



Co-funded by the
Erasmus+ Programme
of the European Union



SOFIA UNIVERSITY
"ST. KLIMENT OHRIDSKI"
EST. 1888



ICT-TEX course on Digital skills

Topic 6: Introduction to Software Engineering

The course is developed under Erasmus+ Program Key Action 2:
Cooperation for innovation and the exchange of good practices [Knowledge Alliance](#)

ICT IN TEXTILE AND CLOTHING HIGHER EDUCATION AND BUSINESS

Project Nr. 612248-EPP-1-2019-1-BG-EPPKA2-KA

The information and views set out in this publication are those of the authors and do not necessarily reflect the official opinion of the European Union. Neither the European Union institutions and bodies nor any person acting on their behalf may be held responsible for the use which may be made of the information contained therein.



Co-funded by the
Erasmus+ Programme
of the European Union



SOFIA UNIVERSITY
"ST. KLIMENT OHRIDSKI"
EST. 1888



6.4. UML Diagrams



These slides are part of the topic on
“Topic 6: Introduction to Software Engineering” of the
course on Digital skills in Textile and clothing industry.

Check also the other themes in this topic:

- 6.1. Software Engineering
- 6.2. Requirements Engineering
- 6.3. Introduction to modeling and UML



Contents

4. Unified Modelling Language (UML) – Process Diagrams

- [UML Diagrams](#)
- [Process Diagrams](#)
- [UML Activity Diagrams](#)
- [State Diagrams](#)
- [Interaction Diagrams](#)
- [Sequence Diagrams](#)
- [Communication Diagrams](#)
- [Time Diagrams](#)
- [Interaction Overview Diagrams](#)
- [UML Process Diagrams - Conclusions](#)

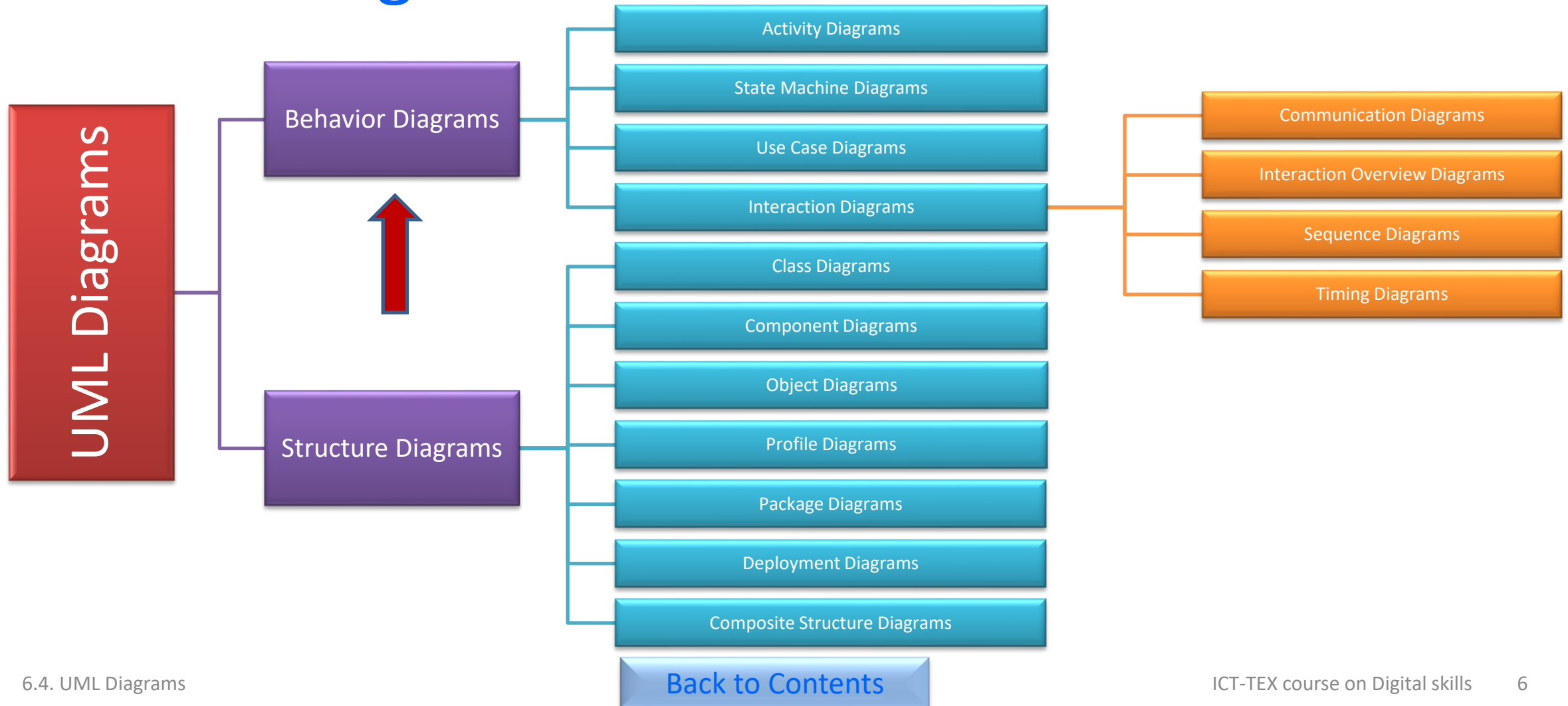


UML diagrams

- UML is the modern, general-purpose modelling approach to specifying, visualizing, constructing, and documenting the artifacts of software systems, business modeling and other non-software systems
- UML facilitates it by diagrams of various types
- UML diagrams represent the OO analysis and design solutions
- You can draw UML diagrams by hand or by using CASE (Computer Aided Software Engineering) tools
- Using CASE tools requires some expertise, training, and commitment by the project management



Process diagrams





The 4+1 views of the software architecture

Scenarios (UML Use Cases)

Logical View
(Object-oriented
Decomposition)

So-called conceptual view
- describes the object
model of the design

Development View
(Subsystem
Decomposition)

Describes the static
organization or structure
of the code in the
development environment

Physical View
(Mapping the
Software to Hardware)

Describes the deployment
of the software on the
hardware

Process View
(Process
Decomposition)

Describes the aspects of
competitiveness and
synchronization

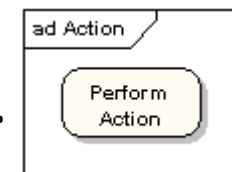
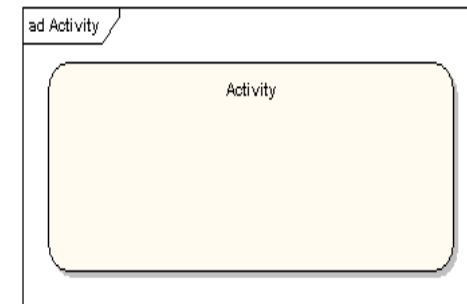


UML Activity diagrams



- While UML use cases diagrams describe the behavior of the target system from an external point of view and capture user requirements, the UML activity diagrams provide a way to model the workflow (sequence of activities producing observable value) of a business process.
- An activity diagram is basically a special case of a state machine in which most of the states are activities and most of the transitions are implicitly triggered by completion of the actions in the source activities.

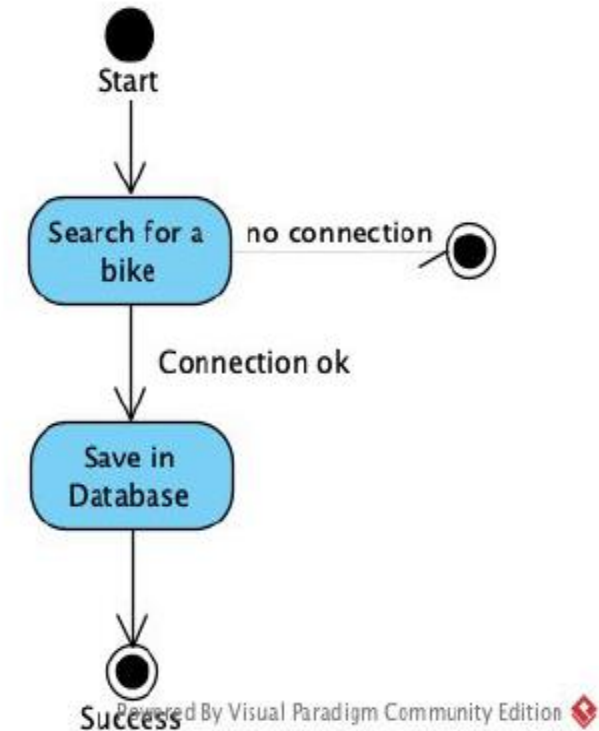
Activities and actions

- An activity diagram may describe *a use case, an operation or a message* – *the purpose* is to describe implementation-oriented details
- An activity diagram is composed by mainly by *activities, actions, transitions, pins, decisions, synchronizations, and swimlanes.*
- An *activity* is a sequence of actions that take finite time and can be interrupted; the specification of a parameterized sequence of behavior. An activity is shown as a round-cornered rectangle enclosing all the actions, control flows and other elements that make up the activity
- An *action* is an atomic task that cannot be interrupted (at least from user's perspective). An action represents a single step within an activity.



Transitions

- Transitions in an activity diagram:
 - Are presented by arrows
 - Indicate the completion of an action or sub-activity and show the sequence of actions or sub-activities
 - Consequently, these transitions are not based on external events
- In the left figure, the ovals represent activities, the arrows show transitions, and  and  present initial and final states, resp.



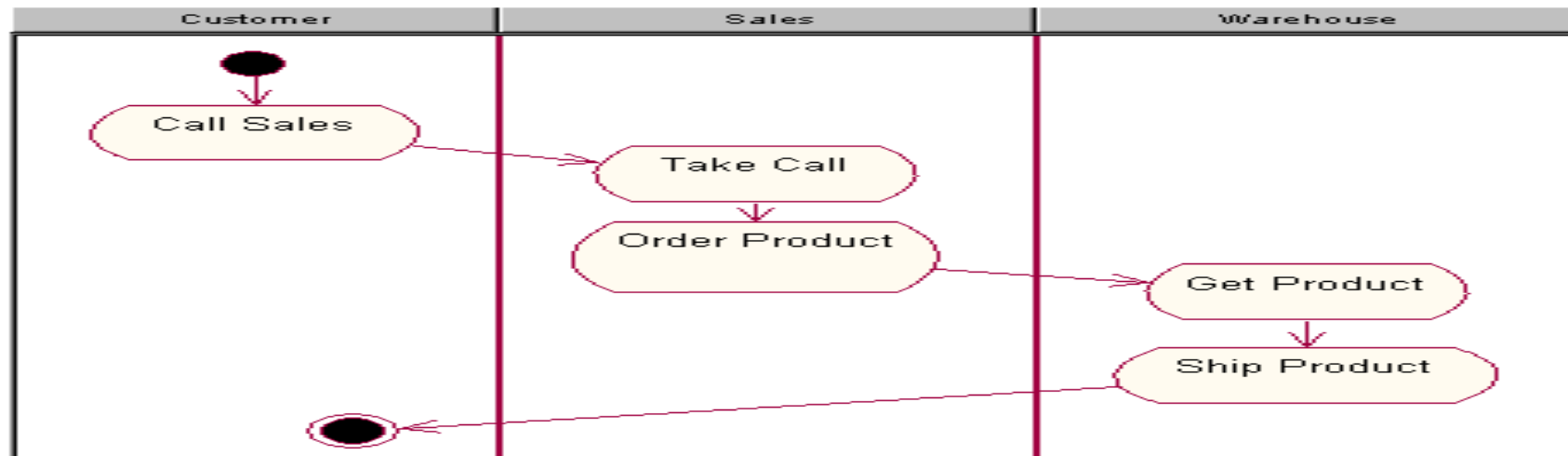
Powered By Visual Paradigm Community Edition 

Used Visual Paradigm Community Edition tool:
<https://www.visual-paradigm.com/> last accessed 04.01.2021



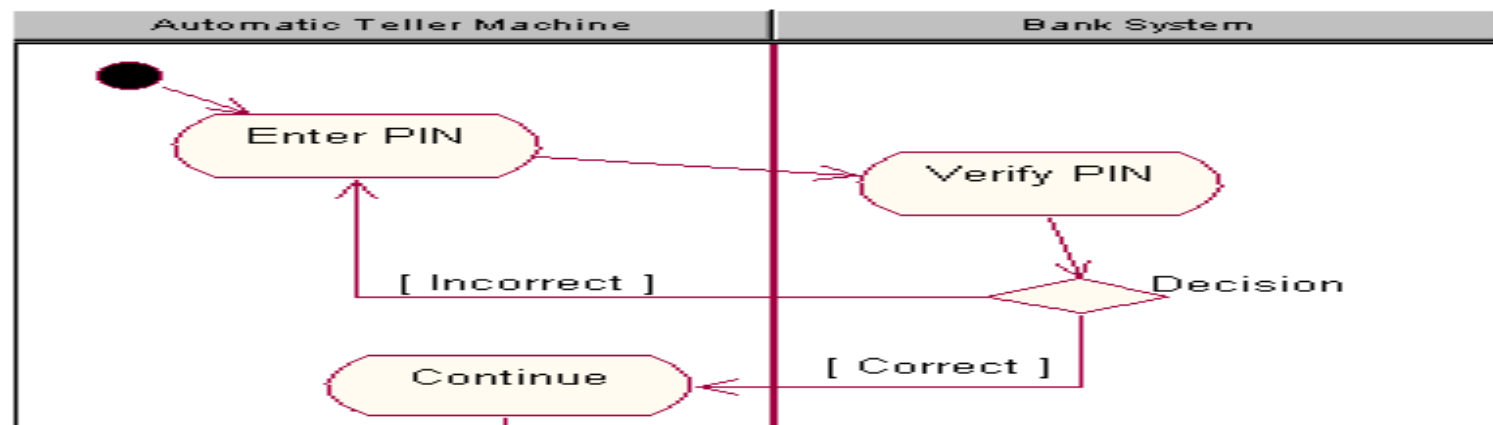
Swimlanes (partitions)

- Swimlanes only appear on activity diagrams and determine which unit is responsible for carrying out the specific activity.
- Example: Get Product and Ship Product activities reside within the Warehouse swimlane indicating that the warehouse is responsible for getting the correct product and then shipping the product to the customer. The workflow ends when the customer (noted through the Customer swimlane) receives the product.



Decision nodes

- A decision represents a specific location where the workflow may branch based upon guard conditions. There may be more than two outgoing transitions with different guard conditions, but for the most part, a decision will have only two outgoing transitions determined by a Boolean expression.
- The following figure displays a decision with [Correct] and [Incorrect] as the guard conditions. If the personal identification number (PIN) is incorrect, the flow of control goes back to the Enter PIN activity. If it is correct, the flow of control moves to the Continue activity.

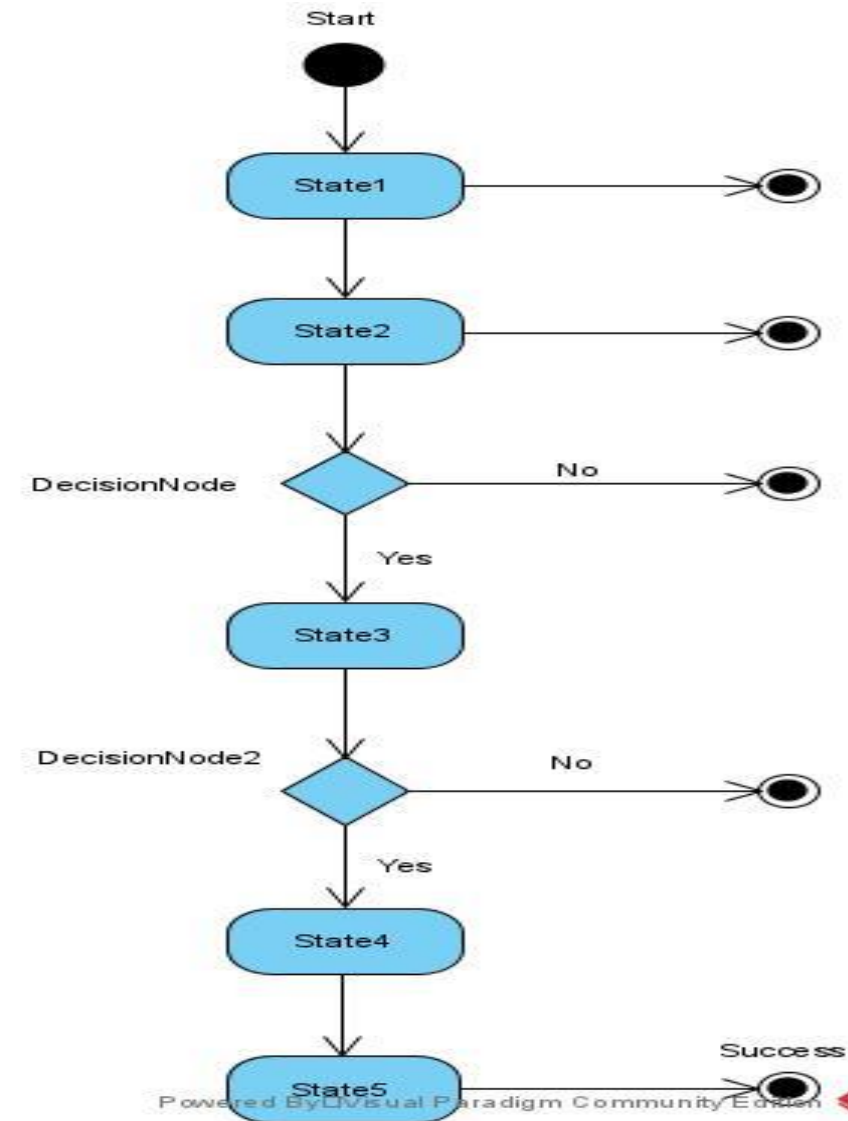




A Sample state diagram

The diagram on the right has:

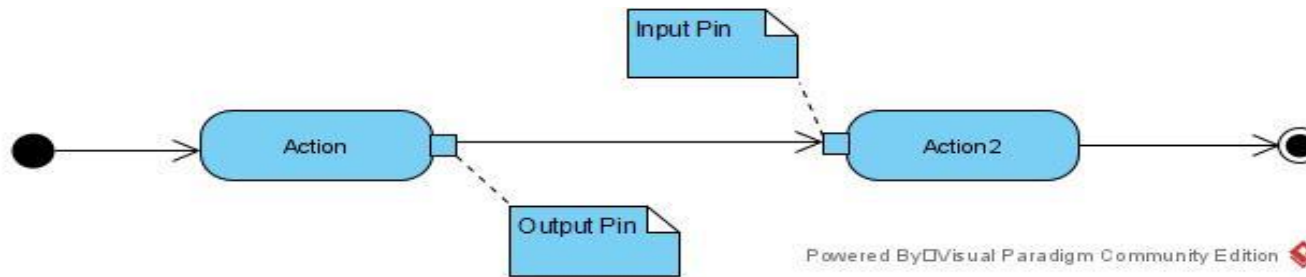
- One start and five final states,
- Four activities
- Two decision blocks
- 12 transitions



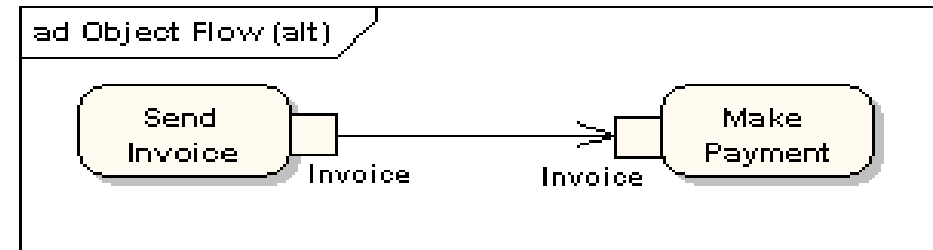
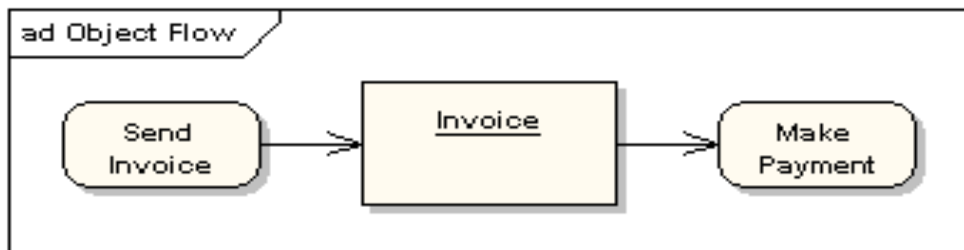
Used Visual Paradigm Community Edition tool:
<https://www.visual-paradigm.com/> last accessed 04.01.2021

Pins (since UML 2.0)

- An *input pin* means that the specified object is input to an action.
- An *output pin* means that the specified object is output from an action.



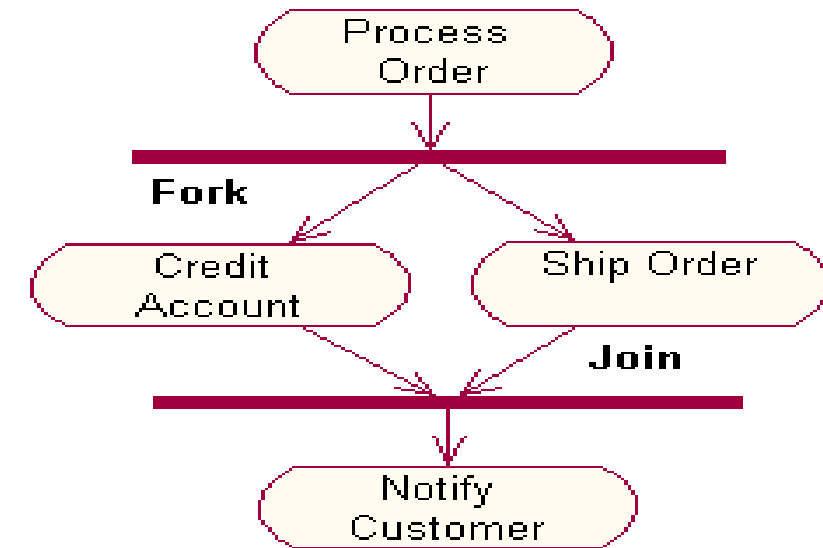
- An object flow must have an object on at least one of its ends (left figure below). A shorthand notation for the above diagram would be to use input and output pins (right figure below).





Synchronizations

- Synchronizations enable you to see a simultaneous workflow. Synchronizations visually define forks and joins representing parallel workflow.
- A fork construct is used to model a single flow of control that divides into two or more separate, but simultaneous flows.
- A join consists of two or more flows of control that unite into a single flow of control. All activities and states that appear between a fork and join must complete before the flow of controls can unite into one.





State diagrams

- A state (statechart) diagram shows a state machine, a dynamic behavior that specifies the sequences of states that an object goes through during its life in response to events, together with its responses and actions.
- A state machine diagram models the behavior of a single object, specifying the sequence of events that an object goes through during its lifetime in response to events.
- It shows the sequences of states that an object goes through, the events that cause a transition from one state to another, and the actions that result from a state change.

States

- State - represents a condition or situation during the life of an object during which it:
 - Satisfies some condition
 - Performs some action
 - Waits for some event
- May be of type:
 - Simple or composite state
 - Real or pseudo-state (like history or junction pseudo-states)
- Each state represents the cumulative history of its behavior. The state icon appears as a rectangle with rounded corners and a name. It also contains a compartment for actions.

State transitions

- A state transition indicates that an object in the source state will perform certain specified actions and enter the destination state when:
 - Non-automatic - a specified event occurs
 - Automatic - when certain conditions are satisfied.
- It takes the state machine from one state configuration to another, representing the complete response of the state machine to an occurrence of an event of a particular type

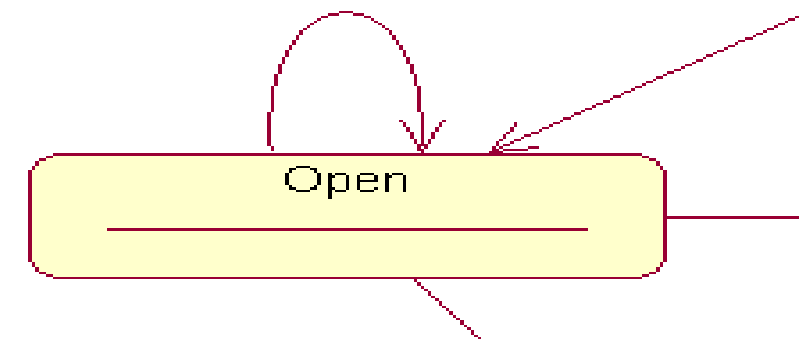
State transitions details

- *Naming*: transitions are labeled with the following syntax:

```
event (arguments) [condition] /  
action ^target.sendEvent (args)
```

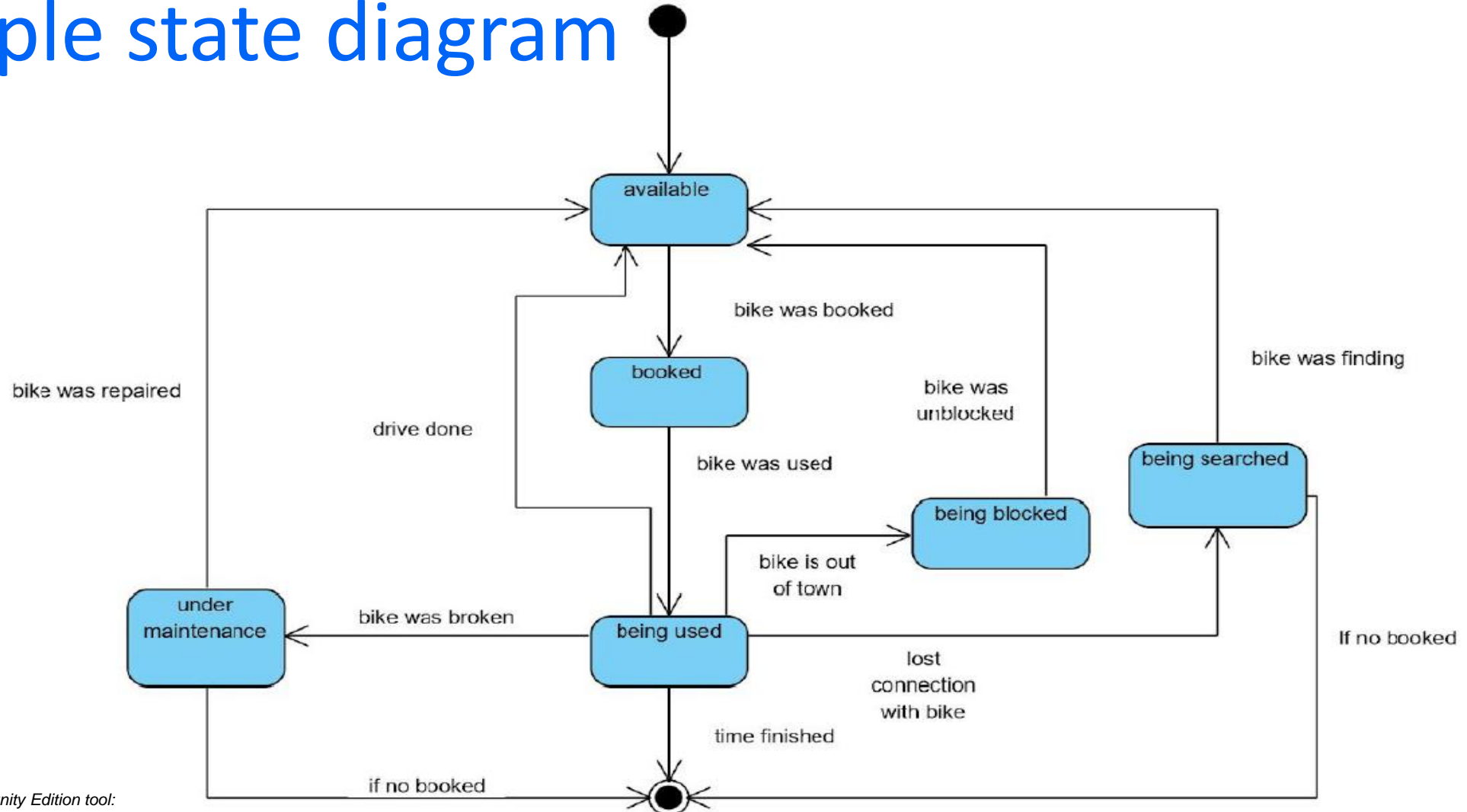
- *Triggering event*: behavior (operation) that occurs in the state transition, like addStudent(Id)
- *Guard condition*: a boolean expression over the attributes that allows the transition, like [count<10]
- *Action*: an action over the attributes, like incrCount
- *Send event*: an event invoked in a target object, like ^CourseRoster.addStrudent(Id)

```
addStudent( Id )[ count<10 ] /  
incrCount  
^CourseRoster.addStudent(Id)
```





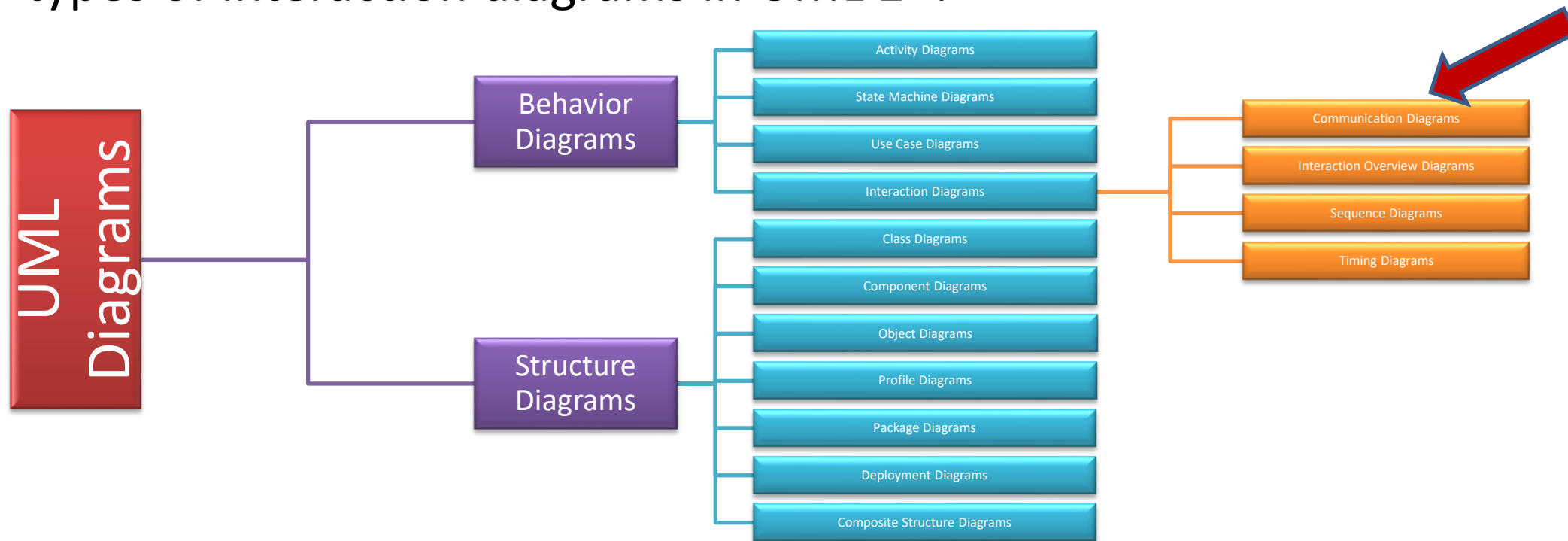
Sample state diagram



Used Visual Paradigm Community Edition tool:
<https://www.visual-paradigm.com/> last accessed 04.01.2021

Interaction diagrams in UML 1/2

For each use-case realization there is one or more interaction diagrams depicting its participating objects and their interactions. There are four types of interaction diagrams in UML 2*:





Interaction diagrams in UML 2/2

1. **Sequence diagrams** - show the explicit sequence of messages and are better for real-time specifications and for complex scenarios;
2. **Communications** (prior UML 2.* - collaboration) diagrams - show the communication links between objects and are better for understanding all of the effects on a given object and for algorithm design.
3. **Interaction overview diagrams** - show a control flow with nodes that can contain interaction diagrams which show how a set of fragments might be initiated in various scenarios. Interaction overview diagrams focus on the overview of the flow of control where the nodes are interactions (sequence diagrams) or interaction use (a reference);
4. **Timing diagrams** - focus on conditions changing within and among lifelines along a linear time axis; paying attention on time of events causing changes in the modeled conditions of the lifelines.

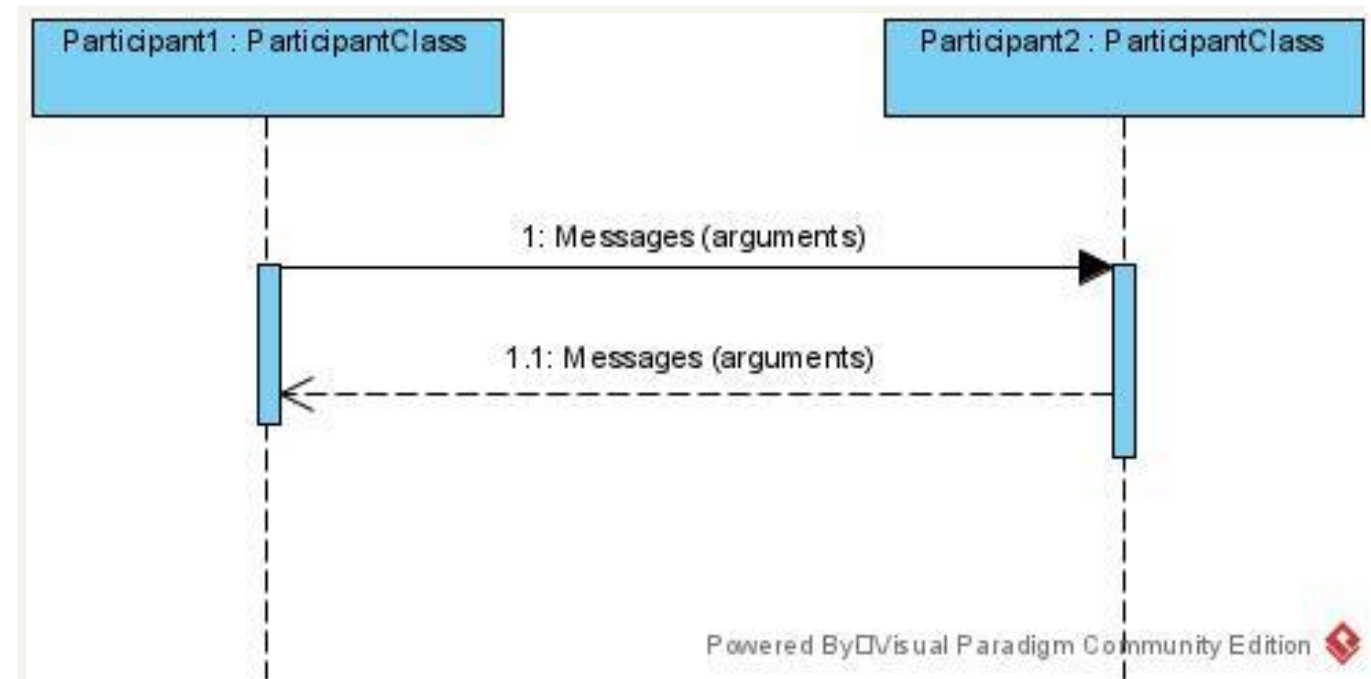
Sequence diagrams

- A sequence diagram describes a pattern of interaction among objects, arranged in a chronological order.
- Sequence diagrams show the objects participating in the interaction by their "lifelines" and the messages that they send to each other, i.e. how objects interact to perform the behavior of a use case.
- Sequence diagrams contain:
 - objects (i.e. class instances), and
 - actor instances
- The diagram describes what takes place in the participating objects, in terms of activations of operations, and how the objects communicate by sending messages.



Messages between objects in sequence diagrams

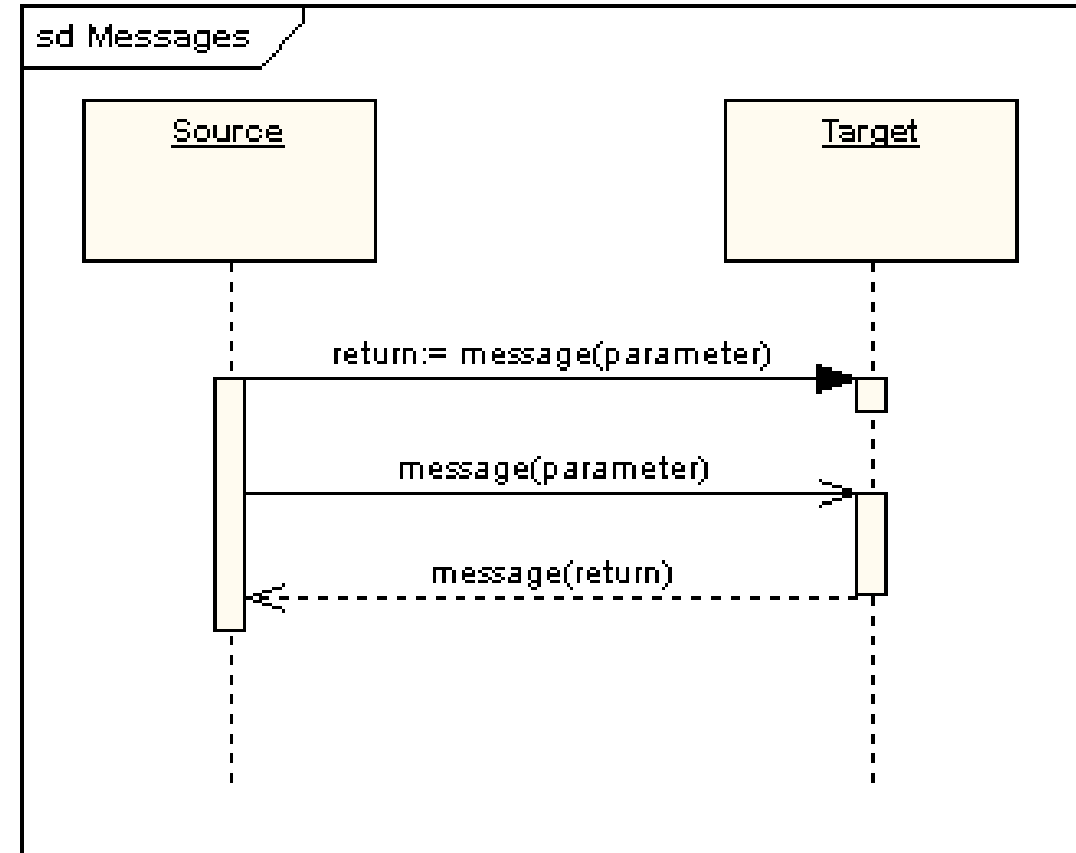
- Objects - shown as a vertical dashed line called the "lifeline".
- An object participant shows the name of the object and its class:
object name : class name
- Interactions on a sequence diagram are shown as messages between participants





Synchronous and asynchronous messages

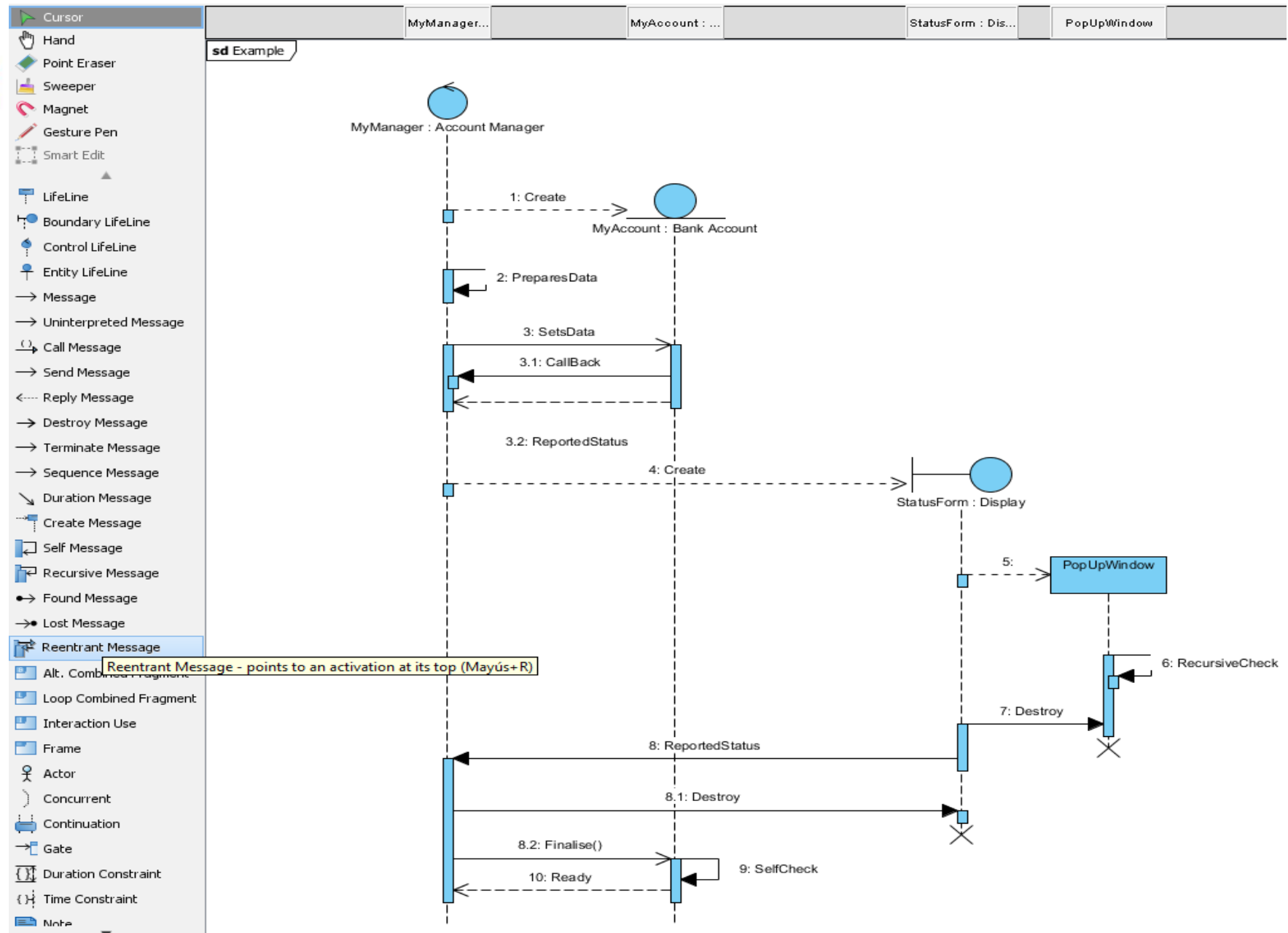
- The first message is a **synchronous** message (denoted by the solid arrowhead) complete with an implicit return message;
- The second message is **asynchronous** (denoted by line arrowhead), and the third is the asynchronous return message (denoted by the dashed line).





Co-funded by the Erasmus+ Programme of the European Union

Sample sequence diagram in the editor of Visual Paradigm™



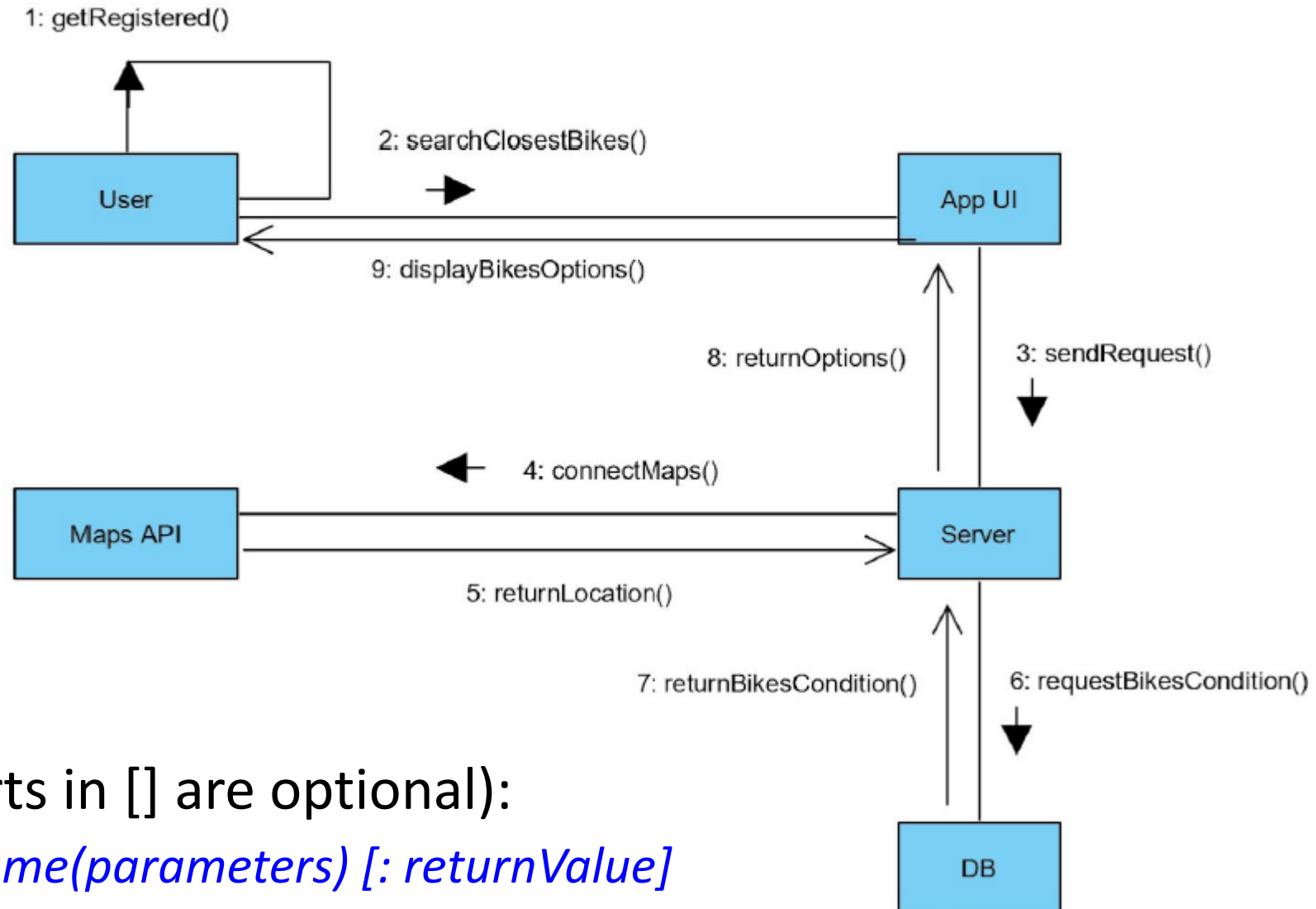


Communication diagrams

- A communication (prior UML 2.0 called collaboration) diagram describes a pattern of objects interaction; it shows the objects participating in the interaction by their links to each other and the messages sent.
- Unlike a sequence diagram, a collaboration diagram shows the relationships among the objects. Sequence and communication diagrams are so similar that most UML tools can automatically convert from one diagram type to the other.
- You can have objects and actor instances in communication (collaboration) diagrams, together with links and messages describing how they are related and how they interact.



A sample communication diagram



Message notation (the parts in [] are optional):
[sequenceNumber:] methodName(parameters) [: returnValue]



Time diagrams

- The UML interaction diagrams shown over do not model detailed timing information
- In UML timing diagrams each event has timing information associated with it
- Timing diagrams describe:
 - when the event is invoked,
 - how long it takes for another participant to receive the event, and
 - how long the receiving participant is expected to be in a particular state.

Events

Events on a timing diagram can even have their own durations, as shown by event1 taking 1 unit of time from invocation by p1:Participant1 and reception by p2:Participant2

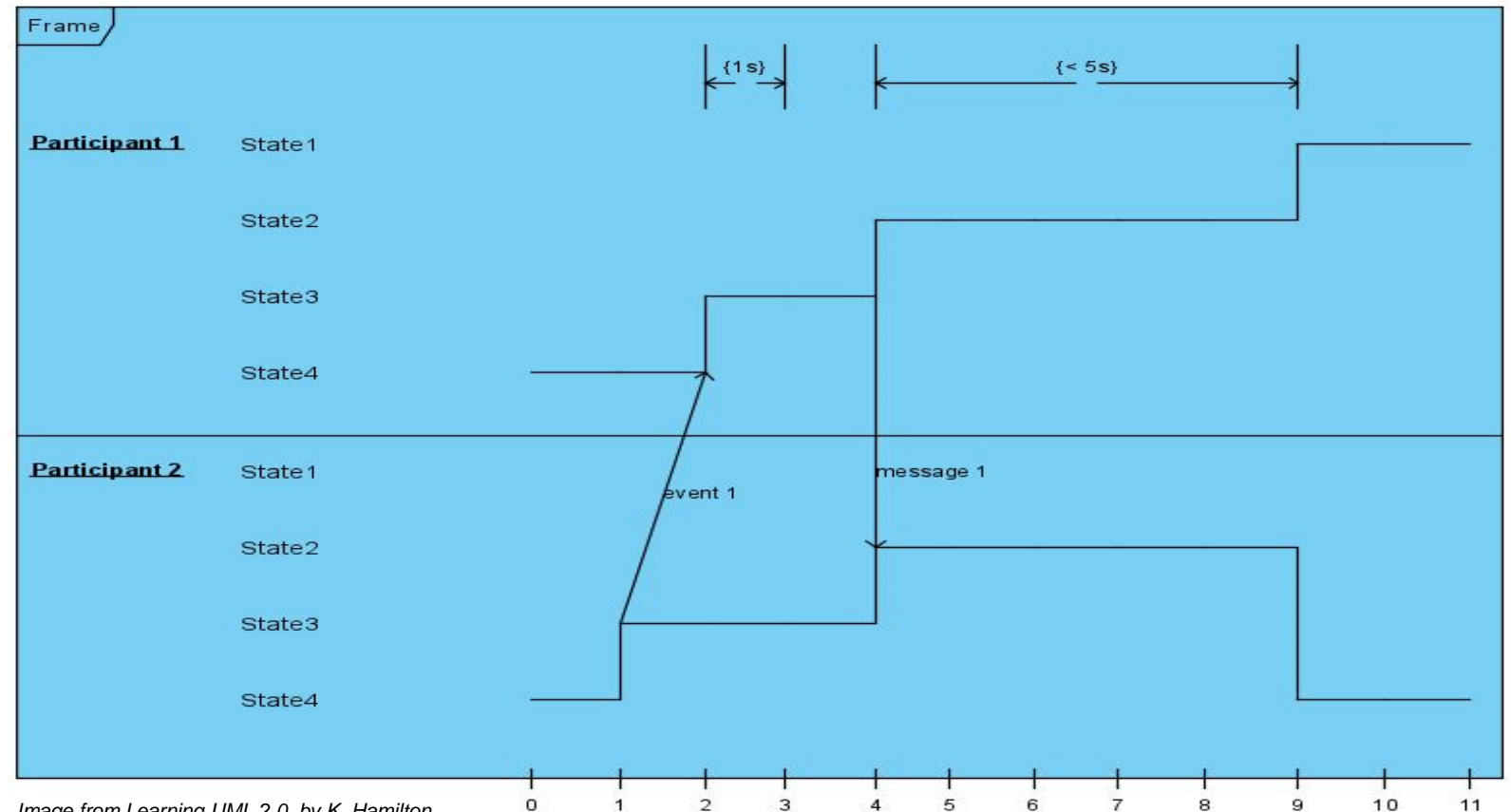
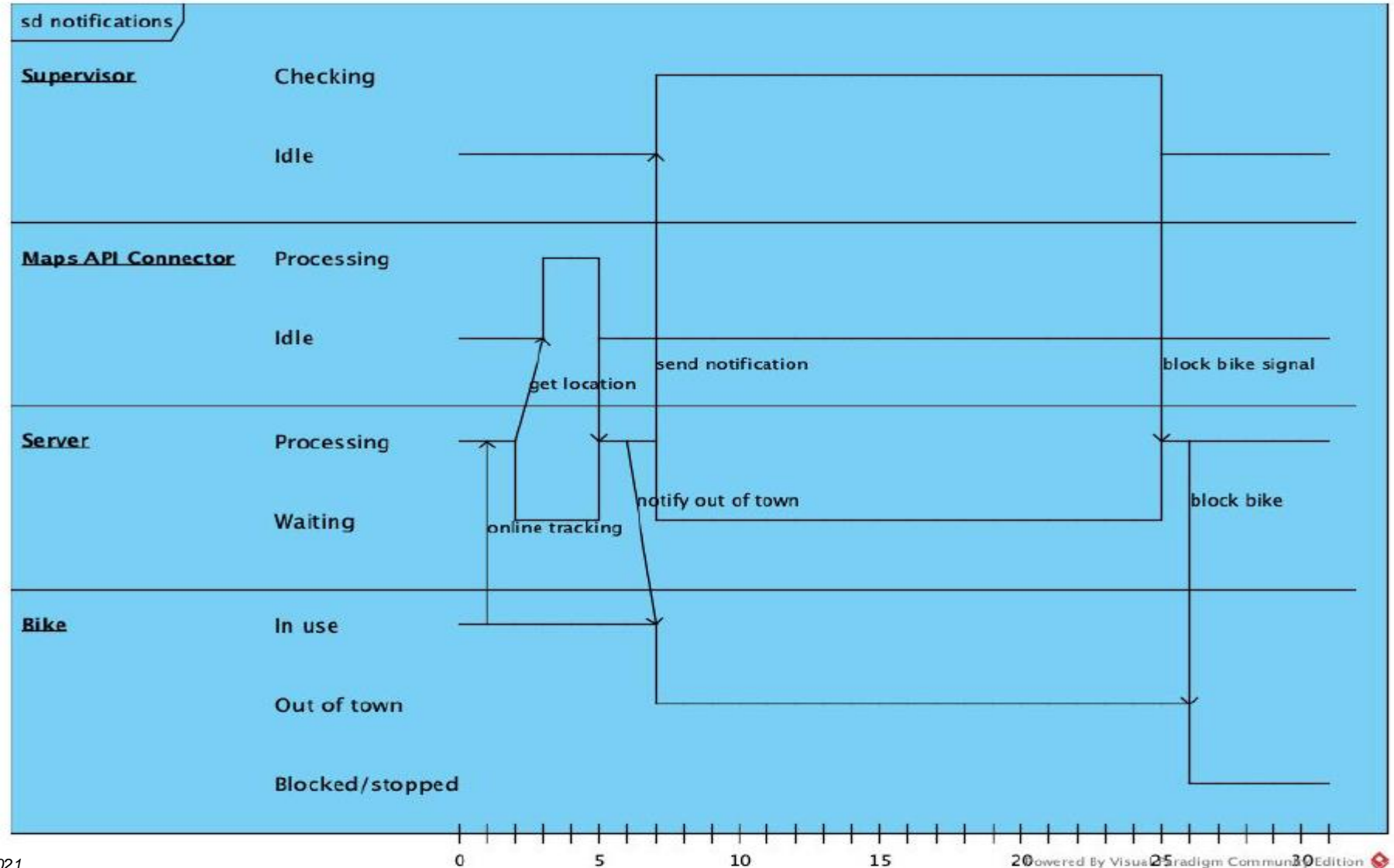


Image from Learning UML 2.0, by K. Hamilton, R. Miles last accessed 04.01.2021



Sample timing diagram



Used Visual Paradigm Community Edition tool: <https://www.visual-paradigm.com/> last accessed 04.01.2021

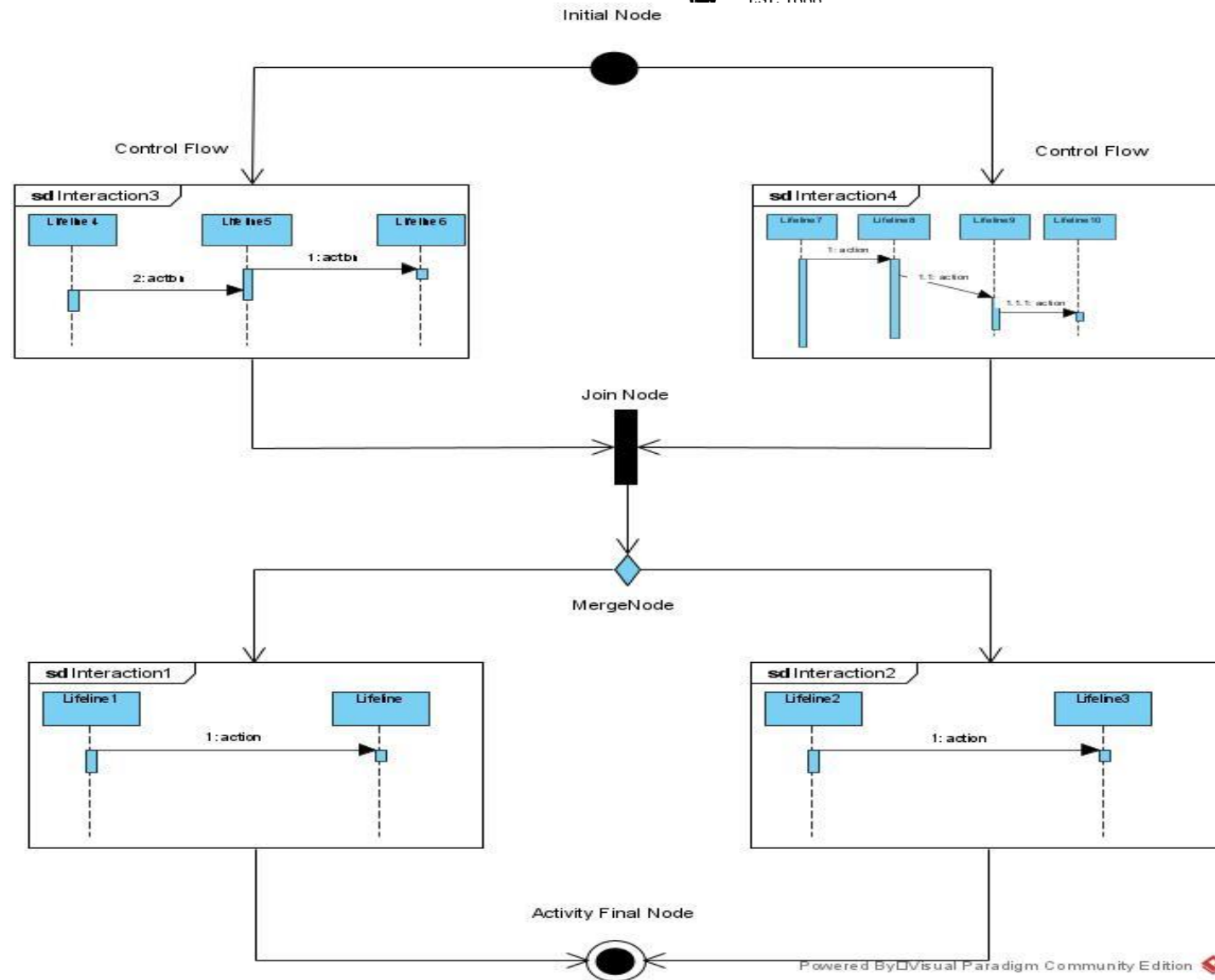


Interaction overview diagrams

- Interaction overview diagrams (since UML 2.0) overview control flow between *interaction diagrams* and show how a set of fragments might be initiated in various scenarios.
- They are variants on UML activity diagrams which represents at a higher level of abstraction the control flow between diagrams
- The nodes within these diagrams are frames of two types:
 - interaction frames depicting any type of UML interaction diagram (sequence diagram: sd, communication diagram: cd, timing diagram: td, etc.)
 - interaction occurrence frames (ref; typically anonymous) which indicate an activity or operation to invoke.



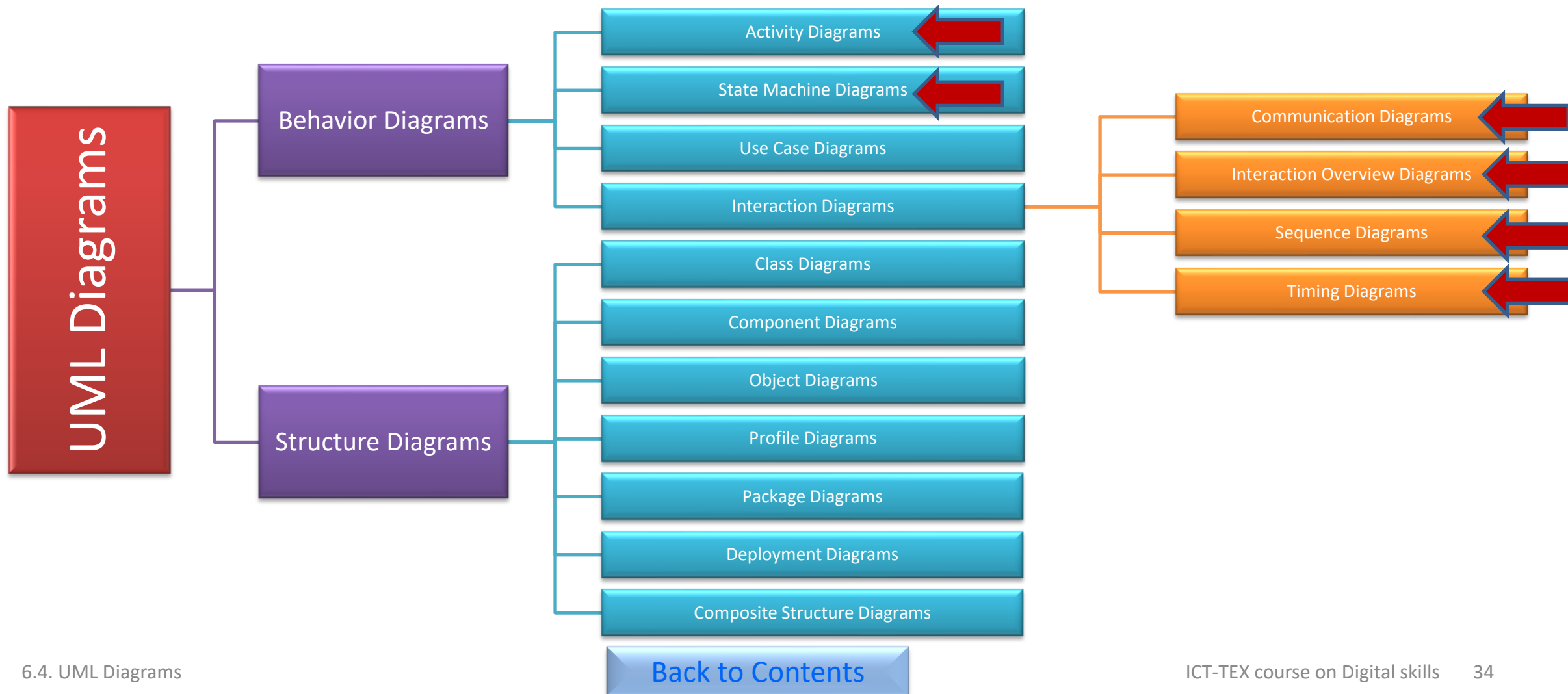
Sample interaction overview diagram



Used Visual Paradigm Community Edition tool: <https://www.visual-paradigm.com/> last accessed 04.01.2021



The six types of UML Process Diagrams



UML process diagrams - conclusions

- UML process diagrams describe the system aspects of competitiveness and synchronization
- They illustrate the flow of events of UML use cases, performed by the set of system object and subsystem instances.



UML



References

- Dan Pilone, Neil Pitman. UML 2.0 in a Nutshell, O'Reilly Media, Inc., June 2005.
- Martin Fowler. UML Distilled: A Brief Guide to the Standard Object Modeling Language, Addison-Wesley Object Technology Series, September 2003.
- Russ Miles, Kim Hamilton. Learning UML 2.0: A Pragmatic Introduction to UML, O'Reilly Media, 1st edition, May 2006.

CONTACTS

Coordinator:
Technical University of Sofia

Project coordinator:
assoc. prof. Angel Terziev, PhD
aterziev@tu-sofia.bg

Web-site: ICT-TEX.eu

Author:
Professor Boyan Bontchev
Sofia University "St. Kliment Ohridski"

Email: bbontchev@fmi.uni-sofia.bg
ResearchGate: <https://www.researchgate.net/profile/Boyan-Bontchev>
Scopus: <https://www.scopus.com/authid/detail.uri?authorId=6506653436>

Assistant professor Yavor Dankov
Sofia University "St. Kliment Ohridski"

Email: yavor.dankov@fmi.uni-sofia.bg
ResearchGate: <https://www.researchgate.net/profile/Yavor-Dankov>
Scopus: <https://www.scopus.com/authid/detail.uri?authorId=57202891597>



Co-funded by the
Erasmus+ Programme
of the European Union

KNOWLEDGE ALLIANCE

ICT-TEX

ICT IN TEXTILE AND CLOTHING
HIGHER EDUCATION AND BUSINESS

These slides and the materials included in these slides (including references) are for educational purposes only. The use of slides should be done with correct citation and only for educational purposes.

The information and views set out in this publication are those of the authors and do not necessarily reflect the official opinion of the European Union. Neither the European Union institutions and bodies nor any person acting on their behalf may be held responsible for the use which may be made of the information contained therein.