ir. Wil van der Aalst
resources can have many roles
resources can be distributed over many locations
work together synchronously or asynchronously

prioritization and allocation can be smart and flexible
Resource classification
Workflow management concepts

A workflow definition is composed out of three parts:

1. **process definition:**
   a description of the process itself

2. **resource classification:**
   a classification of the resources to be used

3. **resource management rules:**
   how to map work onto resources
Why a resource classification?

- Avoid hard coding or resources (otherwise the process needs to be changed in case of personnel changes).
- Organizations have structure (cf. organigram).
- Work distribution needs to be described: Who is doing what?
- Ordering of work items: In what order do we need to do things?
Resource classes

- **Resource**
  (participant, actor, user, agent)
  - A resource can execute certain tasks for certain cases.
  - Human and/or non-human (printer, modem): limited capacity.

- **Resource class**
  A set of resources with similar characteristic(s).

A resource class is typically based on:

- **Role**
  (skill, competence, qualification)
  Classification based on what a resource can do.

- **Group**
  (department, team, office, organizational unit)
  Classification based on the organization.
Example: 8 resource classes

- Secretary
- Manager
- Sales_clerk
- Office_worker

- Purchase
- Sales

Eindhoven

- Kees
- Pim
- Kevin
- Truus
- Jan
- Sjaak
- Anita
- Frank
- Ivone
- Ad

Amsterdam

Groups

Roles
Possible roles: student, professor, programmer
Convention used in this course

Each task executed by a resource (worker) is labeled with one role and one group.
• Insurance company X processes claims which result from traffic accidents with cars where customers of X are involved in. Therefore, it uses the following procedure for the processing of the insurance claims.

• Every claim, reported by a customer, is registered by an employee of department CD (CD = Car Damages). After the registration of the claim, the insurance claim is classified by a claim handler of rank A or B within CD. There are two categories: simple and complex claims.

• For simple claims two tasks need to be executed: check insurance and phone garage. These tasks are independent of each other.
The complex claims require three tasks to be executed: check insurance, check damage history and phone garage. These tasks need to be executed sequentially in the order specified.

Both for the simple and complex claims, the tasks are done by employees of department CD. After executing the two respectively three tasks a decision is made. This decision is made by a claim handler of rank A and has two possible outcomes: OK (positive) or NOK (negative).

If the decision is positive, then insurance company X will pay. An employee of the finance department handles the payment. In any event, the insurance company sends a letter to the customer who sent the claim. An employee of the department CD writes this letter.
Roles and groups

The following roles are identified:

- **Employee** (E)
- **Claim handler** (CH)
- **Claim handler A** (CHA)
- **Claim handler B** (CHB)

The following groups are identified:

- **Car Damages Department** (CD)
- **Finance Department** (FN)
Resulting model
• Each year travel agency Y has to process a lot of complaints (about 10,000). There is a special department for the processing of complaints (department C). There is also an internal department called logistics (department L) which takes care of the registration of incoming complaints and the archiving of processed complaints. The following procedure is used to handle these complaints.
Complaints handling (2)

• An employee of department L first registers every incoming complaint. After registration a form is sent to the customer with questions about the nature of the complaint. This is done by an employee of department C. There are two possibilities: the customer returns the form within two weeks or not.

• If the form is returned, it is processed automatically resulting in a report which can be used for the actual processing of the complaint. If the form is not returned on time, a time-out occurs resulting in an empty report. Note that this does not necessarily mean that the complaint is discarded. After registration, i.e., in parallel with the form handling, the preparation for the actual processing is started.
Complaints handling (3)

- First, the complaint is evaluated by a complaint manager of department C. Evaluation shows that either further processing is needed or not. Note that this decision does not depend on the form handling.

- If no further processing is required and the form is handled, the complaint is archived. If further processing is required, an employee of the complaints department executes the task ‘process complaint’ (this is the actual processing where certain actions are proposed if needed).

- For the actual processing of the complaint, the report resulting from the form handling is used. Note that the report can be empty. The result of task ‘process complaint’ is checked by a complaint manager. If the result is not OK, task ‘process complaint’ is executed again. This is repeated until the result is acceptable. If the result is accepted, an employee of the department C executes the proposed actions. After this the processed complaint is archived by an employee of department L.
Solution (only control flow)

1. Register complaint
2. Evaluate complaint
3. Process complaint
4. Process form
5. Time out
6. Check
7. Execute
8. Archive
9. End
Roles and groups

The following roles are identified:

- **Employee** (E)
- **Complaint manager** (CM)

The following groups are identified:

- **Department C** (DC)
- **Logistics department** (LD)
Roles:
- Employee (E)
- Complaint manager (CM)

Groups:
- Department C (DC)
- Logistics department (LD)
Protos (Pallas Athena)
Process view

Task

Diagram showing a process flow with nodes such as 'register', 'send bill', 'check availability', 'reminder', 'receive payment', 'ship goods', 'replenish', 'archive', and a trigger for 'time'.
Data view

data element
Relating both uses address contained by address
Relating both
Organizational/resource view

- Role
- Group

“is a” relationships between roles and groups.
Relating both
Relating both
Roles and groups

Activity properties register

- **Role**
  - Executor: Employee
  - Responsible: No role
  - Team: Company (C)

- **Group**
  - No activity
Case management (strict form)
4-eyes principle
Assumptions so far ...

- Eventually every work-item is executed by a single resource.
- Every resource is working on one activity at the same time.

Some observations:
- There may be a need to further limit the set of resources (e.g., the 4 eyes principle), i.e., we need to be able to specify further constraints.
- There may be many resources that have the right role/group combination, i.e., work distribution is needed.
- There may be many work items that can be executed by the same resource at a given point in time, i.e., work items need to be ordered.
Further constraints

if compensation > 1000 euro
then role := manager
else role := employee
Push versus pull
Push control

workflow enactment service

push to a selected "victim"
Pull control

workflow enactment service

pull selected work items
Workflow (1)

Process Model:

Pull control

Work items:

John
Clare
Mike
Jody

Click on a work-item to select a piece of work for a specific case.

/ faculteit technologie management
Most WFM systems are hybrid, e.g., Staffware.
Step 2: Specify System Behaviour when Offering a Work Item

The offer process involves choosing which participants should be informed of the offer and eventually do this work. As you have specified the system, manage the offer process to be offered to. Begin by specifying a set of participants and/or to distribute offers of which at runtime will contain a participant’s userid or the name of a role.

Participants:
- Wil van der Aalst (wvdaalst)
- Boudewijn van Dongen (bvdongen)
- Kees van Hee (khee)
- Eric Verbeek (everbeek)

Roles:
- all
- check role
- final tasks role
- register role

Net Parameters:
- Name
- Refers to

offered to individuals

offered to groups
Q1: Who is doing what?

- **Pull control**
  - Determined by people, i.e., a "race" among resources.
  - Select to start and/or select to allocate work item.

- **Push control**
  - Round robin.
  - Weighted round robin.
  - Shortest queue.
  - Select the most specialized resource (i.e., do not allocate "generalists" unless needed).
Q2: In what order?

- **Pull control**
  - Determined by people, i.e., a "race" for attention.
  - System may "suggest" urgent work items or offer multiple views on the work available.

- **Push control**
  - System decides whether the worker can view and/or select multiple work items, i.e., order may be enforced or suggested.

- In both cases possible **queueing disciplines** are:
  - FIFO (First In First Out)
  - LIFO (Last In First Out)
  - SPT (Shortest Processing Time)
  - EDD (Earliest Due Date)
  - PRIO (Prioritization based on case attributes)
shared resource

FIFO (First In First Out)
LIFO (Last In First Out)
SPT (Shortest Processing Time)
EDD (Earliest Due Date)
PRIO (Prioritization based on case attributes)
How is queueing?
Modeling resources (naive view)
• **Limitations:**
  • Resources can only have one role/group.
  • No further constraints (e.g., 4-eyes principle).
  • Resources and cases cannot be identified.
  • Just pull control and no strategies with respect to resource selection and ordering of work-items.
Let us try to model this in CPN Tools

- Limitations addressed:
  - Resources can only have one role/group ☺.
  - No further constraints (e.g., 4-eyes principle).
  - Resources and cases cannot be identified ☺.
  - Just pull control and no strategies with respect to resource selection ☻ and ordering of work-items.

- CPN Model is *not* proposed as an end-user language!
- It is merely used to provide semantics.
- Resource allocation is generic, i.e., selection rather than modeling.
**Generic task model**

**arriving case triggers the creation of a work-item**

**completed activity triggers next task**

**resources that have been selected by offer() and that are free can start working on the work-item.**

**resources are global**

**based on the properties of the task (name, role and group) and the properties of resources (name, roles, groups, and state) a set of possible resources is selected**

1. Arriving case triggers the creation of a work-item.
2. Taskinfo is provided.
3. Taskinfo is sent to task.
4. Taskinfo is processed to determine the next available work-item.
5. Work-item is created and sent to offer.
6. Offer is triggered by the work-item.
7. Offered resources are selected based on the properties of the task and the available resources.
8. Offered resources are scheduled for work.
9. Resources are locked to ensure exclusive access.
10. Resources are unlocked once work is completed.
11. Completed activity triggers the next task.

**Notes:**
- Resources are global and can be used by multiple tasks.
- Offer() selects a set of possible resources based on the task's requirements.
- Offered resources can start working on the work-item once they are free.
- Resources are locked to prevent conflicts and ensure data integrity.
• colset Case = int timed;
• colset ResourceID = with John | Clare | Mike | Jody;
• colset TaskId = with A | B | C | D |E;
• colset Role = with R1 | R2 | R3 | R4 | R5;
• colset Roles = list Role;
• colset Group = with G1 | G2 | G3 | G4 | G5;
• colset Groups = list Group;
• colset State = with free | busy;
• colset Resource = product ResourceID * Roles * Groups * State;
• colset Resources = list Resource;
• colset RIDs= list ResourceID;
• colset Task = product TaskId * Role * Group;
• colset CxRS = product Case * RIDs timed;
• **Signature of function:**
  
  \[
  \text{fun offer}(t: \text{Task}, rs: \text{Resources}) = ... : \text{RIDs};
  \]

• **Example function "pure pull":**
  
  \[
  \text{fun pull}(t,[]) = [] | \\
  \text{pull}((\text{tid}, \text{role}, \text{group}), (\text{rid}, \text{roles}, \text{groups}, \text{state})::l) = (\text{if elt}(\text{role}, \text{roles}) \text{ andalso elt}(\text{group}, \text{groups}) \text{ then [rid] else []}) ^^ \text{pull}((\text{tid}, \text{role}, \text{group}), l);
  \]

• **Example function "random push":**
  
  \[
  \text{fun push_random}(t, rs) = [\text{pick_random}(\text{pull}(t, rs))];
  \]
Experiments

- Poisson arrival process (average inter-arrival time = 20 minutes).
- Neg. exp. processing time for all four tasks (average processing time = 8 minutes).
- 10 simulations of 10,000 cases.
- pull control
- flow time = 45.6+/-7
- utilization = approx. 0.4
- wip = 2.27+/-0.04
- push control (random)
- flow time = 253+/-33
- utilization = approx. 0.4
- wip = 12.6+/-1.7

- flow time much higher due to high utilization of Jody, she does all D's but also her share of A, B, and C.
Increasing the utilization

• Poisson arrival process (average inter-arrival time = 20 minutes).
• Neg. exp. processing time for all four tasks (average processing time = 12 minutes), i.e., an expected average utilization of 0.6.
• 10 simulations of 10,000 cases.
- pull control
- flow time = 187\(+/-28\)
- utilization = approx. 0.6
- wip = 9.4\(+/-1.5\)
- all resources can do all
- flow time = 52.4 +/- 0.5
- utilization = approx. 0.6
- wip = 2.63 +/- 0.04

```plaintext
val initResources =
[(John, [R1,R2], [G1,G2], free),
 (Clare, [R1,R2], [G1,G2], free),
 (Mike, [R1,R2], [G1,G2], free),
 (Jody, [R1,R2], [G1,G2], free)];
```
- dedicated resource per task
- flow time = 103.7 +/- 1.9
- utilization = approx. 0.6
- wip = 5.2 +/- 0.1
General observations push versus pull

- Pull provides a better performance than push unless there is a strong “specialization effect” and “preferences do not match capabilities”.
- Larger roles/groups (i.e., fewer specialized resources) lead to better performance unless there is a strong “specialization effect”.

![Airport scene with long queues and signs]
Ordering of work-items

- Limitations addressed:
  - Resources can only have one role/group 😊.
  - No further constraints (e.g., 4-eyes principle).
  - Resources and cases cannot be identified 😊.
  - Just pull control and no strategies with respect to resource selection 😊 and ordering of work-items.
Now there is resource manager
Resource manager

- request
  - Request
  - (t,c,tm)
  - put in queue
    - Request
    - []
    - queue
      - Requests
      - []
      - initResources
        - 1`[]
        - capabilities
          - Capabilities
          - 4`
          - 4`
          - 1`(John,[R1],[G1,G2])++
          - 1`(Clare,[R2],[G1,G2])++
          - 1`(Mike,[R3],[G1,G2])++
          - 1`(Jody,[R4],[G1,G2])++
        - allocate
          - Allocation
          - (t,c,tm,r)
          - [t=(tid,role,group), elt(role,roles), elt(group,groups)]
- allocate
  - Allocation
  - (t,c,tm,rx)
  - allocate
    - Allocation
    - (t,c,tm,rx)
  - free-resources
    - free-resources
    - release
      - Resource
      - r
      - fall(initResources)
        - 1`John++
        - 1`Clare++
        - 1`Mike++
        - 1`Jody++
Generic task
General observations priorities

- SPT (Shortest Processing Time) can significantly reduce flow times (although the same amount of work is done) and improve service levels.

- FIFO (First In First Out) and LIFO (Last In First Out) are often comparable in terms of average flow time, but LIFO may lead to higher services level whereas FIFO aims at fairness.

- EDD (Earliest Due Date) may be used to increase service levels.

- Note that a BPM system allows for “smart” ordering of work-items (unlike most physical queues).
Other phenomena (e.g., Yerkes-Dodson Law of Arousal)
Summary

- Roles and groups are used to link tasks and resources.
- Two questions:
  - Q1: Who is doing what?
  - Q2: In what order?
- Two basic mechanisms:
  - pull: people decide
  - push: the system decides
- Many variants are possible, cf. resource patterns.
- Closely linked to performance issues.
Exercises, see OASE
"BPMS-instruction-3-resource.pdf"