Politecnico
di Milano

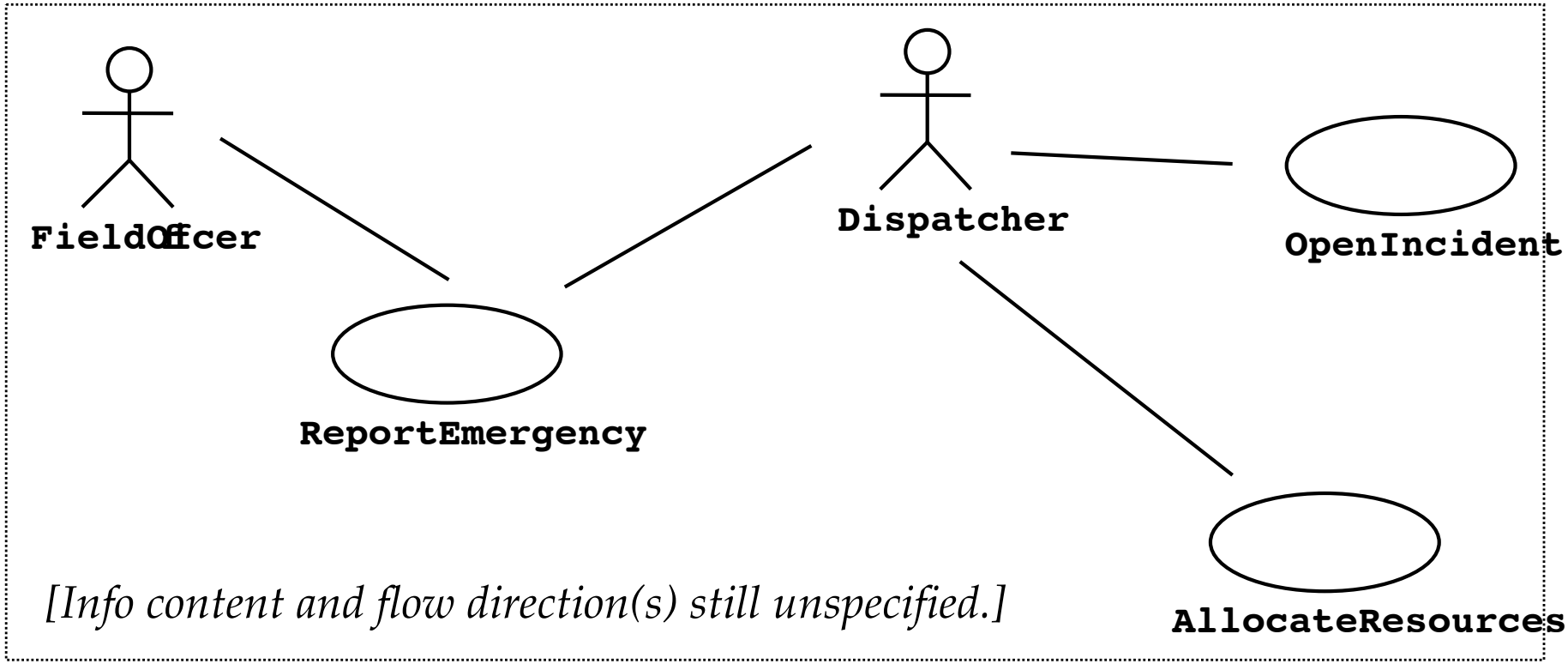
UML and requirements engineering

Use Cases

- A use case is a flow of events in the system, including interaction with actors
- It is initiated by an actor
- Each use case has a name
- Each use case has a termination condition

Use Case Model: The set of all use cases specifying the complete functionality of the system

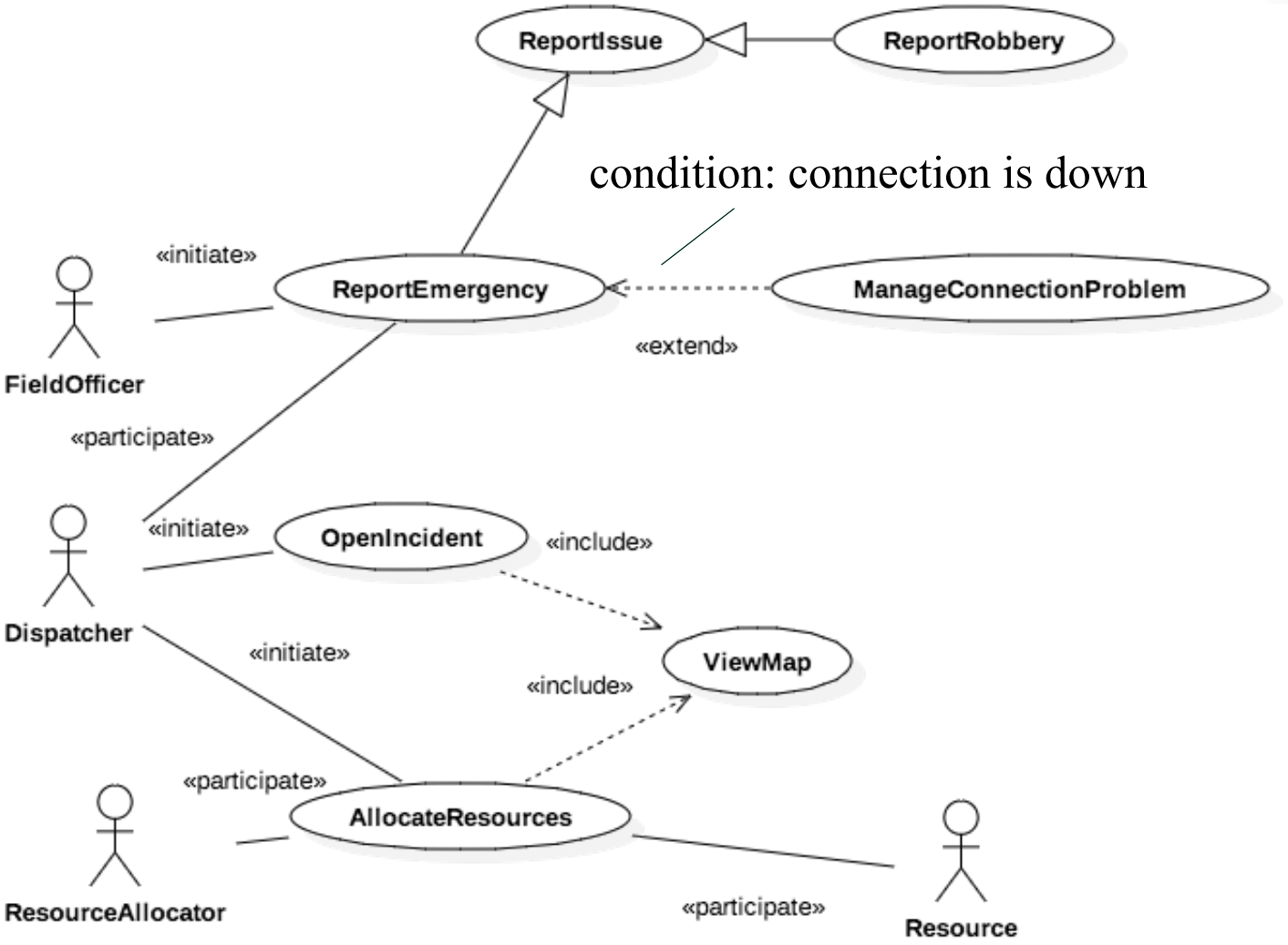
Example: Use Case Model for Incident Management



Use Case Associations

- A use case association is a relationship between use cases
- Important types of use case associations
  - Include
    - A use case uses another use case (“functional decomposition”)
  - Extends
    - A use case extends another use case
  - Generalization
    - An abstract use case has several different specializations

Examples


ReportIssue
ReportRobbery
condition: connection is down
«initiate»
ReportEmergency
ManageConnectionProblem
«extend»
FieldOfficer
«participate»
«initiate»
OpenIncident
«include»
Dispatcher
«initiate»
ViewMap
«include»
«participate»
AllocateResources
ResourceAllocator
«participate»
Resource

Requirements-level class diagrams

- They are conceptual models for the application domain
  - different from OO software design models
- They may model objects that will not be represented in the software-to-be
  - because this is not known at the start of the requirements modelling effort
- Usually, they do not attach operations (methods) to objects
  - it's best to postpone this kind of decisions until software design

How do you find objects and classes?

- Analyze any description of the problem and application domain you may have
- Analyze your scenarios and use cases descriptions
- Finding objects is the central piece in object modeling
- A possible tool to use in the analysis
  - Abbott Textual Analysis, 1983, also called noun-verb analysis
    - Nouns are good candidates for classes
    - Verbs are good candidates for associations and operations

Object modeling: an Example

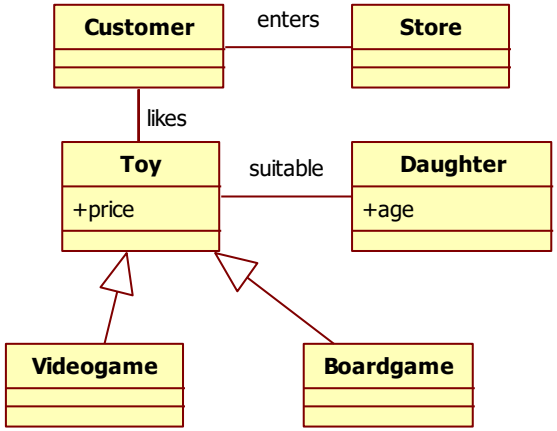
- Consider this text
  - *The customer enters the store to buy a toy. It has to be a toy that his daughter likes and it must cost less than 50 Euros. He tries a videogame, which uses a data glove and a head-mounted display. He likes it.*
  - *An assistant helps him. The suitability of the game depends on the age of the child. His daughter is only 6 years old. The assistant recommends another type of toy, namely Monopoly (a board game). The customer buys the game and leaves the store*

Textual Analysis using Abbot's technique of Natural Language Analysis [OOSE 5.4.1]

Example	*Grammatical construct*	*UML Component*
"Monopoly"	Concrete Person/Thing	Object
"toy"	Noun	Class
"board game"	Noun	Class
"6 years old"	Adjective	Attribute
"enters"	Verb	Operation/Association
"depends on …"	Intransitive Verb	Association
"is a", "either … or", "kind of …"	Classifying Verb	Inheritance
"Has a", "consists of"	Possessive Verb	Aggregation
"must be", "less than …"	Modal Verb	Constraint

Generation of the class diagram for the example

- The customer enters the store to buy a toy. It has to be a toy that his daughter likes and it must cost less than 50 Euros. He tries a videogame, which uses a data glove and a head-mounted display. He likes it.
- An assistant helps him. The suitability of the game depends on the age of the child. His daughter is only 6 years old. The assistant recommends another type of toy, namely Monopoly (a board game). The customer buys the game and leaves the store



Finding Participating Objects in Use Cases

- Pick a use case and look at its flow of events
  - Look for recurring nouns (e.g., Incident),
  - Identify real-world entities that the system needs to keep track of (e.g., FieldOfficer, Dispatcher, Resource),
  - Identify real-world procedures that the system needs to keep track of (e.g., EmergencyOperationsPlan),
  - Identify data sources or sinks (e.g., Printer)
  - Identify interface artifacts (e.g., PoliceStation) [e.g., telecomm link?]
- Be prepared that some objects are still missing and need to be found
- Always use the user's terms

Object modeling and the Report Emergency use case

- Nouns are in red
- Verbs are in green

- The FieldOfficer activates the “Report Emergency” function of her terminal. FRIEND responds by presenting a form to the officer.
- The FieldOfficer fills the form, by selecting the emergency level, type, location, and brief description of the situation. The FieldOfficer also describes possible responses to the emergency situation. Once the form is completed, the FieldOfficer submits the form, at which point, the Dispatcher is notified.
- The Dispatcher reviews the submitted information and creates an Incident in the database by invoking the OpenIncident use case. The Dispatcher selects a response and acknowledges the emergency report.
- The FieldOfficer receives the acknowledgment and the selected response.

Textual analysis

Term	Grammatical construct	UML Component
FieldOfficer	Noun	Class
FRIEND	Noun	This is the S2B object of our analysis. It may be a Class.
Emergency (situation)	Noun	Class
Level, type, location	Noun	Attributes
Brief description	Noun + Adjective	Attribute
Response	Noun	Class
Dispatcher	Noun	Class
Incident	Noun synonym of emergency	No need for a new class
Emergency report	Noun	Class containing the attributes listed above
Form	Noun synonym of Emergency report	No need for a new class

Textual analysis

Term	Grammatical construct	UML Component
Activates	Verb	Association/Operation
Responds	Verb	Reply to Activate
Fills	Verb	Association/Operation
Describes	Verb	Association/Operation
Submits	Verb	Association/Operation
Is completed	Passive verb	State of Form
Is notified	Passive verb	Association/Operation
Creates	Verb	Association/Operation
Selects	Verb	Association/Operation
Acknowledges	Verb	Association/Operation
Receives	Verb	Event

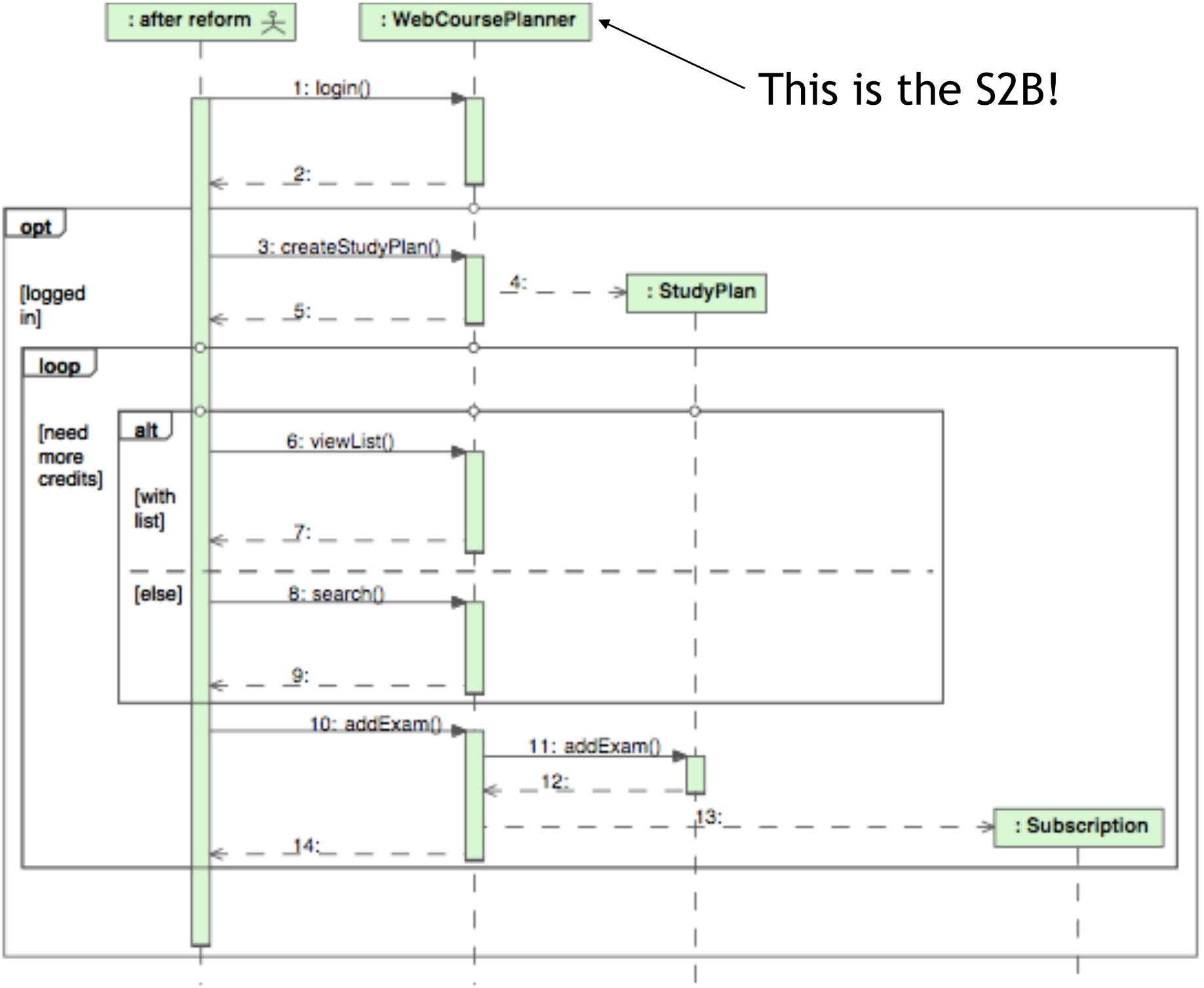
Now you can try to define the class diagram yourself

Dynamic modeling

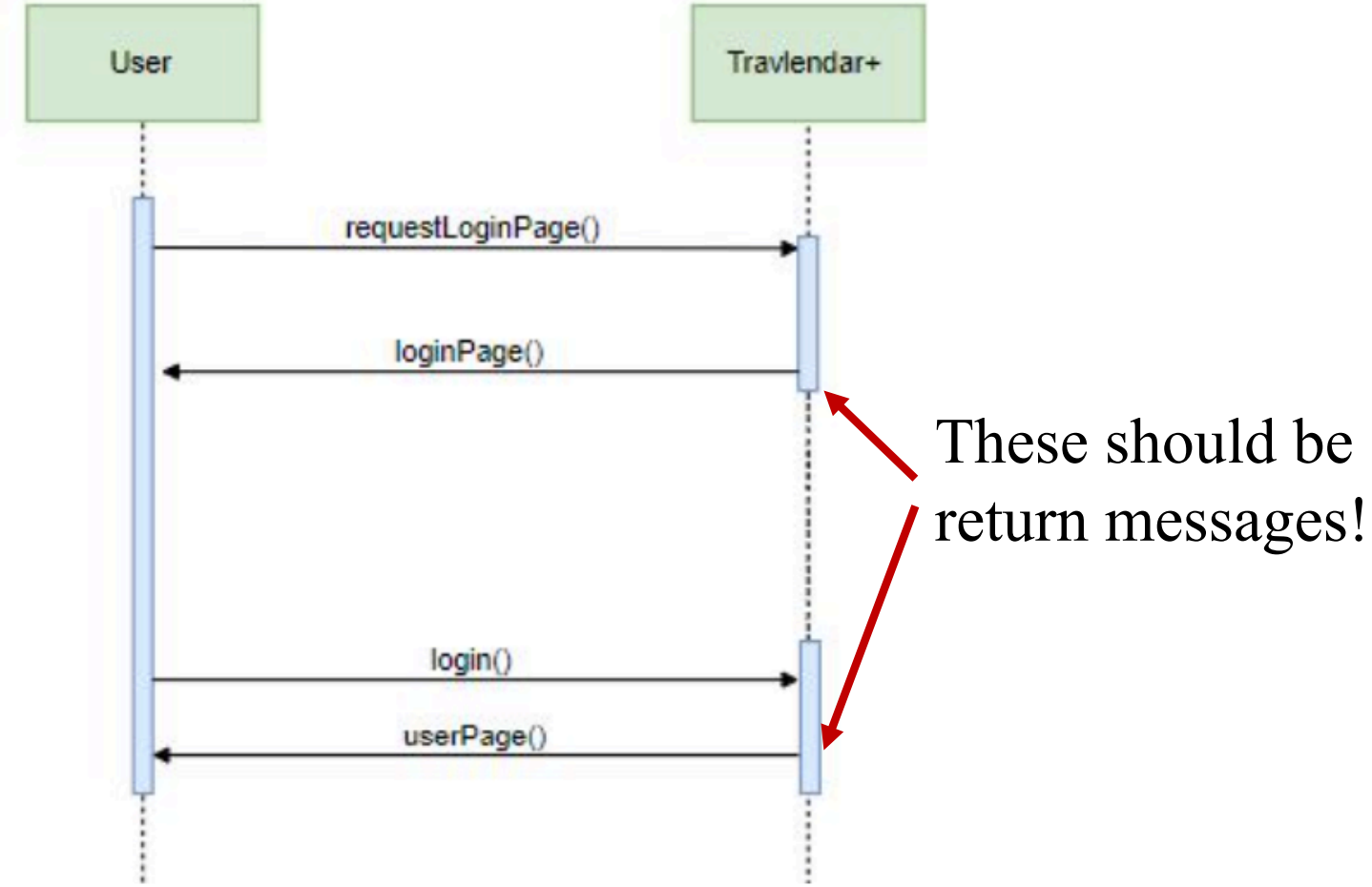
- Purpose:
  - Supply methods to model interactions, behaviors of participants and workflow
- How do we do this?
  - Start with use case or scenario
    - Model interaction between objects => sequence diagram
  - Model dynamic behavior of a single object => state machine diagram (“state diagram” for short)
  - Model workflow => activity diagram

Sequence diagram

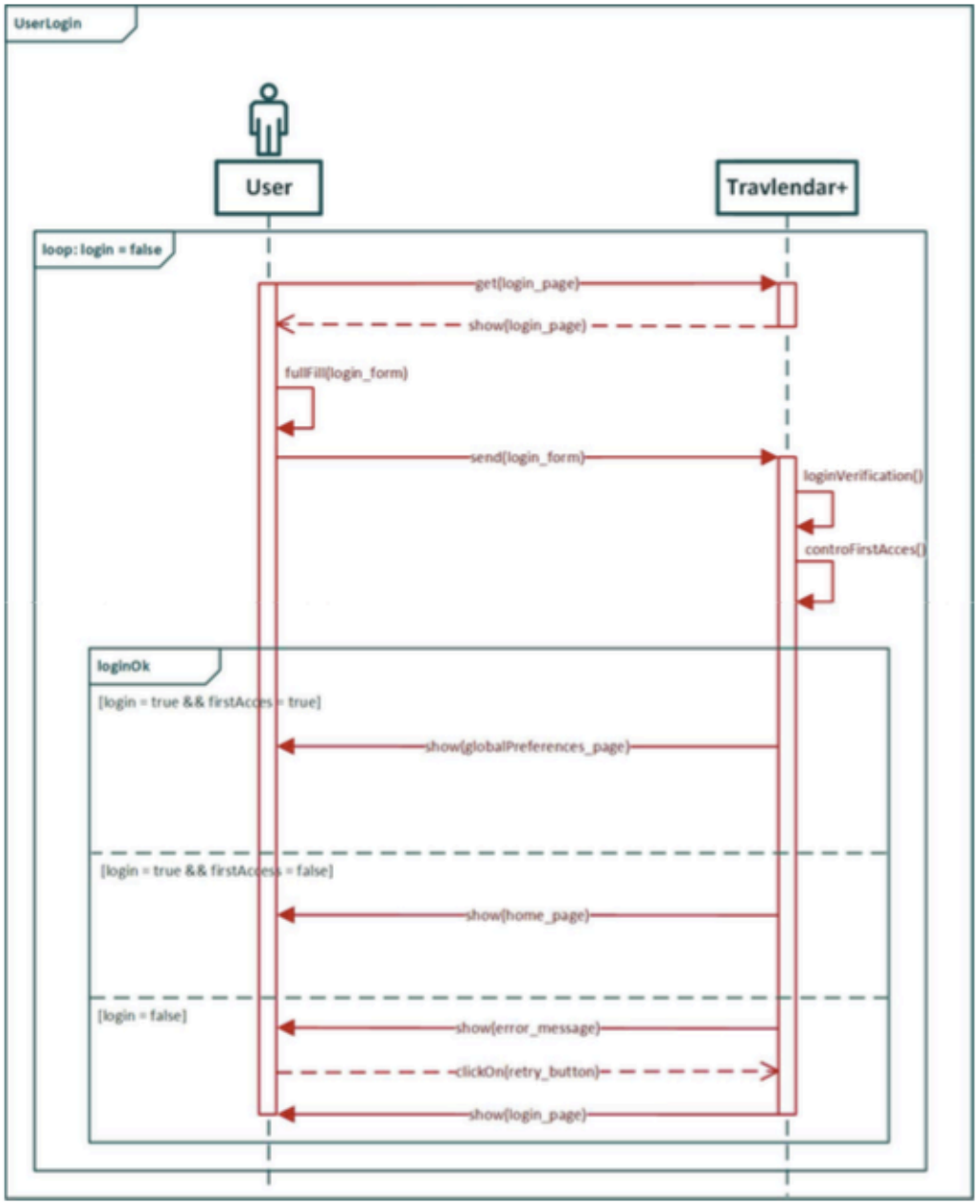
- From the flow of events in the use case or scenario, proceed to the sequence diagram
- A sequence diagram is a graphical description of objects participating in a use case or scenario using a DAG (directed acyclic graph) notation
- Relation to object identification:
  - Objects/classes have already been identified during object modeling
  - Other objects are identified as a result of dynamic modeling
- Heuristic:
  - An event always has a sender and a receiver
  - The representation of the event is sometimes called a message
  - Find sender and receiver for each event $\Longrightarrow$ These are the objects participating in the use case


: after reform
: WebCoursePlanner
This is the S2B!
1: login()
2:
opt
3: createStudyPlan()
4:
: StudyPlan
[logged in]
5:
loop
[need more credits]
alt
6: viewList()
[with list]
7:
[else]
8: search()
9:
10: addExam()
11: addExam()
12:
13:
: Subscription
14:

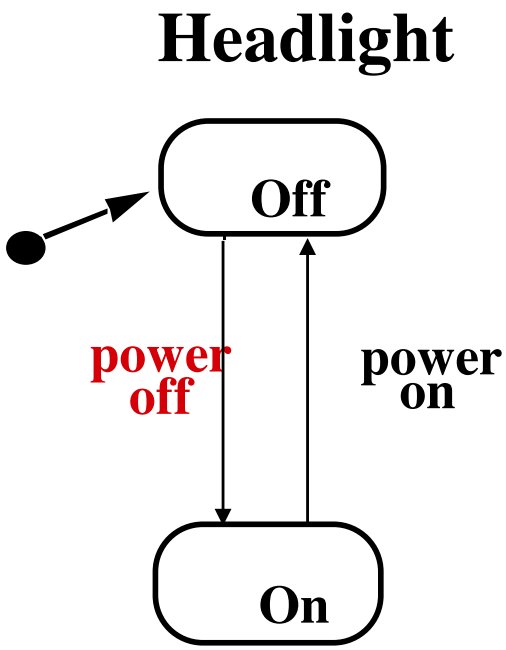
An example of incorrect sequence diagram



Another example of incorrect diagram

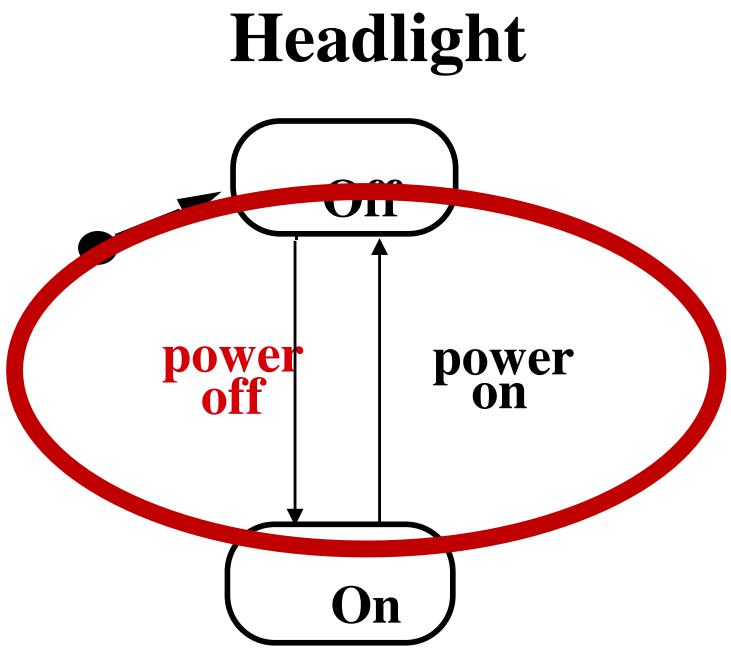


An example of state diagram: Toy car



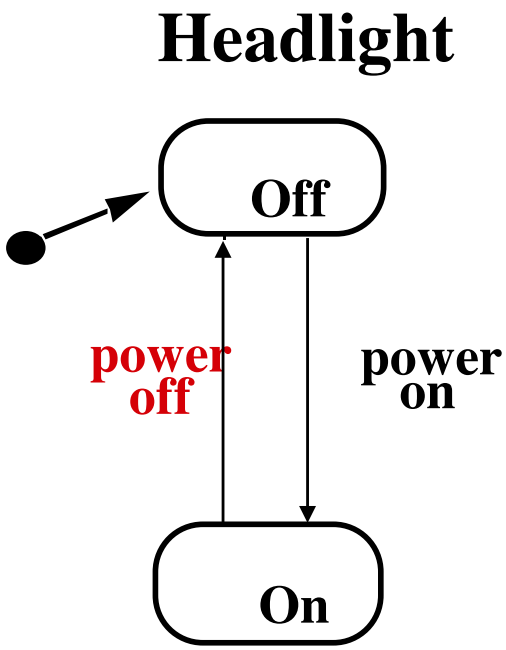
Do you see any problems with this model?

An example of state diagram: Toy car

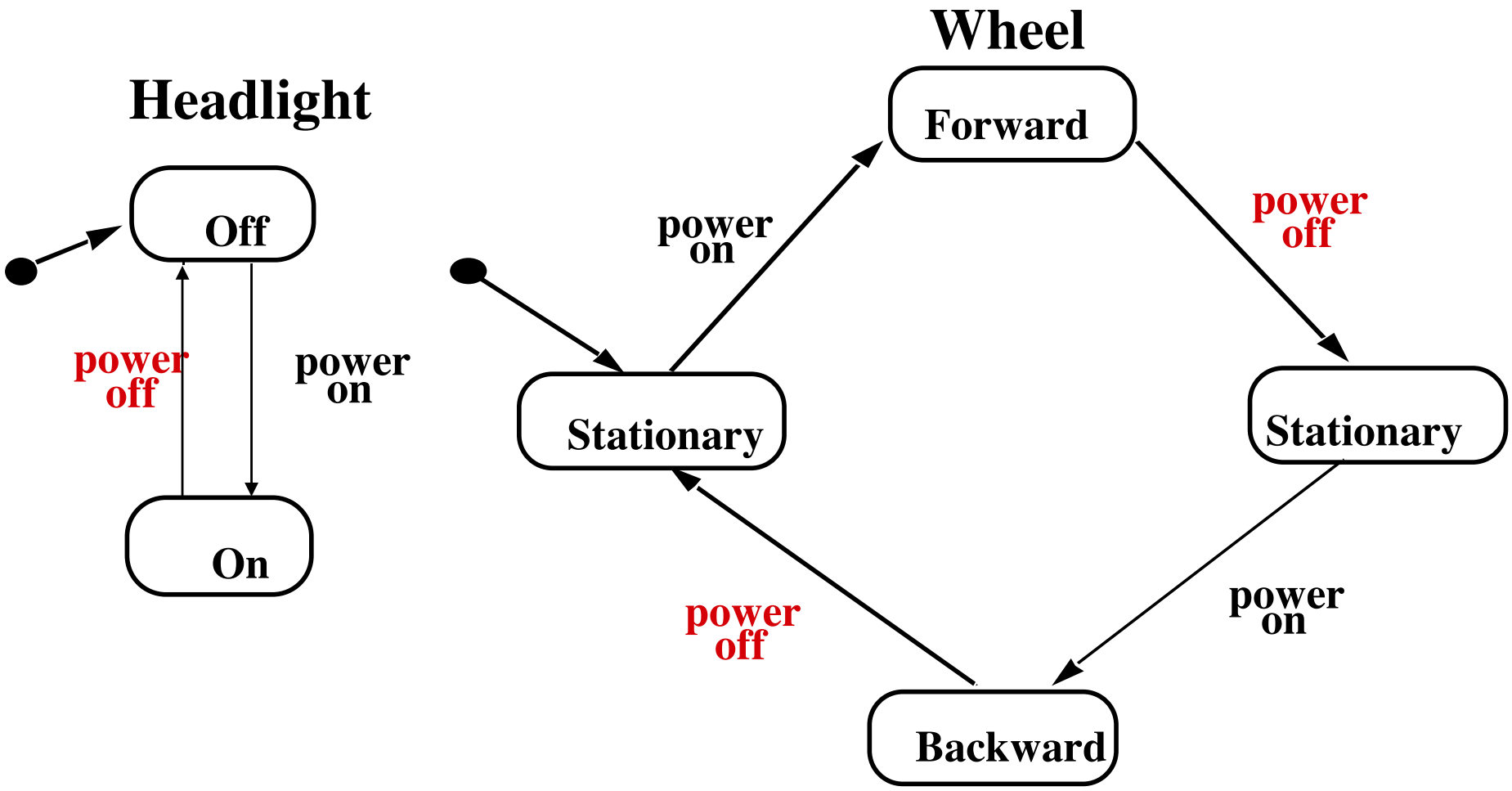


Do you see any problems with this model?

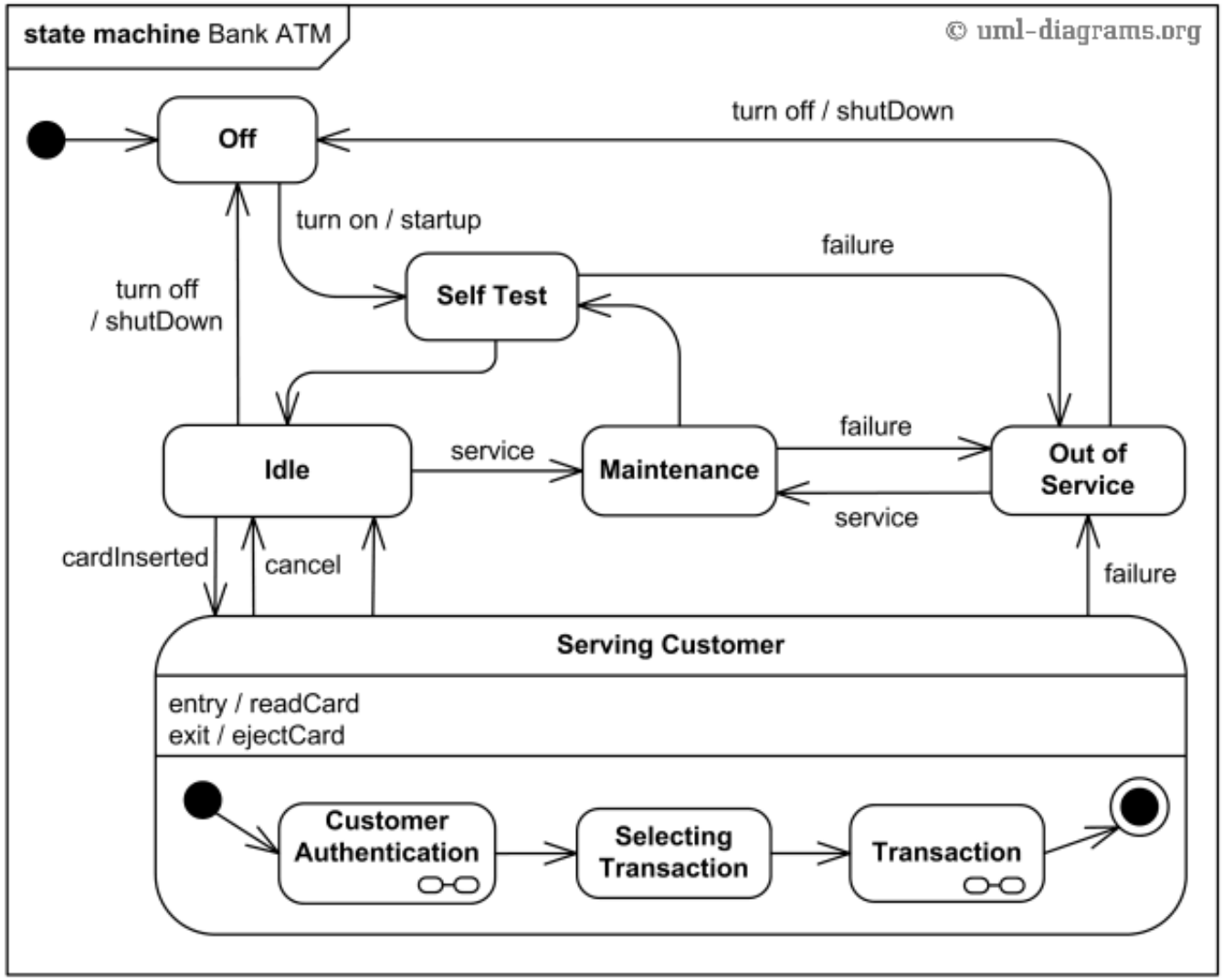
An example of state diagram: Toy car



An example of state diagram: Toy car



Another example of state diagram

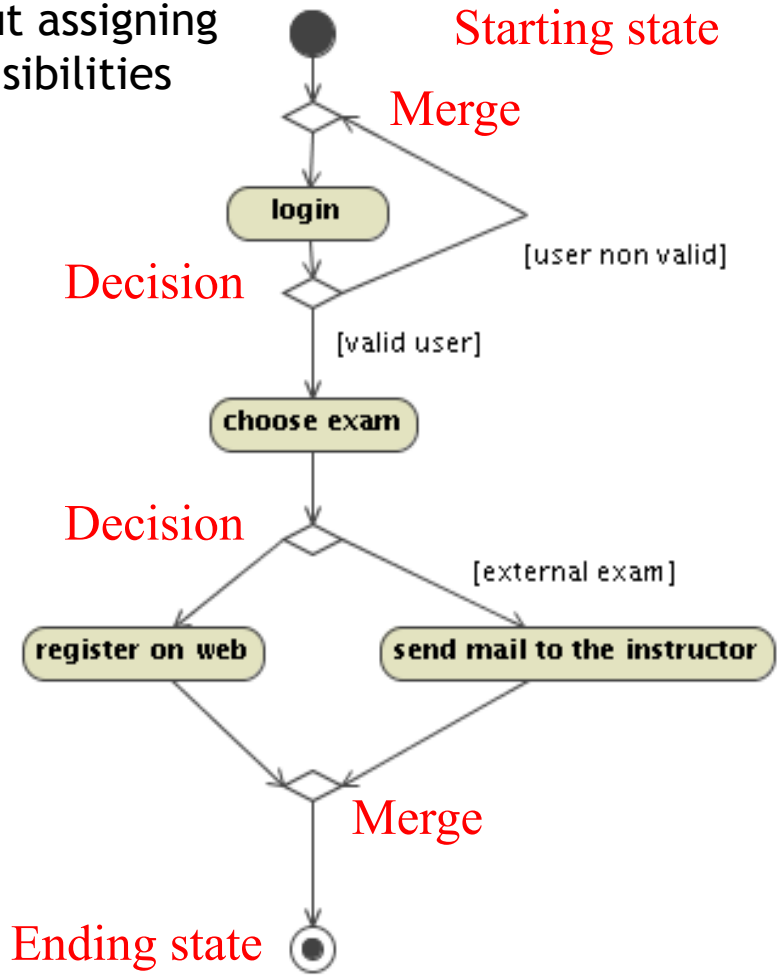


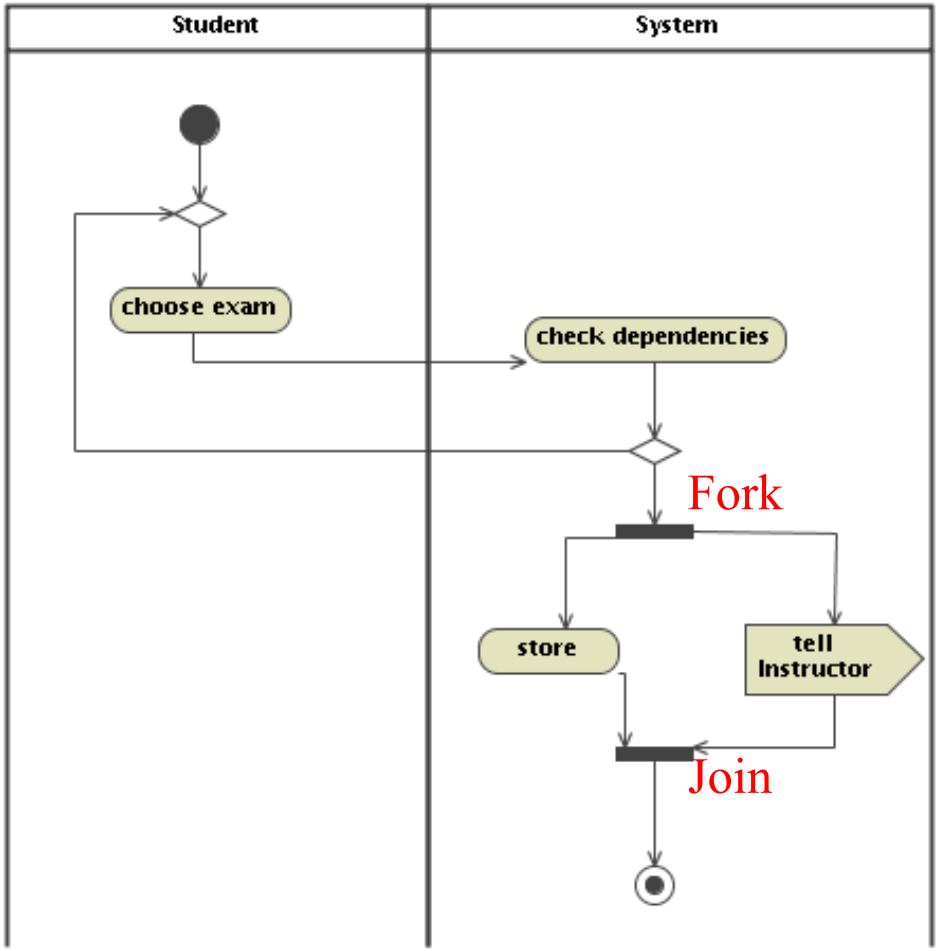
State diagram vs sequence diagram

State diagram	Sequence diagram
Focus on changes in an individual object over time	Focus on the interaction between objects over time
Class-level documentation	Instance-level documentation
We can infer many possible event and order-dependent behaviors	Shows one specific case or a subset of cases

Activity diagrams

Describes an activity without assigning responsibilities





Describes "register on the web" by highlighting the responsibilities of the student and of the system

Sequence diagram vs activity diagram

Sequence diagram	Activity diagram
Focus on object interaction	Focus on activities and flow of activities
Suited for describing an interaction protocol	Suited for describing a process

Tips for dynamic modeling

- Construct dynamic models only for classes with significant dynamic behavior
  - Avoid “analysis paralysis”
- Consider only relevant attributes
- Look at the granularity of the application when deciding on actions and activities
- Reduce notational clutter
  - Try to put actions into state boxes (look for identical actions on events leading to the same state)

Summary: Requirements analysis

- What are the input/output transformations?  **Functional Modeling**
  - Create scenarios and use case diagrams
    - Talk to client, observe, get historical records, do thought experiments
- What is the structure of the world?  **Object Modeling**
  - Create class diagrams [static information models]
    - Identify objects
    - What are the associations between them?
    - What is their multiplicity?
    - What are the attributes of the objects?
    - What operations are defined on the objects?
  - Is there any state change in an object that is to be defined explicitly? If yes, create state diagrams [dynamic class behavior models]

Summary: Requirements analysis (cont)

- How is the expected interaction between the S2B and the environment?

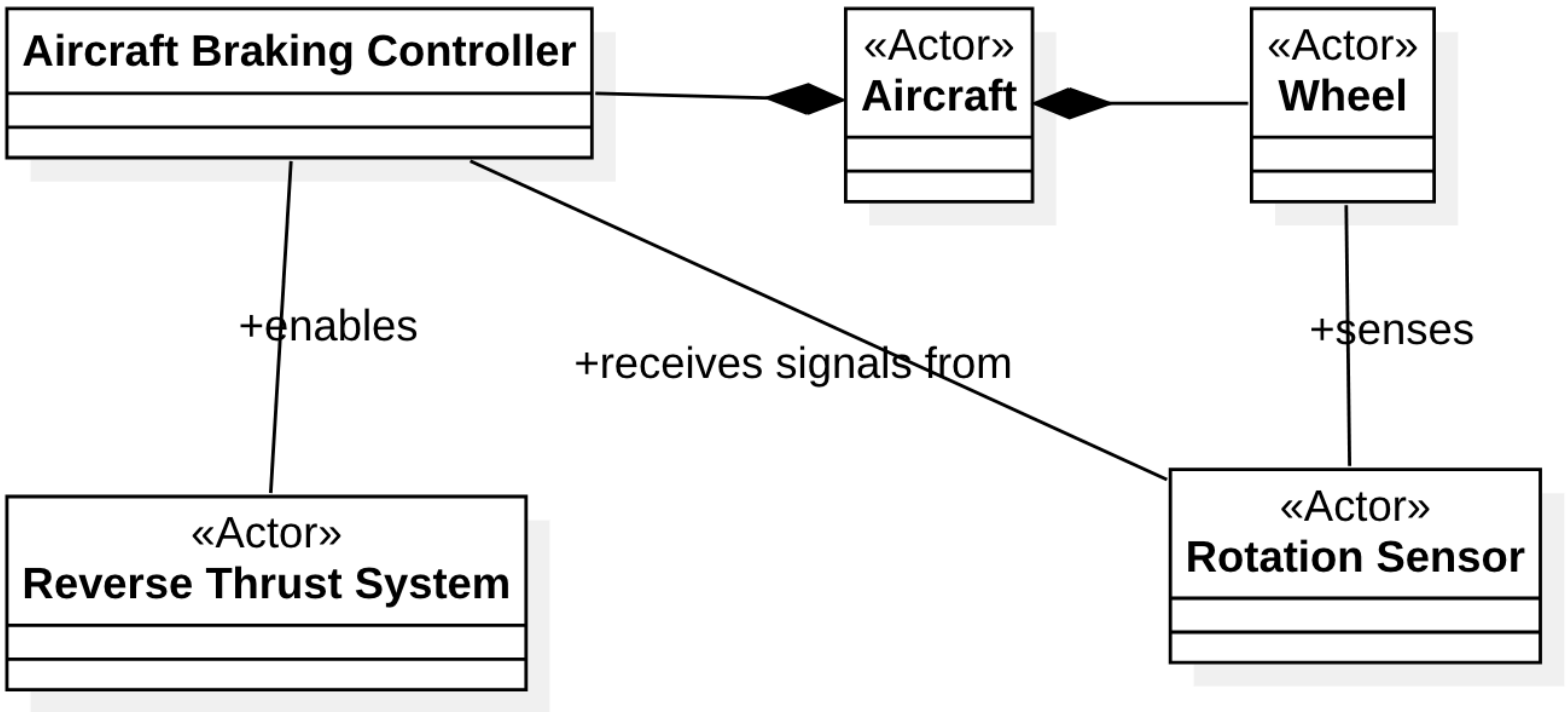
Dynamic Modeling

  - Create sequence diagrams from use cases [dynamic object behavior instance examples]
    - Identify event senders and receivers
    - Show sequence of events exchanged between objects Identify event dependencies and event concurrency
  - Create activity diagrams when you want to highlight important processes [workflow]

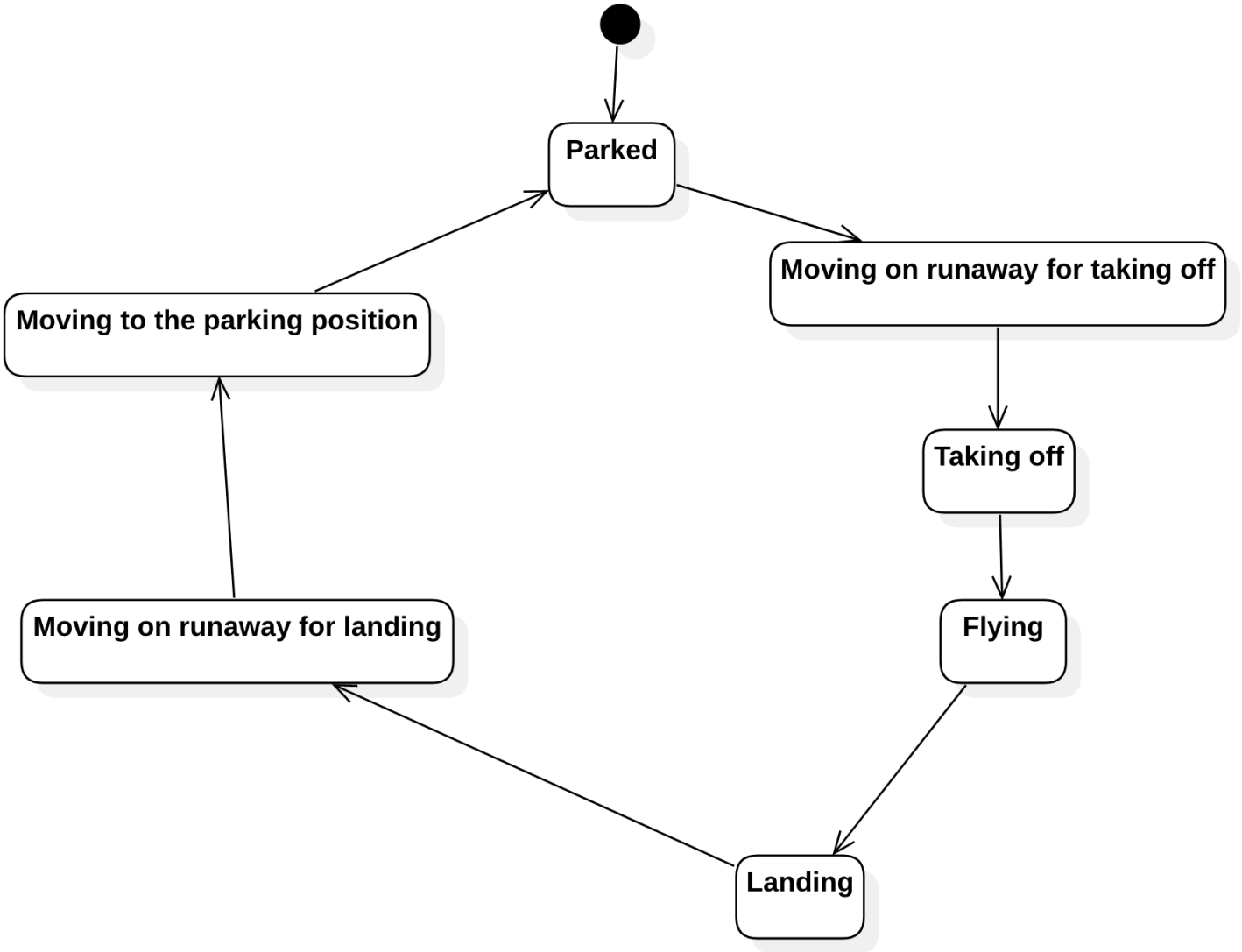
Modeling the Airbus braking logic with UML

What can we model?
What not?

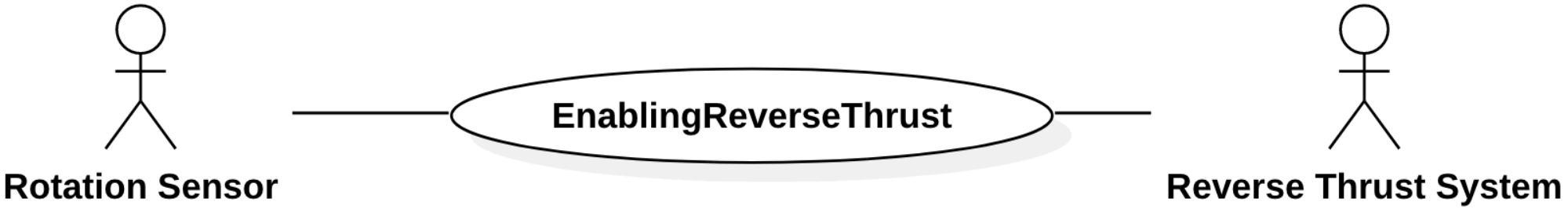
Domain modeling: main entities and relationships

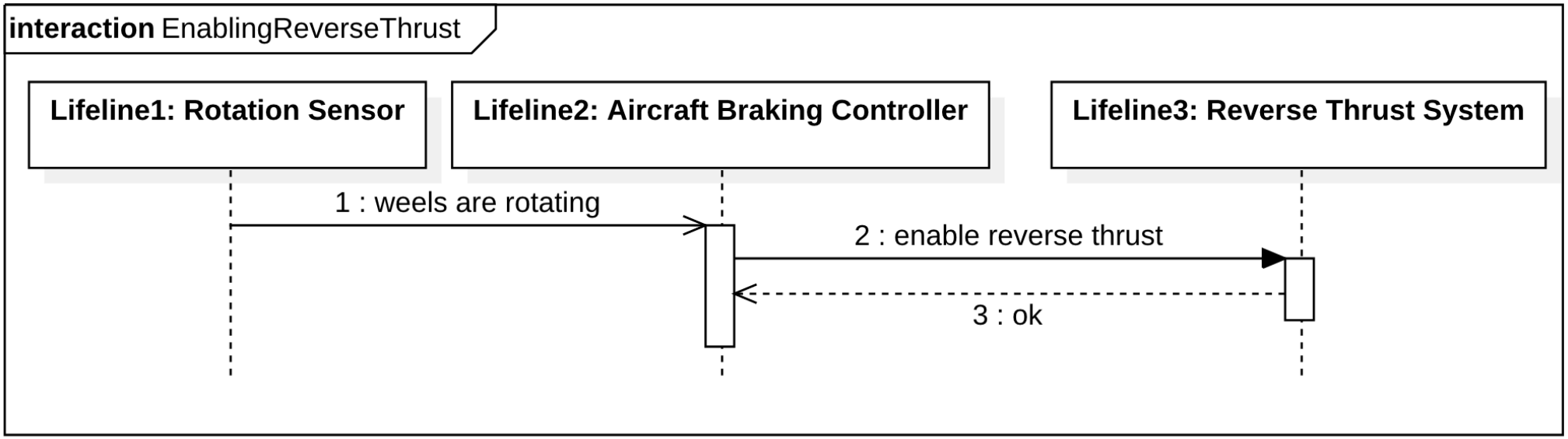


Domain modeling: states of an aircraft



Dynamic behavior: enabling reverse thrust when wheels are rotating





Is the UML spec complete?

- We have described all phenomena
  - Aircraft, wheels, sensor, reverse thrust system
- … and a use case EnablingReverseThrust
- Are we missing something?
- Are we representing goals, domain properties and requirements?
  - Goal
    - Reverse_enabled ⇔ Moving_on_runway
  - Domain properties
    - Wheel_pulses_on ⇔ Wheels_turning
    - Wheels_turning ⇔ Moving_on_runway
  - Requirements
    - Reverse_enabled ⇔ Wheels_pulses_on

Is the UML spec complete?

- Pure UML does not help us in expressing assertions
- We can complement its usage with
  - Some formal or informal description of these assertions

References and interesting sources

- M. Jackson, P. Zave, "Deriving Specifications from Requirements: An Example", Proceedings of ICSE 95, 1995
- M. Jackson, P. Zave, "Four Dark Corners of Requirements Engineering",TOSEM, 1997
- B. Nuseibeh, S. Easterbrook, "Requirements Engineering: A Roadmap", Proceedings ICSE 2000
- A. van Lamsweerde, Requirements Engineering: From System Goals to UML Models to Software Specifications, Wiley and Sons, 2009
- M. Jackson, Software Requirements and Specifications: A Lexicon of Practice, Principles and Prejudices, ACM Press Books, 1995
- S. Robertson and J. Robertson, Mastering the Requirements Process, Addison Wesley, 1999
- Requirements Engineering Specialist Group of the British Computer Society http://www.resg.org.uk/
- B. Bruegge & A.H. Dutoit, Object-Oriented Software Engineering: Using UML, Patterns, and Java, 2nd Edition, Prentice Hall, Upper Saddle River, NJ, September 25, 2003
- T. E. Bell and T. A. Thayer. 1976. Software requirements: Are they really a problem?. In Proceedings of the 2nd international conference on Software engineering (ICSE '76). IEEE Computer Society Press, Los Alamitos, CA, USA, 61-68.
- F. P. J. Brooks, "No Silver Bullet Essence and Accidents of Software Engineering," in Computer, vol. 20, no. 4, pp. 10-19, April 1987. doi: 10.1109/MC.1987.1663532
- Slides by Emmanuel Letier UCL