
ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

# Business Process Management

## Introduction

**Federico Chesani**
Department of
Computer Science and Engineering
University of Bologna

# Credits and sources

These slides have been partly inspired by:

- Work of Prof. Marco Montali, Free University of Bozen http://www.inf.unibz.it/~montali/
- Mathias Weske, *Business Process Management,* Second Edition, Springer, 2012

- Workflow patterns http://www.workflowpatterns.com/
- Wil M. P. van der Aalst, Arthur H. M. ter Hofstede, Bartek Kiepuszewski, Alistair P. Barros: Workflow Patterns. Distributed and Parallel Databases 14(1): 5-51 (2003)
- Russell, N.C.; ter Hofstede, A.H.M.; van der Aalst, W.M.P.; Mulyar, N.A. Workflow control-flow patterns : a revised view. Tech Rep. BPM Center BPM-06-22 https://pure.tue.nl/ws/files/2456784/710796038377571.pdf

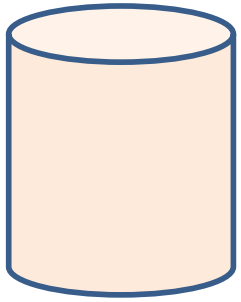
- https://academic.signavio.com/p/login
- https://bpm-book.com/BpmBook/WebHome

- https://www.draw.io/

# Business Processes

From Information Systems...

- Static, conceptual description of the "*thing*"
- Both conceptual models (e.g., E/R schemas, ontologies) as well as facts (data, events, logs)
- Partly captures the flow of your business
- Usually, no assumptions on how the information about the "*thing*" is used



| Instance/Cart ID | Customer ID | Completed? | When processed | When shipped | When delivered |
|---|---|---|---|---|---|
| 12 | 12345 | ✓ | 03/06/2018 | 03/06/2018 | 04/06/2018 |
| 37 | 12345 | ✓ | 03/06/2018 | 04/06/2018 | |
| 56 | 87654 | ✓ | 05/06/2018 | | |
| 89 | 87654 | | | | |

Which information about the process ?

What if I am interested, e.g., to performances of my business?

# Business Processes

... to Business Process Management (BPM):

- Activities to be performed
- Interaction with external stakeholders and customers
- Use cases
- Law and external norms
- Internal policies and best practices
- Object lifecycles
- Organizational issues
- . . .

Notice that:

- Products and Services are provided through the **execution of activities**
- Activity executions call for **coordination**
- (Un)Effective coordination influences **costs**, **time**, **quality** of your business (of your products, of your services, of your industry, of your employees' worklife and satisfaction)

# Business Processes

Business Process Management (BPM), according to Weske:

BPM includes concepts, methods, and techniques to support the **design**, **administration**, **configuration**, **enactment**, and **analysis** of BPs.

# Business Processes – an example

Brian wants to travel to Sydney. He decides to call a cab to get from his apartment to the airport in Frankfurt. After 10 minutes, the cab arrives at his apartment. Then, the cab needs 1.5 hours for the 20 km to terminal C. Once arrived, Brian checks in using the Self-Check-In terminal and receives his boarding pass. Of course, he could have also checked in at the Check-In counter. Since he has no baggage to check in, he goes directly to the security screening that is located around 100 m left of the departure hall. Luckily, the queue is not very long. Already after 5 minutes he can go to the gate. Instead of relaxing in the frequent flyer lounge, he checks out the duty-free shops and buys a newspaper, before returning to the gate 15 minutes later. Another 10 minutes later, he boards the airplane.

Let us abstract from Brian's specific case...

- How to depict the generic process of X flying to Y ? The text is quite boring... any graphical language?
- Make it **easy**... your colleagues should be able to easily understand it
- Make it **unambiguous**: you don't want different people doing different things, when they are supposed to act similarly

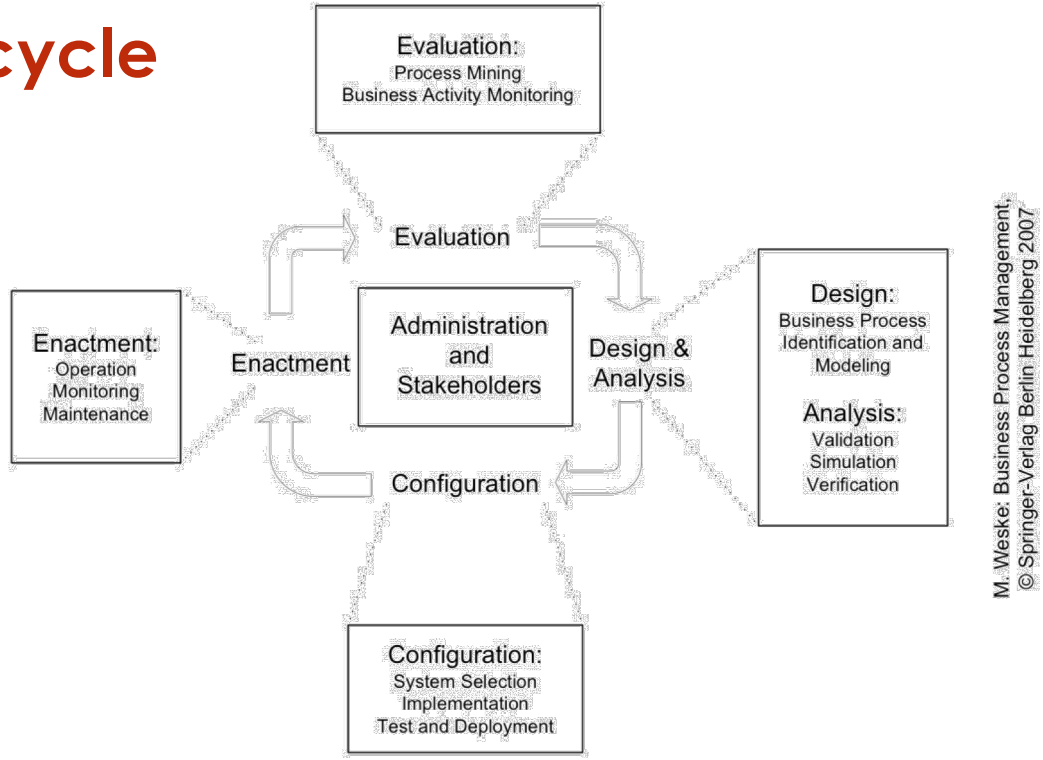
# Business Processes – why you should bother

- IT infrastructures heavily permeated all the possible business types
- From an organizational issue, processes became also a **technical** issue

- Strong pushes to **automation**: whatever can be done automatically, it must be done automatically
- **Decisions** are automatically taken, depending on what you will specify (policies, criteria, rules...) (how? are decisions part of a process?)

- If you can't clearly explain how your "thing" is run, which activities, which decisions, your "thing" is doomed...
- IT infrastructures provide easy logging of any aspect of your business... you might be asked for/explain/defend the way your business run... and logs are there as witnesses...

# Business Process Lifecycle




- **Design&Analysis**: conceptual schema, process models, then validation, simulation and properties verification
- **Configuration**: i.e., the implementation of the business process: database schema, process execution in a socio-technical environment, etc.
- **Enactment**: the business runs! BP instances are initiated, event logs are collected, monitoring of the runs, and *previsions about the future*
- **Evaluation**: process mining, quality and quantitative kpi, and possibly process re-engineering.

# Which stakeholders?

- Chief Process Officer
- Business Engineer (domain expert, usually with an economic background)
- Process Designer (business analyst)
- Process Participant
    - Knowledge Worker
- Process Owner (the guy responsible for its execution)
- System Architect
- Developers

} IT

How to share and communicate knowledge among all these different actors, all with different knowledge, skills and background?

# Business Processes – which types?

From a high level perspective (business-oriented):

- Organizational vs Operational

**Organizational**:

They describe inputs, outputs, expected outcomes, and dependencies (versus other organizational processes). They are described in formal and (more often) semi-formal languages, and capture coarse-grained business functionalities. An organizational process usually corresponds to many operational processes.

**Operational**:

Activities (and their relationships) are specified, while implementation is disregarded. However, they are the basis for the effective implementation. A strong push in the BPM to describe operational processes by means of formal methods (why?)

# Business Processes – which types?

From a high level perspective (business-oriented):

- Intra vs Inter-organizational

**Intra-organization**:

All the activities are executed within the business boundaries, there is information about allocated resources, about execution times, often orchestration is used as coordination model, there is some chance of intervention and improvement

**Inter-organization**:

Part of the activities are executed somewhere else, no control on those activities, no information about them, choreography as a coordination method, legal and business contract complication, technical hurdles, semantical hurdles, needs for a number of (formal) concepts such as dynamic controllability.

# Business Processes – which types?

From an execution-oriented perspective:

- Degree of **automation**: which human intervention? E.g., buying a flight ticket online, vs. opening/claiming a technical issue for a mobile phone (socio-technical systems?)
    - Human presence in a business process directly affect the process deviations
- Degree of **repetition**: buying a flight ticket is repeated many times -> makes sense to fully analyze it. Designing a skyscraper is a one-time process, the focus there is on collaborative aspects and logging of process decisions. What about scientific experiments?
- Degree of **structuring**: from fully predictable processes (*production workflows*) to ad-hoc processes. Degree of structuring greatly affects the choice of formal models: control-flow approaches vs. declarative approaches. E.g.: go to the bar and ask for a coffee… which process? Notice also that structuredness has a formal counterpart (definition of *well-structured* processes, depending on the formalism)

# Business Processes – which types?

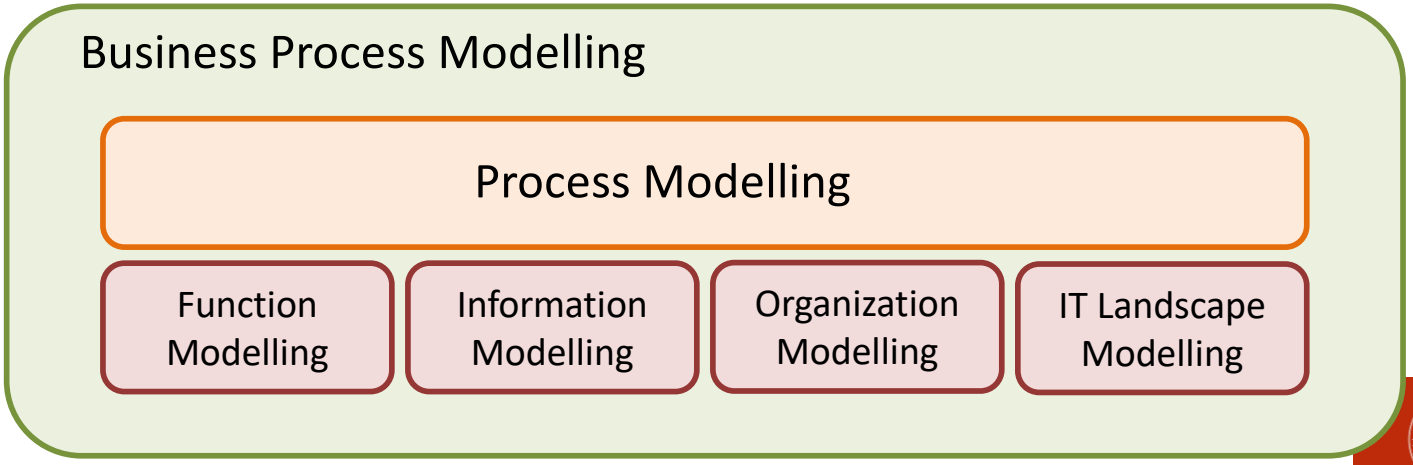
From a high level perspective (business-oriented):

- Organizational vs Operational
- Intra vs Inter-organizational

From an execution-oriented perspective:

- Degree of automation (socio-technical systems?)
- Degree of repetition
- Degree of structuring

Also, according to Weske:



# Summing up – which goals for BPM?

- Better understanding of business operations and their relationships
- Achieve flexibility – i.e., the ability to evolve, to react to external stimuli
- Continuous process improvement
- To narrow the gap between the company view of its business, and its implementation

# A historical note…

- Process management has a very long history, in particular in the economic and management research field (18th century, industrial revolution)
- The advent of IT systems, starting from the 1970s, revolutionized the approaches
- In the 1993, the Workflow Management Coalition (WfMC) was founded, and started to produce formal definitions and languages (http://www.wfmc.org/ )

Def: **Workflow** is the automation of a business process, in whole or in part, during which documents, information or tasks are passed from one participant to another for action, according to a set of **procedural rules**.

Distinction between:

- System workflows (no humans involved)
- Human Interaction Workflows

# A historical note…

The workflow perspective has been also criticized:

- **Lack of adequate support for knowledge workers**. Workflow management systems have massive effects on users daily work and routines: they usually don't care about the technology, but rather how the system constraint them in the process execution. Moreover, the lack of support to flexibility and knowledge pushes the users to avoid/skip the IT systems.
- **Technical Integration Challenges**. Workflows need to be integrated into existing technical solutions; in turn, this call for technical issues, but also about semantical issues. The advent of SOA mitigated the former issue; the latter is still a current problem in real life.
- **Process Support without Workflow Systems**. A number of everyday applications run as production processes, but are directly coded into the application implementation. E.g., the "buying flight ticket" process and its implementation into a web-based application framework: the logic is embedded into the source code.

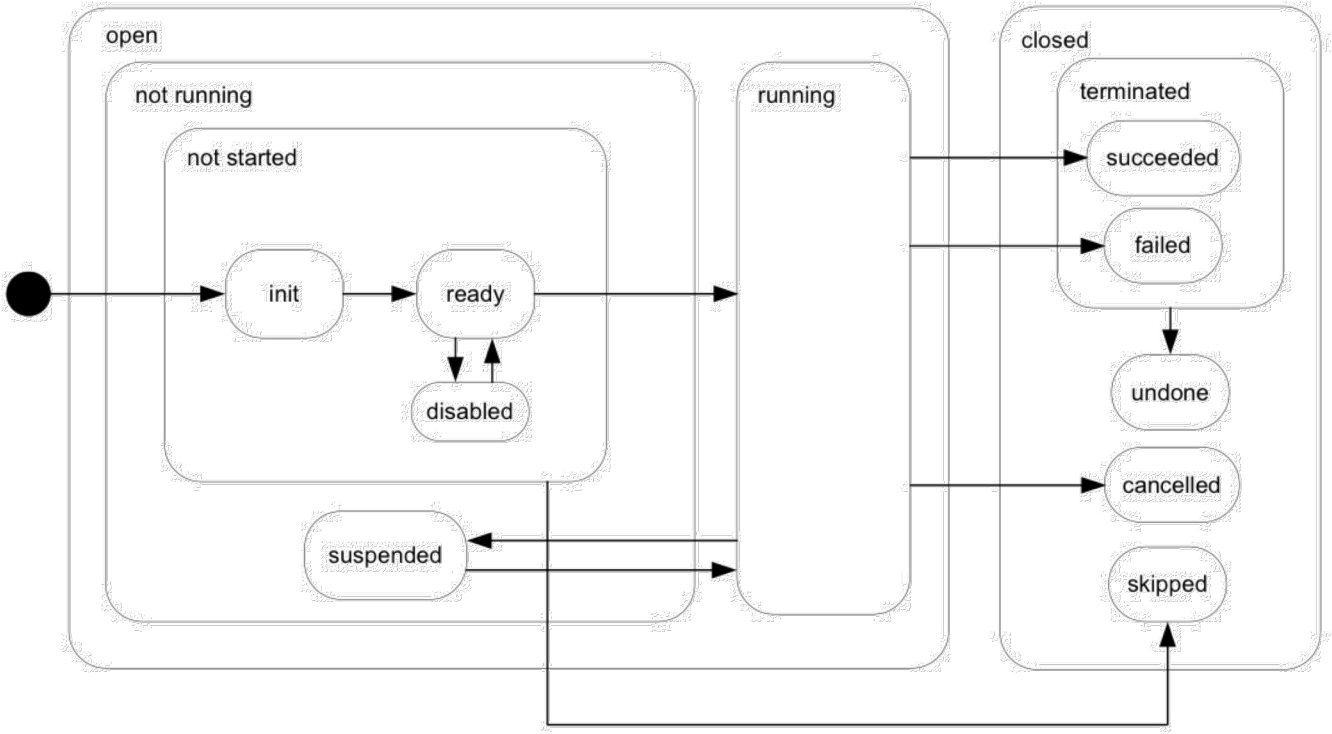
# Business Process Modelling Foundation

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

# Activity Models and Activity Instances

- An **Activity Instance** represents the actual work conducted during the execution of a business *instance*, i.e., the actual (minimal) unit of work.
- An **Activity Model** describes a set of similar Activity Instances.

- An Activity Instance, during its life, passes through different **states**.

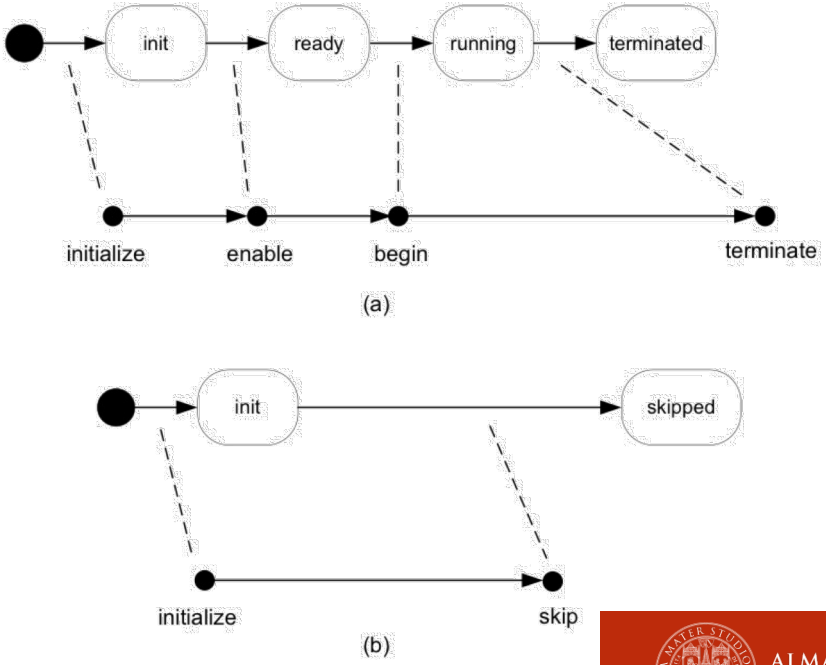



M. Weske: Business Process Management,
© Springer-Verlag Berlin Heidelberg 2007

# Activity Models and Activity Instances

How to characterize the lifetime evolution of an activity instance?

- The notion of events is used: each state transition defined by the happening of an **event**
- Events have a temporal ordering: i.e., they can happen only in certain specified **orders**
- As a consequence, an **activity instance is described by a totally ordered set of events**.
- Events are **points** in time (no duration)
- The time an activity instance stands in a certain state (the duration of that state) is given by the corresponding entering and the exiting events




M. Weske: Business Process Management,
© Springer-Verlag Berlin Heidelberg 2007

# Activity Models and Activity Instances

- Activity instances then are described as (partial) sets of events:

E.g.:

$$i = (E_i, <_i), \qquad with\ E_i \subseteq \{i_i, e_i, b_i, t_i\}, <_i \subseteq \{(i_i, e_i), (e_i, b_i), (b_i, t_i)\}$$

- There is a function *model(i)* that maps an activity instance into an activity model

# Process Instances and Process Models

How to define a process instance?

-> let us exploit the definition of an activity instance

Weske definition (revisited):

A process instance $i = (E_i, <_i)$ based on a process model XXX is defined by a partially ordered set of events $E_i$ such that:

- $E_i$ consist of events for all activity instances j such that model(j) is *allowed* by the process model
- $<_i$ is an event ordering in $E_i$ that satisfies the ordering of events:
    - In each activity instance
    - (In each gateway instance (???) )
    - The ordering of events between activities satisfies the execution constraints as defined by the process model

## Which Process Model?

# Three main aspects of Process Modelling

- **Control-flow**: captures the allowed orderings induced by the relevant dynamic constraints over the events that trace the execution of activity instances.
- **Data** that are produced, transferred and manipulated by the activity instances, in accordance to the constraints of a conceptual schema that models the domain.
- **Resources** and the organizational structure enable the execution of BPs, following specific allocation strategies and policies that define responsibilities and duties of the involved stakeholders.

# Data Modelling and Activities

Why data conceptual models (e.g., ER schema) are not enough?

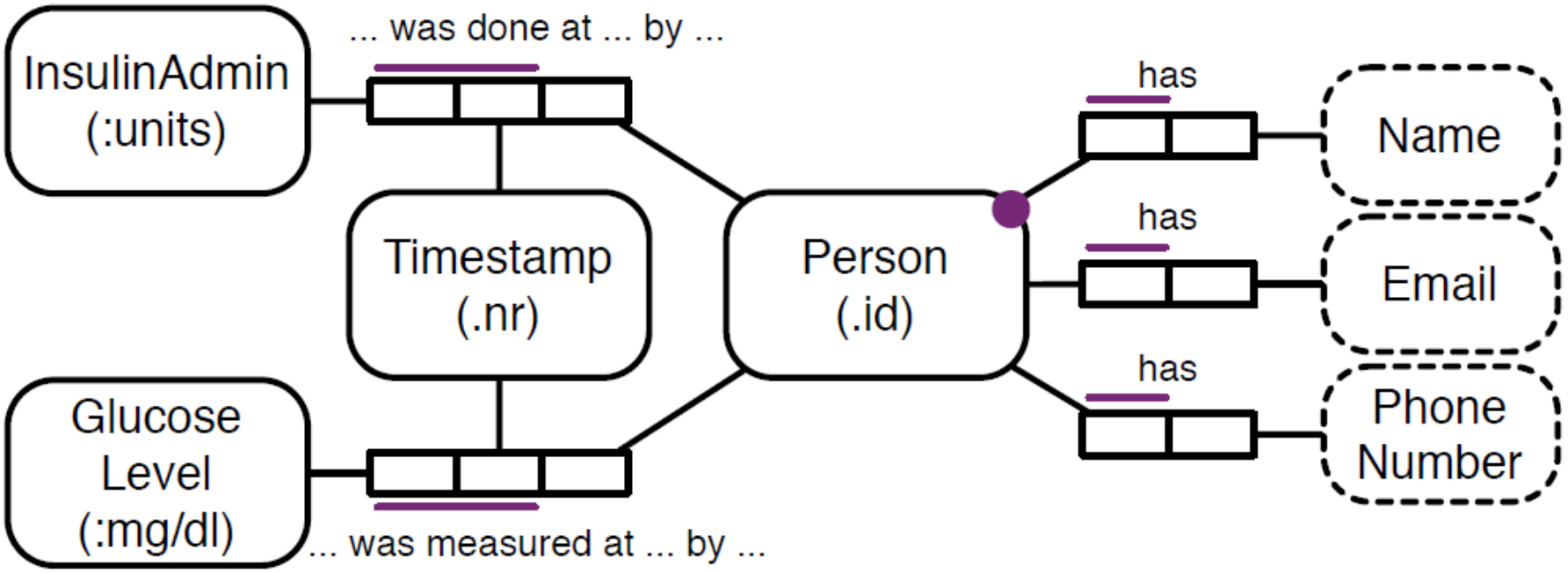
**Conceptual schemas** induce data dependencies among the activities of the BPs.

- The transfer of data among activities is called **data flow**.
- Data can be coupled with the process following different paradigms (**workflow data patterns**).

**Conceptual modeling** is also the first step towards data integration and reconciliation of heterogeneous information.

- This is particularly important in B2B scenarios and collaborative/inter-organizational BPs.

# Data Modelling and Activities




Definition of activity `Do glucose test`.

# Workflow Data Patterns

They are **recurrent paradigms** used to specify how BPs handle data.

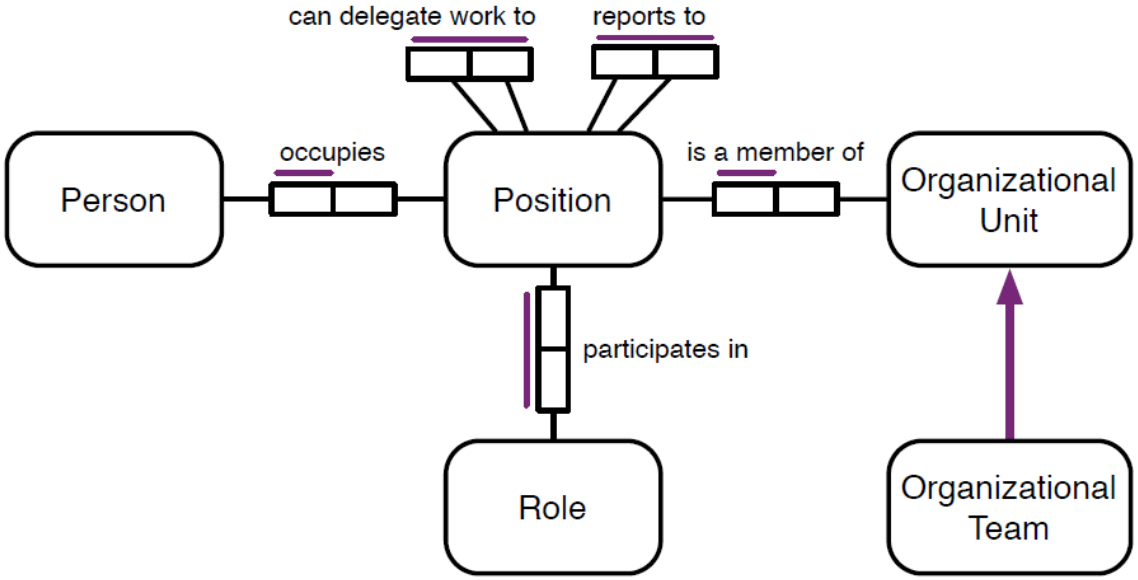
- **Data visibility**: definition of the scope of data in a BP.
  - Task data: data local to a specific activity.
  - Block data: data visible to all activities of a given sub-process.
  - Workflow data: data visible to all activities of a BP, with access determined by the BP model.
  - Environment data: data part of the execution environment in which the BP is enacted; they can be accessed at runtime.
- **Data interaction**: modalities of communication of data objects.
  - Communication between activities and activities/sub-processes of the same BP, communication between activities of different BPs (message passing), communication between the BP and the management system.
- **Data transfer**: how data objects are communicated.
  - Passing values vs passing references.
- **Data-based routing**: how data affect the enactment of a BP.
  - Enablement of an activity based on data, data-aware conditions and data-driven choices.

See http://workflowpatterns.com/patterns/data/

# Resource Modelling (Organizational modelling)

The work (execution of activities) must be coordinated inside a company.

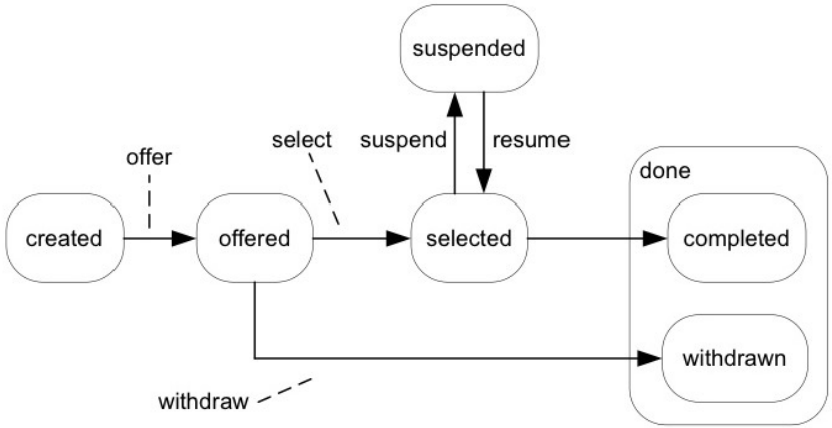
- Resources (humans, devices, trucks, warehouses) are entities that can perform work for the company.
- Persons are part of an organization, and work to fulfill the business goal of the company.
- Position is used to decouple the person and duties/privileges.



# Resource Modelling and Work Items

Organizational modelling captures the link between activity instances to be performed and knowledge workers that can perform it.

- An Activity Instance in ready state is ***offered*** as a work item to a set of workers.
- Selection is then done following specific *resource allocation mechanisms*.
- In human interaction workflows, work items have a transition diagram that evolves as a result of their manipulation by humans.




M. Weske: Business Process Management,
© Springer-Verlag Berlin Heidelberg 2007

# Workflow Resource patterns

They are **recurrent allocation strategies** of work items to resources: they model how to interconnect the organizational model and the activities of the BP.

- **Direct allocation**: all activity instances of an activity to a single, specific Person (an individual).
- **Role-based allocation**: activity bound to a role, all members of the role will be candidate for the execution of its instances. Role resolution is dynamically applied to determine the members of the role. Two possibilities: dynamic allocation to only one member of the role, or instance offered to all members, then selection + withdrawn of work items for the other members.
- **Deferred allocation**: Decision completely taken at runtime (made as an explicit step in the workflow).
- **Authorization Allocation**: based on the persons' positions.
- **Separation of duties**: imposes an exclusive constraint between the person selected in different allocations.
- **History-based**: Allocation based on what the persons worked on previously.

# …ok… but… what about control flow modelling?

A step back…

… when considering the flow aspects, at least two dimensions must be considered:

- **Procedural** vs. **Declarative** process modelling
- **Open** vs. **Closed** modelling

These two dimensions can be combined in any way… but… afaik, you can find only:

- Procedural closed process modelling approaches
- Declarative open process modelling approaches

# Process Modeling: Procedural vs Declarative

**Procedural** approaches focus on the order of steps composing the process

- based on concepts such as (**gates:**) **sequence**, **or** and **xor** choices, **parallel** execution, possibly (infinite?) **loops**, (**events:**) **start** and **end**
- they try to ensure general properties of process models, such for example termination, deadlocks, ...
- they implicitly impose **time constraints** between activity executions
- they allow to compute time properties such as maximum completion time
- mainly based on the **orchestration** perspective

- easy to deploy: once an action execution ("a piece of software") is assigned to an activity, the process can be automatically executed by a workflow engine (robotization of the process)
- historically, mild support to data and resources
- **Suffer the "spaghetti-like process problem"**

# The spaghetti-like processes…

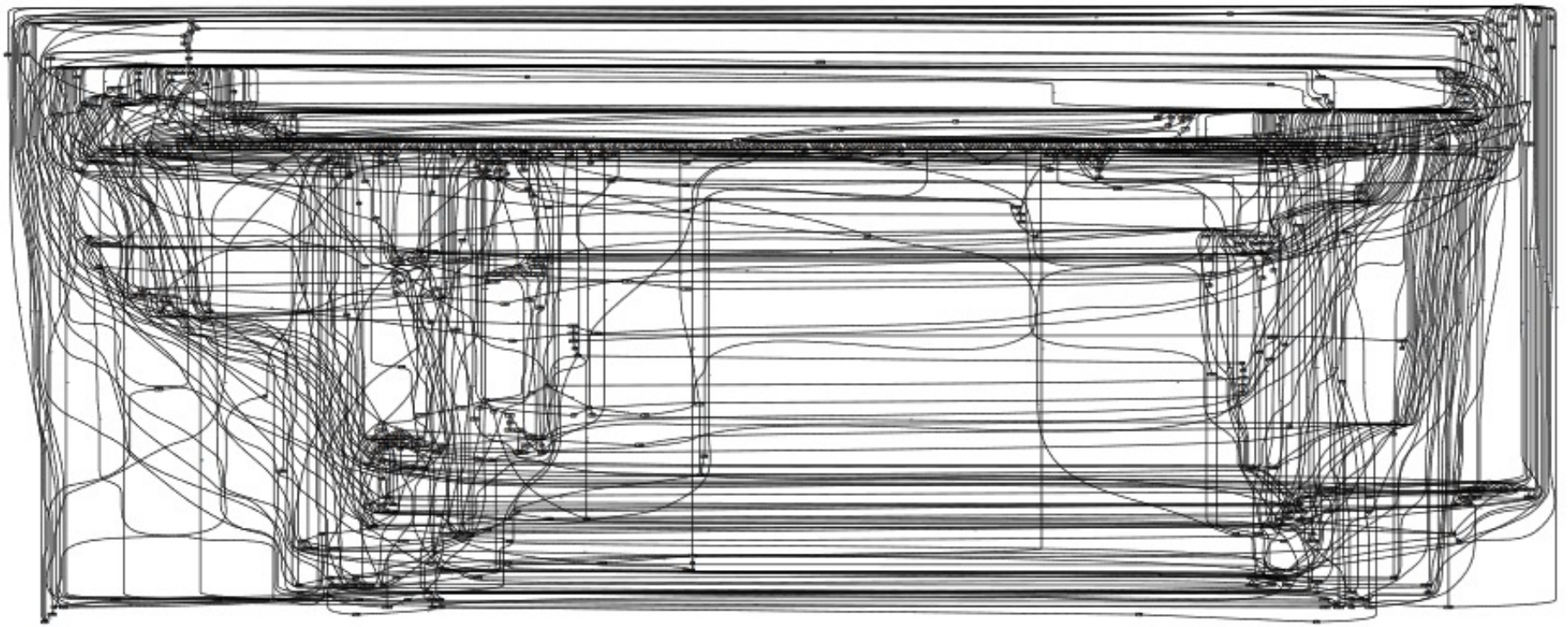


Fig. 4: Spaghetti process describing the diagnosis and treatment of 2765 patients in a Dutch hospital. The process model was constructed based on an event log containing 114,592 events. There are 619 different activities (taking event types into account) executed by 266 different individuals (doctors, nurses, etc.)

Wil M.P. van der Aalst. Process Mining: Discovering and Improving Spaghetti and Lasagna Processes. CIDM keynote
http://wwwis.win.tue.nl/~wvdaalst/publications/p615.pdf

# Process Modeling: Procedural vs Declarative

**Declarative** approaches focus on the properties that should hold along the process execution

- based on concepts such as **expected** executions, **prohibited** executions, and (**not expected AND not prohibited**) executions (three-value logic)
- implicitly support parallel execution, but sequences can be imposed as well
- they implicitly impose time constraints between activity executions

- guarantee the domain-related properties of the business process

- loose support for automatic execution
- more oriented to the monitoring of the process
- historically, mild support to data and resources
- easier to define (it's more rule-like...)

# Process Modeling: Procedural vs Declarative

Exercise: Model the following process

A customer enters a bar, she can have a coffee or a cappuccino, (but not both), a croissant, an orange juice, and she is expected to pay. After paying and consuming, the customer exits the bar.

Which activities?

- start (enter the bar)
- have a coffee
- have a cappuccino
- have an orange juice
- have a croissant
- pay
- end (exit the bar)

In which order these activities should be executed?

Further issue: what does it mean "... she **can** have ..."?

# Process Modeling: Open vs. Closed

**Closed** process models allows the execution of only the activities that are indeed envisaged by the process, at the right moment along the process execution (its sequence diagram)

- activities not envisaged are prohibited by default: if any non-envisaged activity should happen, the process instance is faulty
- envisaged activities happening out of the prescribed order will make the process faulty
- easy to implement/deploy
- Almost all business/implemented processes

**Open** process models clearly specify:

- which activities can be done
- which activities are prohibited (within specific time windows)
- nothing about the "other" activities ???

- practically, the "other" activities are defined any way as a finite set...

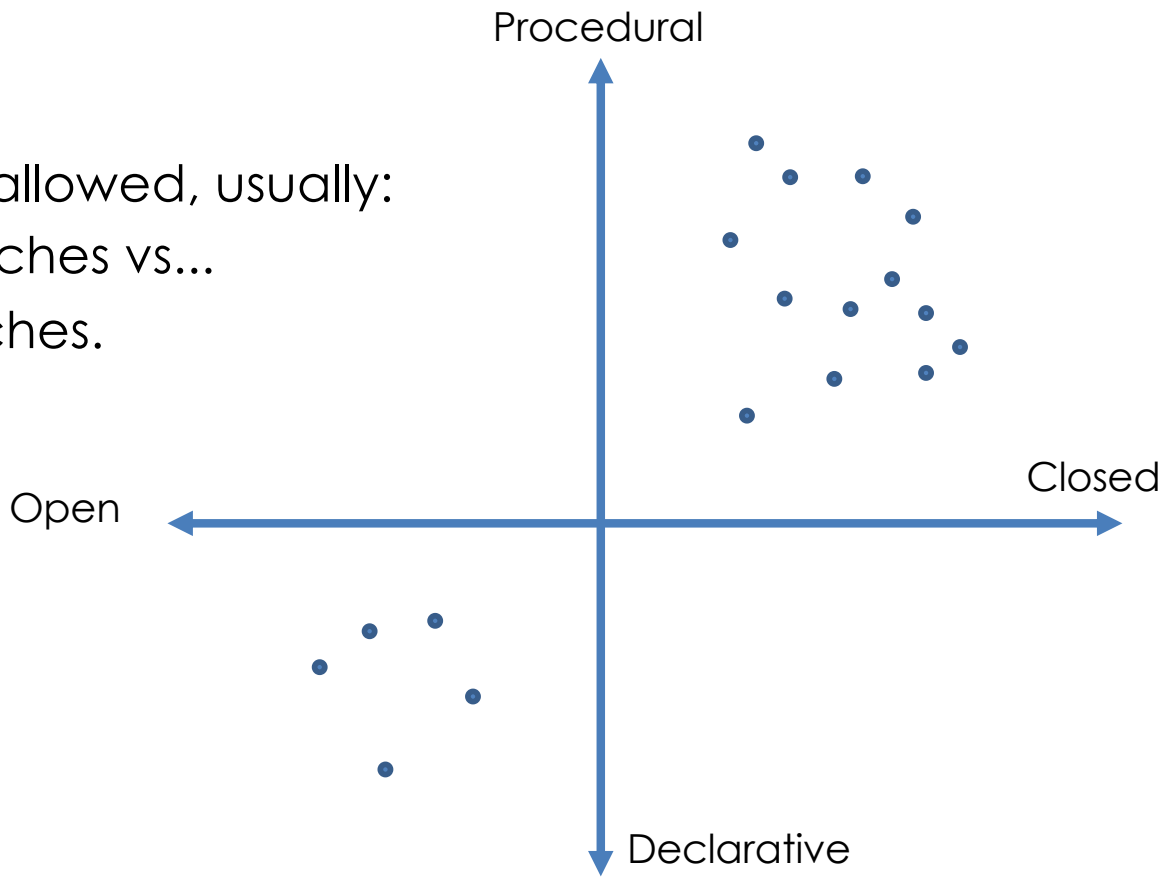
# Process Modeling

Two dimensions:

- Procedural vs. Declarative
- Open vs. Closed

Although any combination is allowed, usually:

- **Procedural, closed** approaches vs...
- **Declarative, open** approaches.



# Procedural, Closed Process Modelling

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

# Procedural, Closed Process Modelling

Reference metamodel:

- **Process Model**: a blue print for a set of process instances with a similar structure. Each process model consists of nodes and directed edges.
- Directed Edges: used to express the relationship between nodes. Usually, the direction of the edge is used to indicate the temporal ordering constraint that should be respected by the process instances (i.e., by the activity instances events)
- Node: a model of
    - **Activity Models**: Units of work; usually each activity model appears only once in a BP; they have exactly one incoming edge, and one outgoing edge.
    - **Event Models**: capture the occurrence of states relevant for a BP. E.g., a process instance starts and ends with two (distinct)events, thus a process model start and ends with event models.
    - **Gateway models**: used to express control flow constructs. Like activity models, *each gateway model has a start and an ending event*.

# Control flow patterns

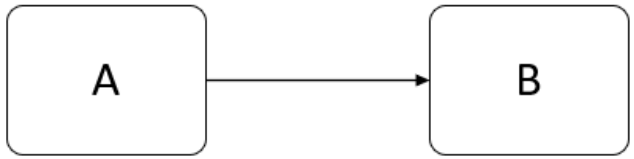
- A number of patterns for expressing the control flow aspects
- There are a set of basic patterns (**sequence**, **and split**, **and join**, **exclusive or split**, **exclusive or join**): all procedural closed process modelling approaches support these basic patterns
- Other patterns are built on top of the basic ones

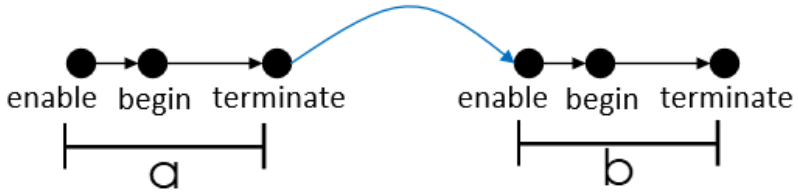
Which semantics?

- Although procedural closed process models capture the same control flows, they greatly differ for the underlying semantics
- A simple, intuitive semantics is the one based on the definition of activity instances as set of ordered events -> it is referred as "**execution semantics**"

# Control flow patterns - Sequence

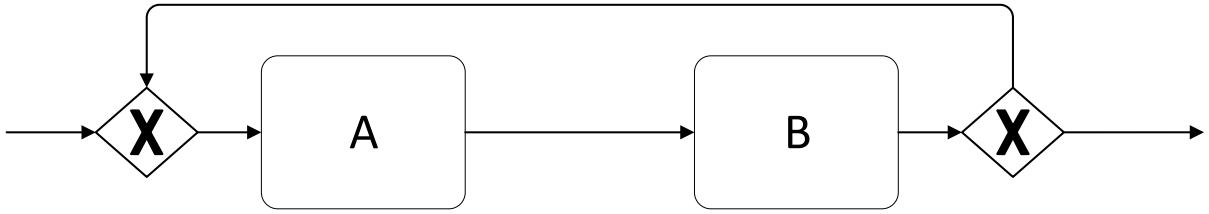
- A sequence pattern states that an activity instance *b* in a process instance *p* is enabled after the completion of an activity instance *a* in *p*.
- Activities model A and B are connected through a gate which type is *sequence*.

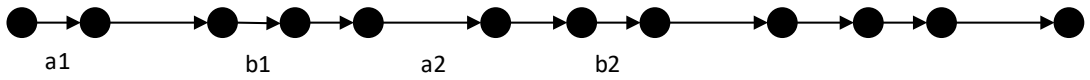


# Control flow patterns – Sequence in loops

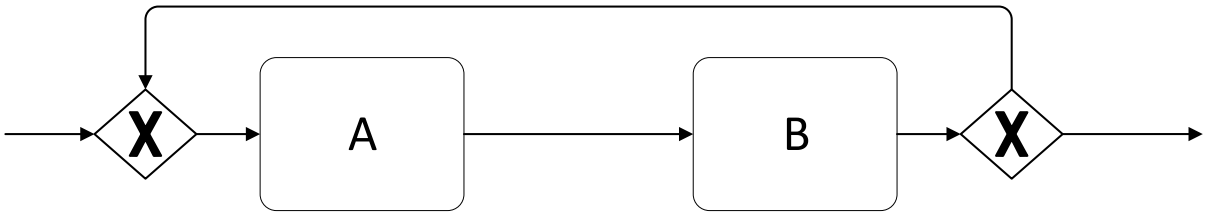
- The semantics is fine if the sequence does not appear in any loop… otherwise?
- In loops there can be multiple repetitions (multiple instances) of activity models A and B…
- Better semantic: for each termination event of *a* there is an enable event of *b* such that $t_a < e_b$.

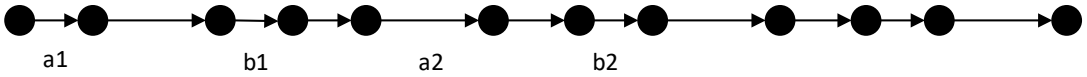



- This semantics is still problematic…

# Control flow patterns – Sequence in loops



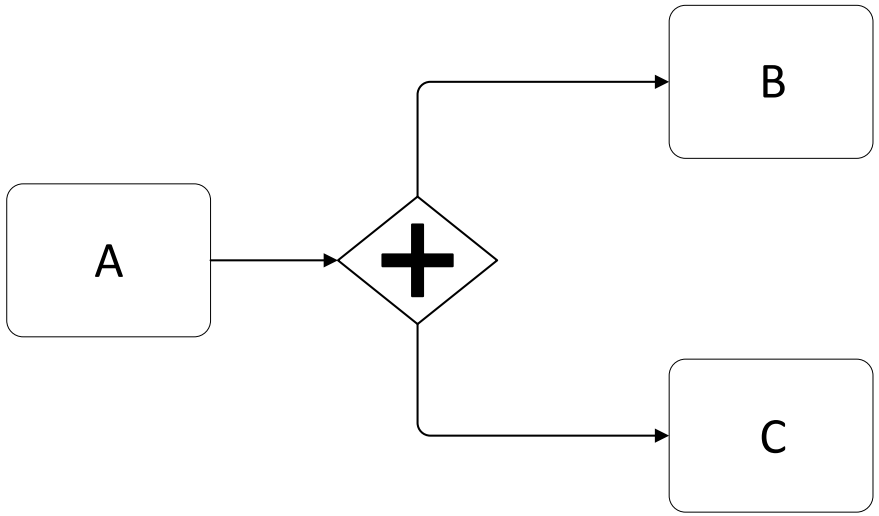

- For each termination event of *a* there is an enable event of *b* such that $t_a < e_b$.
- What about the sequence:

$$e_{a1}, t_{a1}, e_{a2}, t_{a2}, e_b, t_b$$

- Is it allowed by the model?
- What is the purpose of the semantics?
- For each termination event of *ai* there is an enable event of *bi* such that $t_{ai} < e_{bi}$, and no event $e_{aj}$ appears in between.

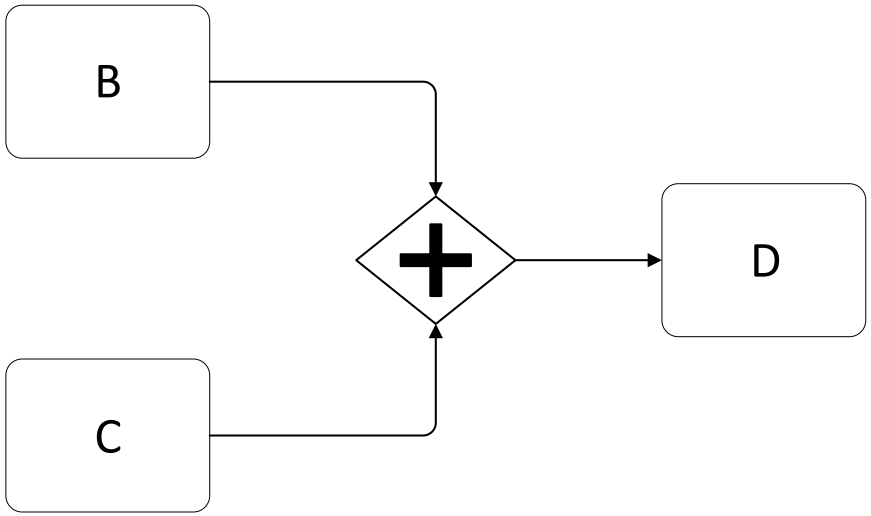
# Control flow patterns – And Split (parallel split)

- A parallel split gate in a process model indicates the point where the single thread of control splits into multiple threads of control, which are executed concurrently.
- Activity model A, and B, C are connected through a gate which type is *and-split*.



# Control flow patterns – And Join (parallel join)

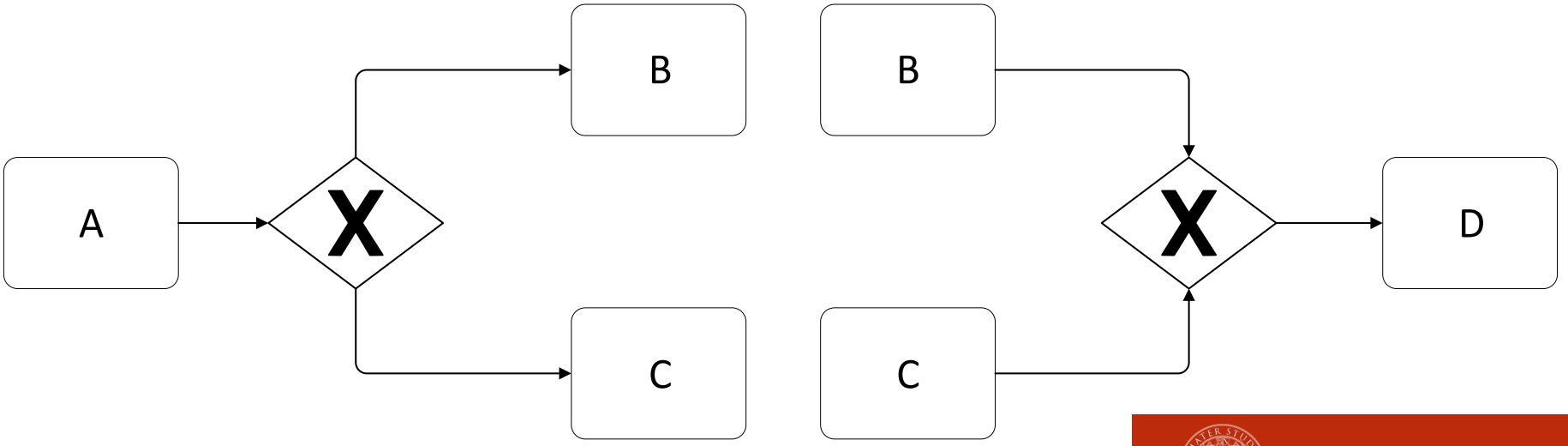
- A parallel join gate in a process model indicates the point where multiple concurrent threads converge into one single thread of control
- *It is an assumption that each incoming branch is executed exactly once… well structured processes?*
- Activity model B, C and D are connected through a gate which type is *and-join*.



# Control flow patterns – Xor split and Xor Join

- A xor split gate in a process model indicates the point where exactly **one** of several branches is chosen.

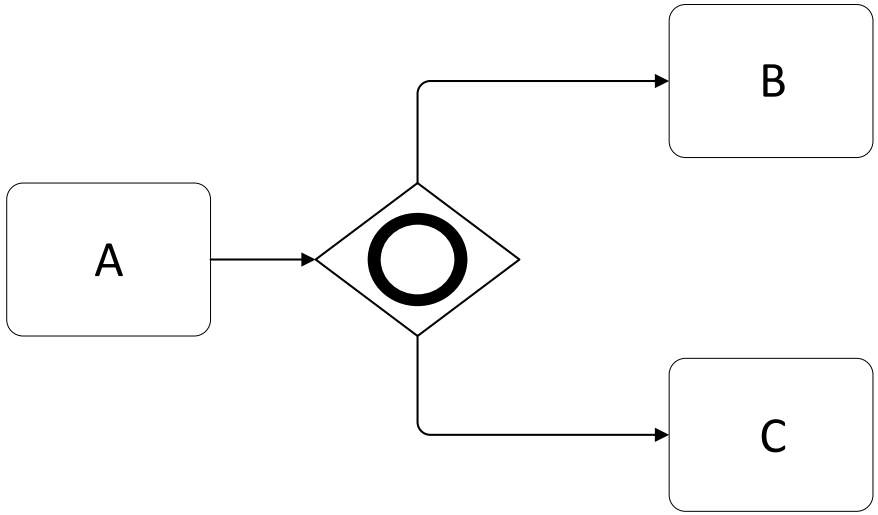
- A xor join gate is a point where two or more alternative threads come together without synchronization.
- It is an assumption that exactly one of the alternative branches is executed… well structured processes?



# Control flow patterns – Or split

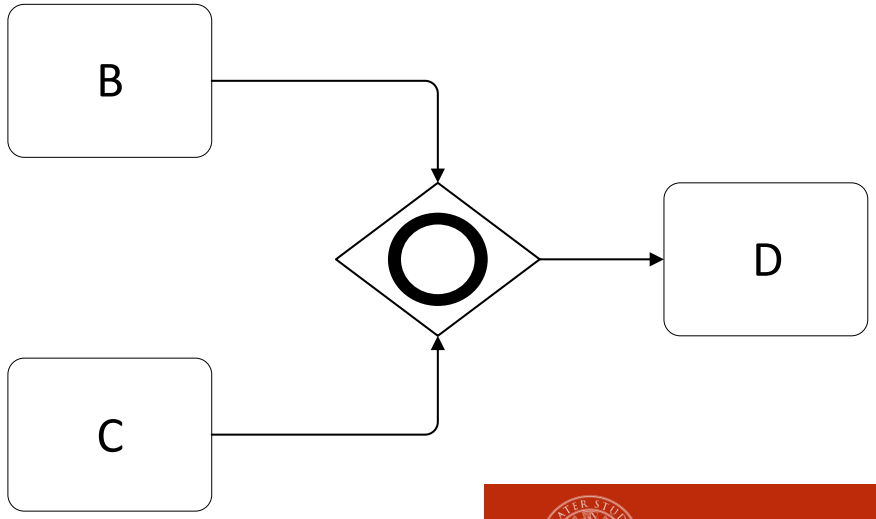
- A Or split gate in a process model indicates the point where **at least one** of several branches is chosen.

- Often, the equivalent semantics in disjunction of conjunctions is preferred, because of minimality



# Control flow patterns – Or Join

- An Or Join gate in a process model indicates the point where **multiple threads of control converge into one single thread**.
- Assumption: a branch that has already been activated cannot be activated again while the merge is still waiting for other branches to complete (well-structured processes?)
- It forces a synchronization point between the activated branches…
- … BUT … which branches have been activated?



# Control flow patterns – Or Join

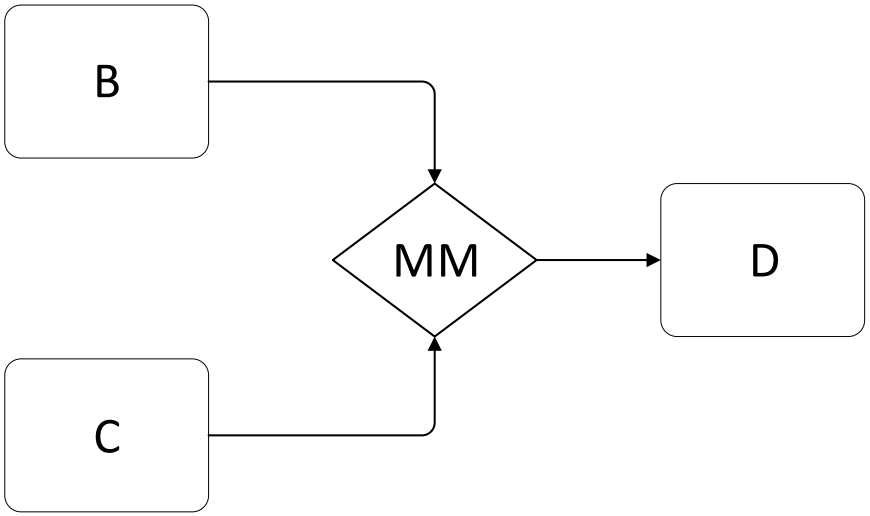
- The Or join gate **cannot be decided locally**…
- Two possible behaviors:
    - **wait** until all the activated branches terminate
      how long shall we wait?
    - **trigger** as soon as the first branch terminates
      shall we trigger at the first termination, at the last termination, or many times (once for each enabled branch)?
- In the case of wait, the entire process might end up in deadlocks…

- Often, the equivalent semantics in disjunction of conjunctions is preferred, because of minimality and clearer semantics

- Is it a common situation? E.g., final evaluation of this course…

# Control flow patterns – Multiple Merge

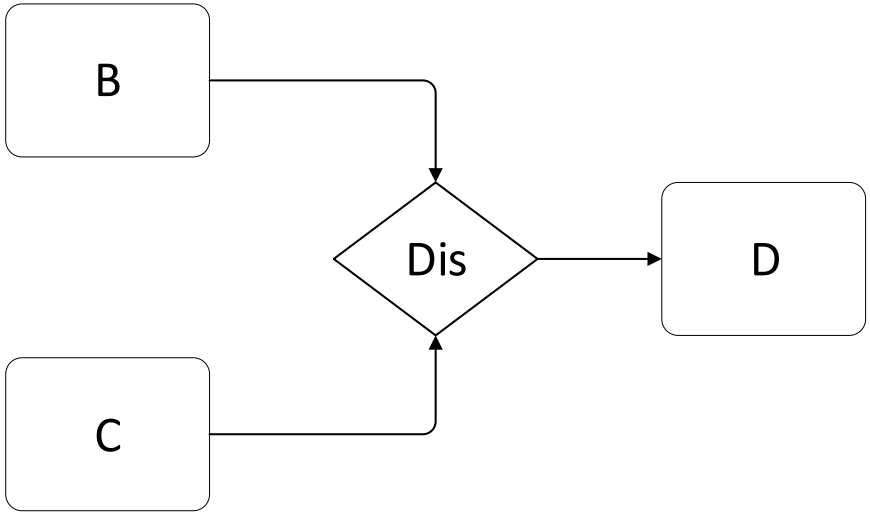
- A Multiple Merge gate in a process model indicates the point where multiple concurrent threads (two or more) of control join without synchronization.
- I.e., the activity following the gate is enabled for every activation of every incoming branch.




- Multiple Merge gates span off new threads of control
- Every new thread needs an id, otherwise it would not be possible to correctly synchronize different threads on the same flow paths.

# Control flow patterns – Discriminator

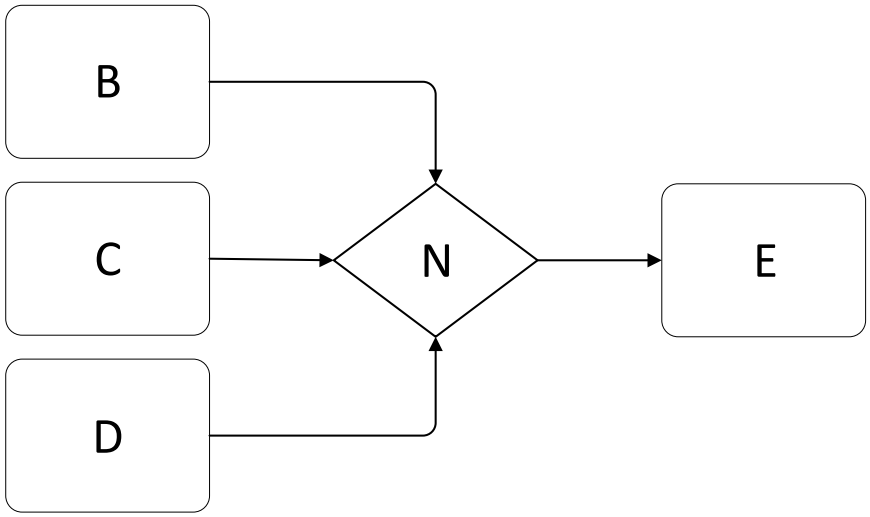
- A Discriminator gate in a process model waits for one of the incoming branches to complete, and then triggers the subsequent activity.
- Then it waits for the other branches to terminate, and ignores them (i.e., their termination does not have any other effect)
- Only when all the incoming branches have terminated, it resets and can be enabled again -> loop safeness…




- Again, also this pattern suffer of the non-locality problem…
- … how does it know which branches have been activated?
- However, at least the triggering behavior is defined…

# Control flow patterns – N-out-of-M Join

- A N-out-of-M join has a number M of incoming paths.
- It triggers when N (<=M) paths have terminated.
- The M-N activities can complete unharmed, but their completion does not trigger anything
- When M-N activities terminate, it reset itself (loop safeness condition)

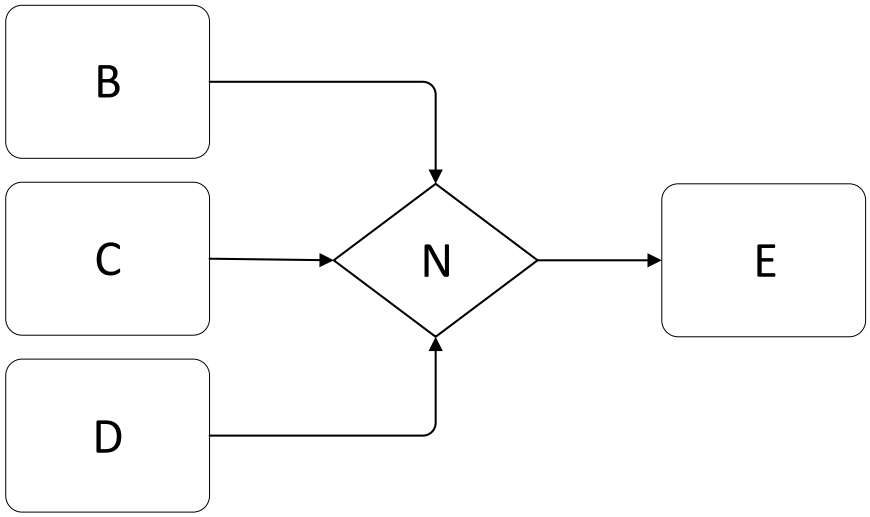



- Again, also this pattern suffer of the non-locality problem…
- … how does it know how many and which branches have been activated?
- What if less than N branches have been enabled?

# Control flow patterns – N-out-of-M Join

- If N==M, it becomes an and join
- If N==1, it implements the discriminator

- It is a fairly commonly used pattern

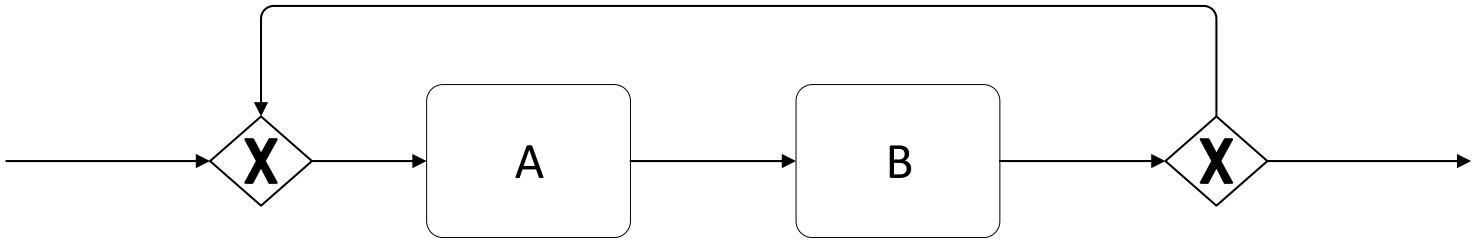



- N can be decided:
  - Statically
  - Dynamically
  - Deadline

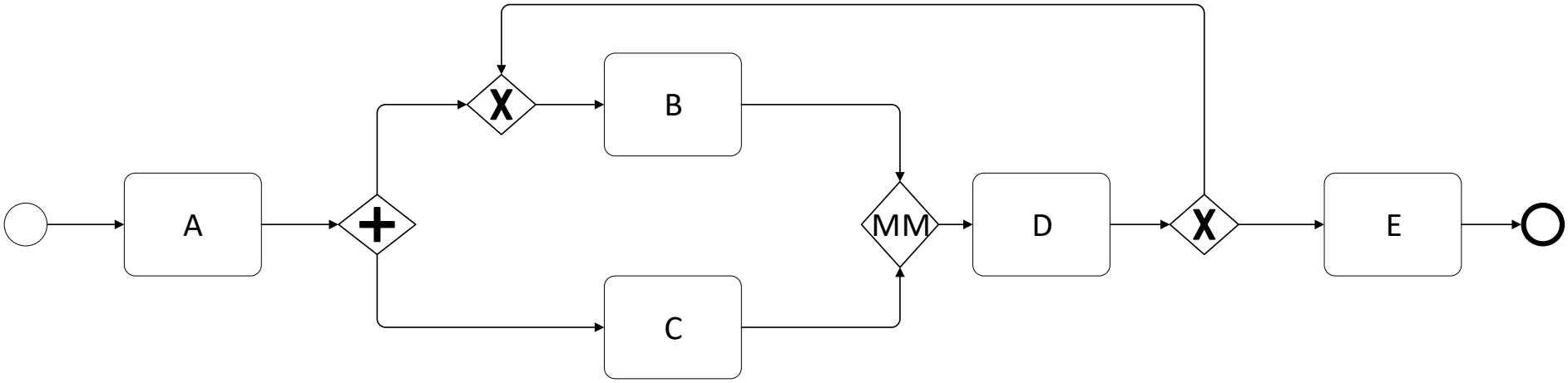
Example: How do we usually coordinate for lunch?

# Control flow patterns – Arbitrary cycles

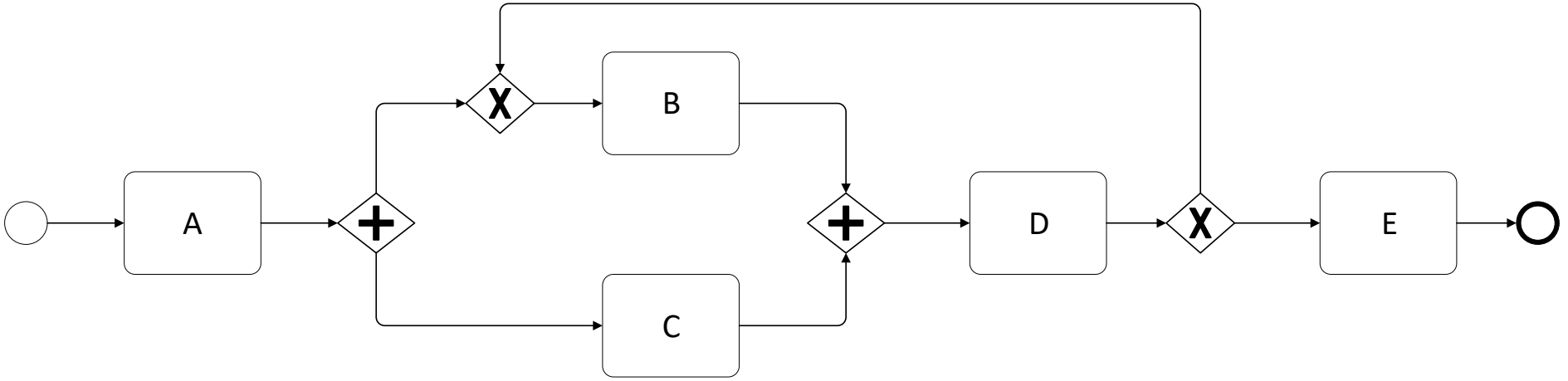
- Repetition of one or more activities
- Usually done through xorsplit/xorjoin gates used in combination



# Control flow patterns – Examples



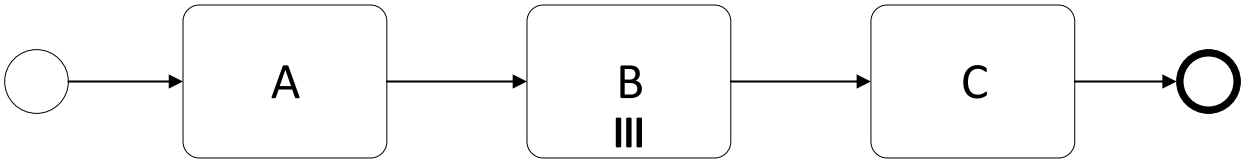

- How many times does this process terminate?




- What's wrong?

# Control flow patterns – Multiple Instances

- Multiple Instances without synchronization
- Multiple Instances wit a priori design time knowledge
- Multiple Instances wit a priori run time knowledge
- Multiple Instances without a priori run time knowledge




- Example: for each item in the Amazon order, wrap it and then send it.

# How many control patterns?

Resource: http://www.workflowpatterns.com

**Basic Control Flow Patterns**

Sequence
Parallel Split
Synchronization
Exclusive Choice
Simple Merge

**Advanced Branching and Synchronization Patterns**

Multi-Choice
Structured Synchronizing Merge
Multi-Merge
Structured Discriminator
Blocking Discriminator
Cancelling Discriminator
Structured Partial Join
Blocking Partial Join
Cancelling Partial Join
Generalised AND-Join
Local Synchronizing Merge
General Synchronizing Merge
Thread Merge
Thread Split

**Multiple Instance Patterns**

Multiple Instances without Synchronization
Multiple Instances with a Priori Design-Time Knowledge
Multiple Instances with a Priori Run-Time Knowledge
Multiple Instances without a Priori Run-Time Knowledge
Static Partial Join for Multiple Instances
Cancelling Partial Join for Multiple Instances
Dynamic Partial Join for Multiple Instances

**State-based Patterns**

Deferred Choice
Interleaved Parallel Routing
Milestone
Critical Section
Interleaved Routing

**Cancellation and Force Completion Patterns**

Cancel Task
Cancel Case
Cancel Region
Cancel Multiple Instance Activity
Complete Multiple Instance Activity

**Iteration Patterns**

**Arbitrary**

Cycles
Structured Loop
Recursion

**Termination Patterns**

Implicit Termination
Explicit Termination

**Trigger Patterns** Transient Trigger
Persistent Trigger

# Summing up

Flows can become very strange…

- Need for a formal semantics: the presented execution semantics is easy to understand, but can lead to ambiguities
- Workflow patterns were born from real business cases: a number of very difficult situations can come up
- Some sort of "structure" property is desirable… shall we loose expressivity?

- Flows might incur in deadlocks
- In general, we might want to check for a number of **formal properties**
- This would be possible only in presence of formal semantics

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

**Federico Chesani**

DISI – Department of Computer Science and Engineering

federico.chesani@unibo.it

www.unibo.it