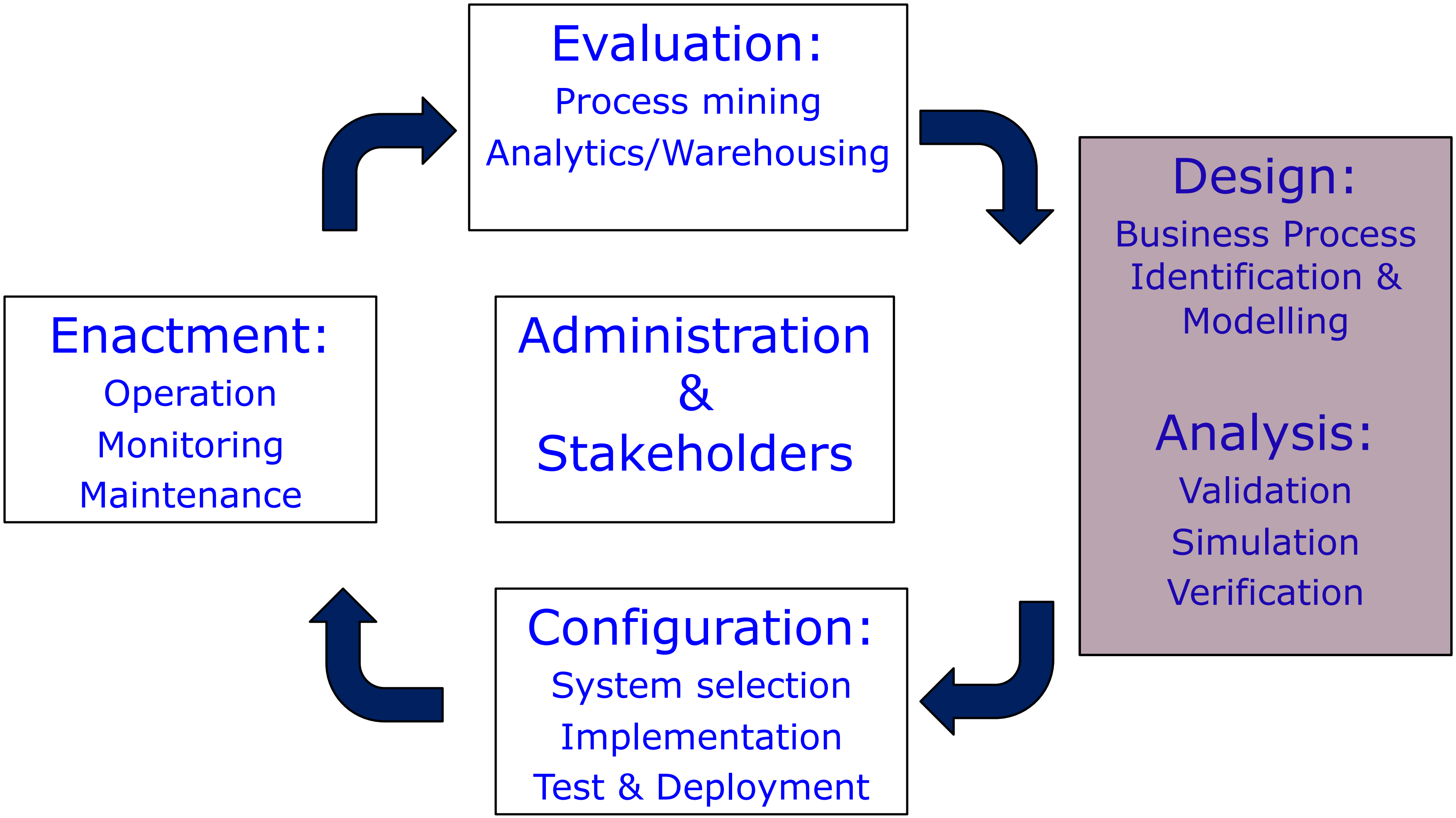
Lecture 2

Business Process Modelling

BUSINESS PROCESS LIFECYCLE



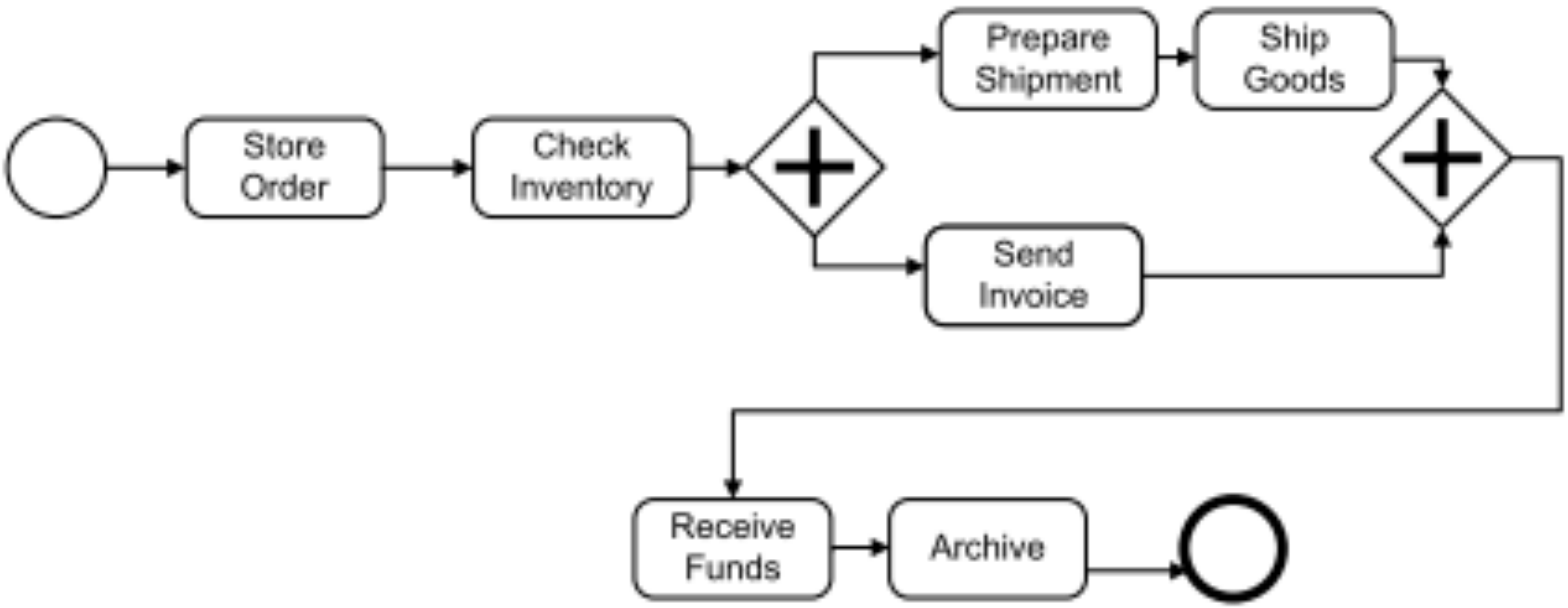
BUSINESS PROCESS MODELLING

- The model of a BP can be described via a specific language and be specified through BP modelling tools (part of a BPMS)
- Can be performed at different levels:
  - Organizational (coarse-grained, textual forms)
  - Operational (fine-grained, semi-structured/formal models)
- Explicit representation:
  - Usually through graphical notations
  - Ideal for internal communication between stakeholders -> flexibility

BUSINESS PROCESS MODELLING

- Various types of languages have been proposed:
  - Procedural: The procedural aspects determine the order of steps (tasks, events, and gateways) that are needed to achieve the relevant process goals. This is comparable with algorithms or operating instructions (Petri Nets, BPMN, BPEL)
  - Declarative: processes are based on declarative elements like complex decisions, relationships between variables, or data constraints. This information is expressed using business rules and some kind of functional or logical language.

PROCEDURAL BUSINESS PROCESS MODELLING

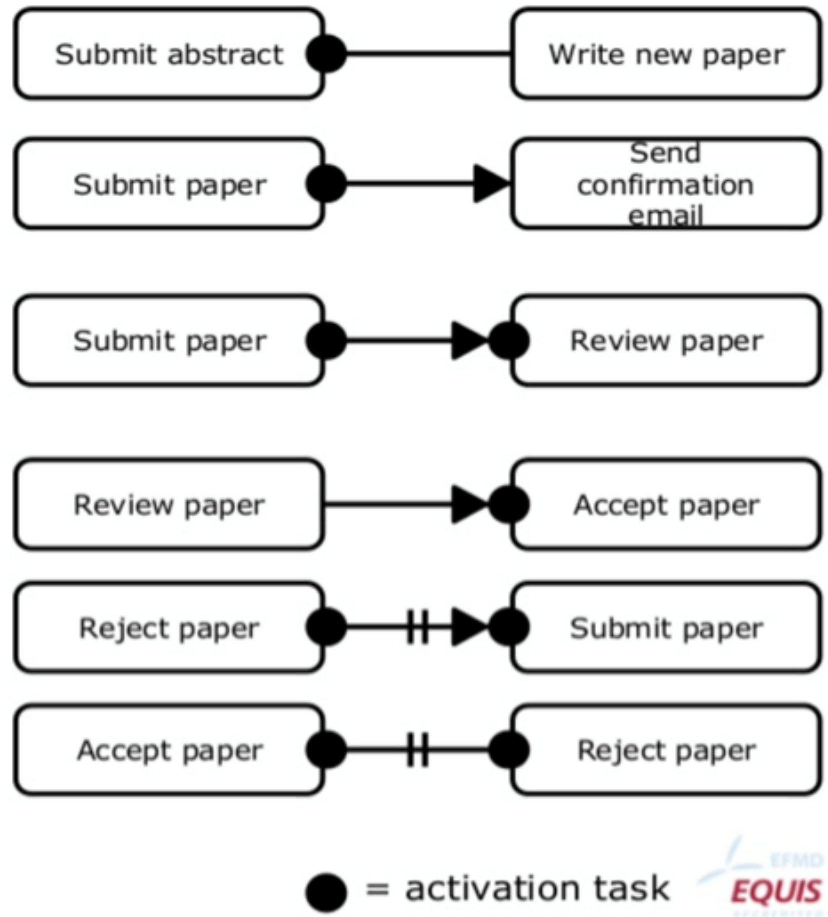


DECLARATIVE BUSINESS PROCESS MODELLING

- If an abstract is submitted, a new paper had been written or will be written
  Responded existence(Submit abstract, Write new paper)
- After the paper submission, a confirmation email is sent
  Response(Submit paper, Send confirmation email)
- After the paper submission, the paper will be reviewed;
  there can be no review without a preceding submission
  Succession(Submit paper, Review paper)
- A paper can be accepted only after it has been reviewed
  Precedence(Review paper, Accept paper)
- After the rejection, no further submission follows
  Not succession(Reject paper, Submit paper)
- Paper cannot be both accepted and rejected
  Not co-existence(Accept paper, Reject paper)

Template Tasks

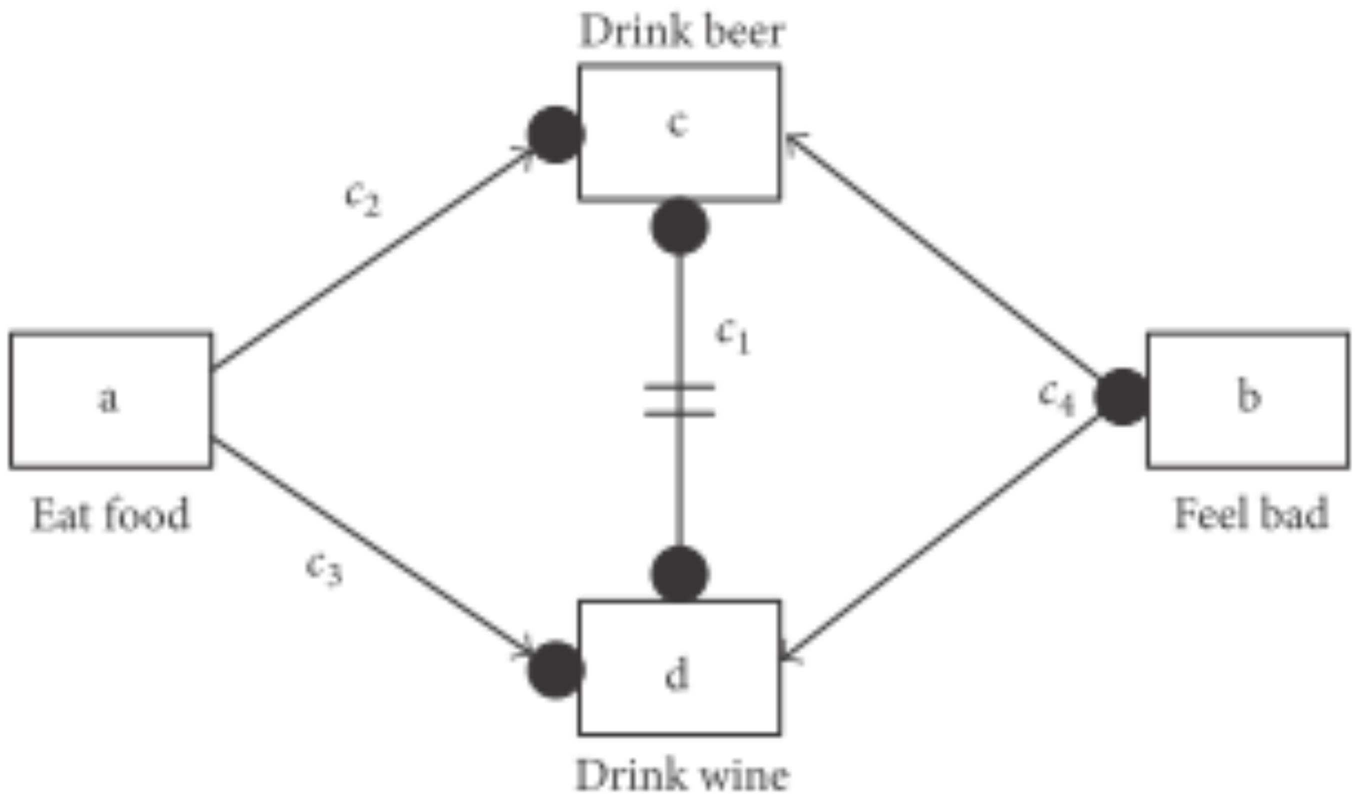
SEITE 6



EQUIS

Source: http://www.slideshare.net/cdc08x/semantical-vacuity-detection-in-declarative-process-mining

DECLARATIVE BUSINESS PROCESS MODELLING



DECLARATIVE BUSINESS PROCESS MODELLING

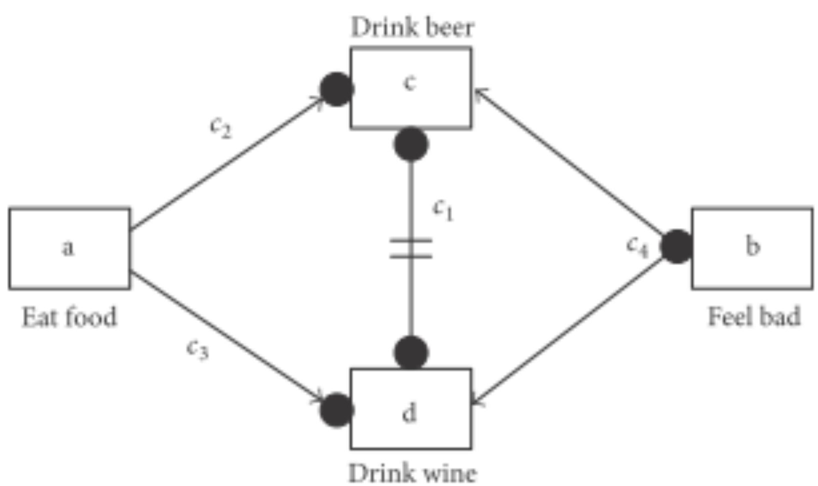
- Declare is one declarative language
  - Grounded in Linear Temporal Logic (LTL)
  - Finite-trace semantics
  - Each constraint is mapped to an LTL formula using operators such as: always $\square$, eventually $\Diamond$, until $\sqcup$, weak until W & next time $\bigcirc$

$$\varphi ::= \mathrm{true} \mid a \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \bigcirc\,\varphi \mid \varphi_1 \,\mathsf{U}\, \varphi_2$$

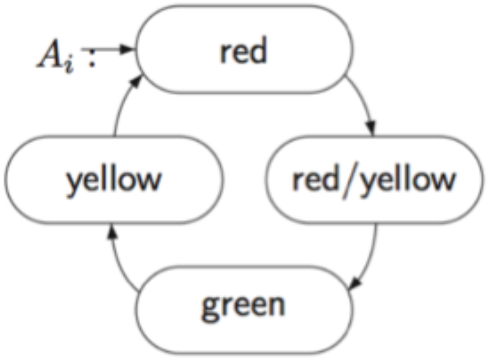
- a = atomic proposition
- $\bigcirc$ = “next”: ϕ is true at next step
- U = “until”: ϕ2 is true at some point, ϕ1 is true until that time

LTL: RESTAURANT EXAMPLE

- c1 $\equiv \neg((\Diamond c) \wedge (\Diamond d))$: indicates that tasks c and d cannot be true for the same case
- c2 $\equiv (\neg c) W a$, c3 $\equiv (\neg d) W a$: second task cannot happen before first occurs (but only first can just occur or no task of the two)
- c4 $\equiv$ "$\Box(b \Rightarrow (\Diamond c \vee \Diamond d))$: every occurrence of b should be followed by c or d (but not always one-to-one correspondence – b can occur multiple times)



LTL: TRAFFIC LIGHT EXAMPLE



- **System description**
  - Focus on lights in on particular direction
  - Light can be any of three colors: green, yellow, red
  - Atomic propositions = light color
- **Ordering specifications**
  - Liveness: "traffic light is green infinitely often"

$$\Box\Diamond \text{green}$$

- **Chronological ordering**: "once red, the light cannot become green immediately"

$$\Box\,(\text{red} \rightarrow \neg \bigcirc \text{green})$$

- **More detailed**: "once red, the light always becomes green eventually after being yellow for some time"

$$\Box(\text{red} \rightarrow (\Diamond\, \text{green} \wedge (\neg\, \text{green}\ \mathsf{U}\ \text{yellow})))$$

$$\Box(\text{red} \rightarrow \bigcirc\, (\text{red}\ \mathsf{U}\ (\text{yellow} \wedge \bigcirc\, (\text{yellow}\ \mathsf{U}\ \text{green}))))$$

FORMAL BUSINESS PROCESS LANGUAGES

- Have unambiguous semantics
- Allow for BP analysis
- Require some expertise in Mathematics or Computer Science (wrt the formalism used)
- Abstract from implementation details
- Different formalisms have been employed:
  - Markov Chains, Queuing Networks, Turing Machines, Transition Systems, Petri Nets, Temporal Logic and Process Algebras

CONCEPTUAL BUSINESS PROCESS LANGUAGES

- More comprehensive & easy to use
- Do not have well-defined semantics
- Do not allow for analysis
- The respective specifications cannot be executed
- Approximate description of the desired behaviour
- Examples: Business Process Modelling Notation (BPMN), Event-Driven Process Chains (EPCs), UML Activity diagrams

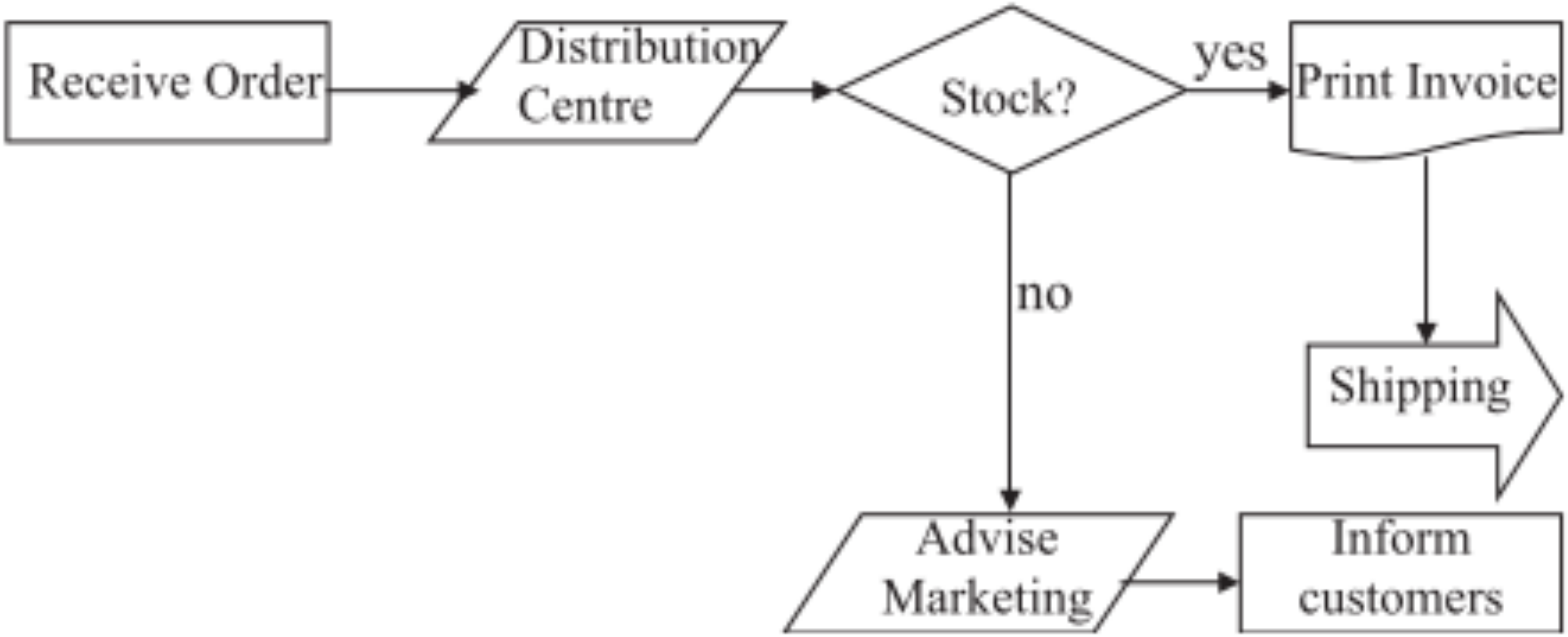
BUSINESS PROCESS EXECUTION LANGUAGES

- Workflow-based languages
- Provide the appropriate level of detail for making specifications executable
- Precise definition of the desired behaviour
- Example: BPEL

FLOW CHARTS

- Formalised graphic representation of a program/work logic sequence
- Sequential flow of actions with no activity breakdown
- Characteristics:
  - Flexibility (various ways for process description)
  - Easy to use – perfect for communication
- Drawbacks:
  - Process boundaries may not be clear
  - Tend to be very big
  - No difference between main & sub-activities
  - Hard to navigate (no sub-layers)

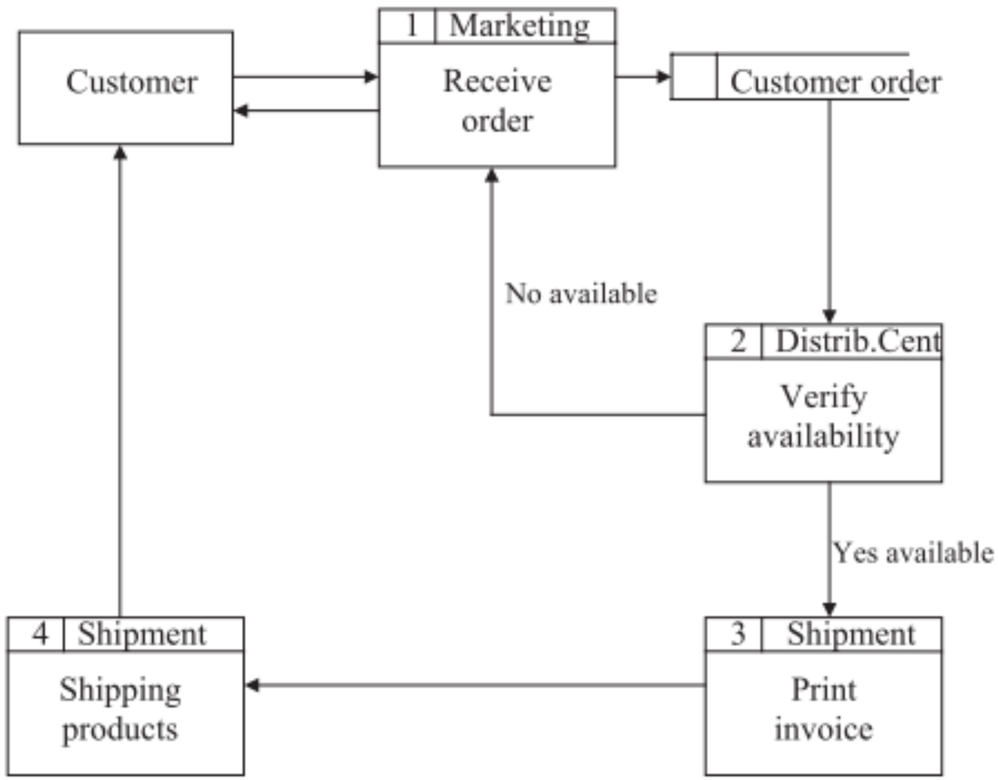
FLOW CHART EXAMPLE



DATA FLOW DIAGRAMS (DFDS)

- Show flow of data/information from one place to another one
- Link processes to data stores & indicate their relation to users and outside world
- Describe what the process will do but not how
- Used for structured analysis
- Characteristics:
  - Comprehensible, verifiable, easy to draw & amend, process breakdown
- Drawbacks: only flow of data is represented

DFD EXAMPLE

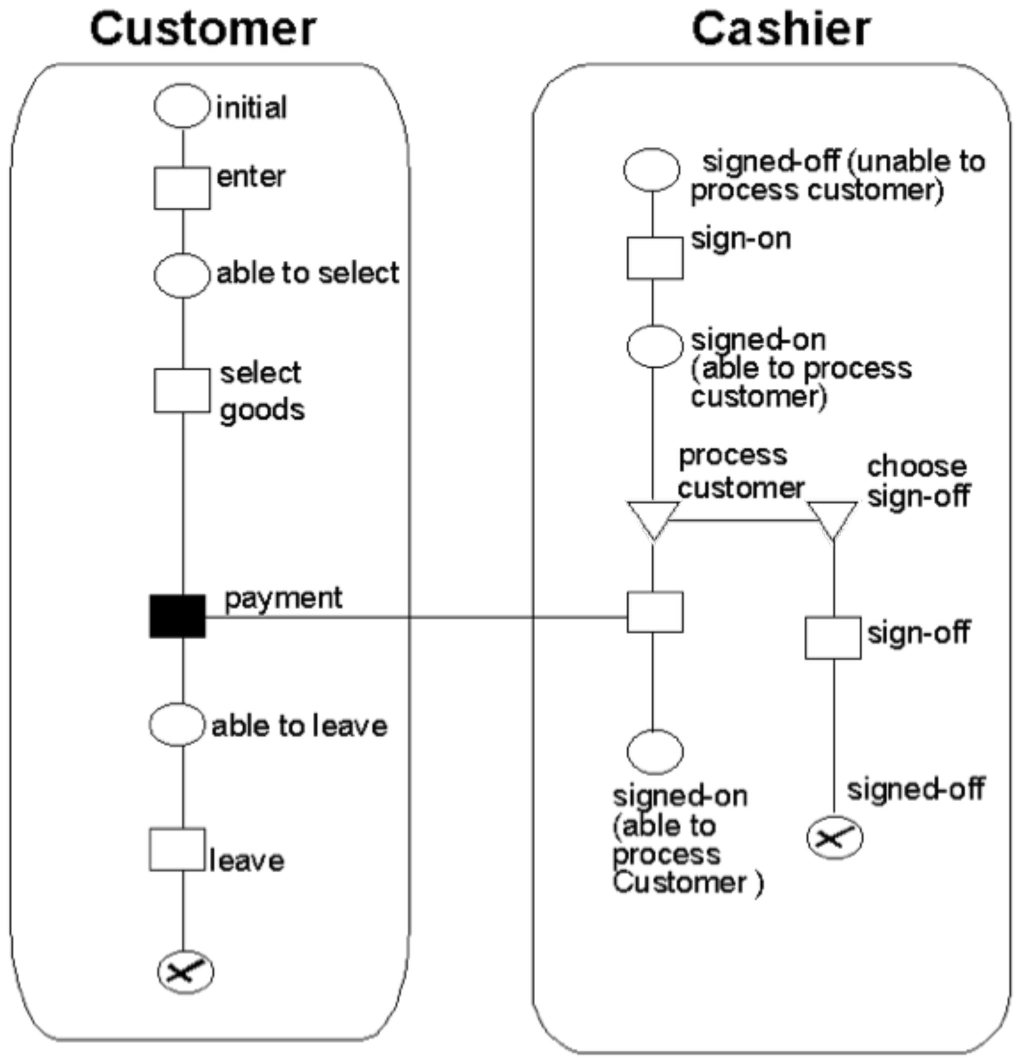


Customer
1
Marketing
Receive order
Customer order
No available
2
Distrib.Cent
Verify availability
Yes available
4
Shipment
Shipping products
3
Shipment
Print invoice

ROLE ACTIVITY DIAGRAMS (RADS)

- Graphic view of process from perspective of roles
- Focus on roles' responsibility & their interactions
- Roles include organisational functions, sw systems, customers & suppliers
- Characteristics:
  - Useful in communication
  - Easy & intuitive to use and understand
  - Detailed process view
  - Activity parallelization
- Disadvantages:
  - Business objects exclusion
  - No process decomposition

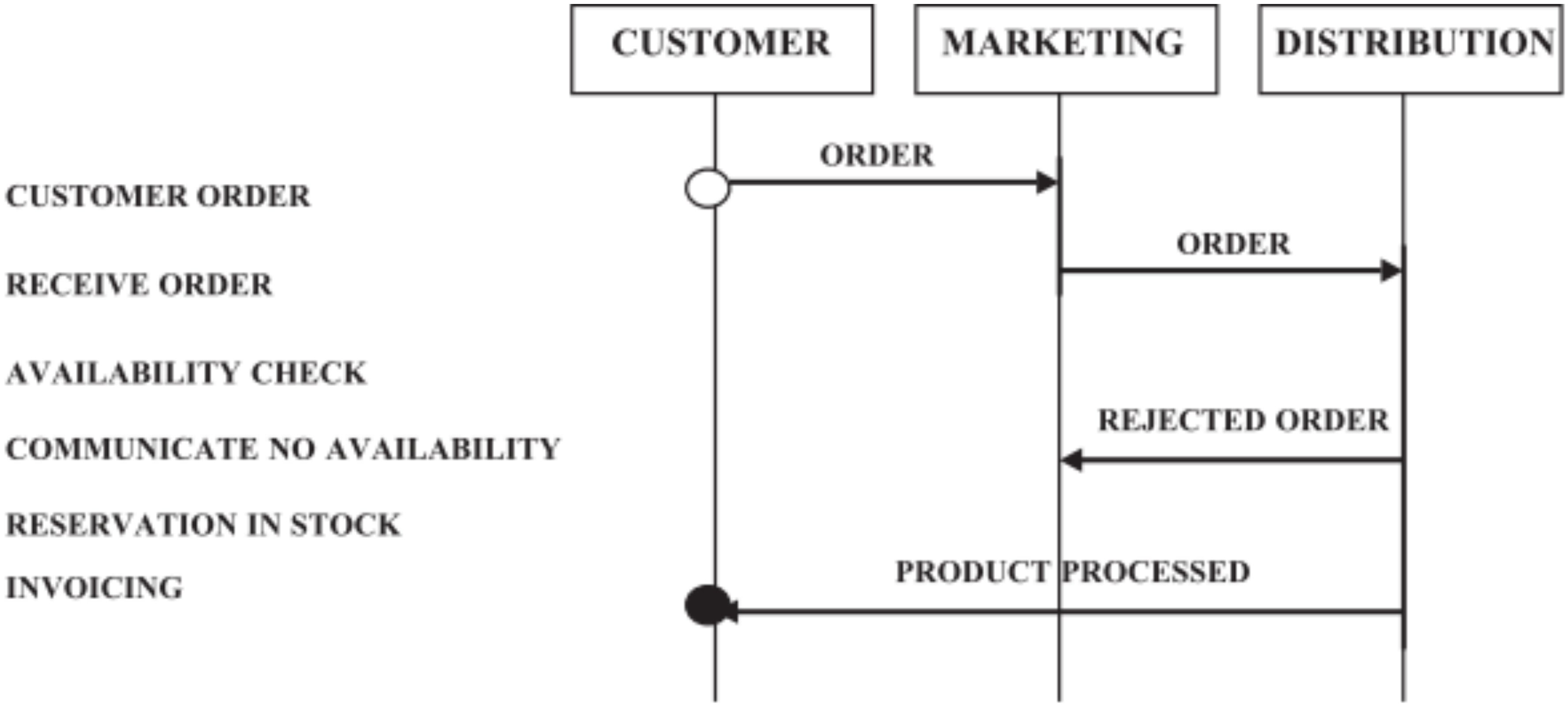
ROLE ACTIVITY DIAGRAMS (RADS)



ROLE INTERACTION DIAGRAMS (RIDS)

- Resulted from combination of RADs & object interaction diagrams
- Matrix used to connect activities with roles
- Horizontal lines indicate human/role interactions
- Characteristics:
  - Intuitive to understand
  - Easy to use
  - Well-definition of responsibilities
  - Activity breakdown
- Drawbacks:
  - Tend to be messy, hard to build & update, no I/O modelling

RID EXAMPLE


CUSTOMER
MARKETING
DISTRIBUTION
ORDER
CUSTOMER ORDER
ORDER
RECEIVE ORDER
AVAILABILITY CHECK
REJECTED ORDER
COMMUNICATE NO AVAILABILITY
RESERVATION IN STOCK
PRODUCT PROCESSED
INVOICING

UNIFIED MODELLING LANGUAGE (UML) DIAGRAMS

- Object-oriented methods used for modelling
- Collection of engineering practices proven successful for large & complex system modelling
- Covers both conceptual (BPs & system functions) & concrete elements (programming language classes, DB schemas, sw components)
- UML diagrams:
  - Class diagram: system structure (concepts & relations)
  - Statechart diagram: states of a class or system
  - Activity diagram: activities and actions
  - Sequence diagram: messages sent between set of objects
  - Collaboration diagram: complete collaboration between objects

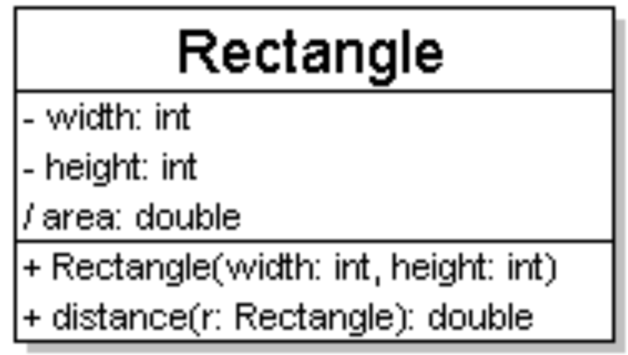
UML CLASS DIAGRAMS

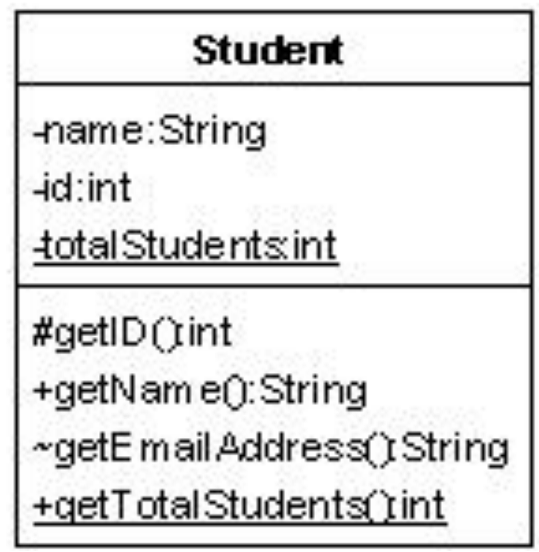
- **UML class diagram**: a picture of the classes in an OO system, their fields and methods, and connections between the classes that interact or inherit from each other

- details of how the classes interact with each other
- algorithmic details; how a particular behavior is implemented

CLASS DIAGRAMS

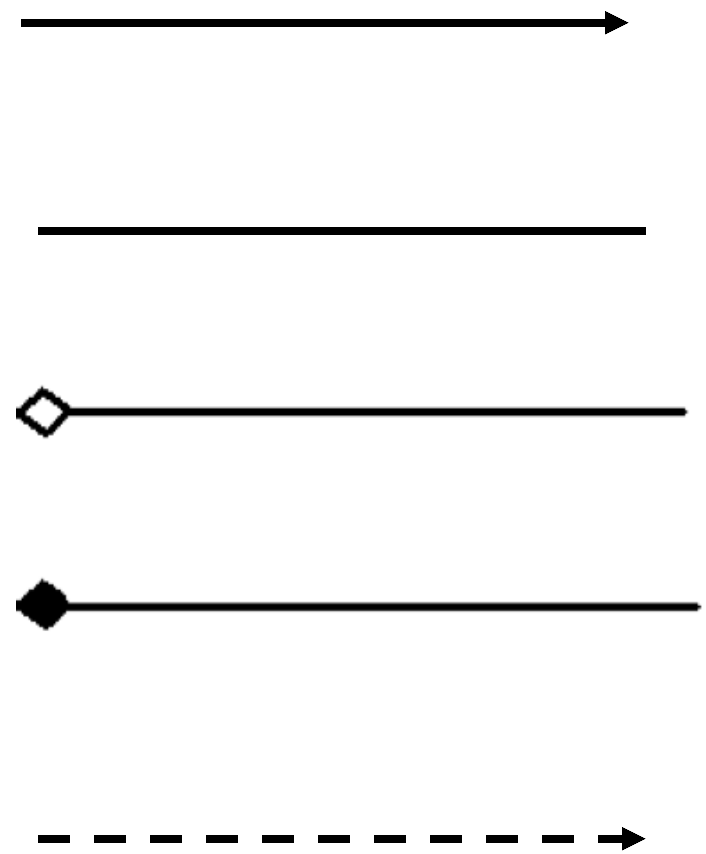
- class name in top of box
  - write <<interface>> on top of interfaces' names
  - use *italics* for an *abstract class* name
- attributes (optional)
  - should include all fields of the object
- operations / methods (optional)
  - may omit trivial (get/set) methods
    - but don't omit any methods from an interface!
  - should not include inherited methods





RELATIONSHIPS BETWEEN CLASSES

- **generalization**: an inheritance relationship
  - inheritance between classes
  - interface implementation
- **association**: a usage relationship
  - dependency
  - aggregation
  - Composition
- **aggregation**: "is part of"
  - symbolized by a clear white diamond
- **composition**: "is entirely made of"
  - stronger version of aggregation
  - the parts live and die with the whole
  - symbolized by a black diamond
- **dependency**: "uses temporarily"
  - symbolized by dotted line
  - often is an implementation detail, not an intrinsic part of
    that object's state



SEQUENCE DIAGRAMS

A <u>sequence diagram</u> depicts a scenario by showing the interactions among a set of objects in *temporal order*.

Objects (not classes!) are shown as *vertical bars*. *Events* or message dispatches are shown as horizontal (or slanted) *arrows* from the sender to the receiver.

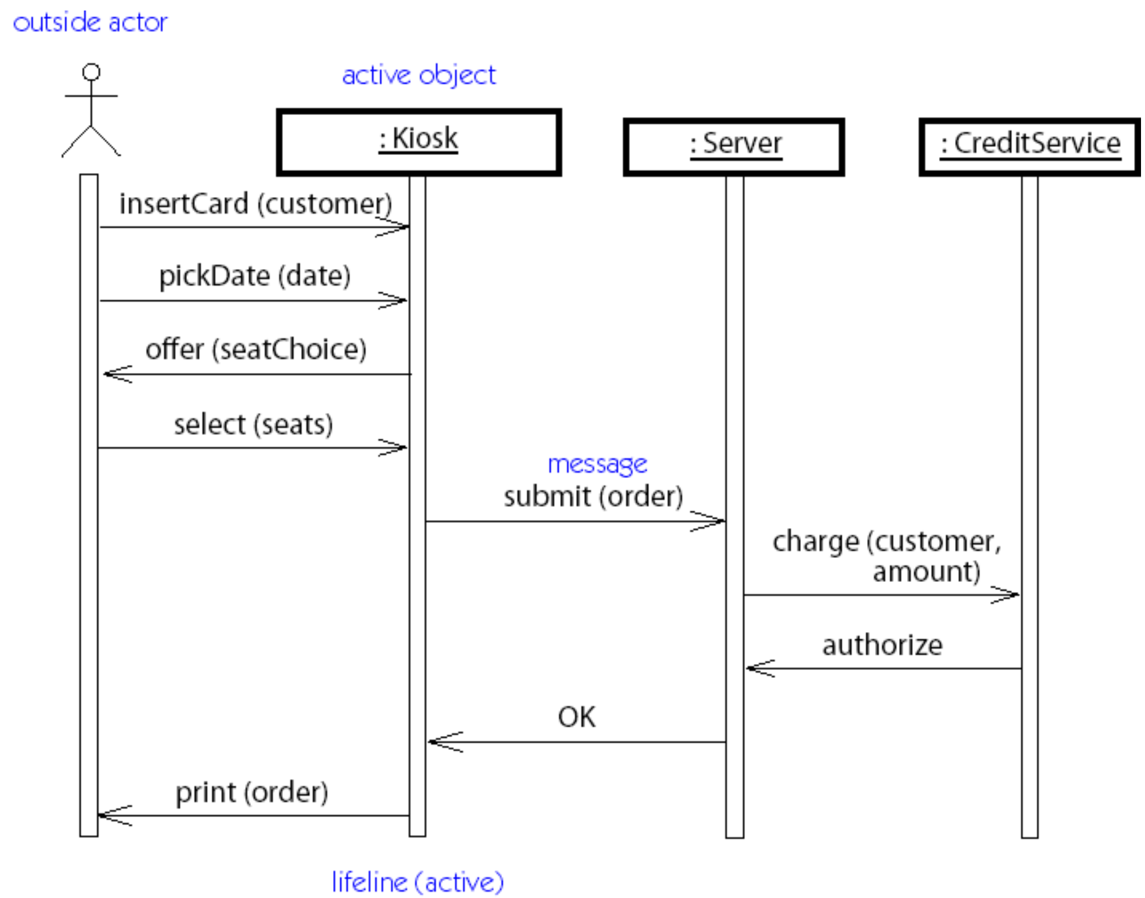
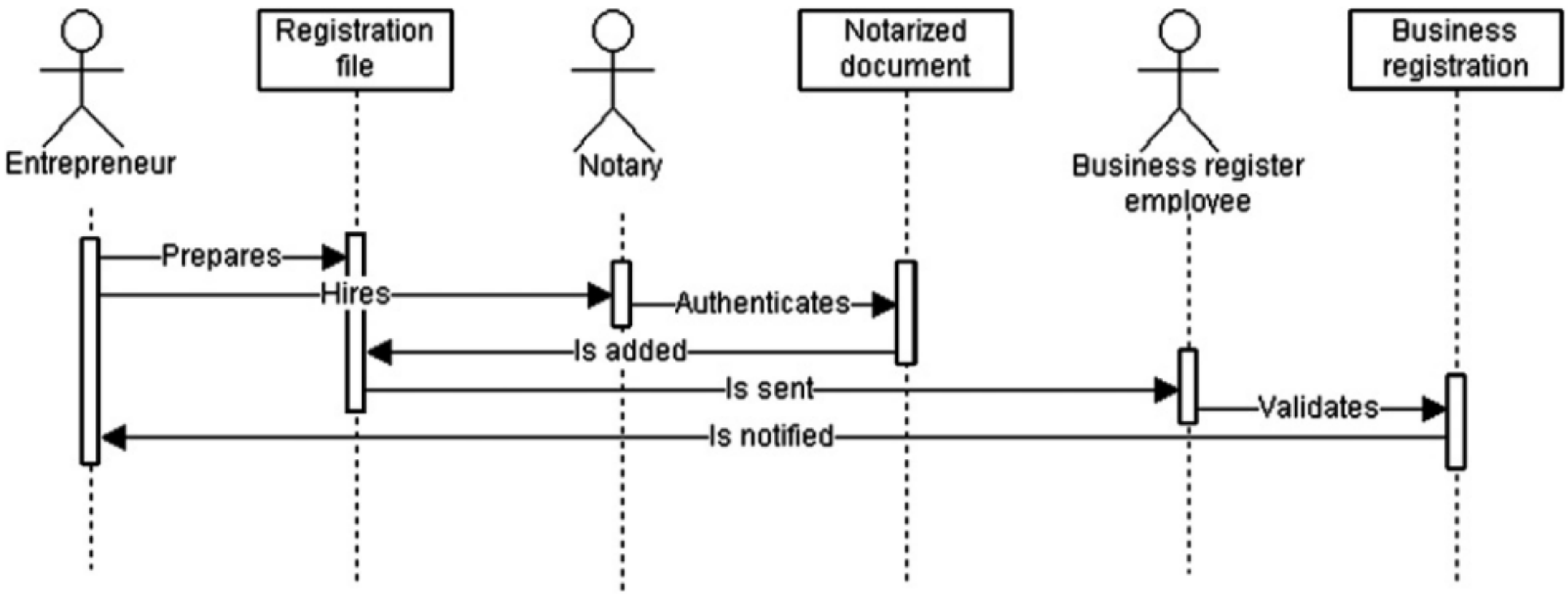


Figure 8-1. *Sequence diagram*

UML SEQUENCE DIAGRAM EXAMPLE



STATECHART DIAGRAMS

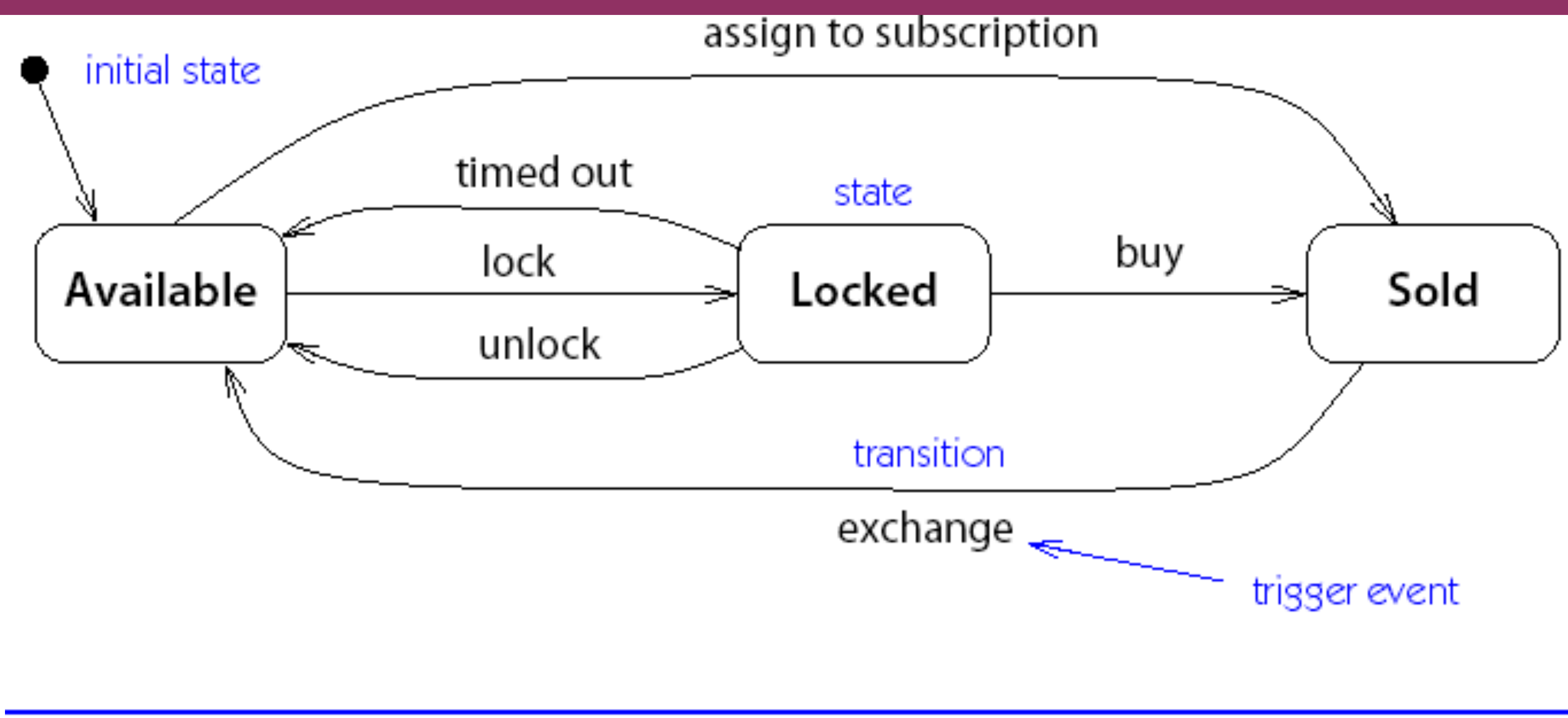
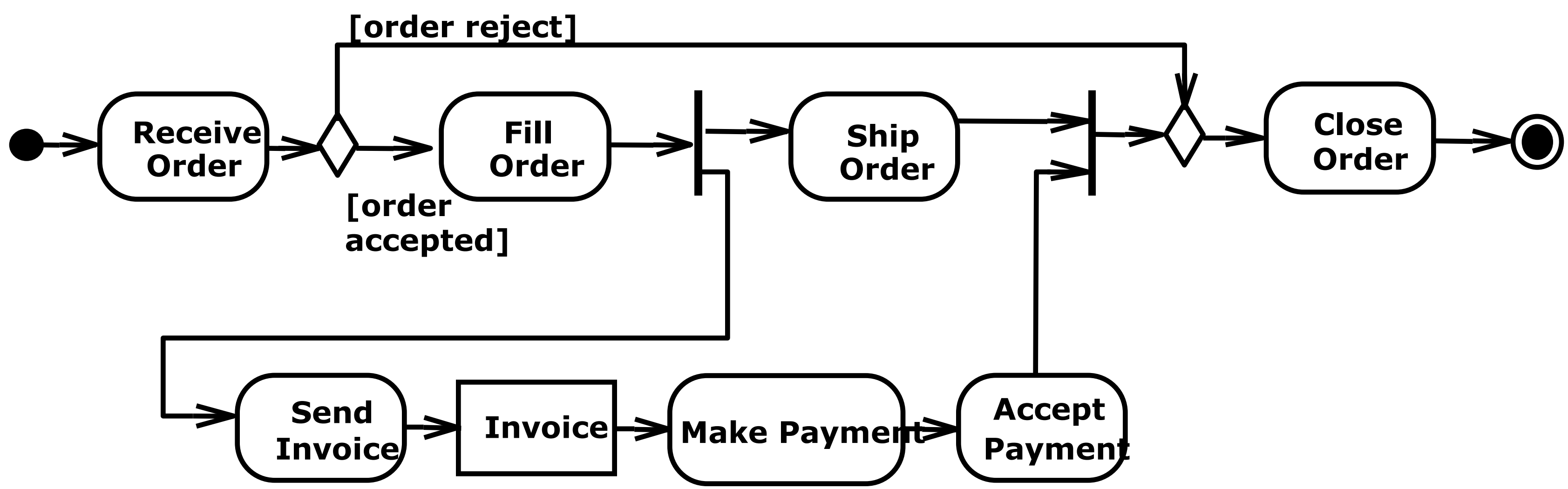


Figure 3-5. *Statechart diagram*

A <u>Statechart Diagram</u> describes the *temporal evolution* of an object of a given class in response to *interactions* with other objects inside or outside the system.

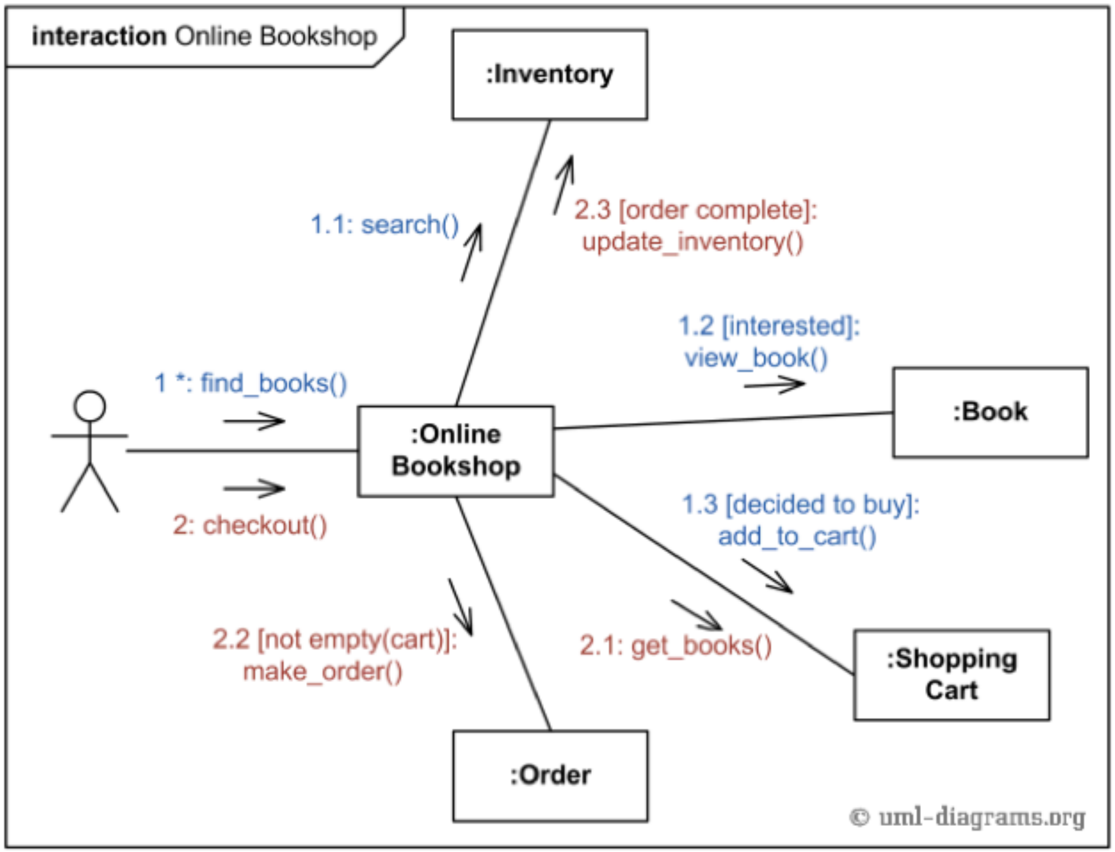
ACTIVITY DIAGRAMS

- Useful to specify software or hardware system behaviour
- Based on data flow models – a graphical representation (with a Directed Graph) of how data move around an information system



COLLABORATION DIAGRAMS

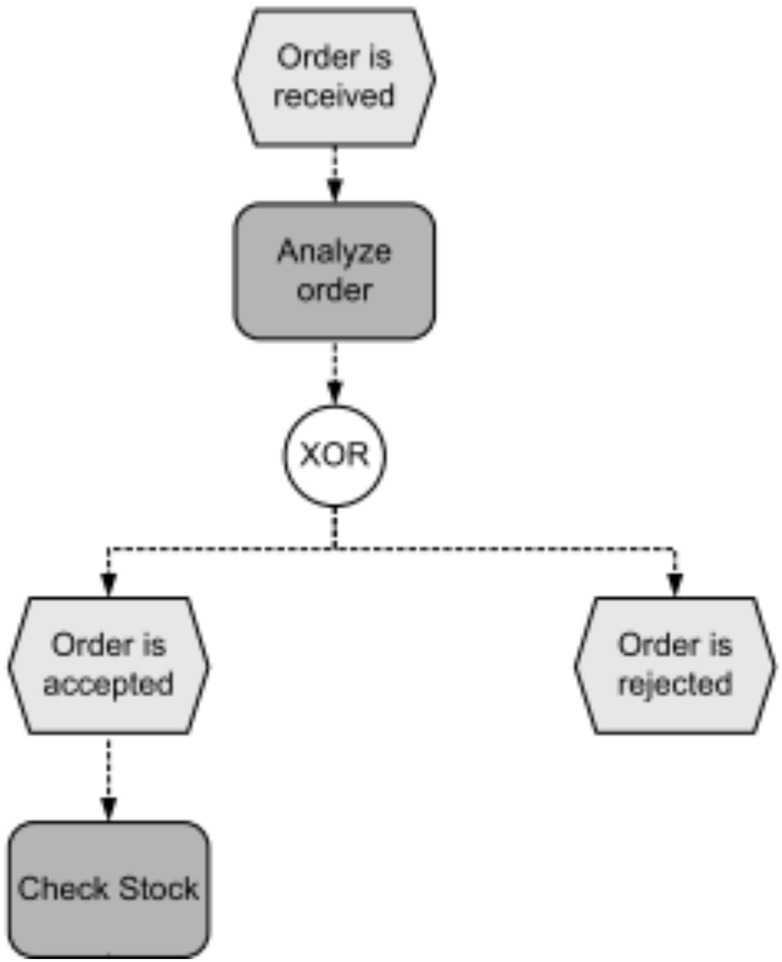
<u>Collaboration diagrams</u> (called *Communication* diagrams in UML 2.0) depict scenarios as *flows of messages* between objects:



EVENT PROCESS CHAINS (EPCS)

- Informal notation for representing domain concepts & processes
- Not focused on technical realization
- Part of a holistic modelling approach called the ARIS framework
- Main building blocks: events, functions (low-level of granularity), connectors (process logic) & control flow edges
- Framework also includes interaction flow diagrams (high-level view of organisational entities & their interactions) & function flow diagrams (refinement of interaction flow diagrams with interaction ordering & interaction representation via coarse-grained functions)
- EPC drawbacks:
  - Verbose & quite complex diagrams
  - Semi-formal representation -> problems with transformation to executable format

EPC EXAMPLE



Order is received
Analyze order
XOR
Order is accepted
Order is rejected
Check Stock

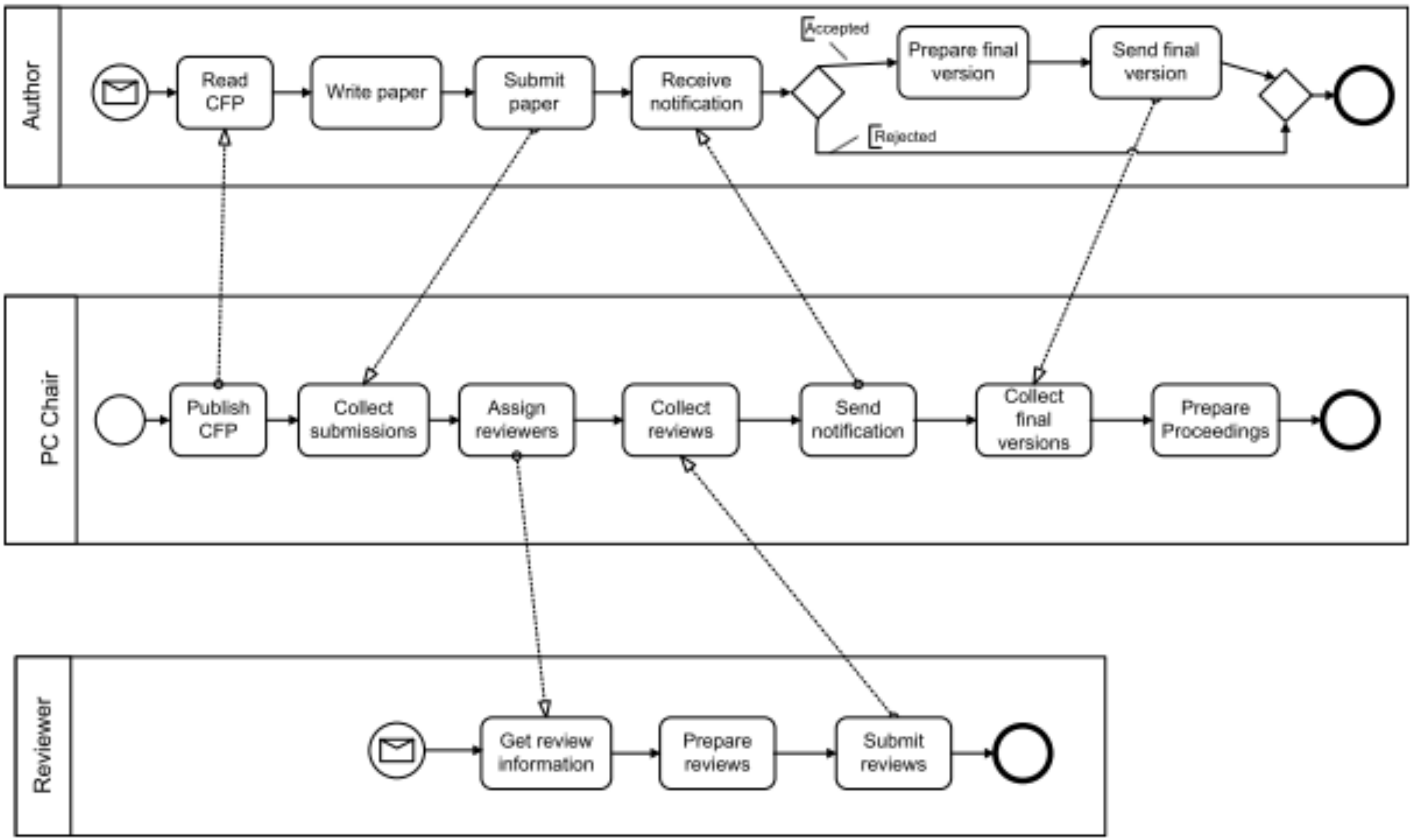
BUSINESS PROCESS MODELLING NOTATION (BPMN)

- Based on flowcharting techniques for processes
- Graphical BP diagram with flow & connecting objects, swimlanes, and artefacts
- Explicitly indicates organisational information
- Covers both orchestrations & choreographies
- Characteristics:
  - Flexibility (well-structured technique with process breakdown & rich set of control flow constructs)
  - Ease of use for both inexperienced and expert stakeholders
  - Understandability
  - Supports the construction of simulation models

BPMN

- Drawbacks:
  - Data Handling (data structures are not covered – could be exploited in conditions)
  - Message flow (two-level hierarchy of swimlanes)
  - No Representation of states
  - Under-representation of systems
  - Not all workflow patterns are covered + for advanced patterns, expertise in filling in no graphical information is needed
  - No formal semantics
  - Minimal support to resource modelling
  - Missing support for business-specific terms & business rules

BPMN EXAMPLE



RECOMMENDED READING

- Ruth Sara Aguilar-Saven. Business Process Modelling: Review and Framework. Int. J. Production Economics 90 (2004) 129–149.
- M. Ould "Business Processes", chapters 1, 2
- Koubarakis & Plexousakis, "A Formal Model for Business Process Modeling and Design", Information Systems 27(2002), pp. 299-319.
- http://www.cds.caltech.edu/~murray/courses/afrl-sp12/L3_ltl-24Apr12.pdf
- https://www.youtube.com/watch?v=UfBFIAMOYgg&list=PLrAWWpbaj-7JIEV3_BfBLNYdYKrpqoC68&index=2
- https://www.youtube.com/watch?v=ztZsEI6C-mI
- https://www.youtube.com/watch?v=UfBFIAMOYgg
- https://creately.com/blog/diagrams/class-diagram-relationships/