Dynamic Modeling

Lecture 8

Analysis model

1. Entity object

   (persistent information)

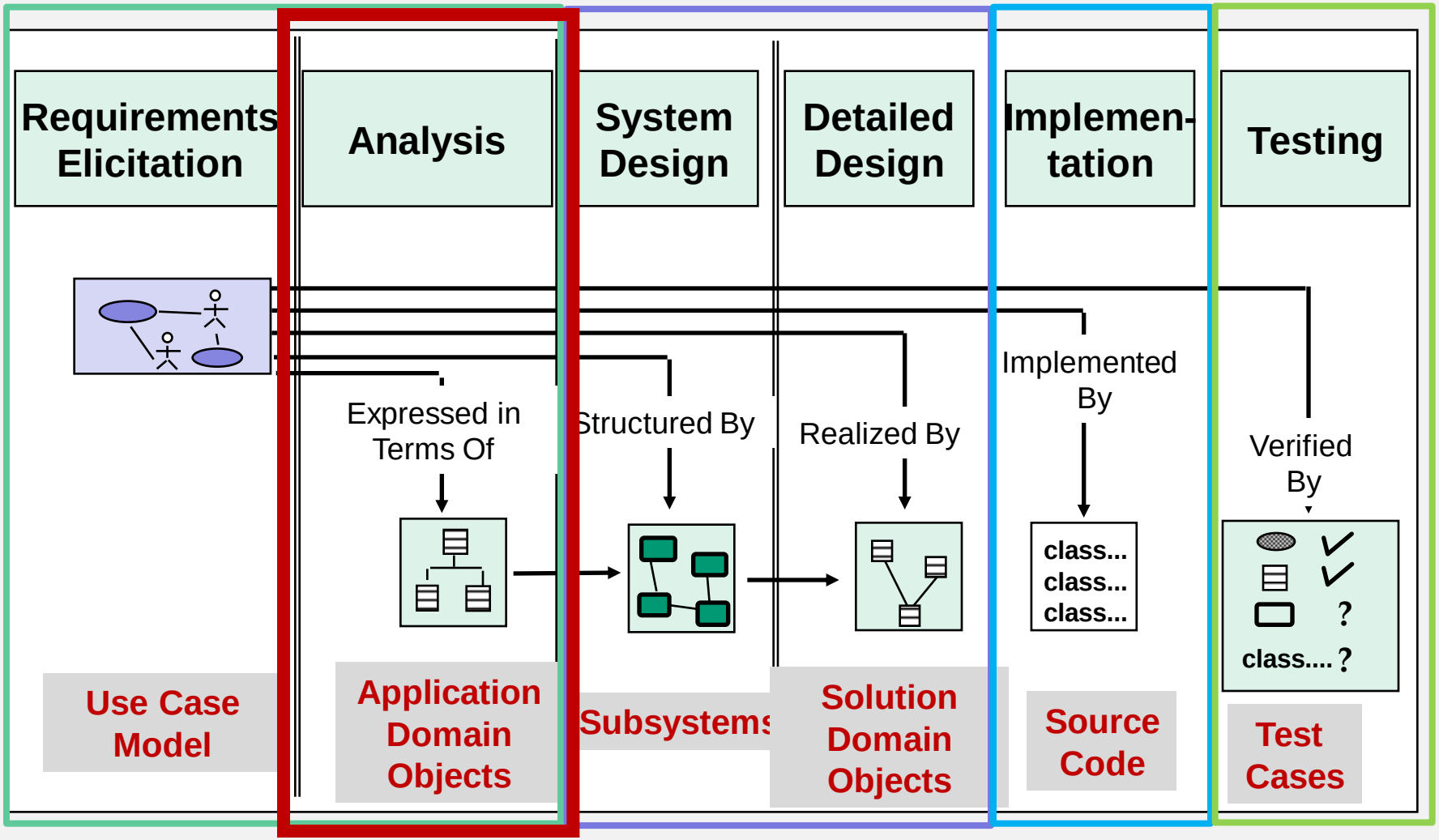
2. Boundary object

   (interaction: interface, forms, notices and message)

3. Control object

   (control task: one control object per object in the use case)

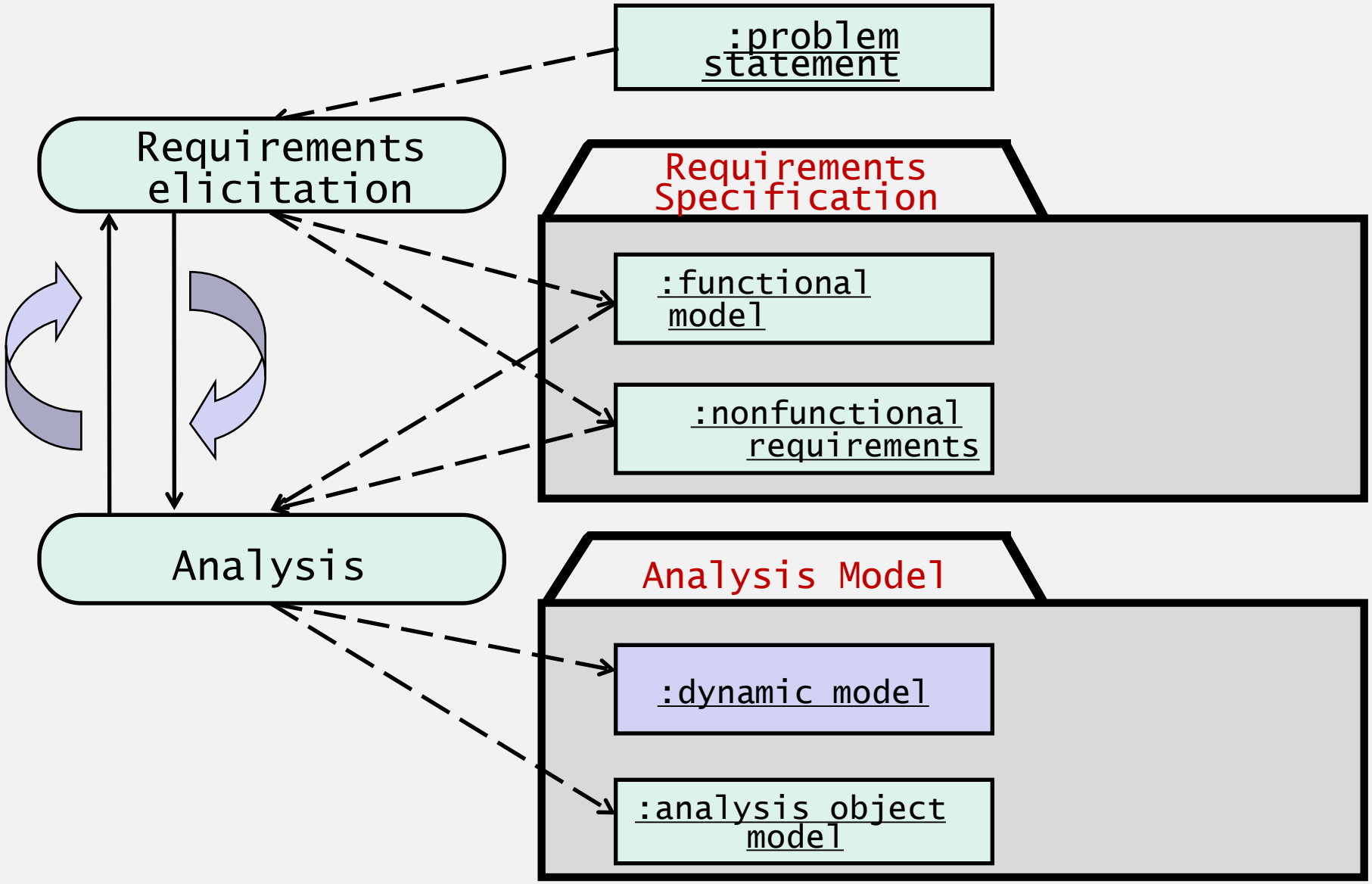
Software Lifecycle Activities



Analysis Activities: From Use Cases to Objects

1	Identifying Entity Objects
2	Identifying boundary Objects
3	Identifying Control Objects
4	Mapping use case to Object with Seq Diagram
5	Modeling interaction among object with CRC
6	Identifying Association
7	Identifying Aggregation
8	Identifying Attributes
9	Modeling State-Dependent Behavior of Individual Objects

Requirements Process



Dynamic Modeling with UML

Definition of a dynamic model:

- Describes the components of the system that have interesting dynamic behavior

Two UML diagrams types for dynamic modeling:

- Interaction diagrams describe the dynamic behavior *between* objects
- Statechart diagrams describe the dynamic behavior *of a single object.*

Purpose:

- Detect and supply operations for the object model.

UML Interaction Diagrams

Two types of interaction diagrams:

- Sequence Diagram:
  - Describes the dynamic behavior of several objects over time
  - Good for real-time specifications
- Collaboration Diagram:
  - Shows the sequential relationship among objects
  - Position of objects is based on the position of the classes in the UML class diagram.
  - Does not show time.

UML State Chart Diagram

State Chart Diagram:

A state machine that describes the response of an object of a given class to the receipt of outside stimuli.

Activity Diagram:

A special type of state chart diagram, where all states are action states.

How do we detect Operations?

We look for objects

- who are interacting and extract their “protocol”
- who have interesting behavior on their own

Good starting point: Flow of events in a use case description

Event is something that happens at a point in time

An event sends information from one object to another

Sequence Diagram

A sequence diagram is a graphical description of the objects participating in a use case.

Heuristic for finding participating objects:

- A event always has a sender and a receiver
- Find them for each event => These are the objects participating in the use case.

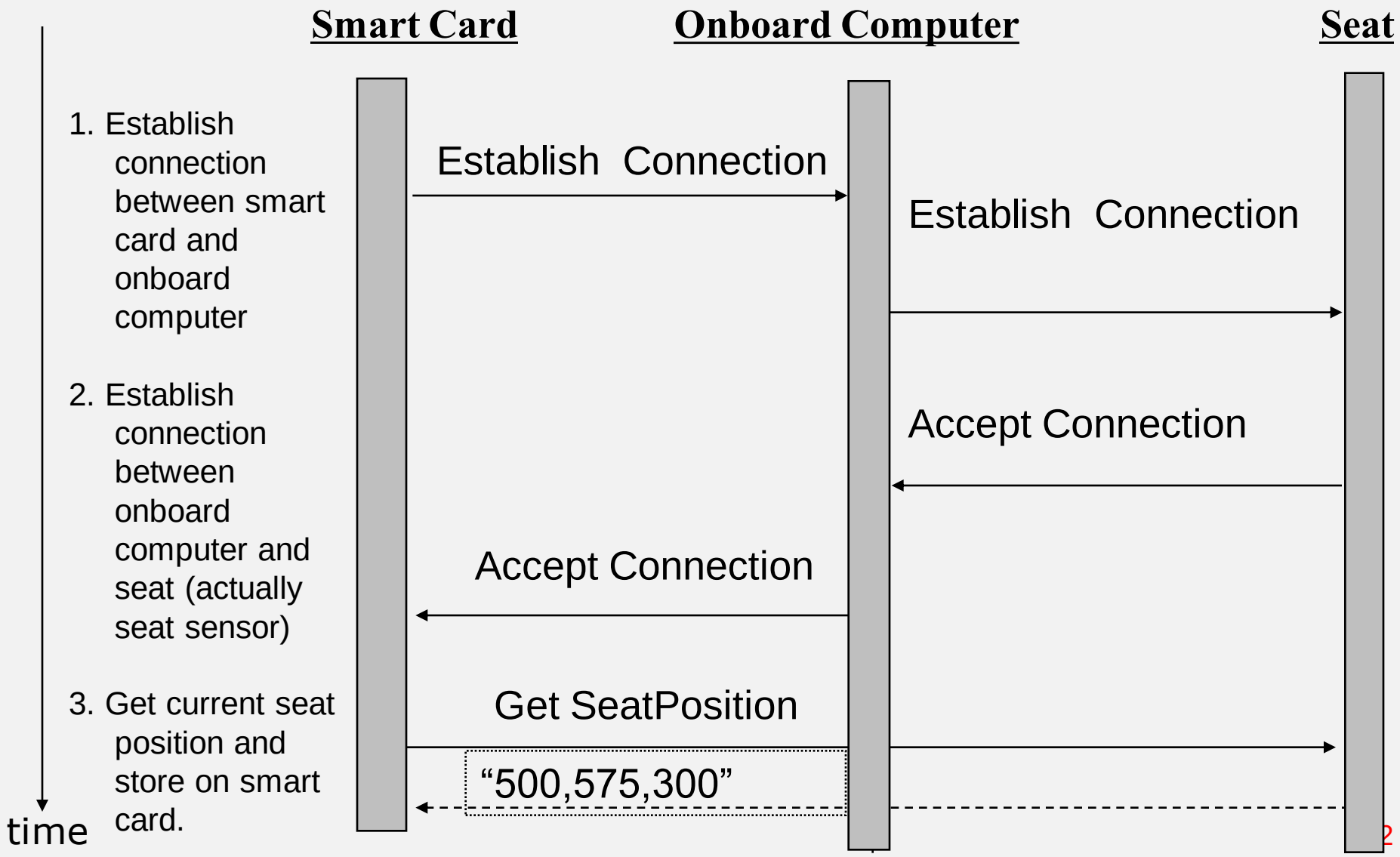
An Example

Flow of events in “Get SeatPosition” use case :

1. Establish connection between smart card and onboard computer
2. Establish connection between onboard computer and sensor for seat
3. Get current seat position and store on smart card

Where are the objects?

Sequence Diagram for "Get SeatPosition"



Heuristics for Sequence Diagrams

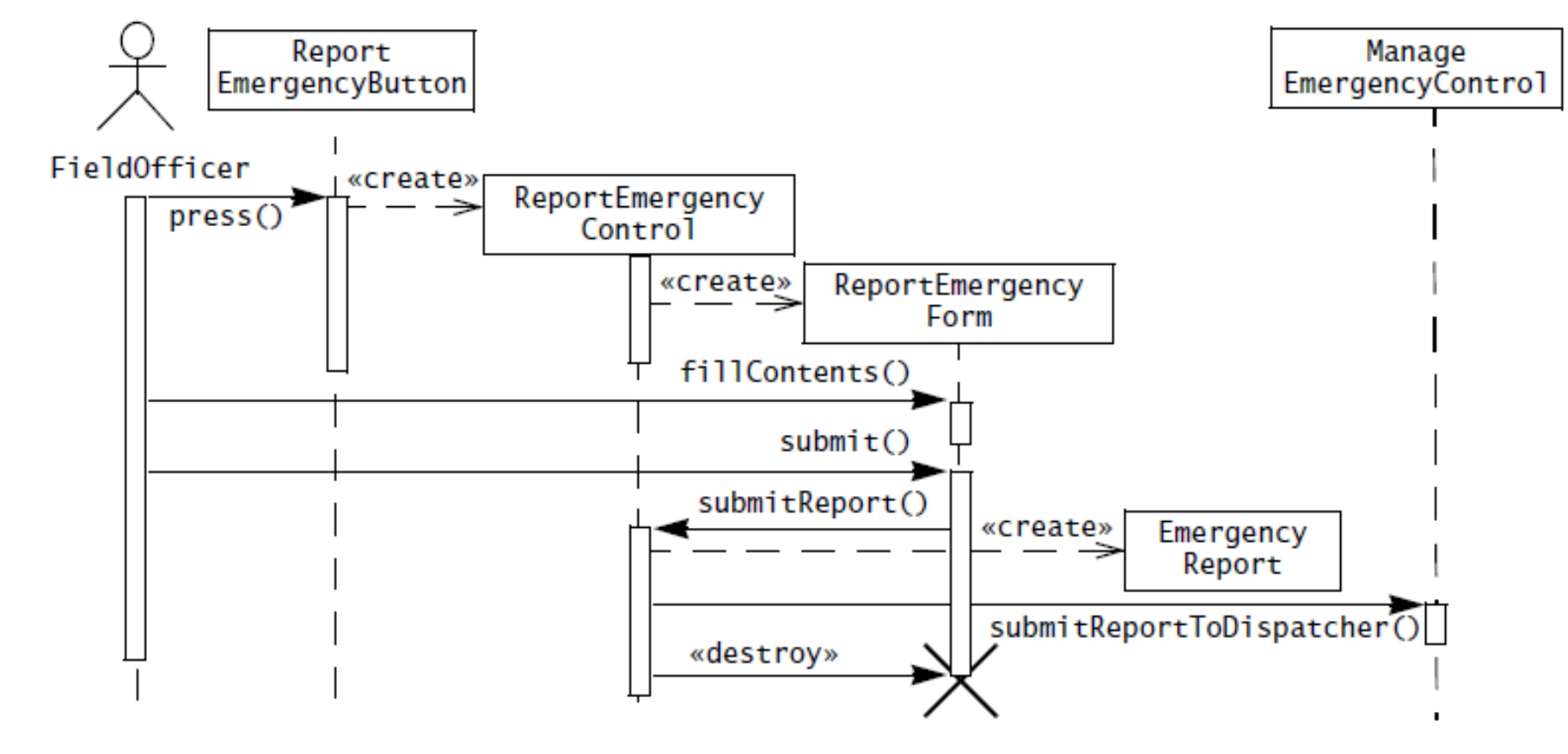
- Layout:
  1st column: Should be the actor of the use case
  2nd column: Should be a boundary object
  3rd column: Should be the control object that manages the rest of the use case

- Creation of objects:
  - Create control objects at beginning of event flow
  - The control objects create the boundary objects
- Access of objects:
  - Entity objects can be accessed by control and boundary objects
  - Entity objects should not access boundary or control objects.

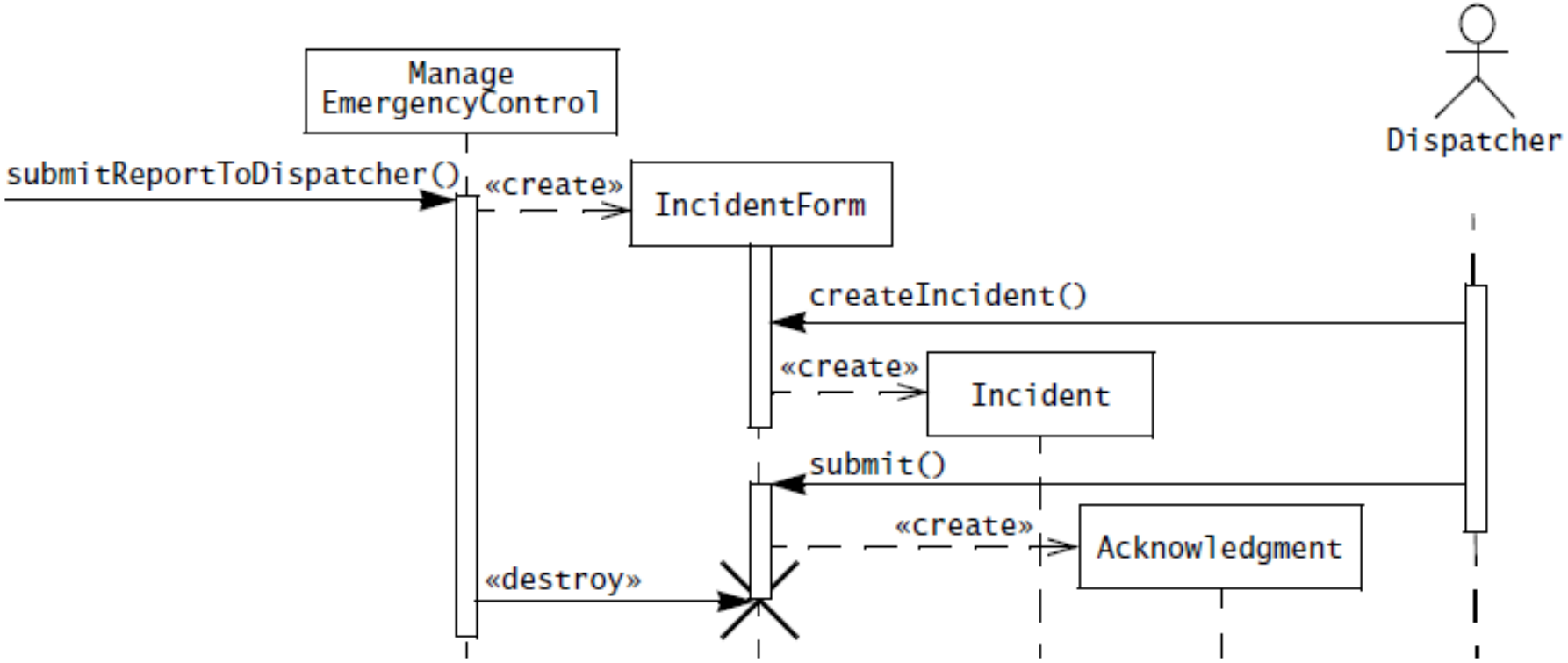
One use case

Use Case Name		Report Emergency
Entry Condition	1	The FieldOfficer activates the “Report Emergency” function of her terminal.
Flow of Event	2	FRIEND responds by presenting a form to the officer. The form includes an emergency type menu (general emergency, fire, transportation), a location, incident description, resource request, and hazardous material fields.
	3	The FieldOfficer completes the form by specifying minimally the emergency type and description fields. The FieldOfficer may also describe possible responses to the emergency situation and request specific resources. Once the form is completed, the FieldOfficer submits the form by pressing the “Send Report” button, at which point, the Dispatcher is notified.
	4	The Dispatcher reviews the information submitted by the FieldOfficer and creates an Incident in the database by invoking the OpenIncident use case. All the information contained in the FieldOfficer’s form is automatically included in the incident. The Dispatcher selects a response by allocating resources to the incident (with the AllocateResources use case) and acknowledges the emergency report by sending a FRIENDgram to the FieldOfficer.
Exit condition	5	The FieldOfficer receives the acknowledgment and the selected response.

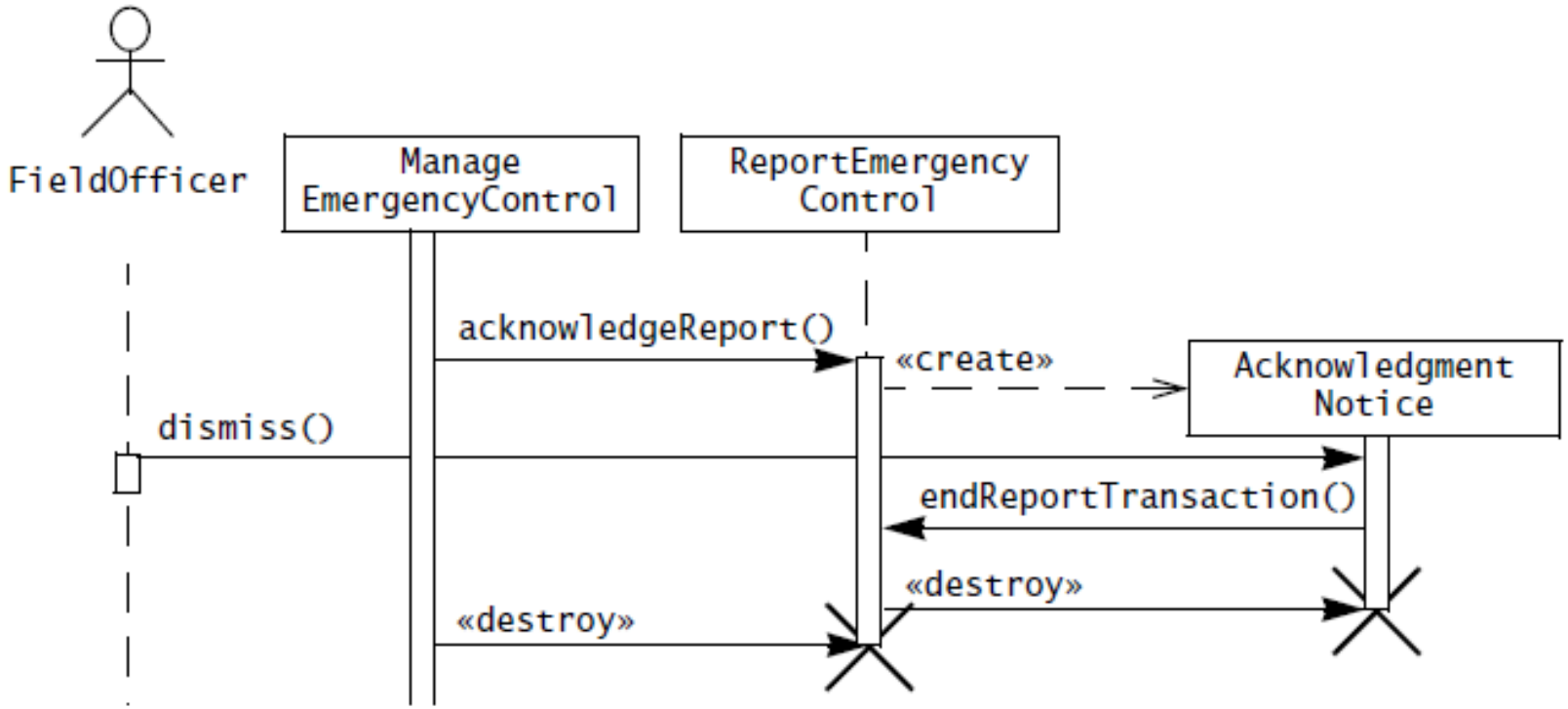
Sequence Diagram: ReportEmergency



Sequence Diagram: ReportEmergency



Sequence Diagram: ReportEmergency



What else can we get out of Sequence Diagrams?

Sequence diagrams are derived from use cases

The structure of the sequence diagram helps us to determine how decentralized the system is

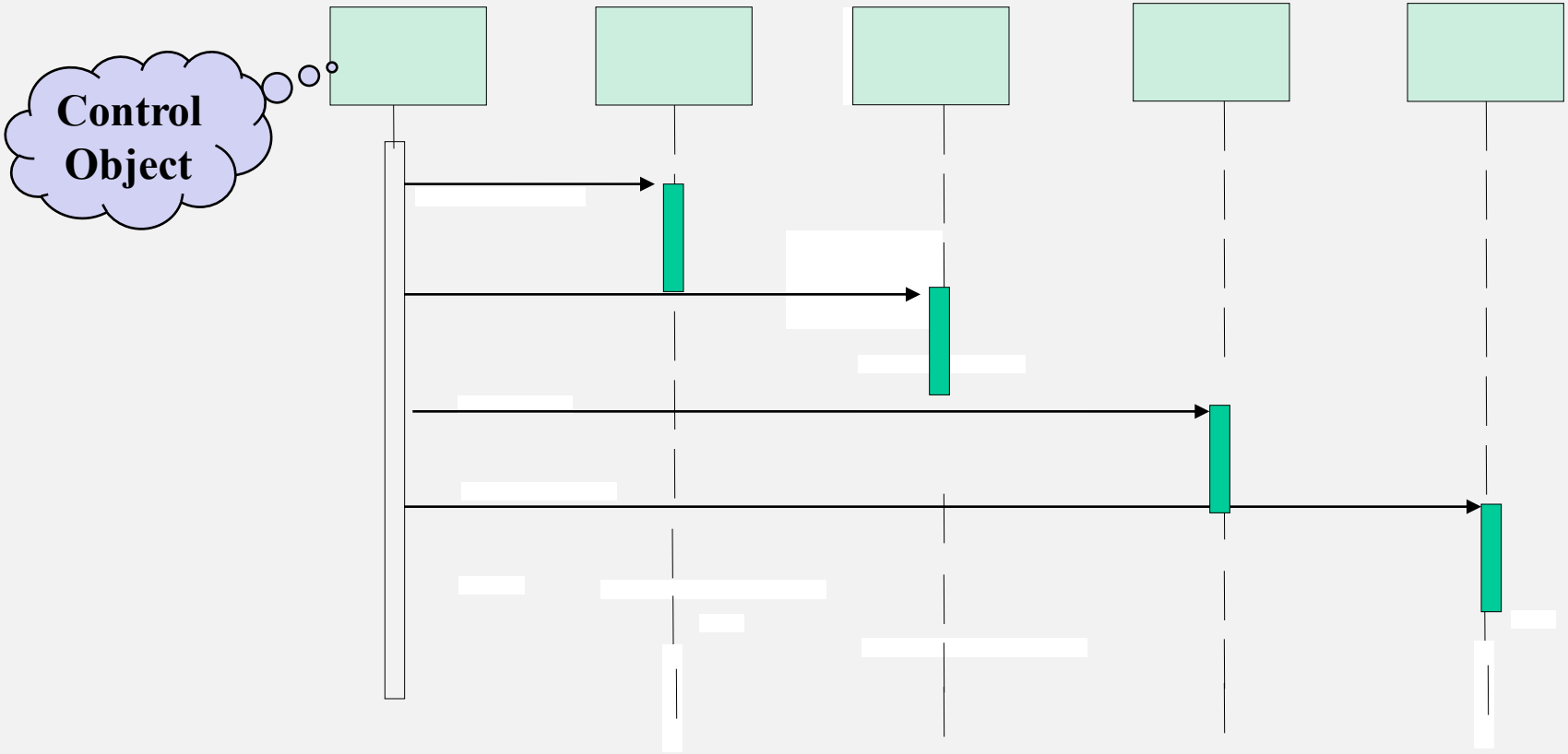
We distinguish two structures for sequence diagrams

Fork Diagrams and Stair Diagrams

Fork Diagram

The dynamic behavior is placed in a single object, usually a control object

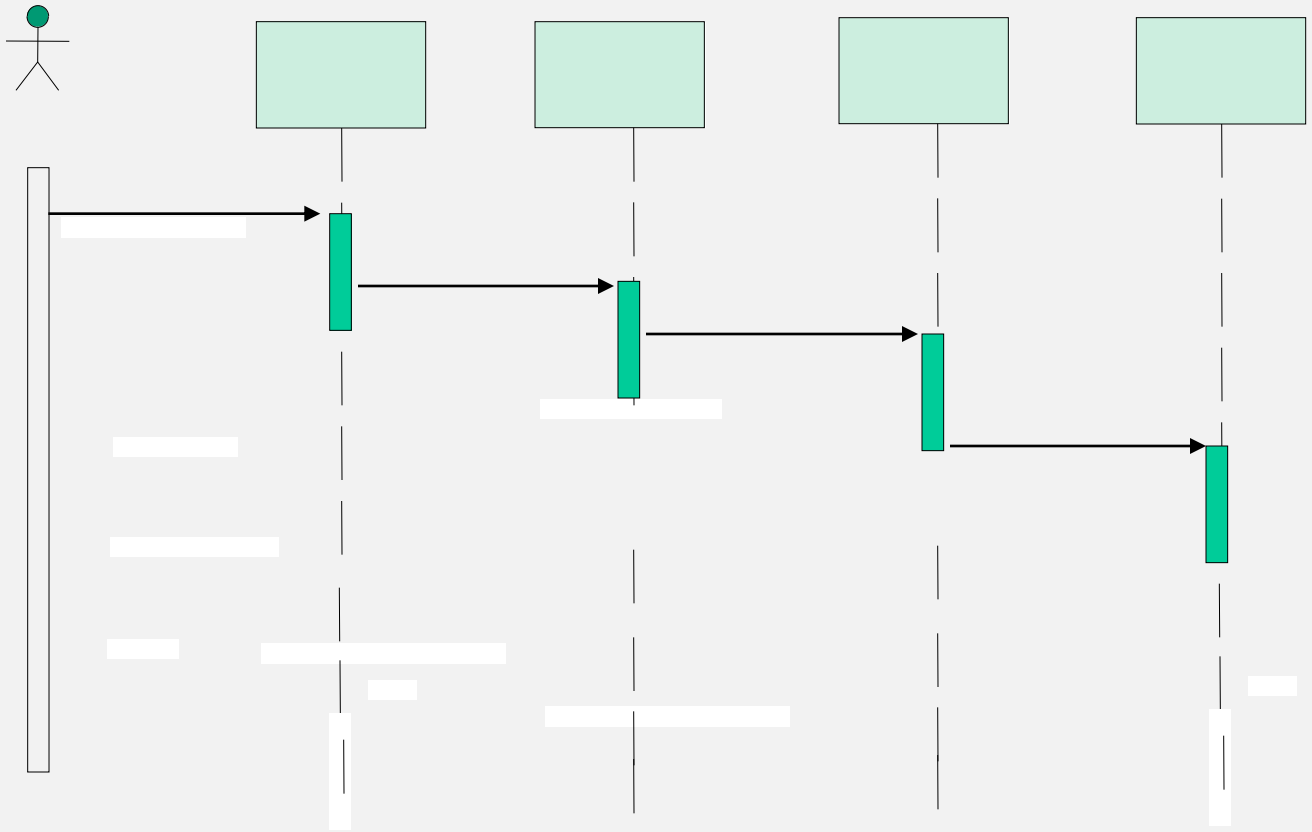
It knows all the other objects and often uses them for direct questions and commands



Stair Diagram

The dynamic behavior is distributed. Each object delegates responsibility to other objects

Each object knows only a few of the other objects and knows which objects can help with a specific behavior



Fork or Stair?

Object-oriented supporters claim that the stair structure is better

Modeling Advice:

Choose the stair - a decentralized control structure - if

- The operations have a strong connection
- The operations will always be performed in the same order

Choose the fork - a centralized control structure - if

- The operations can change order
- New operations are expected to be added as a result of new requirements.

Class-Responsibility-Collaborator (CRC) Modeling

Develop a set of index cards that represent objects

One class per card

Cards are divided into three sections

- class name
- class responsibilities
- class collaborators

Once a complete CRC card set is developed it is reviewed examining the usage scenarios

CRC Card

Class Name	
Super Class	
Sub class	
Responsibility	Collaborators

Criteria CRC Class Inclusion

- Class information (data) should be retained
- Class Provides needed services
- Contains multiple attributes
- Common set of attributes apply to all object instances
- Common set of operations apply to all object instances
- External entities that produce or consume information

Allocating Responsibilities to Classes

- Distribute system intelligence evenly
- State each responsibility as generally as possible
- Information and its related behaviors should reside within the same class
- Localize all information about one entity in a single class
- Share responsibilities among related classes when appropriate

Criteria for Defining Collaborators

Any time a class cannot fulfill a responsibility on its own it needs to interact with another class

A server object interacts with a client object to fulfill some responsibility

One use case

Use Case Name		Report Emergency
Entry Condition	1	The FieldOfficer activates the “Report Emergency” function of her terminal.
Flow of Event	2	FRIEND responds by presenting a form to the officer. The form includes an emergency type menu (general emergency, fire, transportation), a location, incident description, resource request, and hazardous material fields.
	3	The FieldOfficer completes the form by specifying minimally the emergency type and description fields. The FieldOfficer may also describe possible responses to the emergency situation and request specific resources. Once the form is completed, the FieldOfficer submits the form by pressing the “Send Report” button, at which point, the Dispatcher is notified.
	4	The Dispatcher reviews the information submitted by the FieldOfficer and creates an Incident in the database by invoking the OpenIncident use case. All the information contained in the FieldOfficer’s form is automatically included in the incident. The Dispatcher selects a response by allocating resources to the incident (with the AllocateResources use case) and acknowledges the emergency report by sending a FRIENDgram to the FieldOfficer.
Exit condition	5	The FieldOfficer receives the acknowledgment and the selected response.

CRC Card: ReportEmergencyControl

ReportEmergencyControl			
Responsibility		Collaborators	
	Collect input from field officer		Emergency control form
	Control sequence of forms during emergency reporting		Emergency Report
			Acknowledgement notice

CRC Card: Incident

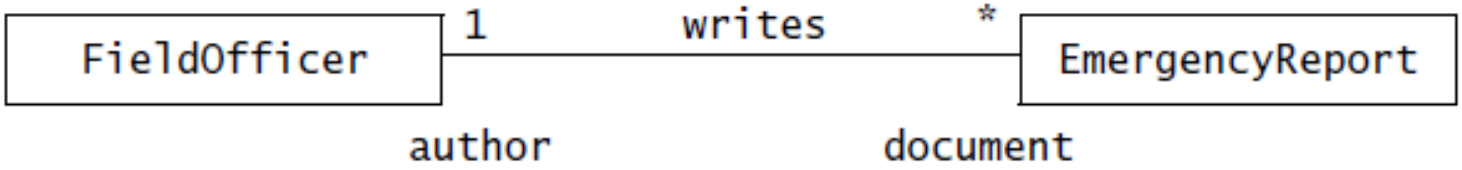
Incident			
Responsibility		Collaborators	
	Track all information related to one incident		Resources

Identifying Association

An Association shows the relationship between two or more classes.

Association has several properties

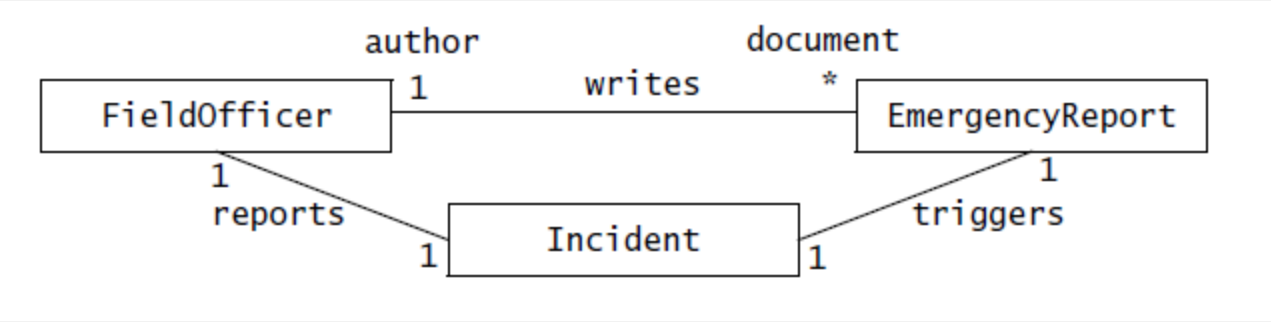
- Name: to describe the association between two classes (writes)
- Role: identify the function of each class wrt association (author,..)
- Multiplicity: identify possible number of instances (1..*)



Heuristic to identify Association

Heuristic to identify Association

- Examine verb phrases.
- Name associations and roles precisely.
- Use qualifiers as often as possible to identify namespaces and key attributes.
- Eliminate any association that can be derived from other associations.
- Do not worry about multiplicity until the set of associations is stable.
- Too many associations make a model unreadable.

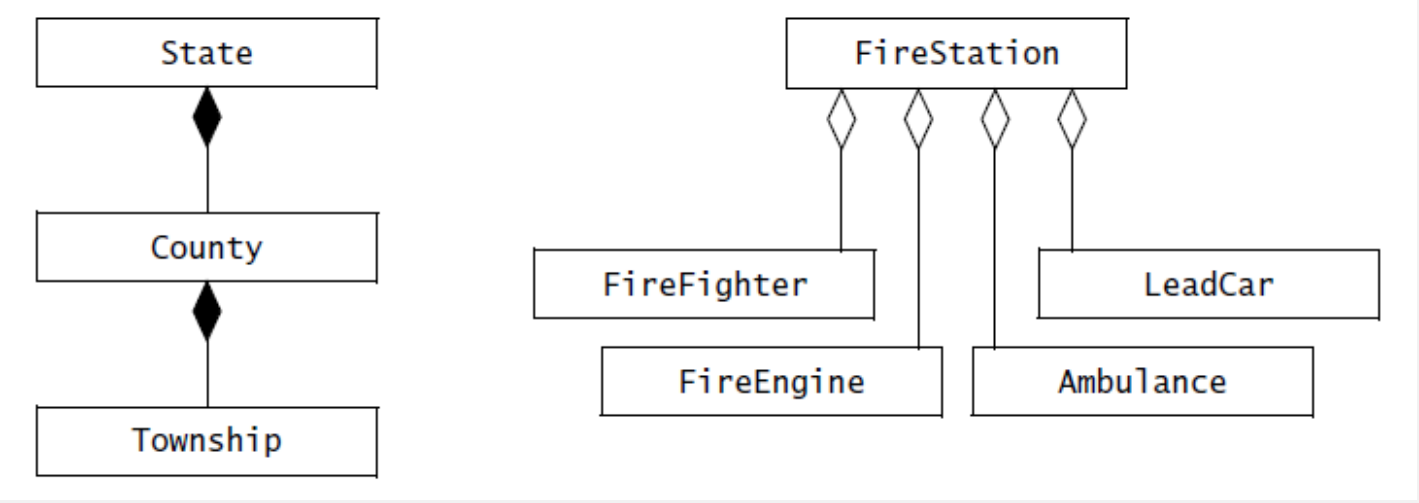


Identifying Aggregation

Aggregation associations are used in the analysis model to denote whole–part concepts.

Aggregation associations add information to the analysis model about how containment concepts in the application domain can be organized in a hierarchy

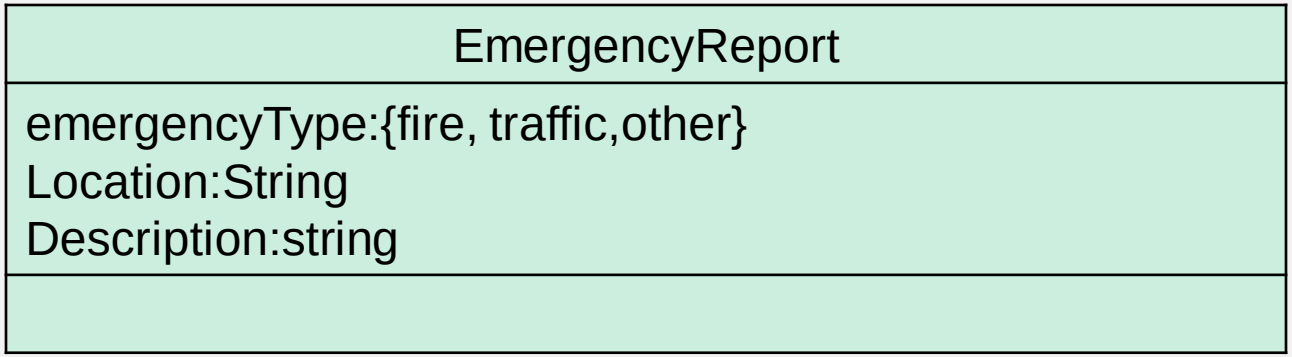
Two types of aggregation, composition and shared.



Identifying Attributes

Attributes are properties of individual objects.

A name and Type



Dynamic Modeling: Statechart

A state chart diagram relates events and states for one class

An object model with several classes with interesting behavior has *a set* of state diagrams

We distinguish between two types of operations:

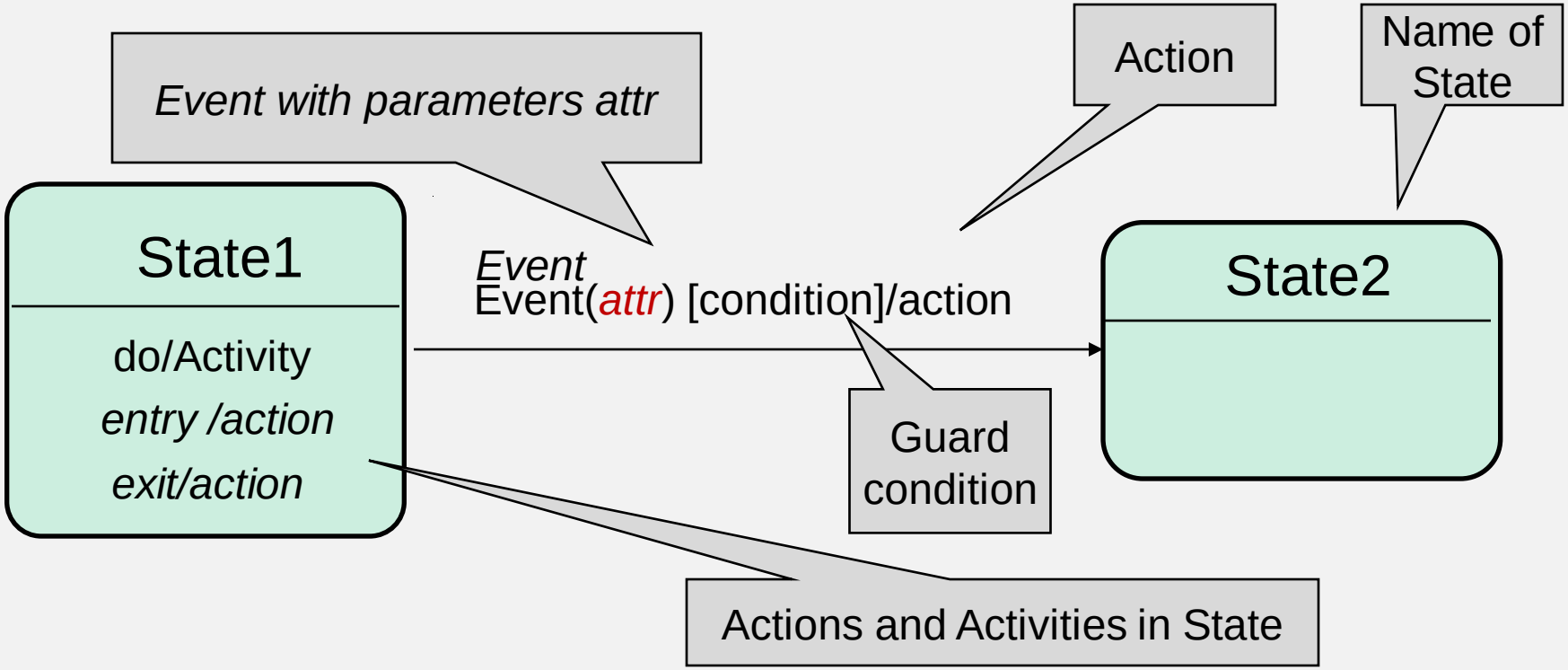
Activity: Operation that takes time to complete

associated with states

Action: Instantaneous operation

associated with events

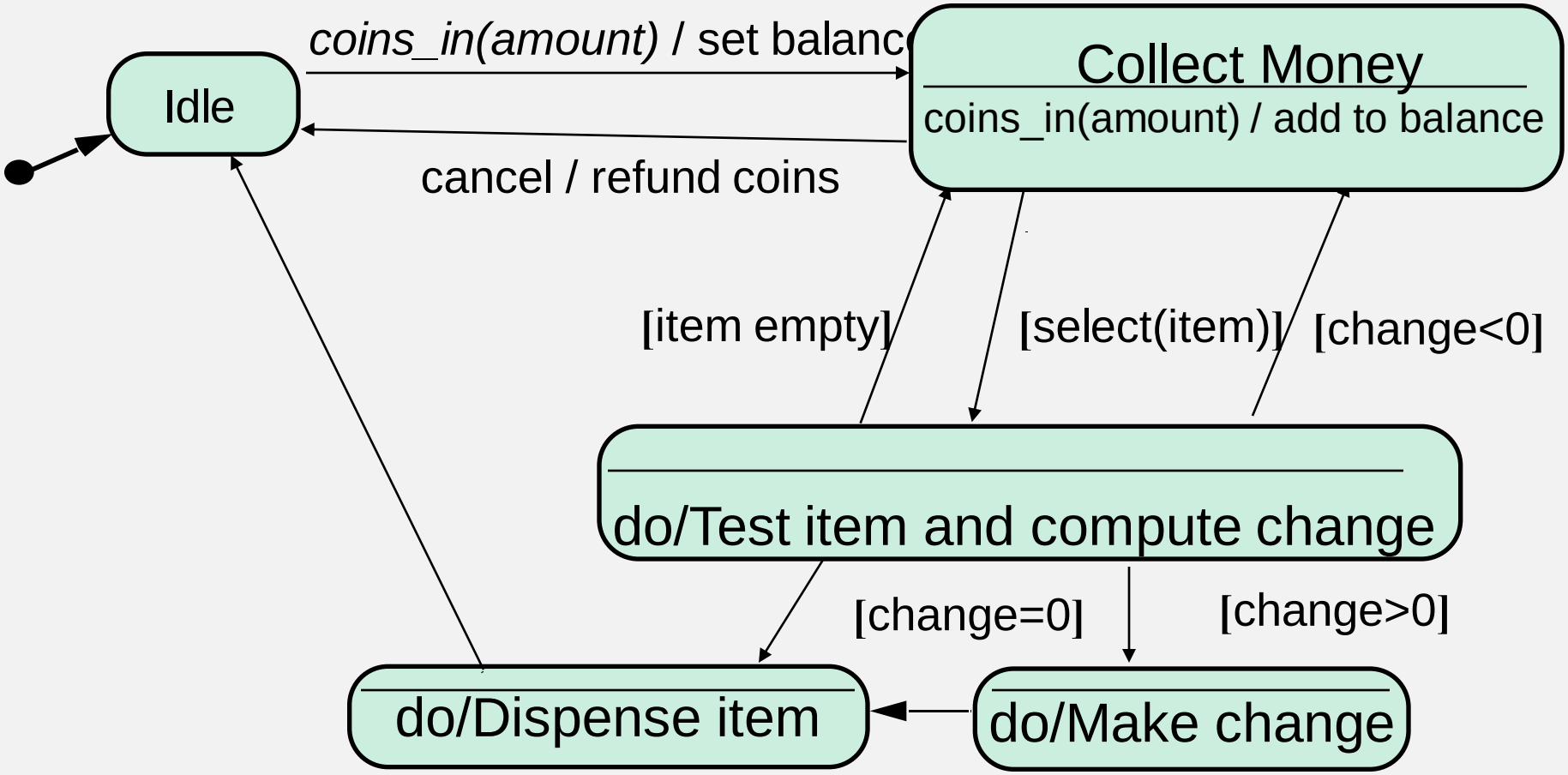
UML Statechart Diagram Notation



Note:

- *Events are italics*
- Conditions are enclosed with brackets: []
- Actions and activities are prefixed with a slash /

Example of a StateChart Diagram



State

An abstraction of the attributes of a class

State is the aggregation of several attributes a class

A state is an equivalence class of all those attribute values and links that do no need to be distinguished

Example: State of a bank

- State has duration

State Chart Diagram vs Sequence Diagram

State chart diagrams help to identify:

- Changes to an individual object over time

Sequence diagrams help to identify:

- The temporal relationship of between objects over time
- Sequence of operations as a response to one ore more events.

Analysis: A Toy Example

Analyze the problem statement

- Identify functional requirements
- Identify nonfunctional requirements
- Identify constraints (pseudo requirements)

Build the functional model:

- Develop use cases to illustrate functional requirements

Build the dynamic model:

- Develop sequence diagrams to illustrate the interaction between objects
- Develop state diagrams for objects with interesting behavior

Build the object model:

- Develop class diagrams for the structure of the system

Problem Statement: Direction Control for a Toy Car

Power is turned on

- Car moves forward and car headlight shines

Power is turned off

- Car stops and headlight goes out.

Power is turned on

- Headlight shines

Power is turned off

- Headlight goes out

Power is turned on

- Car runs backward with its headlight shining

Power is turned off

- Car stops and headlight goes out

Power is turned on

- Headlight shines

Power is turned off

- Headlight goes out

Power is turned on

- Car runs forward with its headlight shining

Find the Functional Model: Use Cases

Use case 1: System Initialization

Entry condition: Power is off, car is not moving

Flow of events:

1. Driver turns power on

Exit condition: Car moves forward, headlight is on

Use case 2: Turn headlight off

Entry condition: Car moves forward with headlights on

Flow of events:

1. Driver turns power off, car stops and headlight goes out.
2. Driver turns power on, headlight shines and car does not move.
3. Driver turns power off, headlight goes out

Exit condition: Car does not move, headlight is out

Use Cases continued

Use case 3: Move car backward

Entry condition: Car is stationary, headlights off

Flow of events:

1. Driver turns power on

Exit condition: Car moves backward, headlight on

Use case 4: Stop backward moving car

Entry condition: Car moves backward, headlights on

Flow of events:

1. Driver turns power off, car stops, headlight goes out.
2. Power is turned on, headlight shines and car does not move.
3. Power is turned off, headlight goes out.

Exit condition: Car does not move, headlight is out

Use Cases Continued

Use case 5: Move car forward

Entry condition: Car does not move, headlight is out

Flow of events

1. Driver turns power on

Exit condition:

Car runs forward with its headlight shining

Use Case Pruning

Do we need use case 5?

Let us compare use case 1 and use case 5:

Use case 1: System Initialization

Entry condition: Power is off, car is not moving

Flow of events:

1. Driver turns power on

Exit condition: Car moves forward, headlight is on

Use case 5: Move car forward

Entry condition: Car does not move, headlight is out

Flow of events

1. Driver turns power on

Exit condition:

Car runs forward with its headlight shining

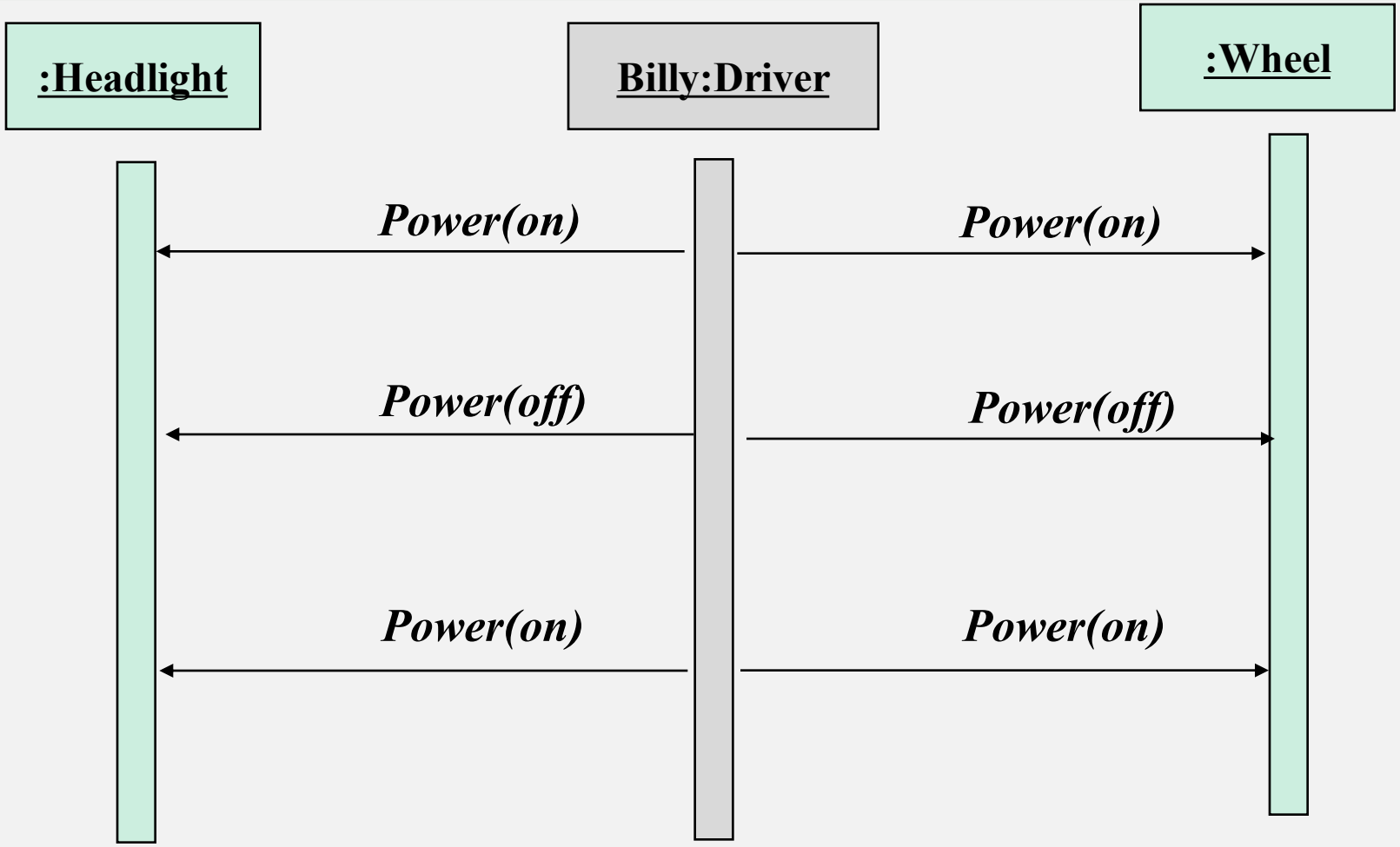
Dynamic Modeling: Create the Sequence Diagram

Name: Drive Car

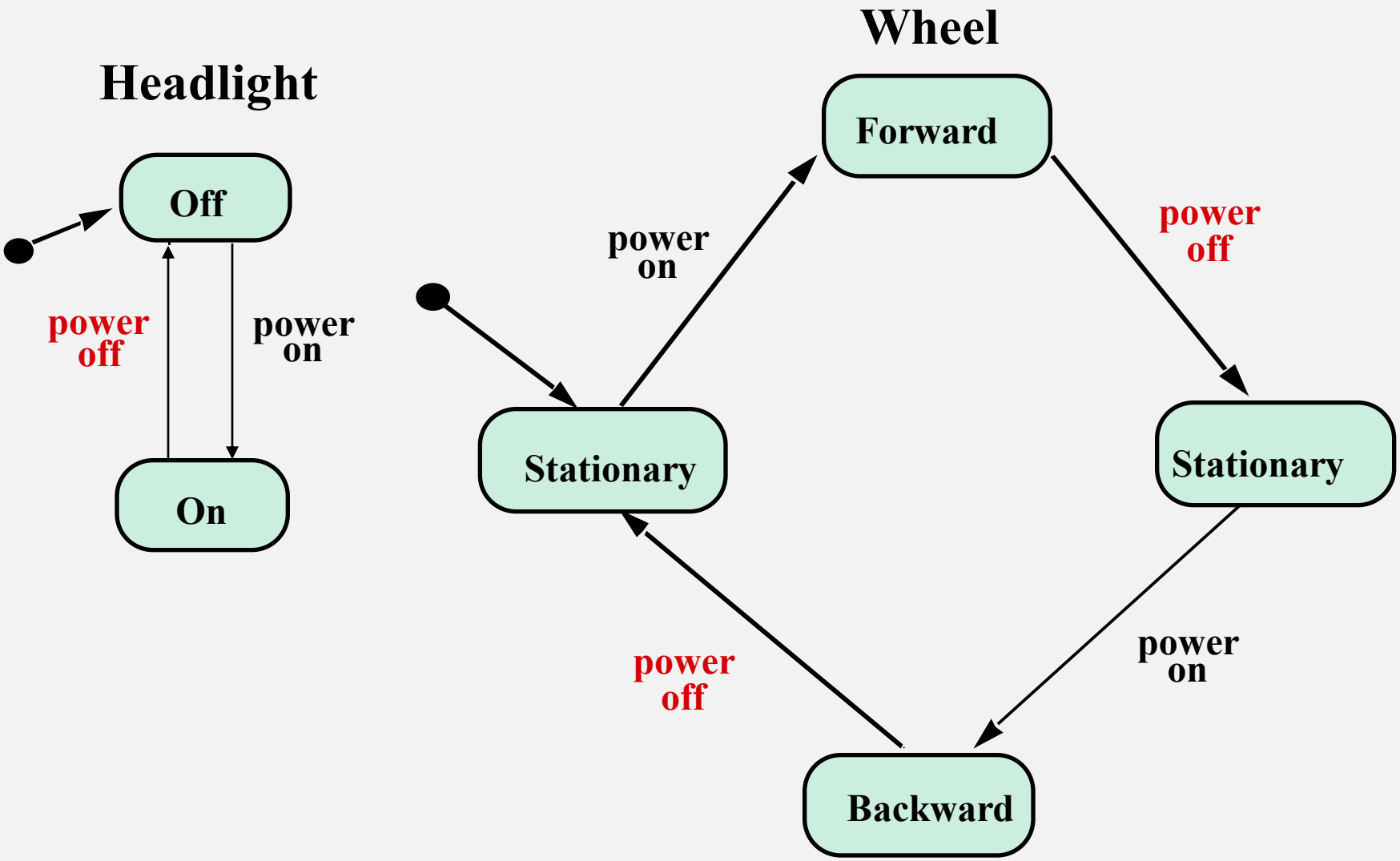
Sequence of events:

- Billy turns power on
- Headlight goes on
- Wheels starts moving forward
- Wheels keeps moving forward
- Billy turns power off
- Headlight goes off
- Wheels stops moving
- . . .

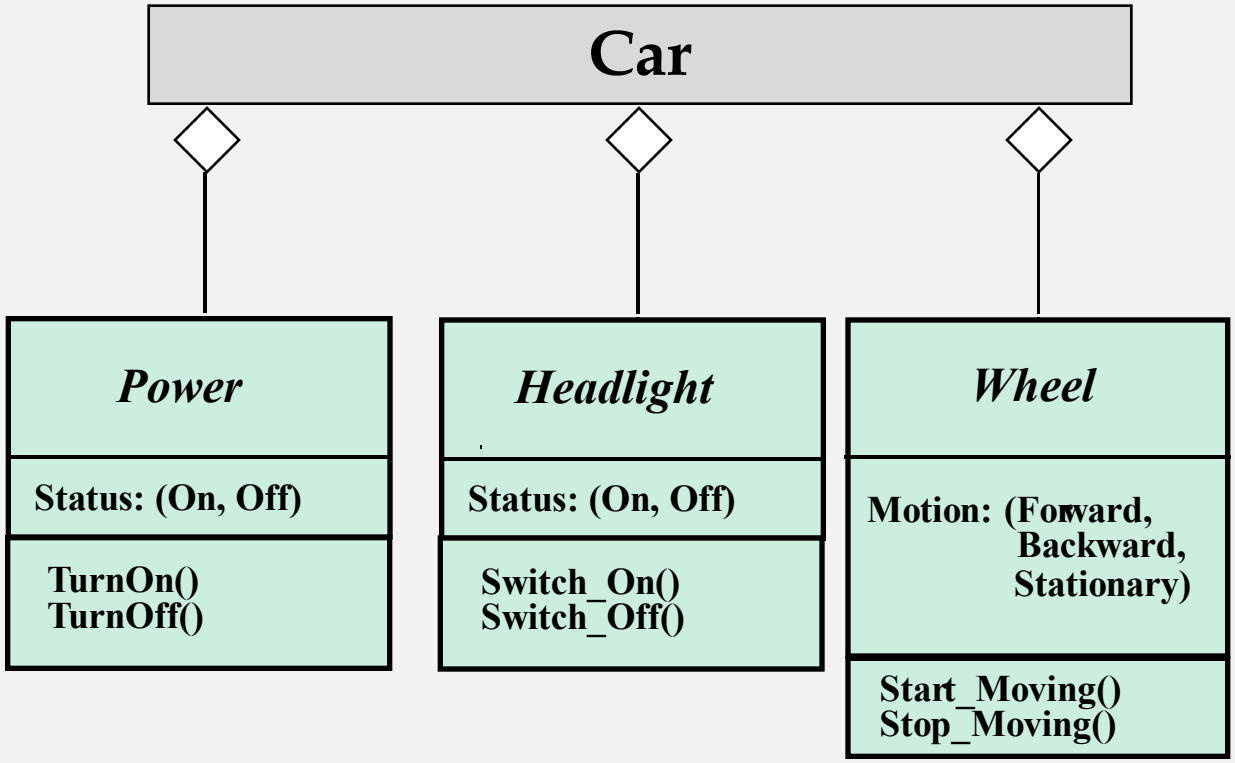
Sequence Diagram for Drive Car Scenario



Toy Car: Dynamic Model



Toy Car: Object Model



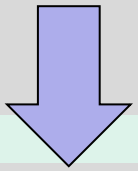
Requirements Analysis Questions

1. What are the transformations?

Functional Modeling

Create *scenarios and use case diagrams*

- Talk to client, observe, get historical records

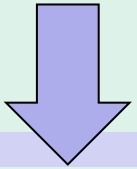


2. What is the structure of the system?

Object Modeling

Create *class diagrams*

- 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?



3. What is its behavior?

Dynamic Modeling

Create *sequence diagrams*

- Identify senders and receivers
- Show sequence of events exchanged between objects.
- Identify event dependencies and event concurrency.

Create *state diagrams*

- Only for the dynamically interesting objects.

Thank You