Requirements modeling according to the IREB standard

Prof. Dr. Ralf Lämmel
University of Koblenz-Landau
Faculty of Computer Science
Software Languages Team

Source: Handbook of Requirements Modeling IREB Standard

Requirements

- The system must represent the data of a person.
- A person has a date of birth.
- Do not grant an insurance policy for a fast car.
- Use the birthday to disambiguate persons with the same name.
- Show the correspondence address of the person.
- Show for a person the company for which it is the contact person.
- Select an address by clicking a location on a map.
- …

Introduction

Source: Handbook of Requirements Modeling IREB Standard

Textual requirements

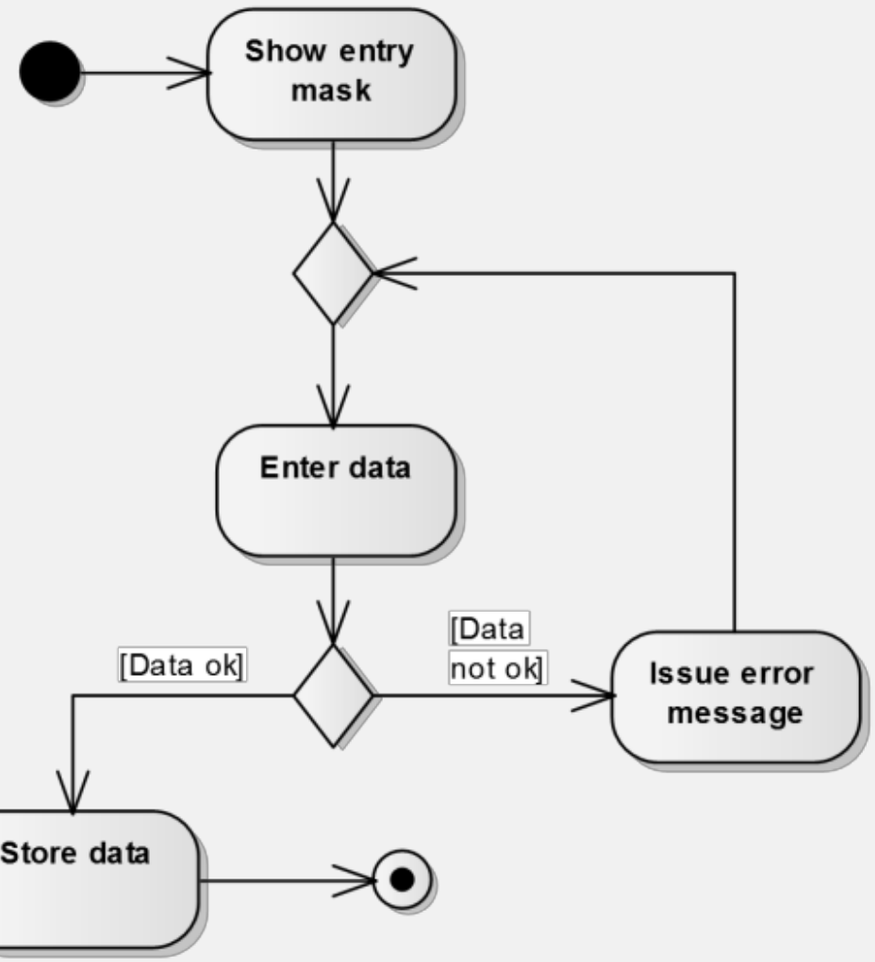
Req-1: The system shall show the entry mask

Req-2: After the action "Show entry mask" is completed, or after the action "Show error" is completed, the system shall offer the user the option to enter data

Req-3: After the action "Enter data" is completed and if the data is ok, the system shall store the data

Req-4: After the action "Enter data" is completed and if the data is not ok, the system shall issue an error message

Modeled requirements



Source: Handbook of Requirements Modeling IREB Standard

IREB Glossary of Requirements Engineering Terminology

Requirement

A requirement is (1) a need that is perceived by a stakeholder or (2) a capability or property that a system shall have.

Source: Handbook of Requirements Modeling IREB Standard

IREB Glossary of Requirements Engineering Terminology

Requirements Engineering

Requirements Engineering is responsible for ensuring that the requirements of the system to be developed are formulated as completely, correctly and precisely as possible, thereby optimally supporting the other development disciplines and activ- ities in the life cycle of the system.

Source: Handbook of Requirements Modeling IREB Standard

The role of requirements

In the life cycle of systems, requirements play a fundamental role. In particular, the various development disciplines (such as **architecture, design, implementation and testing**) are mainly based on the requirements of the system that are specified in requirements engineering, and are largely dependent on the quality of these requirements. In addition to the development disciplines also activities like **maintenance** and **service** up to **decommissioning** of the system and development of upstream activities, such as assessment of risks and costs of the development project, depend highly on the requirements and their quality.

Source: Handbook of Requirements Modeling IREB Standard

Modeling languages for requirements modeling

- OMG SysML (e.g., state machine diagrams)
- OMG UML (e.g., class or activity diagrams)
- MATLAB / SimuLink (e.g., stateflow diagrams)
- BPMN

Source: Handbook of Requirements Modeling IREB Standard

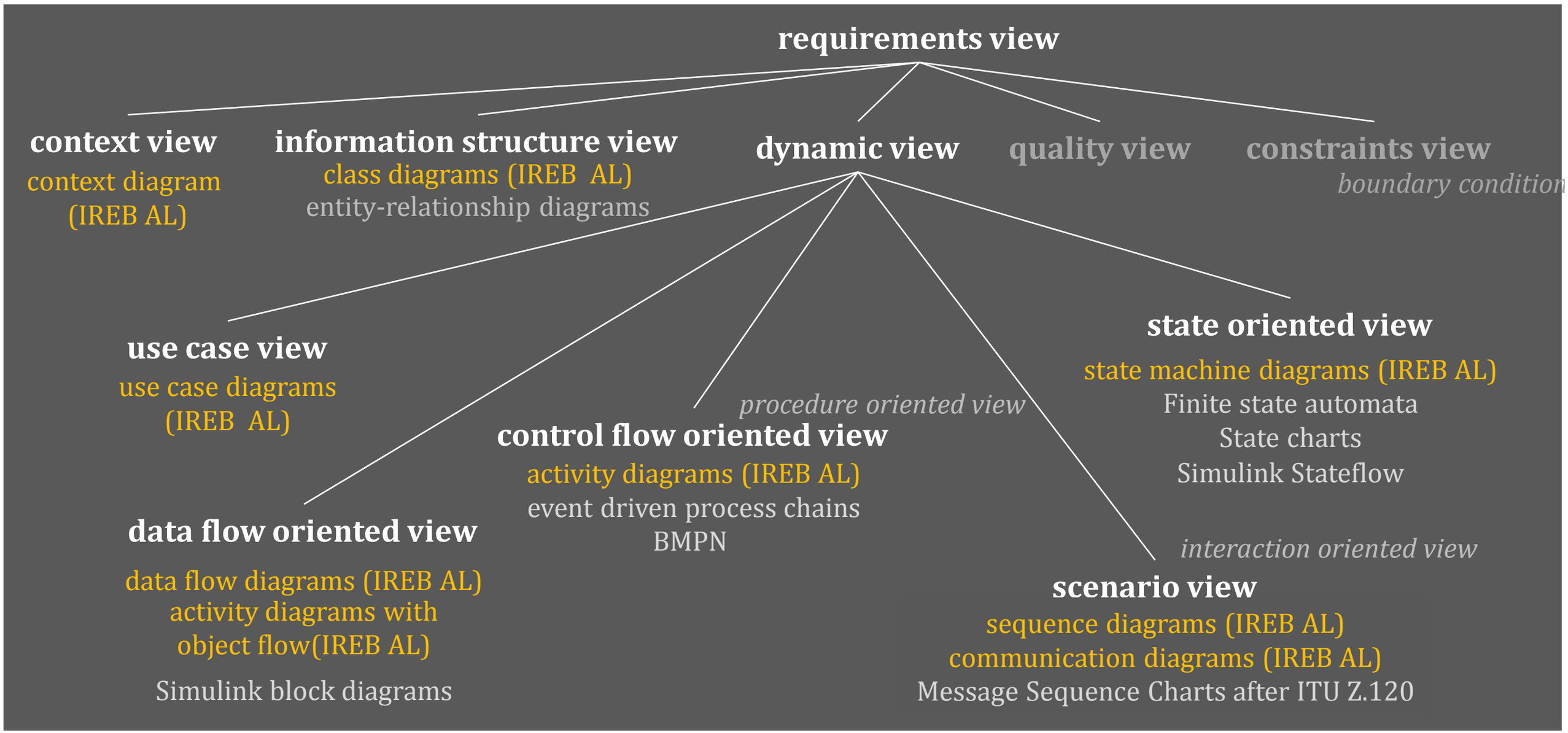
Classification of systems

- Type of system
  - operational information system
  - embedded system
- Application domain
  - banks and insurance
  - production
  - vehicle and aircraft

For instance, the reactive behavior forms the focus of requirements engineering

Source: Handbook of Requirements Modeling IREB Standard

IREB views in requirements modeling



Source: Handbook of Requirements Modeling IREB Standard

IREB views in requirements modeling

- **Context view**: identification of the necessary interfaces between the system under consideration and its context.

- **Information structure view**: identification of static and structural aspects of the functionality such as the structure of data.

- **Dynamic view**: identification of dynamic aspects of the functionality.

Source: Handbook of Requirements Modeling IREB Standard

Views of the dynamic view

- **Use case view**: user functions and dependencies from the system context.
- **Data flow-oriented view**: system functions and data dependencies as expressed by data flow or activity diagrams.
- **Control flow-oriented view**: flow logic of processes as expressed by UML or SysML activity diagrams or BPMN diagrams.
- **State oriented view**: states and state changes as expressed by finite automata, Harel state charts or UML state machine diagrams.
- **Scenario view**: interaction sequences between actors and the system as expressed by typically sequence diagrams of UML / SysML or Message Sequence Charts of ITU.

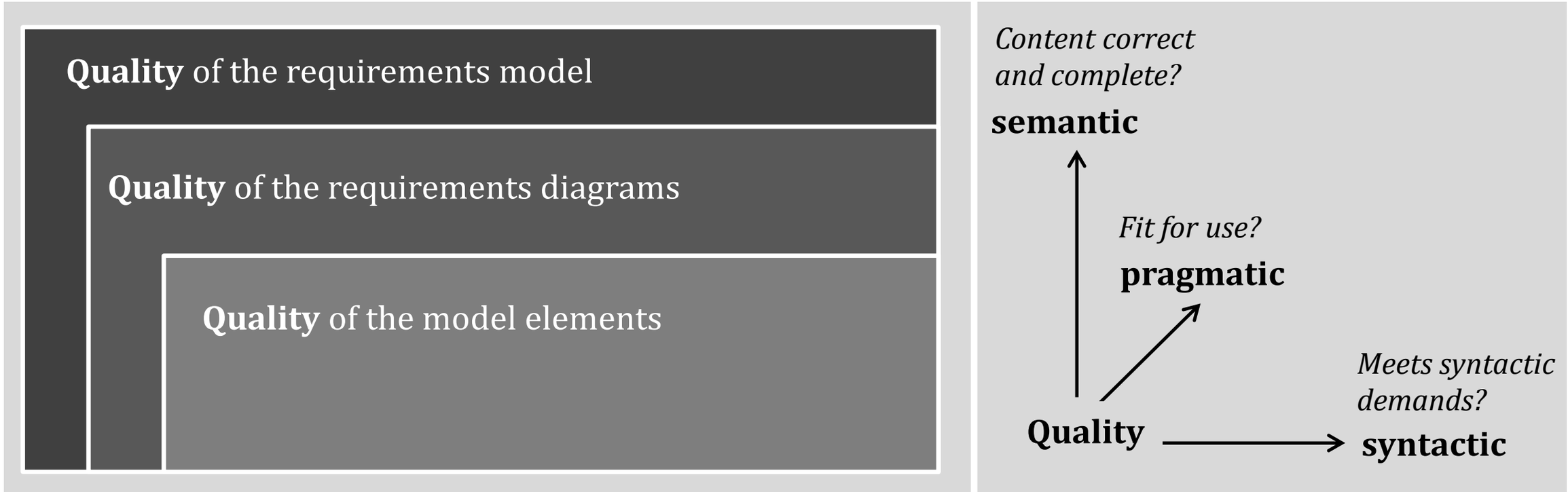
Source: Handbook of Requirements Modeling IREB Standard

Benefits of requirements modeling

- Better comprehensibility of the requirements
- Support of the principle of "separation of concerns”
- Support of the principle „divide and conquer"
- Reduced risk of ambiguity
- Higher potential for automated analysis of requirements
- Higher potential for automatic processing of requirements
- Requirements in Context

Source: Handbook of Requirements Modeling IREB Standard

Quality of requirements models



Context modeling

Source: Handbook of Requirements Modeling IREB Standard

Context modeling

- Which roles and persons interact with the system in operation?
- What other systems are related with the system under consideration in the operation?
- What the interfaces are between the system under consideration and the people and systems?

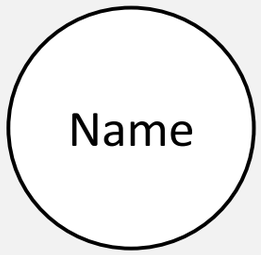
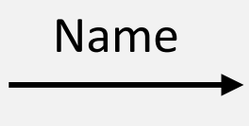
Source: Handbook of Requirements Modeling IREB Standard

Representing context

- **Context diagram from Structured Analysis**
- SysML block diagram
- UML class diagram
- UML component diagram
- **UML use case diagram**

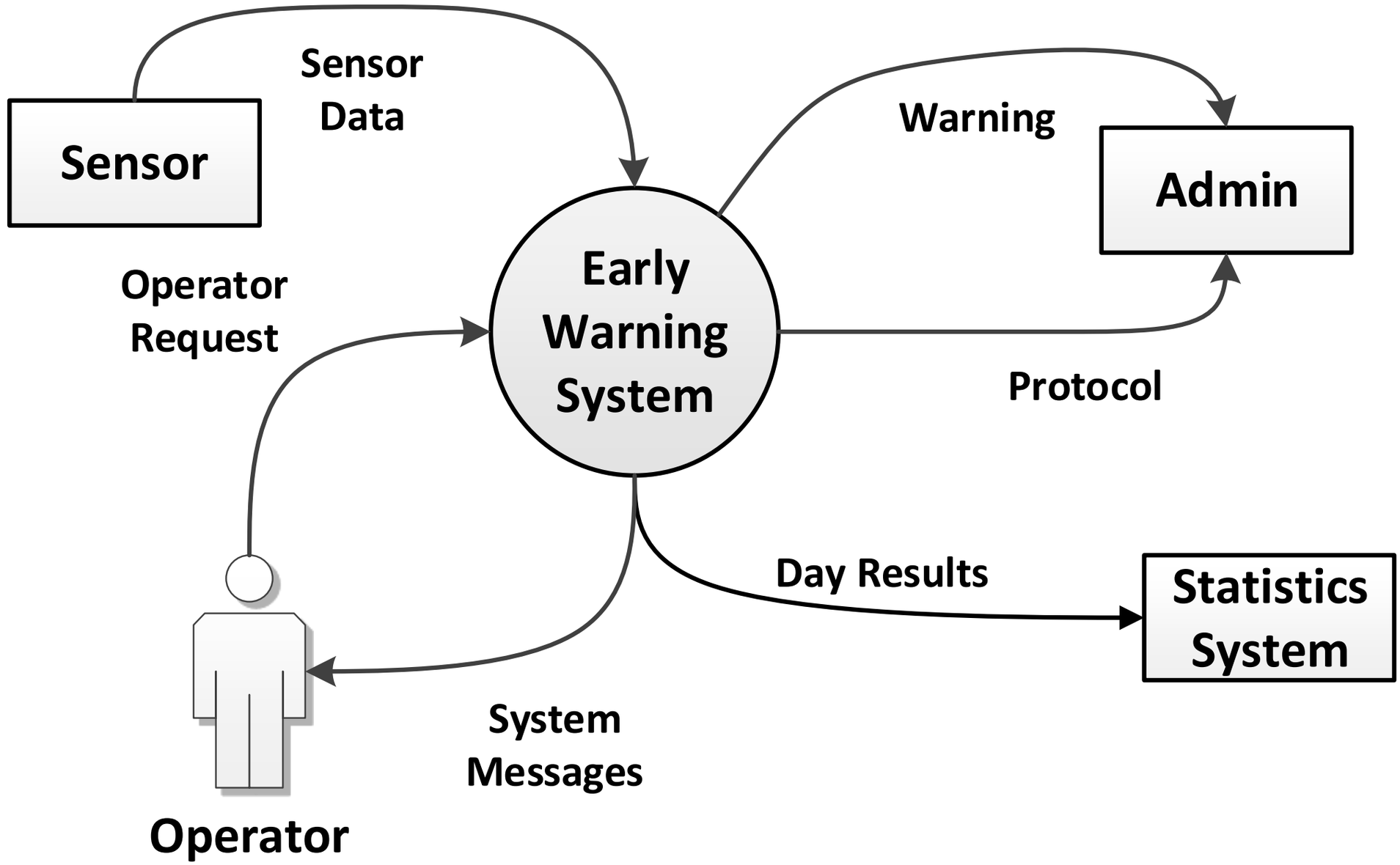
Source: Handbook of Requirements Modeling IREB Standard

Modeling elements of context diagrams

Name	Notation	Explanation
System (SuD)		*the system considered in the scope of analysis/development*
neighboring system / actor		*neighboring system or actor in system context*
data flow		*flow of data between system and system context*

Source: Handbook of Requirements Modeling IREB Standard

A context diagram (SA) for an early warning system in the mining industry



Source: Handbook of Requirements Modeling IREB Standard

Pragmatic rules for context modeling

- All neighboring systems that interact with the system, should be included in the dia- gram (completeness of the communication partners).

- All neighboring systems should be named (to clearly specify where the inputs come from and where the output flow goes to).

- All inputs and outputs should be labeled with a logical name of the data flows (because unnamed arrows point to a lack of understanding of the interface).

Information structure modeling

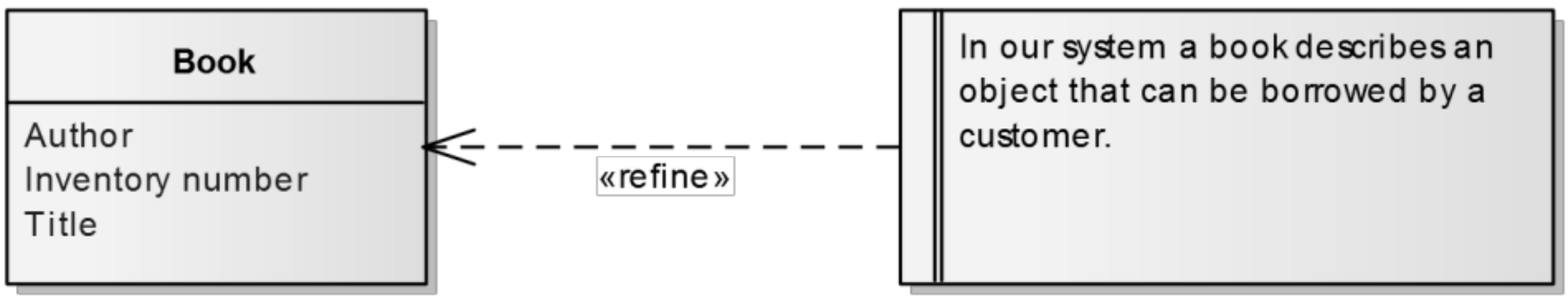
Source: Handbook of Requirements Modeling IREB Standard

Central role of information structure modeling

- Specification of business terms and data
- Specification of requirements that relate to business terms
- Modeled aspects
  - Relationship between terms
  - Attributes of terms

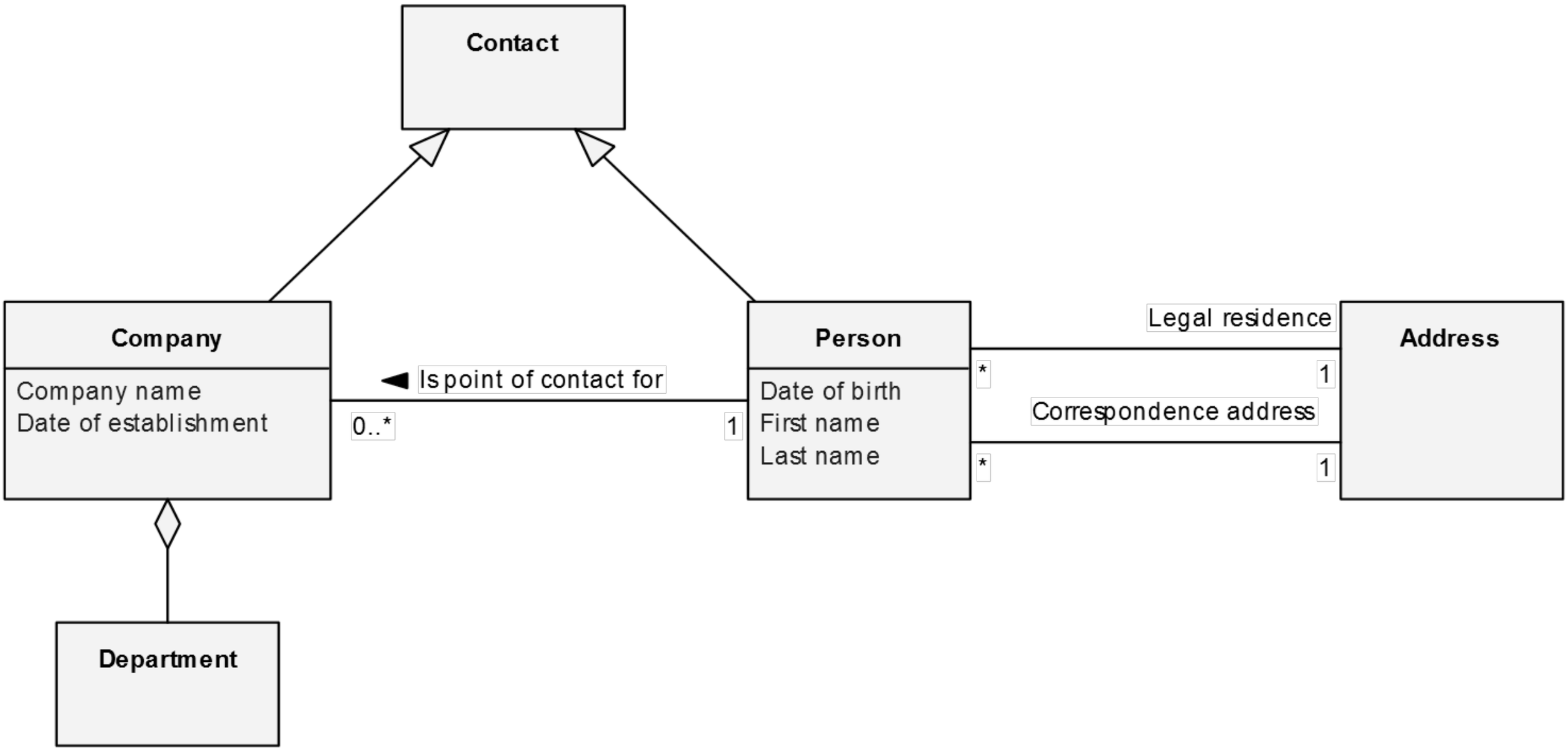
Source: Handbook of Requirements Modeling IREB Standard

Example of a UML class diagram



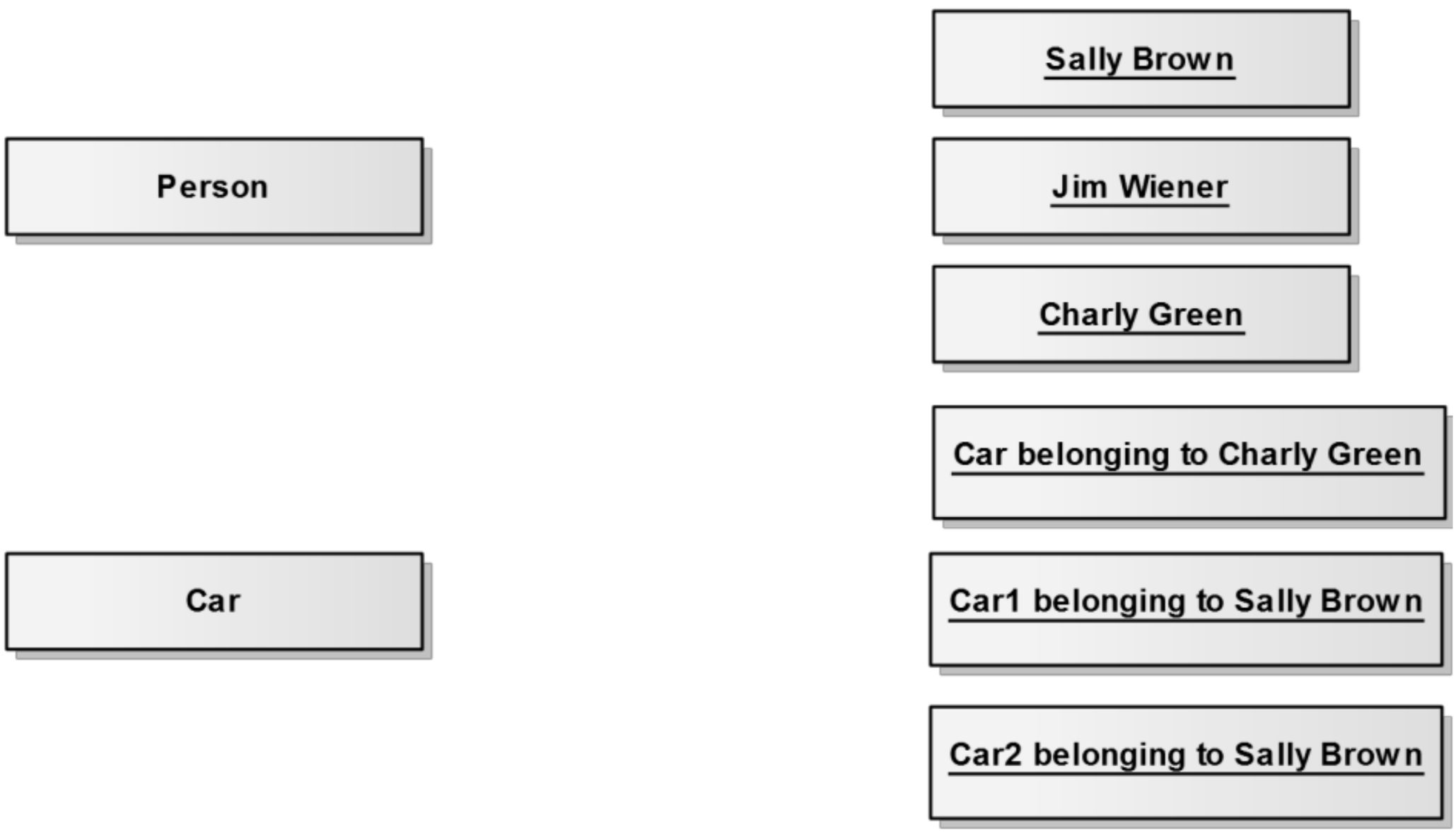
Source: Handbook of Requirements Modeling IREB Standard

Example of a UML class diagram



Source: Handbook of Requirements Modeling IREB Standard

Classes versus objects



Source: Handbook of Requirements Modeling IREB Standard

Identification of classes within (for example) textual requirements

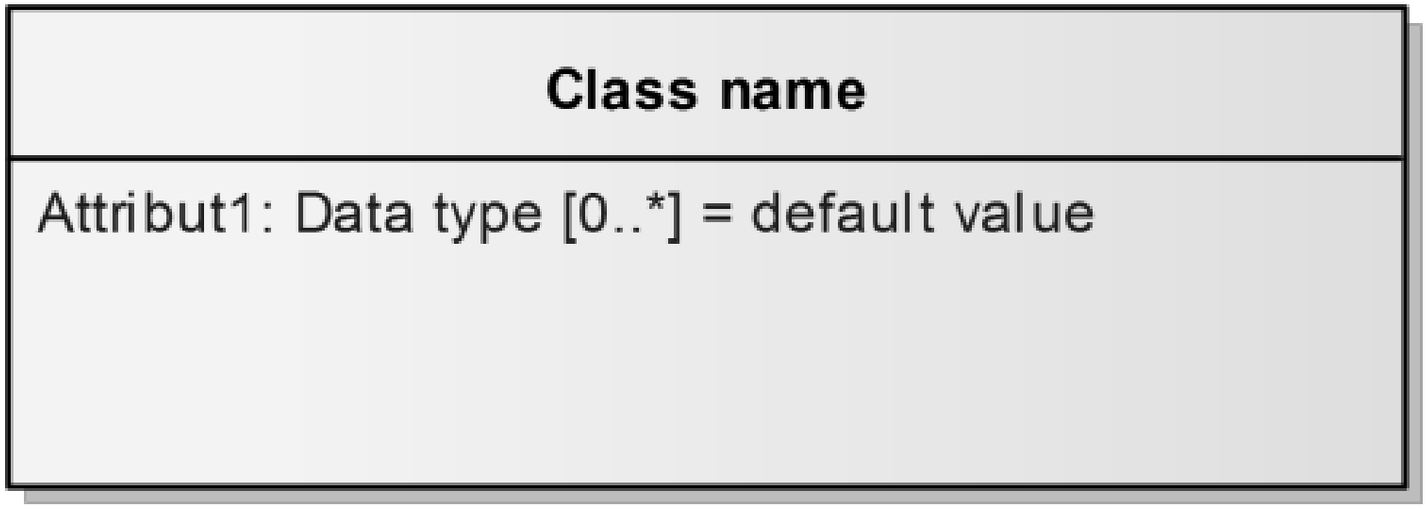
Suppose that the following nouns would have been identified in a first step: person, age, car, gender, color, vehicle, man. In this list, there appear only two terms that are worth, modeling as classes: person and vehicle. For the other terms applies:

- Man: a synonym for person
- Age: property of a person
- Car: Synonym for Vehicle
- Gender: property of a person
- Color: Property of a vehicle

Assumptions: (1) The concept of person must be used consistently and not human. (2) The concept vehicle must be used consistently and not car. (3) The term color refers to the color of a vehicle.

Source: Handbook of Requirements Modeling IREB Standard

Attribute declarations

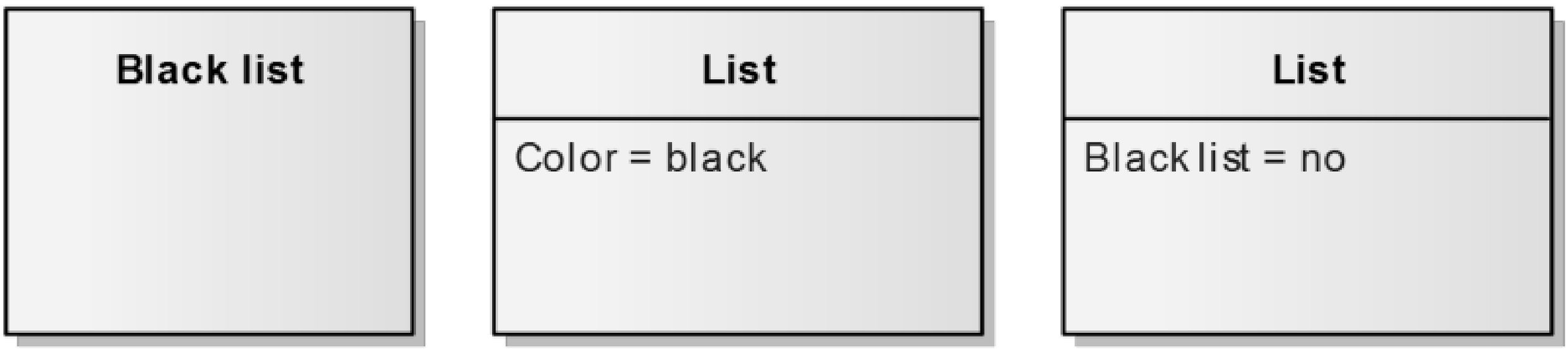


[/] Name [: type] [[multiplicity]] [= default]

- Name: The name of the attribute, which is obligatory
- Data Type: The data type of the attribute. This is optional.
- Default: The value of the attribute set on creation of a new object of the class
- Multiplicity: Can be used if the attribute can take on multiple values simultaneously (Example: Several first names). The same multiplicities are used as in the relationships.
- Derived: The leading "/" indicates that the attribute value can be derived from other values (Example: The age of a person can be derived from the date of birth).

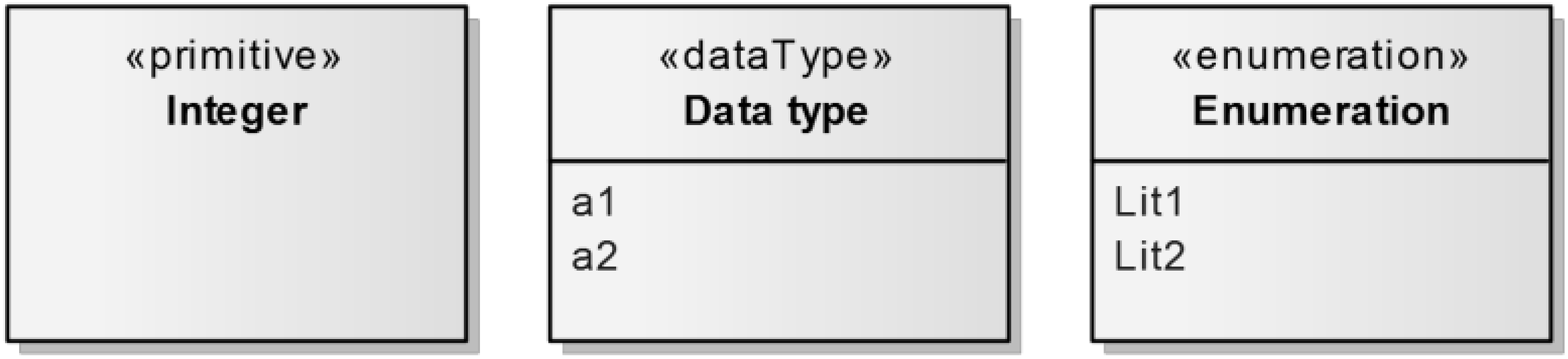
Source: Handbook of Requirements Modeling IREB Standard

Modeling adjectives with nouns



Source: Handbook of Requirements Modeling IREB Standard

Data types

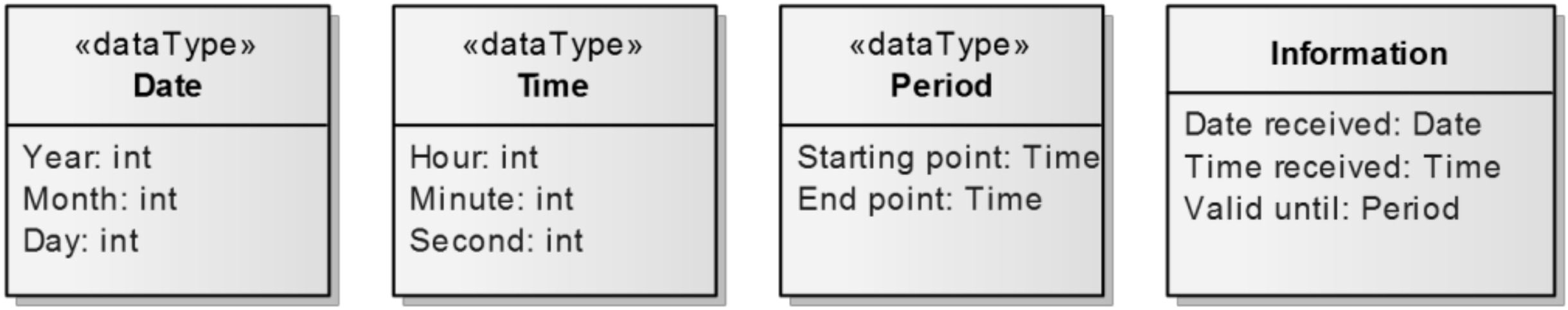


Pre-defined primitive data types:

- Boolean: A Boolean value, can be TRUE or FALSE.
- Integer: A whole number
- Float: A floating point number
- Character: A single character
- String: A sequence of characters

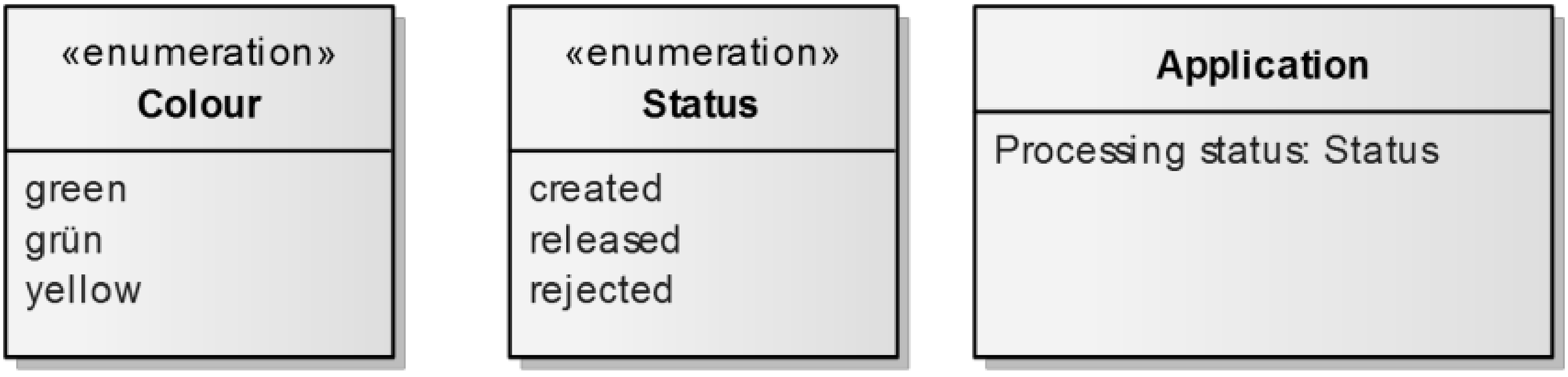
Source: Handbook of Requirements Modeling IREB Standard

Structured data types



Source: Handbook of Requirements Modeling IREB Standard

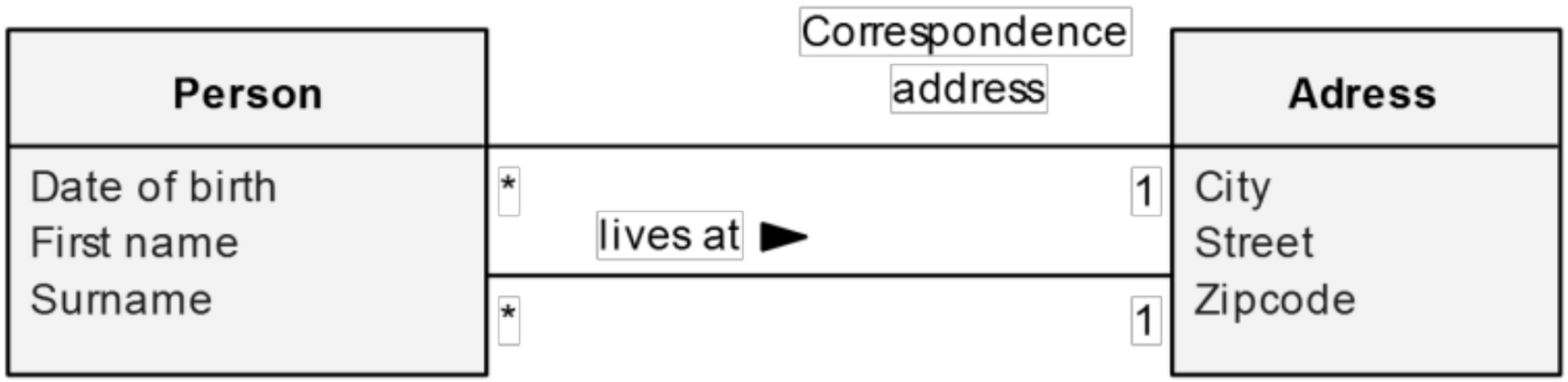
Enumerations (data types)

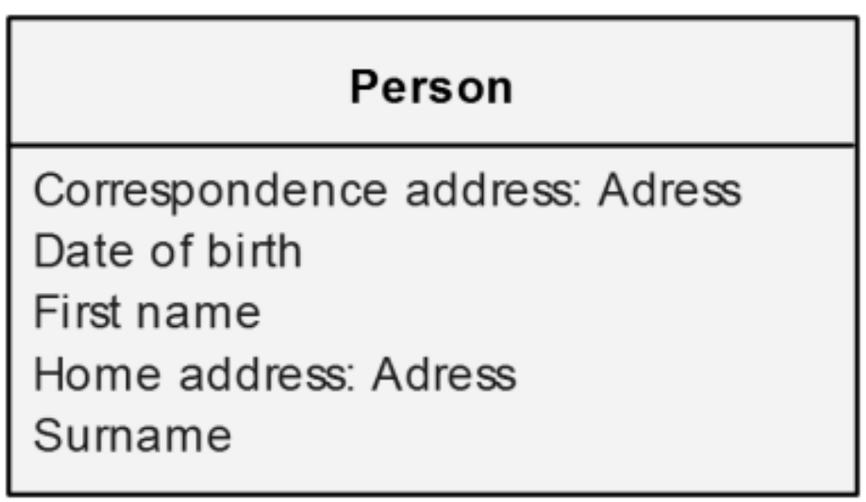


Source: Handbook of Requirements Modeling IREB Standard

Class or attribute

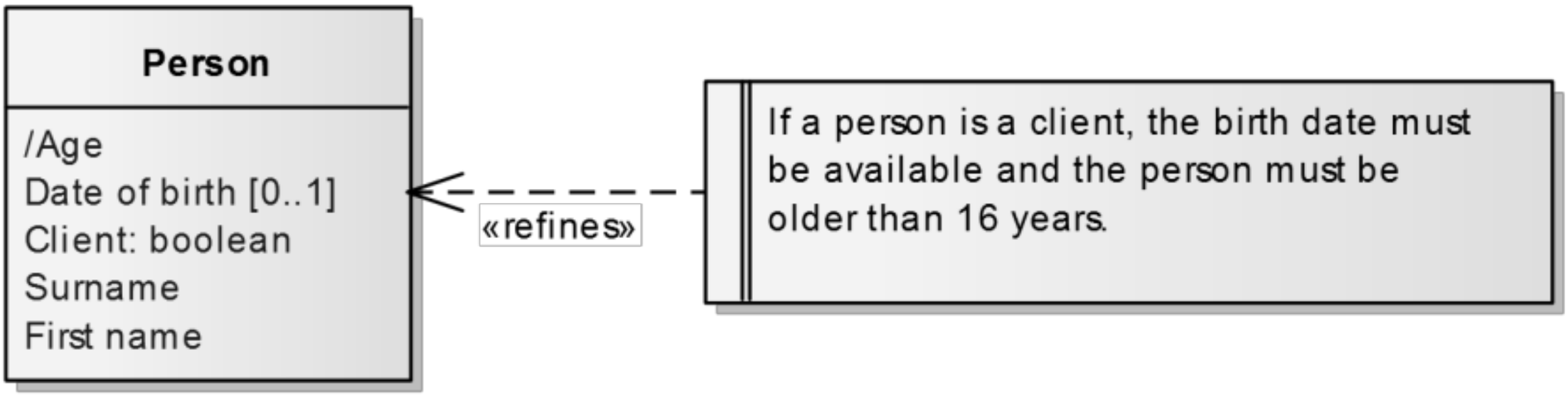
For structured information, the following heuristic is helpful: as soon as a structured form of this information belongs to more than one other object, then it should be modeled as a separate class.





Source: Handbook of Requirements Modeling IREB Standard

Attribute constraints



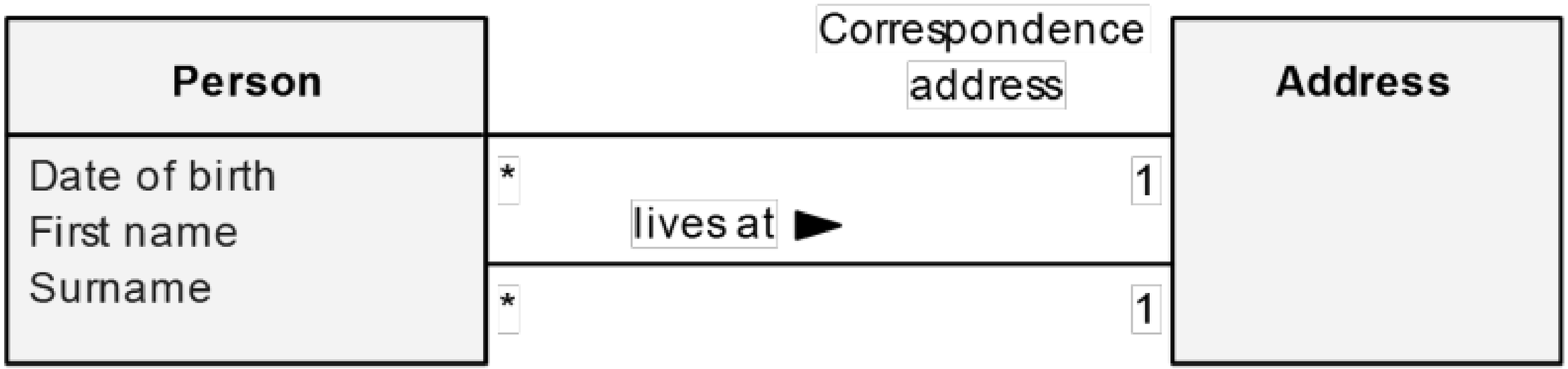
Formal constraints (OCL) may be used, too:

```
context Person inv:
    self.Client=true implies self.age >= 16
```

Modeling relationships

Source: Handbook of Requirements Modeling IREB Standard

Simple relationships

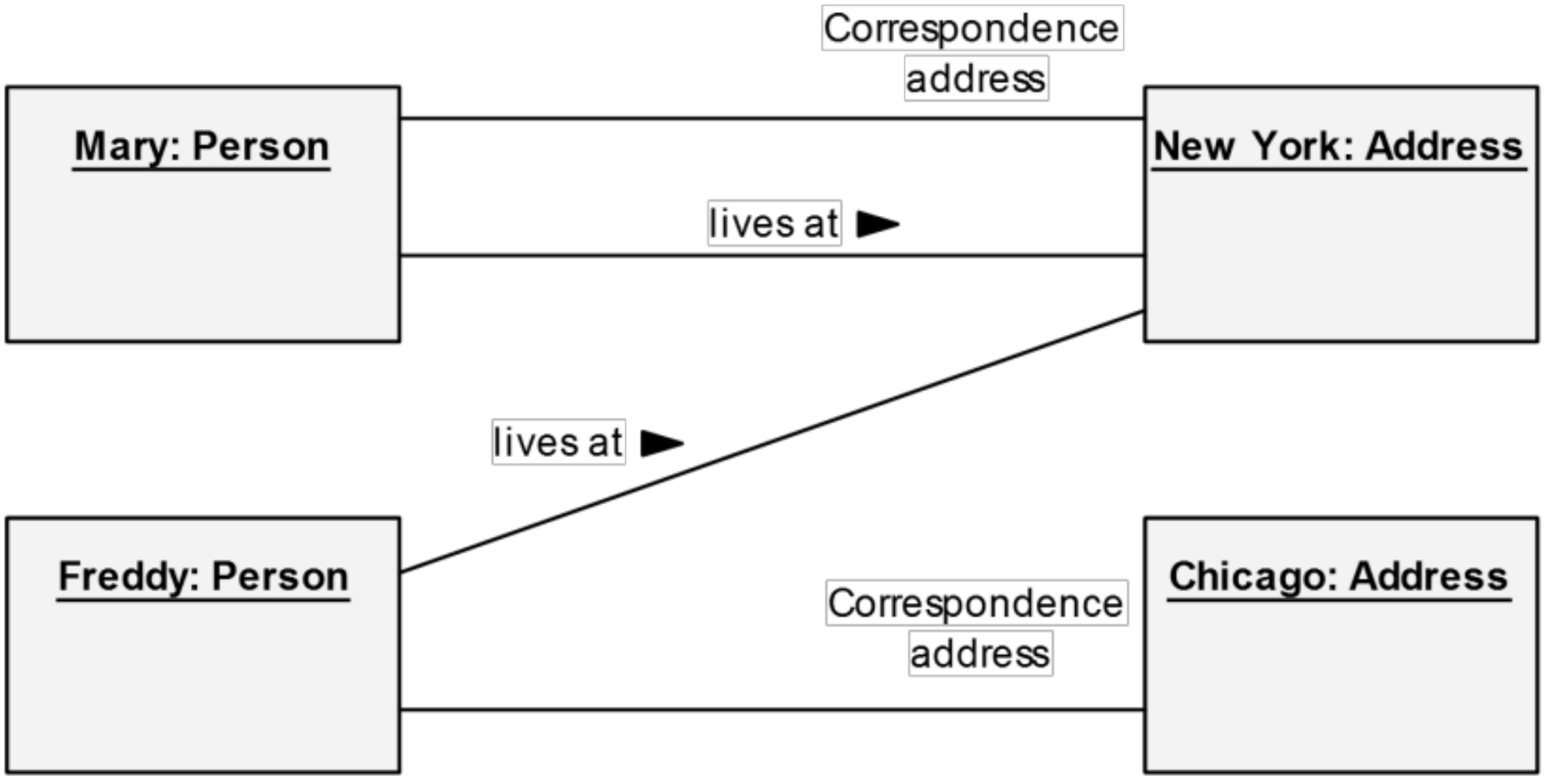


Aspects of relationships:

- Name: Specifies the name (meaning/semantics) of the association in verb phrase
- Reading direction: direction in which the name is to be read
- Multiplicity: Is listed on each end of the association and indicates how many objects the other object may be or must be related to
- Role: Refers to the role played by the object to which the role is attached with respect to the other object

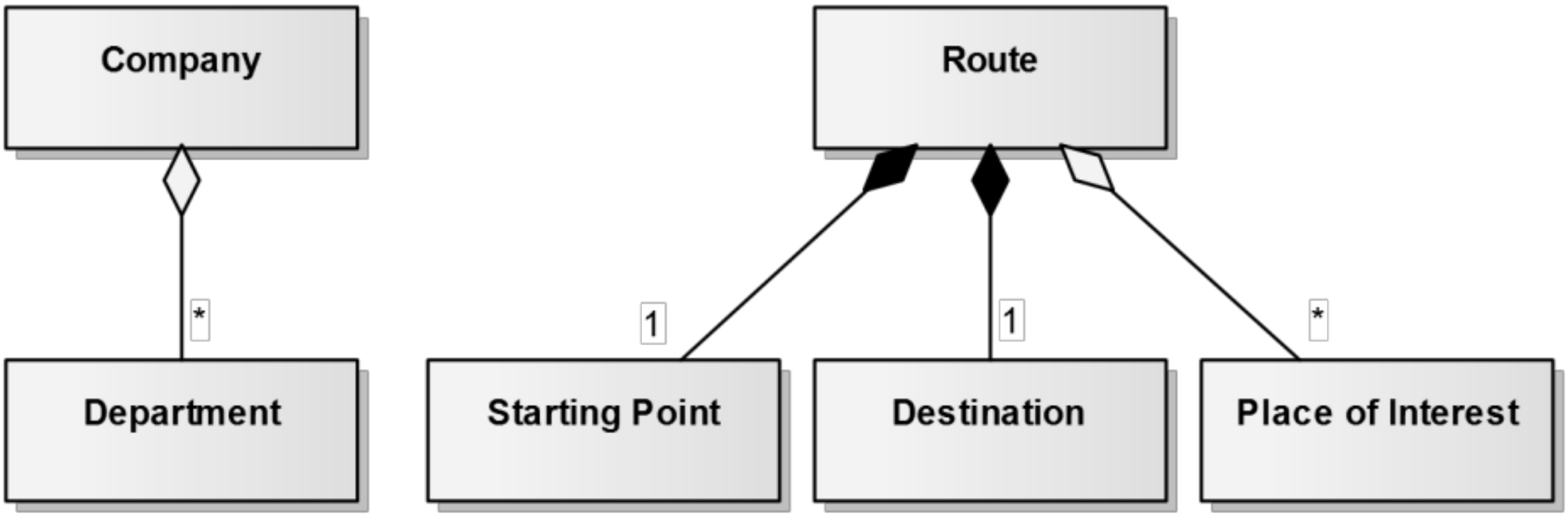
Source: Handbook of Requirements Modeling IREB Standard

Relationships at the object level



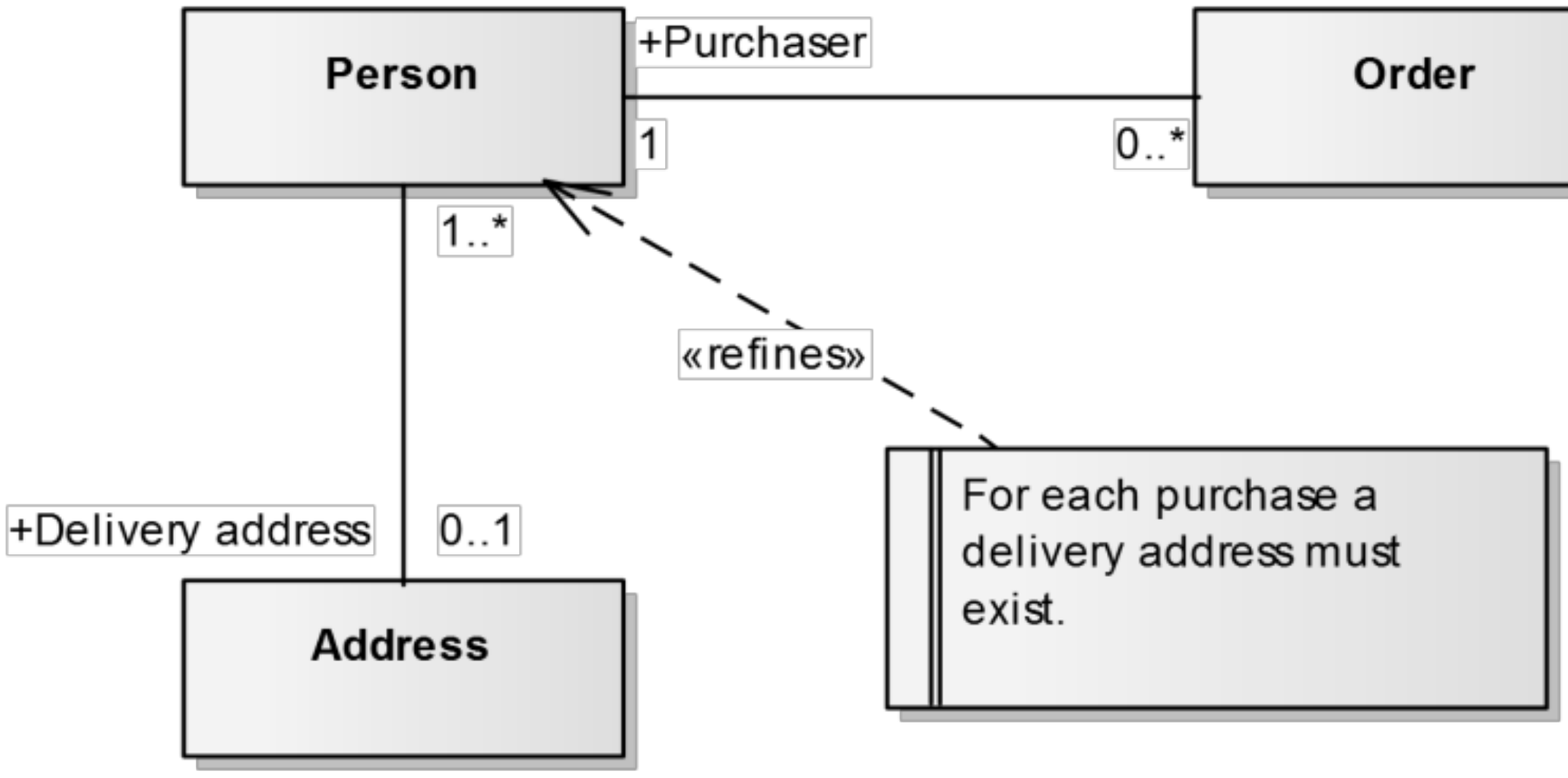
Source: Handbook of Requirements Modeling IREB Standard

Part-of/part-whole relationships



Source: Handbook of Requirements Modeling IREB Standard

Modeling constraints of relationships

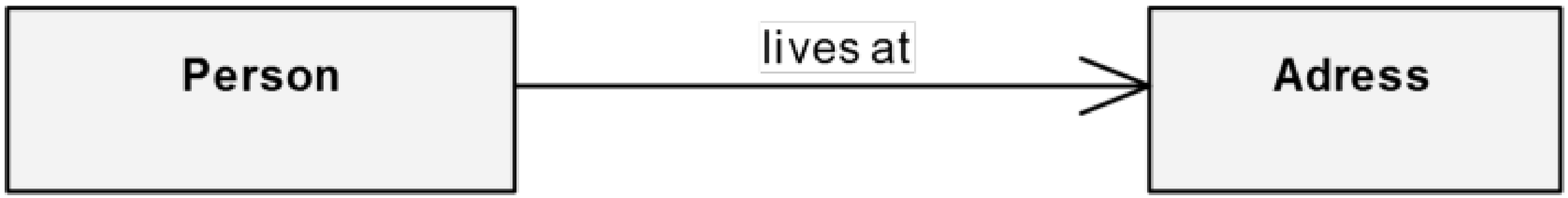


In OCL:

```
context order
inv:self.purchaser->notEmpty()implies
self.Purchaser.DeliveryAddress->notEmpty()
```

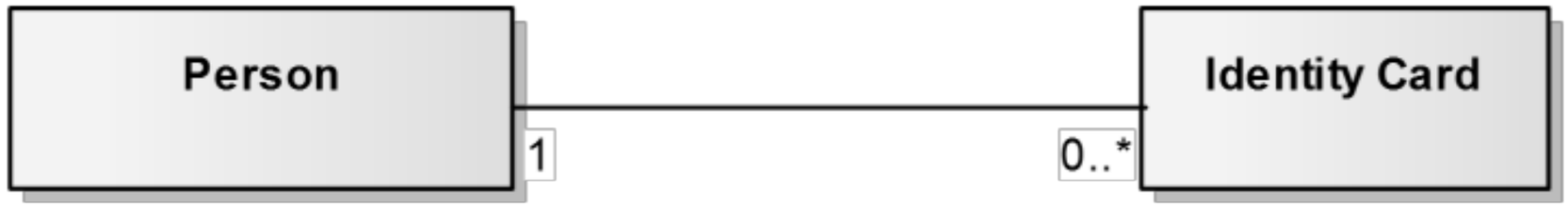
Source: Handbook of Requirements Modeling IREB Standard

Reading direction versus navigability



Source: Handbook of Requirements Modeling IREB Standard

Challenges related to multiplicities

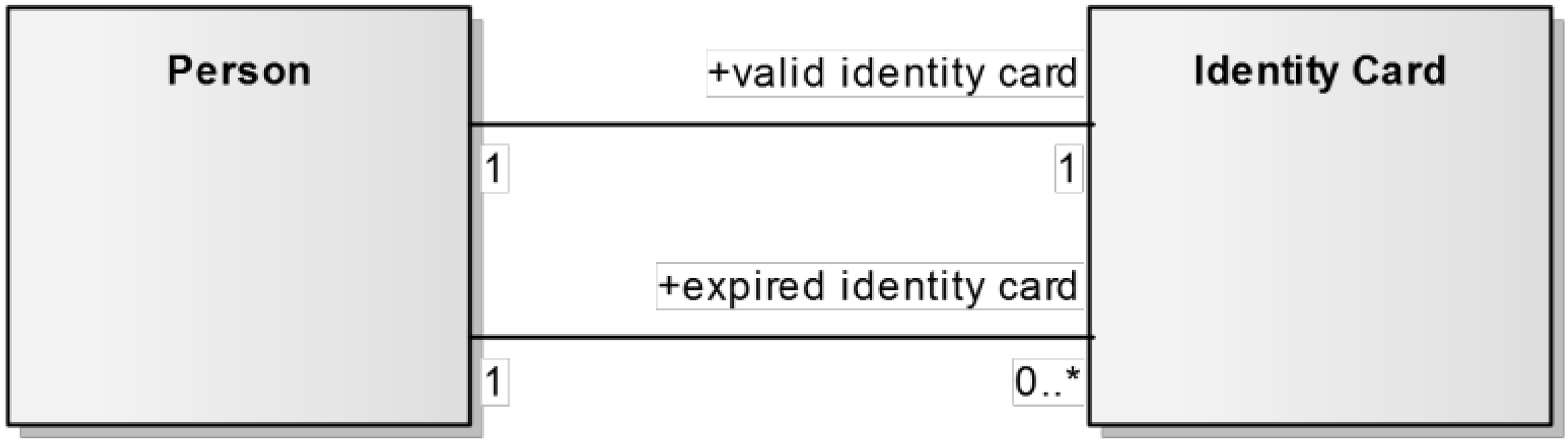


0..* interpreted as:

- *: Person has (over time) many identity cards (expired, lost)
- 0: does not need an identity card (does not have one or has lost it)
- 0: A person always has an identity card, but first a person is created, then the card. So there is a period, before the Identity Card is attached and a person exists without an identity card.

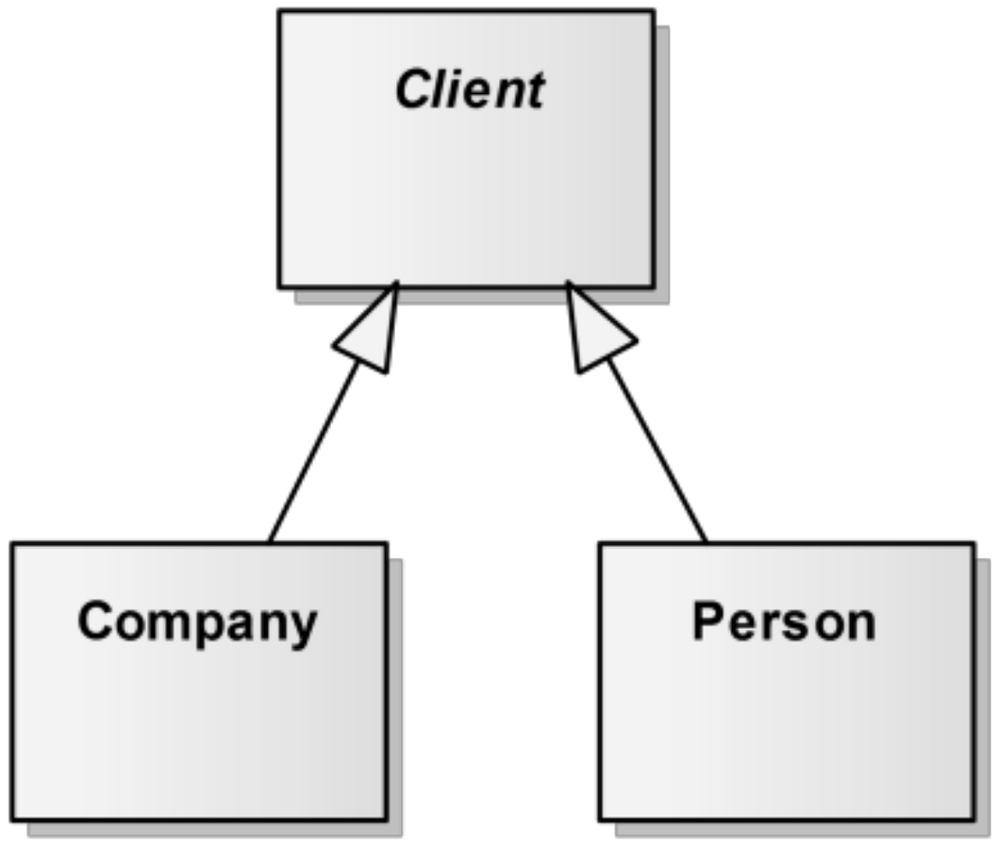
Source: Handbook of Requirements Modeling IREB Standard

Resolved ambiguities



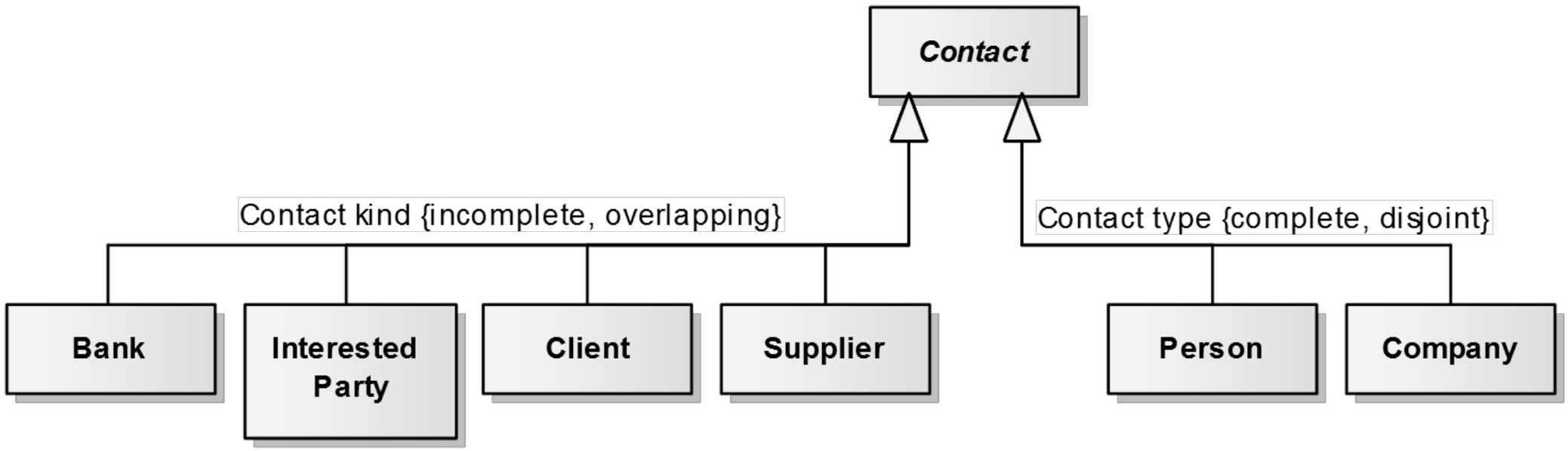
Source: Handbook of Requirements Modeling IREB Standard

Modeling generalizations



Source: Handbook of Requirements Modeling IREB Standard

Constraints on generalizations



- Incomplete: The modeled subtypes are not necessarily complete. For example, manufacturer could be added as kind of contact.
- Complete: The modeled subtypes are complete. No other contact types are possible.
- Disjoint: An instance can only be one of the subtypes. For example, a contact is either a person or a company, but never both.
- Overlapping: An instance can belong to more than one subtype. For example, a contact may be a customer and be a supplier.

Source: Handbook of Requirements Modeling IREB Standard

Identification of generalizations

“The dog is a kind of animal.”

“A kind of animal is a dog.”

“The boss is a special employee.”

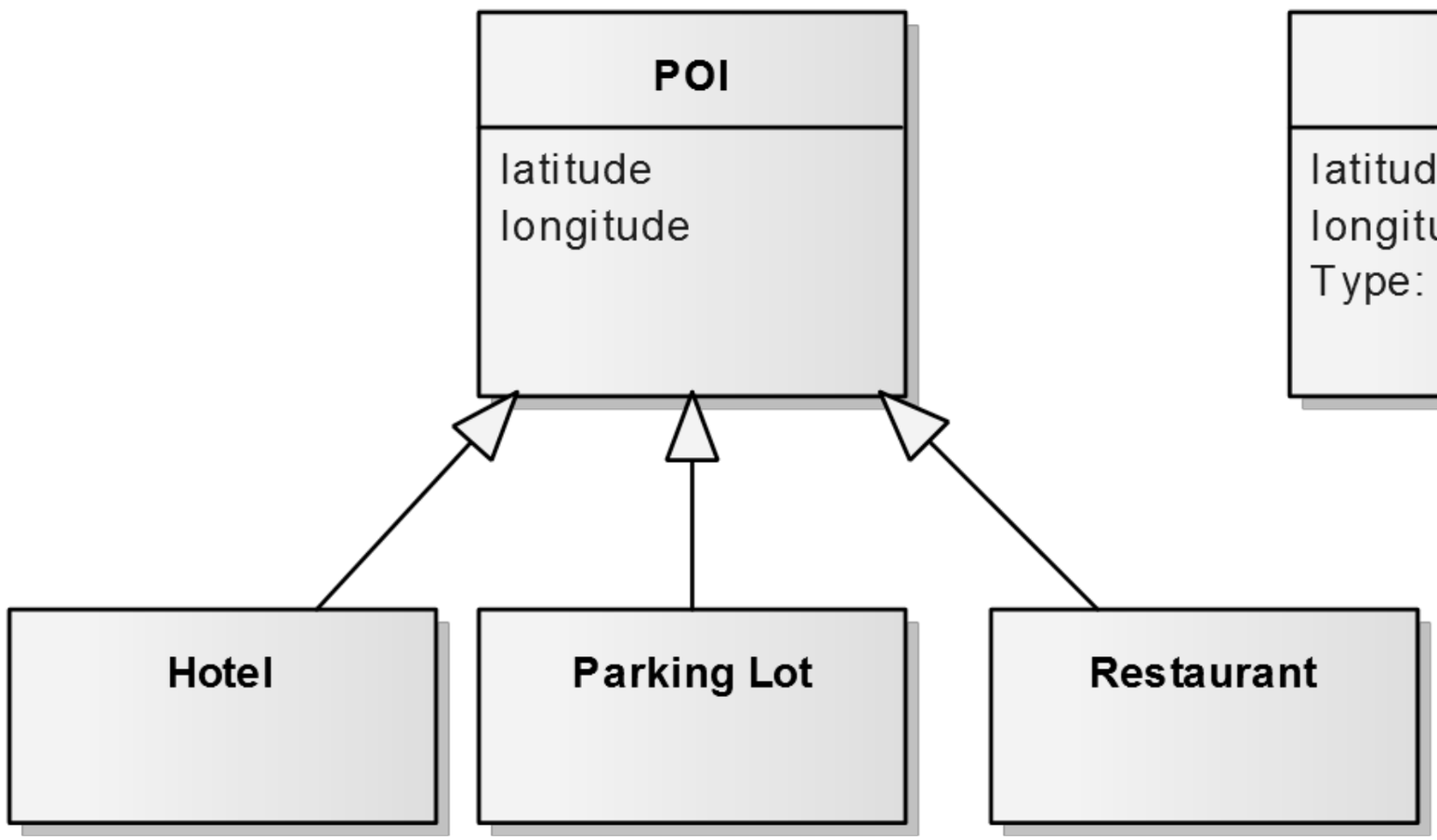
“Typical payment methods are bank transfer or billing.”

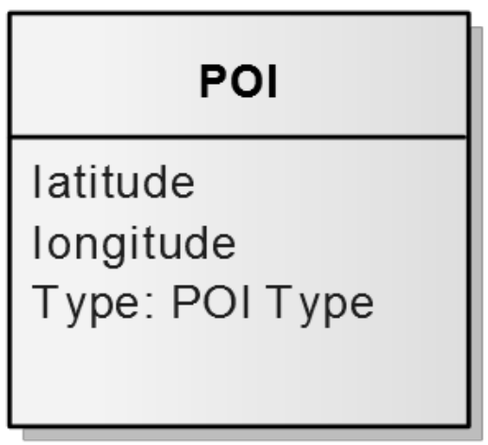
Source: Handbook of Requirements Modeling IREB Standard

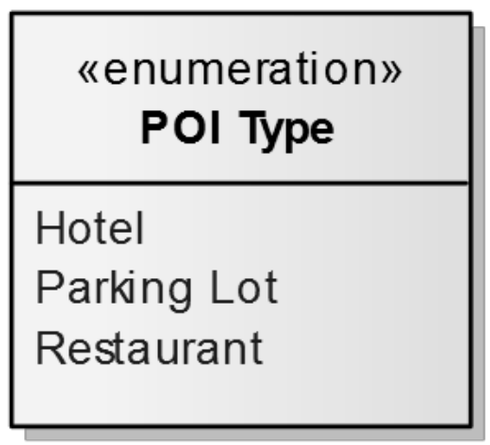
Specializations without attributes

If all specializations are without attributes,

modeling via a property "type" or "kind of" is possible.



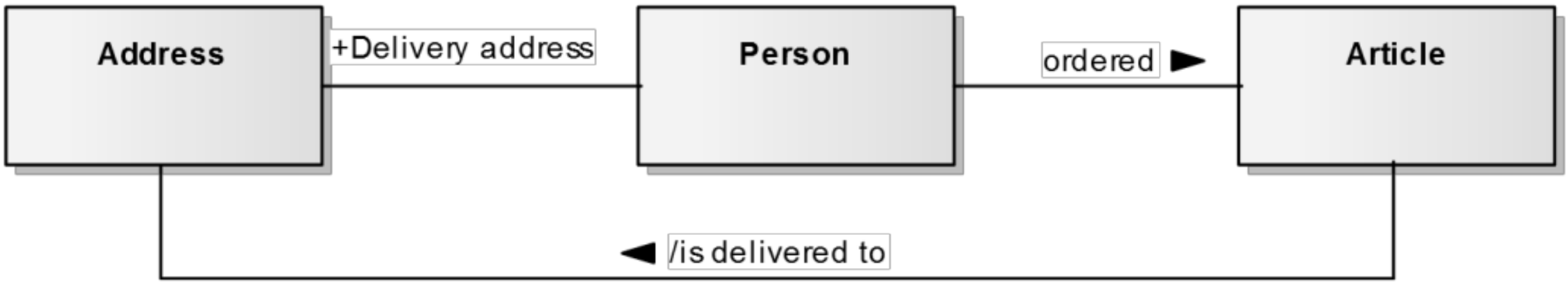




Source: Handbook of Requirements Modeling IREB Standard

Derived associations

Derived associations are associations that can be derived from available associations and are therefore redundant. Similar to derived attributes these associations require a derivation rule.



Dynamic views

Source: Handbook of Requirements Modeling IREB Standard

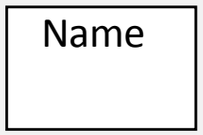
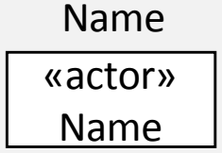
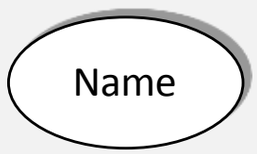
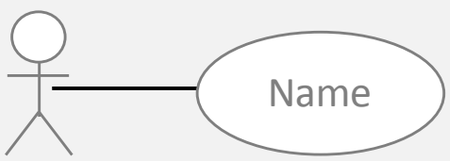
Dynamic views

View	Significance
Use case view	Decomposition of the functionality of the entire system from user perspective in external (or by time) triggered processes (or interactions or sequences of functions), each leading to a specific added business value for one or more actors in the system context, presented in the form of use case diagrams including textual use case specifications for each use case.
Control flow oriented view	Specification of sequences of required functions of a system, where the emphasis is on the sequence of execution. This view is mainly represented by UML activity diagrams with explanatory activity descriptions.
Data flow oriented view	The Specification of the required functions of a system including Input/Output data dependencies is represented classically by data flow diagrams with explanatory descriptions of the functions and data flows between the functions. It can also be represented by UML activity diagrams by using appropriate extensions.
State oriented view	Specification of the event-driven behavior of a system including states of the system, events and conditions for state transitions. Presented by state transition diagrams or state charts with explanatory descriptions of states, functions, conditions and events that trigger state transitions.
Scenario view	Specification of interactions between actors (people, systems) in the system context and the System-under-Development (SuD) that lead to a added business value for one or more actors. Scenario modeling can be done for some examples only (e.g. to support the elicitation of requirements) or with a claim to completeness, i.e. all the scenarios are modeled which are to be supported by the SuD.

Use case diagrams

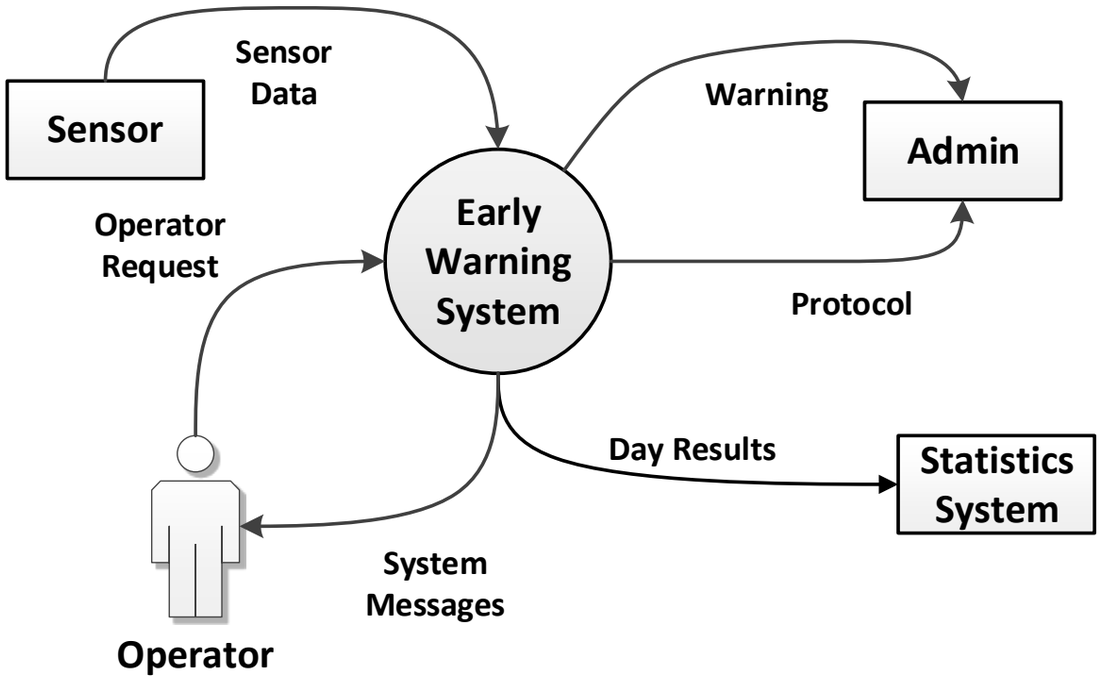
Source: Handbook of Requirements Modeling IREB Standard

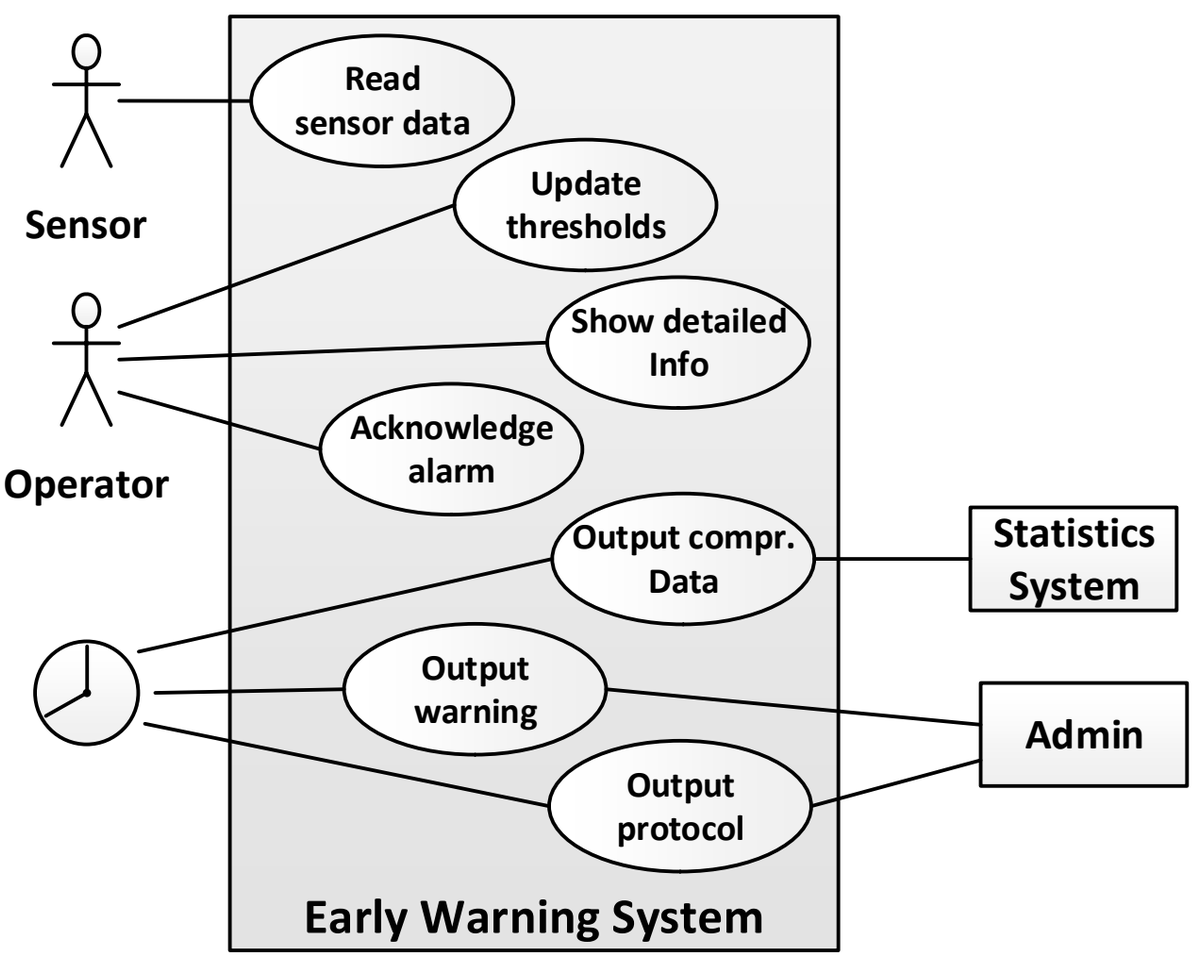
Modeling elements of use case diagrams

Notation	Name	Meaning
	System Boundary	The rectangle depicts the scope of the system. Actors are outside the scope. Use cases are inside the scope .
 Name  *(Alternative)*	Actor	An actor can be a person, a firm or organization, a software or system element (hardware, software or both).
	Use Case	Functionality of the system, needed by an actor that provides value to the actor. The name should contain a verb, as it describes a functionality, and an object, to which the functionality refers, e.g. "monitor velocity" .
	Association	The (unnamed) line between actor and use case indicates that this actor interacts with this use case.

Source: Handbook of Requirements Modeling IREB Standard

Context versus use case diagram





If a context diagram exists, in which all neighboring systems and actors of the considered system are shown, it may be enough, to create a use case diagram that only contains actors, which trigger the execution of use cases. Those actors are called process-triggering actors. These "actors" justify the existence of use cases. In other words, without the respective actor there would be no demand for this use case. So if a context diagram exists, further actors who are involved in the use case (i.e. during the execution of the process after the trigger by an actor) are not necessarily drawn in the use case diagram. They only increase the complexity of the use case diagram, and detract attention from the fact, that the use case view mainly serves to decompose the overall functionality of a system from user perspective into disjunct processes.

Source: Handbook of Requirements Modeling IREB Standard

Finding use cases

- External triggers: An actor (e.g. a neighboring system) wants to trigger a process in our system. Our system will notice this when data coming from the neighboring system crosses the system boundary. For example, "A guest wants a room in a hotel system". Once the request is received (i.e. the corresponding event in the system context hap- pens), the hotel system should offer a suitable room to the guest.

- Time triggers: It is time to execute a process in our system, for example, at specific times or on specific calendar days. By using time events to start a process, there is no need for data to cross the system boundary. It is only necessary that the specified point in time reached. E.g. in the hotel system: "It is 6pm and thus time to cancel all no- shows and make the rooms available for sales again." Monitoring of internal system re- sources is also considered as time event. For example, "It is time to reprint our hotel catalog".

Source: Handbook of Requirements Modeling IREB Standard

Template for the textual specification of use cases

Section	Content
ID	Unique identifier of the use cases in the development project or program
Name	Name of the use case in the model (this name is shown in the use case diagrams)
Trigger	Event that triggers the execution of the use case
Preconditions	Preconditions that must be fulfilled before execution of the use case
Postconditions	Set of postconditions that are fulfilled after successful execution of the use case
Input data	Input data of the use case
Output data	Output data of the use case
Result	Result of the use case, i.e. the added business value, which is provided to the actors after execution of the use case
Primary actor	Actor who receives the significant part of the added value of the use case
Further actors	Actors who are involved in the execution of the use case
Main scenario	Normal sequence of activities (execution flow, for example, in 70% of all cases).
Alternative scenarios	Set of alternative activities. Each alternative process leads also to a successful execution of the use case (e.g. in 30% of cases).
Exception scenarios	Set of exception scenarios. These scenarios are executed after entering an exceptional situation in the use case process. These scenarios ensure a controlled error and exception handling.

Data flow diagrams

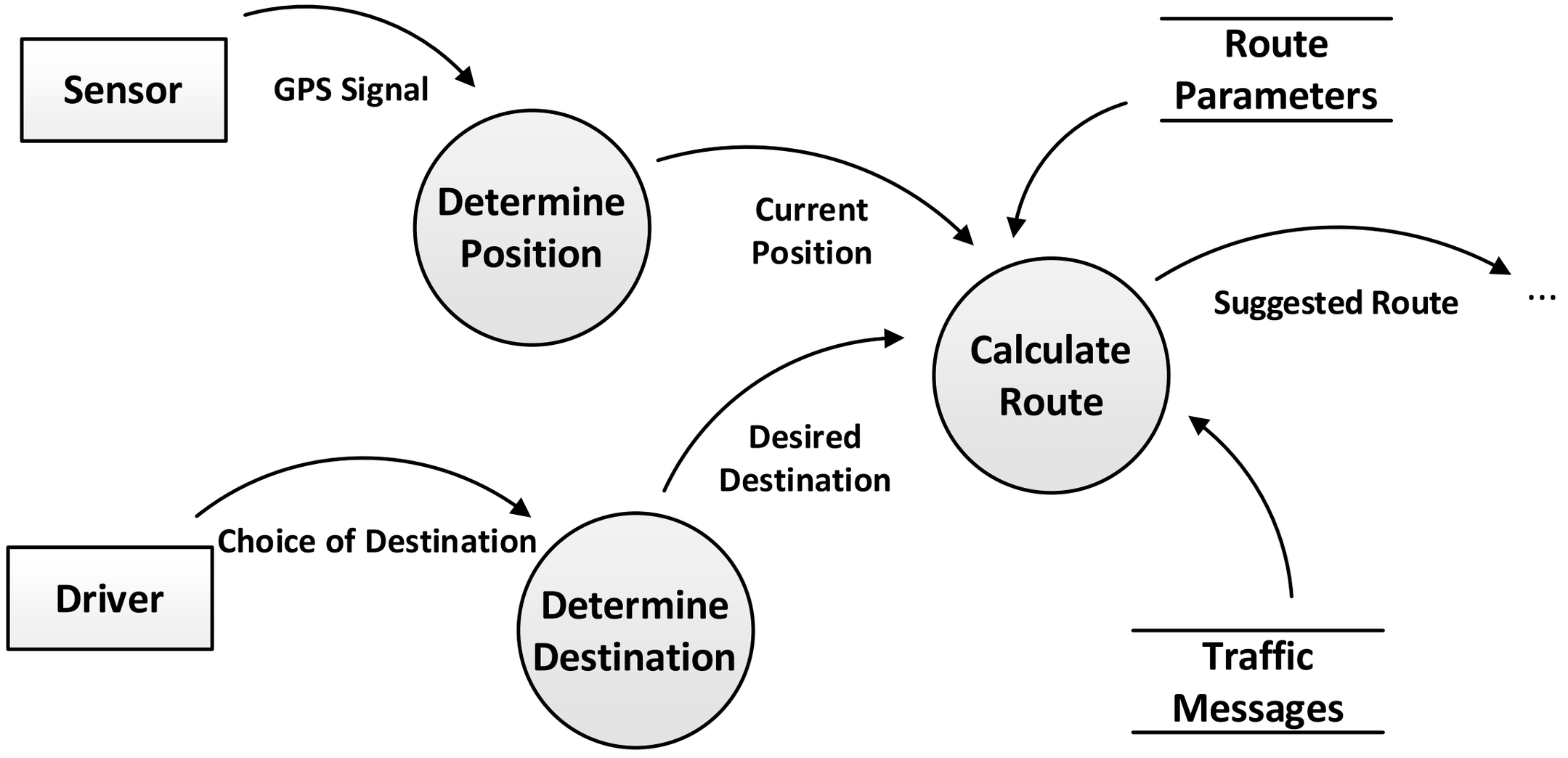
Source: Handbook of Requirements Modeling IREB Standard

Modeling elements of data flow diagrams

Name	Notation	Meaning
Neighboring system / Acor (also terminator, source or sink)	Name	Depicts persons, organizations of technical systems, equipment, sensors, actuators from the system environment, that are source of sink for the information to / from the system
Nodes (process, function of the system)	Name	Depicts a desired functionality in the system
Data flow	Name →	Depicts moving data (inputs, outputs, intermediate results). Not only data flows can be depicted but also material flow or energy flow.
Data store	Name	Depicts data at rest, i.e. information that is stored for a certain period and that is not directly flowing between functions

Source: Handbook of Requirements Modeling IREB Standard

Data flow in a navigation system



Source: Handbook of Requirements Modeling IREB Standard

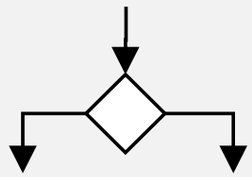
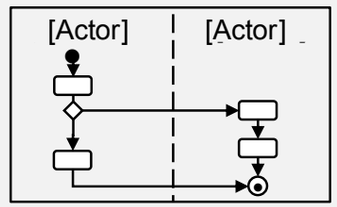
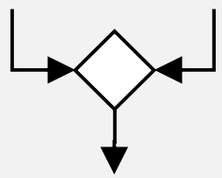
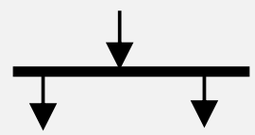
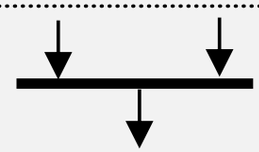
Relationship between data flow modeling and the rest

The data flow oriented modeling of requirements using data flow diagrams has a substantial connection with the context diagram, the use case view and the information structure view. Use Cases are a tool to specify systematically the functions within a defined scope from the user perspective and on a high level. During Requirements Engineering activities, these functions need to be detailed and decomposed in more detailed system functions and its dependencies. The system functions of a Use Case including data dependencies among each other and with actors (terminators) can be modeled using data flow diagrams. The more detailed system functions can be identified during the functional analysis of the Use Case scenarios. The structure of data, which is modeled in the data flow diagrams as data flows (i.e. "data in motion") and as data store (i.e. "data at rest"), is defined in the diagrams of the information structure view.

Activity diagrams

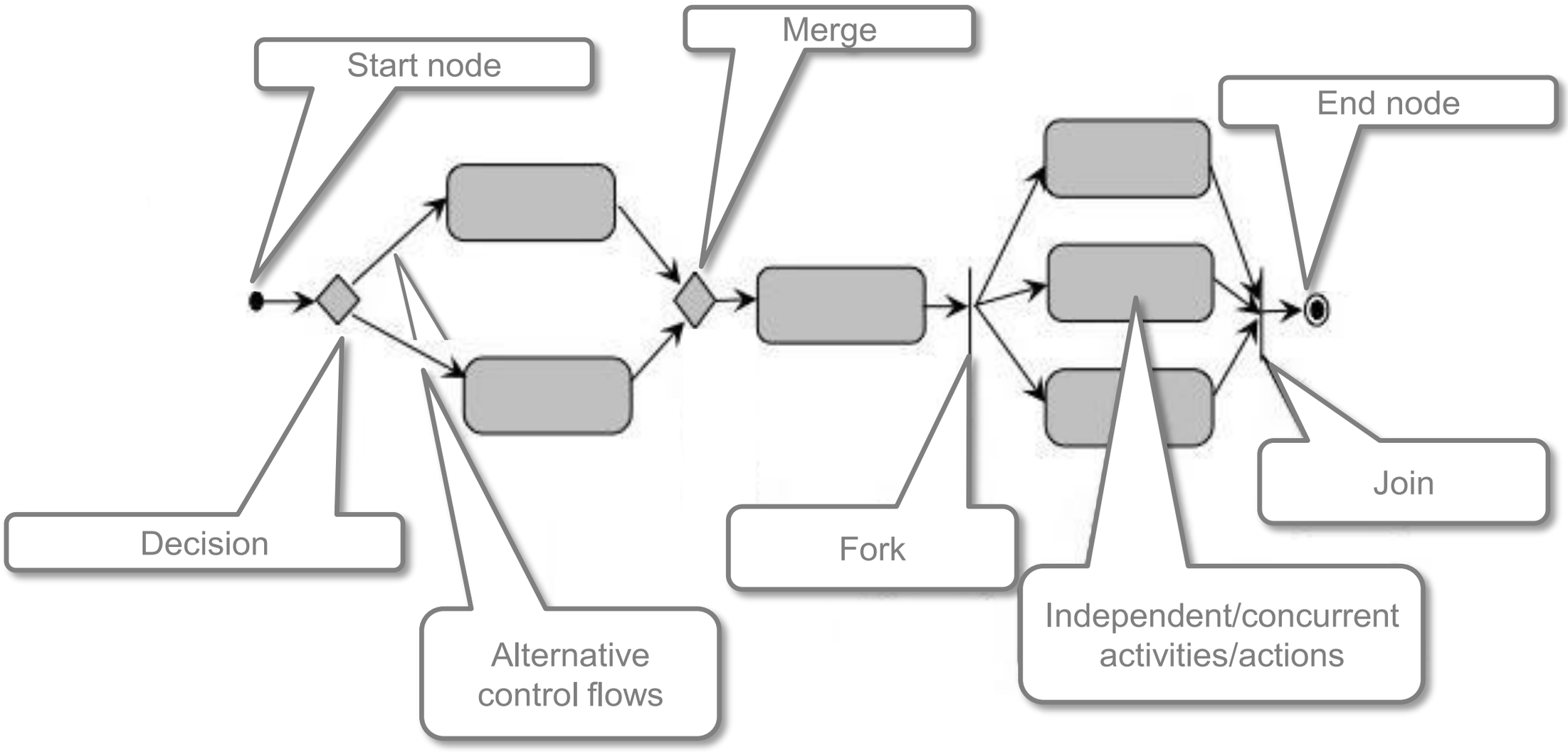
Source: Handbook of Requirements Modeling IREB Standard

Modeling elements of activity diagrams

Notation	Name	Notation	Name	Notation	Name
Name	Acivity/ action		Decision		Partitions (activity partitions)
●	Start node		Merge (of alternative control flows)		
◉	End node				
↓	Control flow		fork		
[condition]	Condition		join		Concurrency (Synchronisation bar)

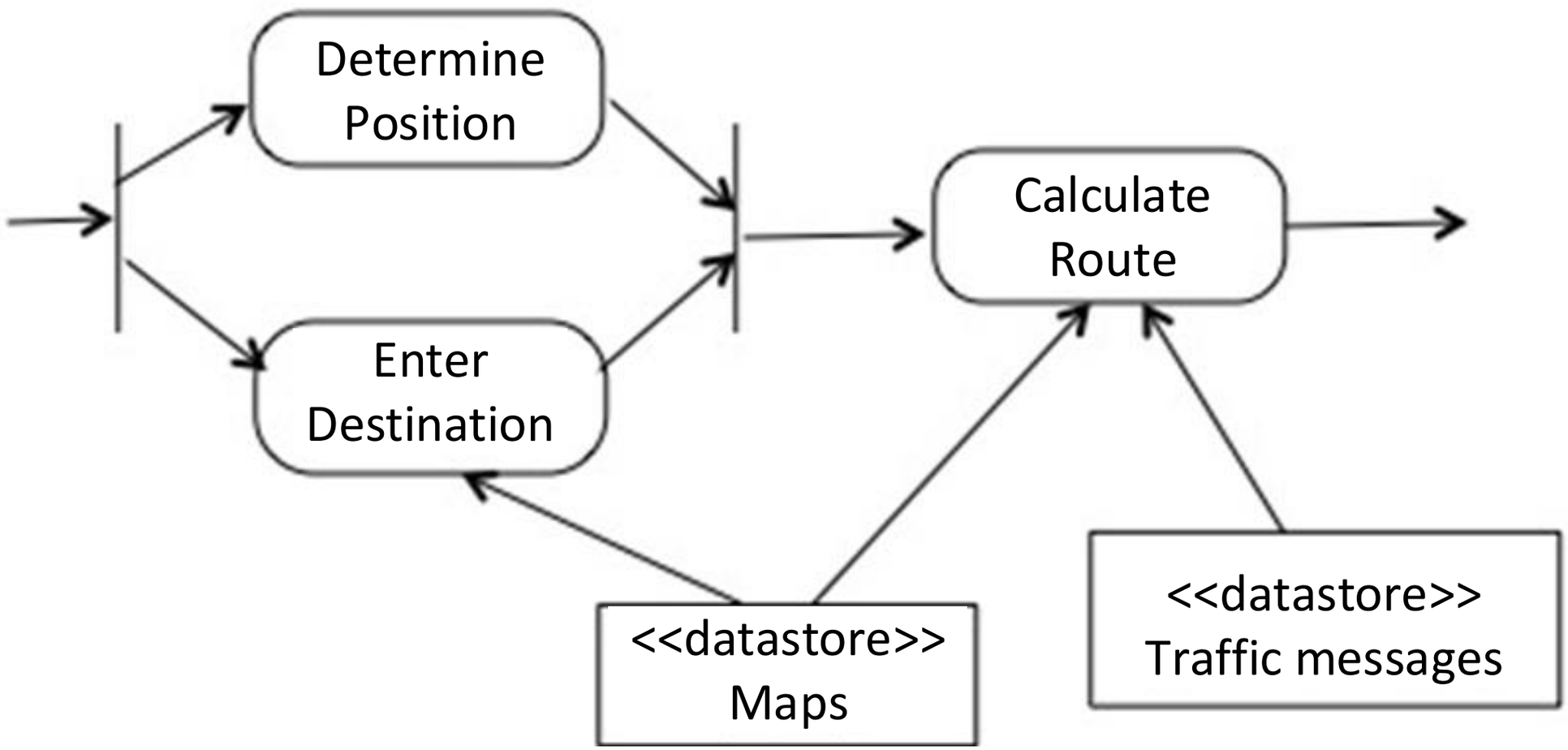
Source: Handbook of Requirements Modeling IREB Standard

Using the modeling elements



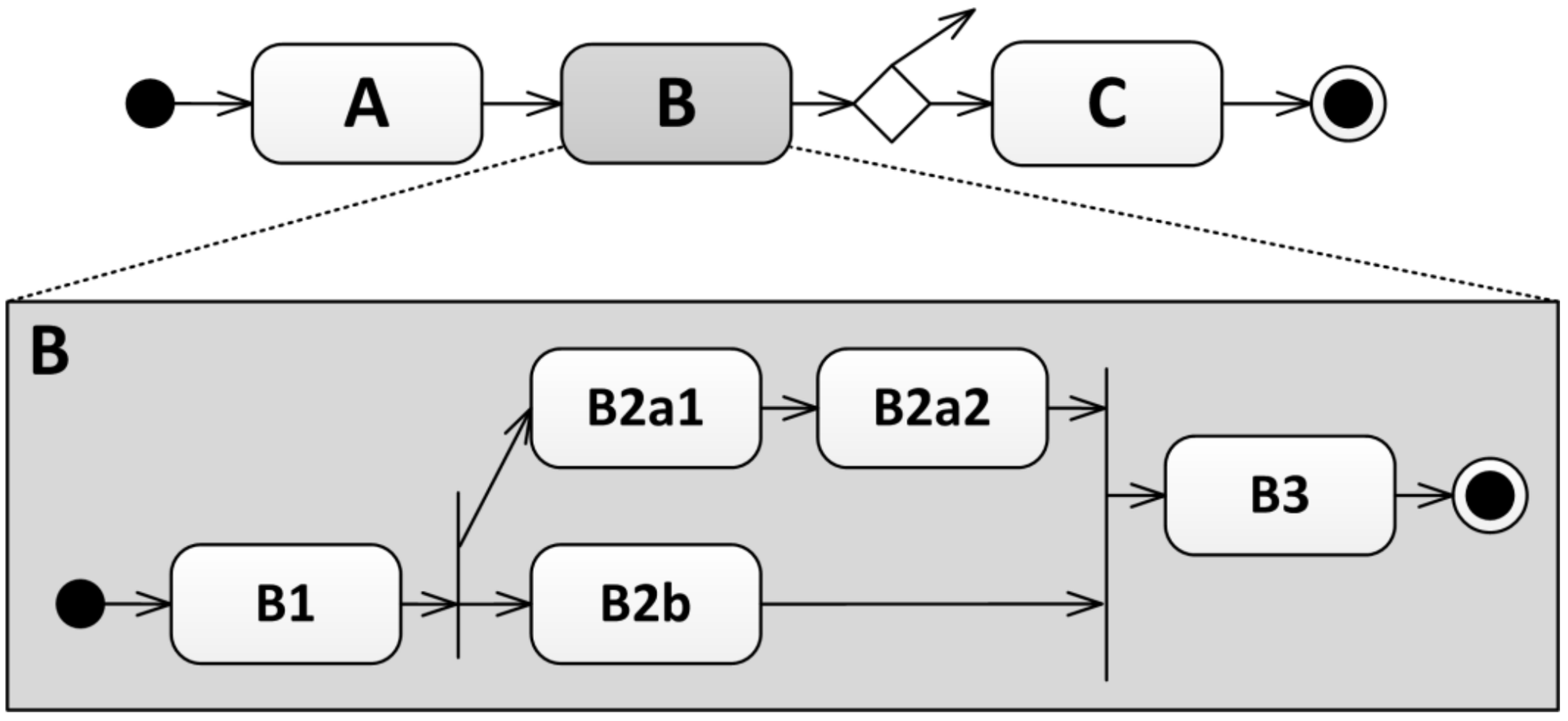
Source: Handbook of Requirements Modeling IREB Standard

Modeling of object flow in activity diagrams applied to the example of a navigation system



Source: Handbook of Requirements Modeling IREB Standard

Decomposition of a function in an activity diagram



Source: Handbook of Requirements Modeling IREB Standard

Textual description

Example: Textual description of the function “identify destination”

Function: identify destination

Input: destination selection (done by the user of the navigation system), map

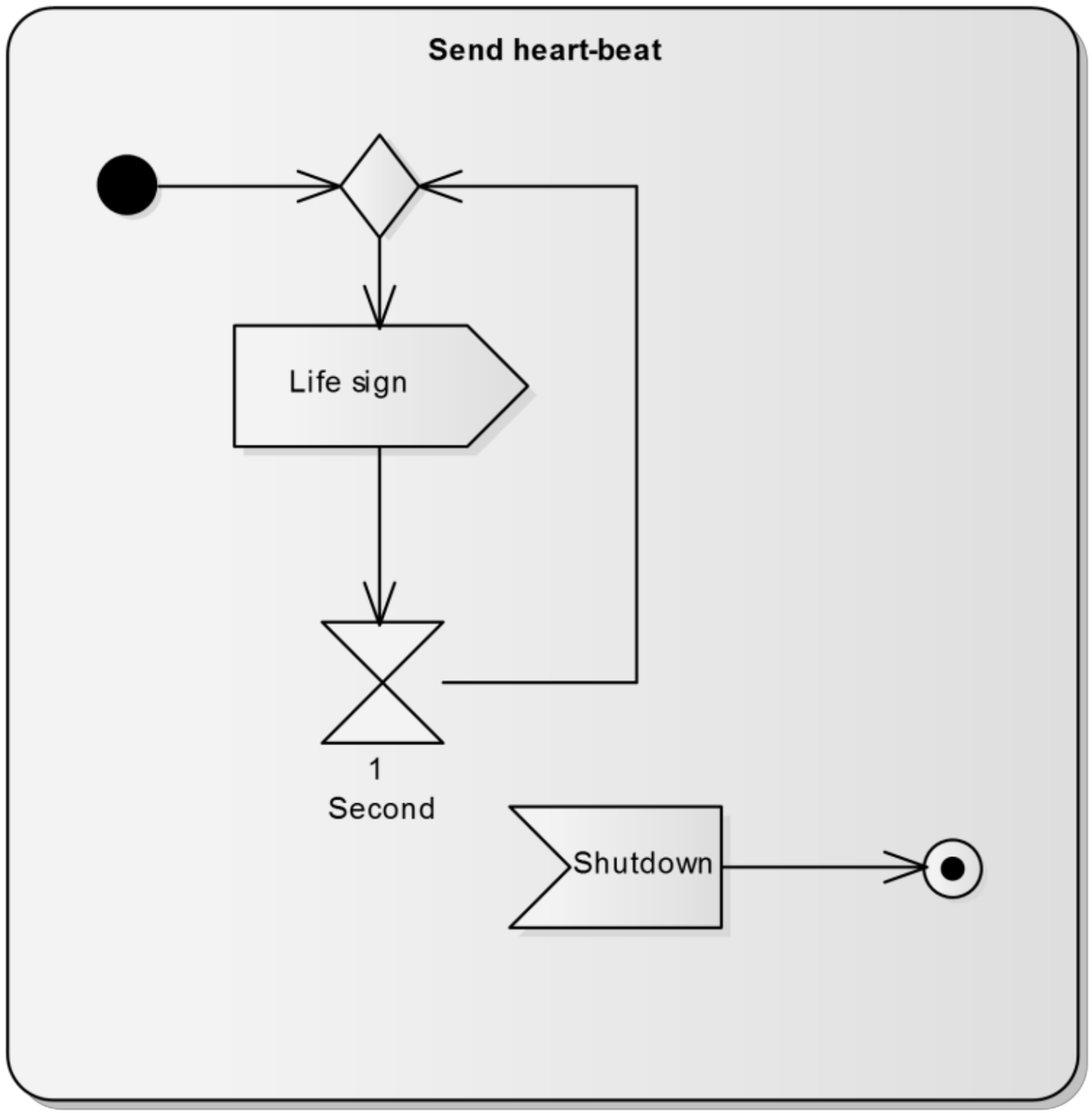
Output: desired destination

The function is to provide to the user four options to select a destination:

- *by entering an address using the keyboard*
- *by entering an address using voice entry*
- *by selecting from a list of stored addresses, or*
- *if a map is displayed by selecting a destination via the touch screen*

Source: Handbook of Requirements Modeling IREB Standard

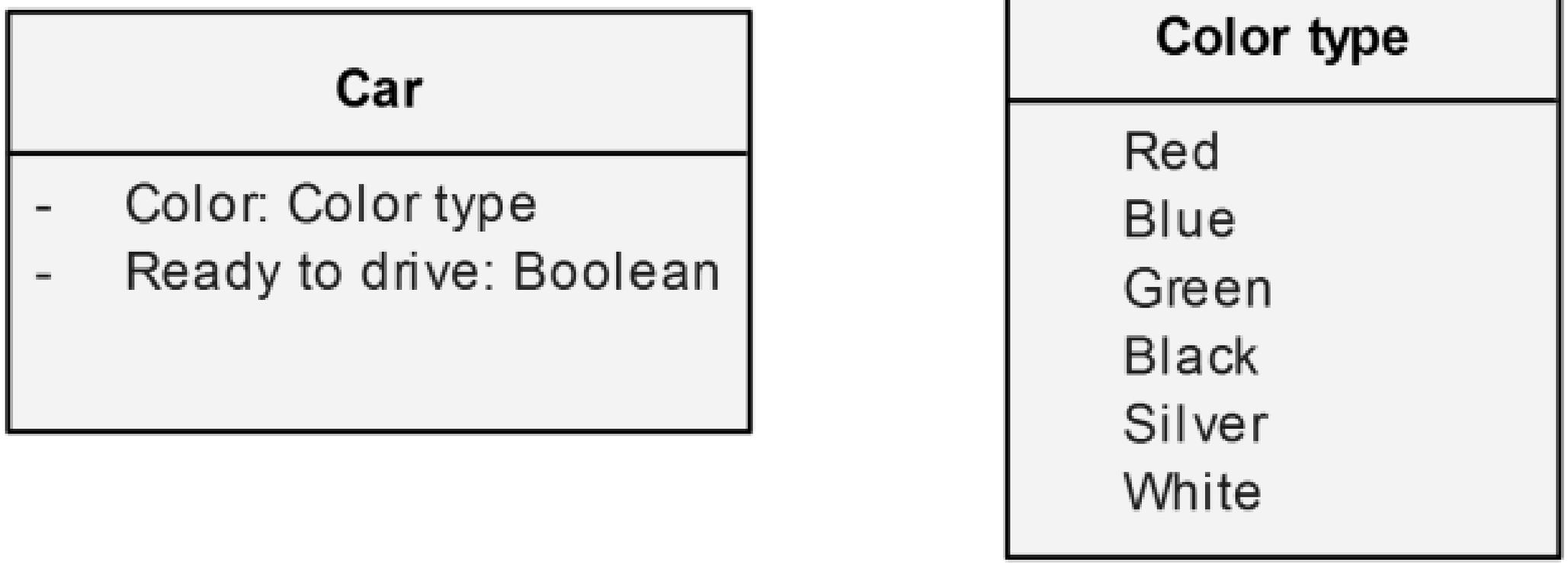
Receiving and sending messages (signals / interrupts)



State-oriented modeling

Source: Handbook of Requirements Modeling IREB Standard

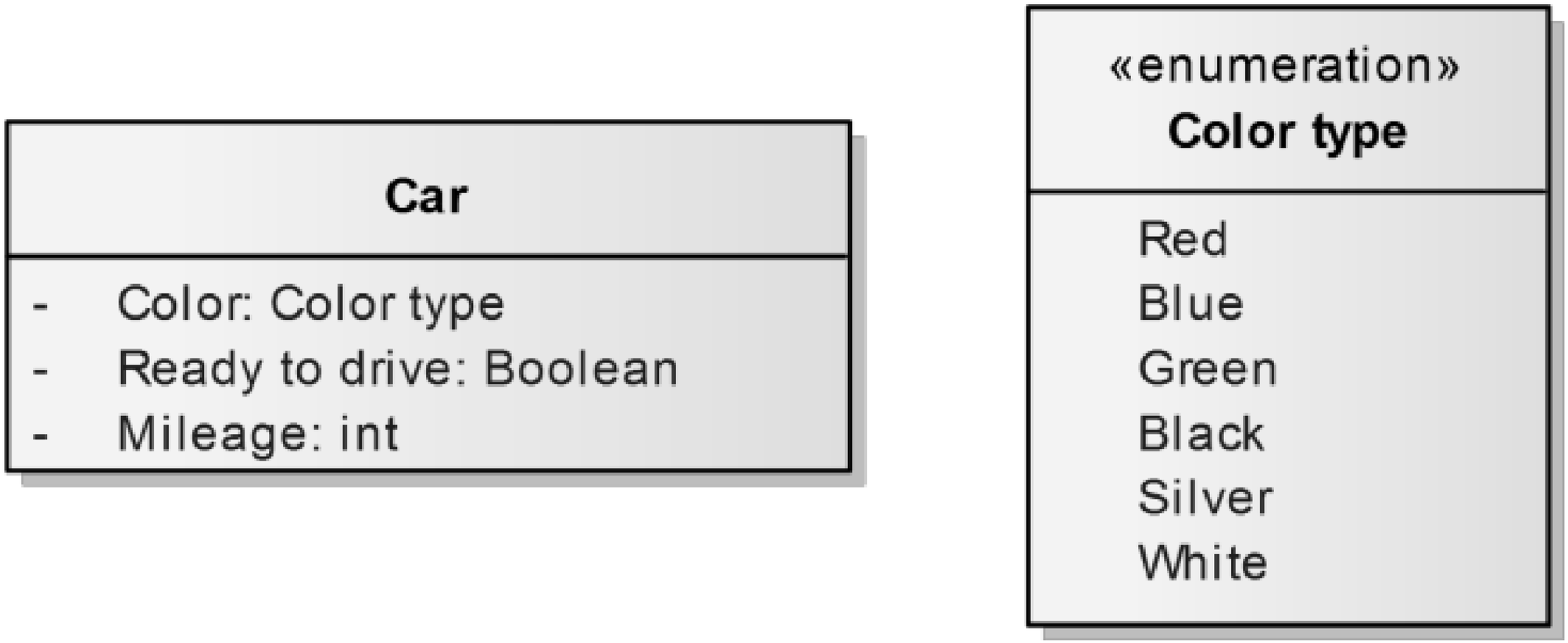
12 potential states of a car



State-oriented modeling could be applied.

Source: Handbook of Requirements Modeling IREB Standard

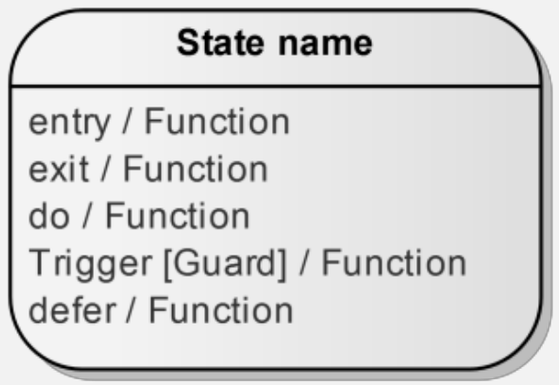
Too many states



State-oriented modeling cannot be applied.
Thus, omit attributes that are irrelevant for the state observation.
Also, form equivalence classes of possible attribute values.

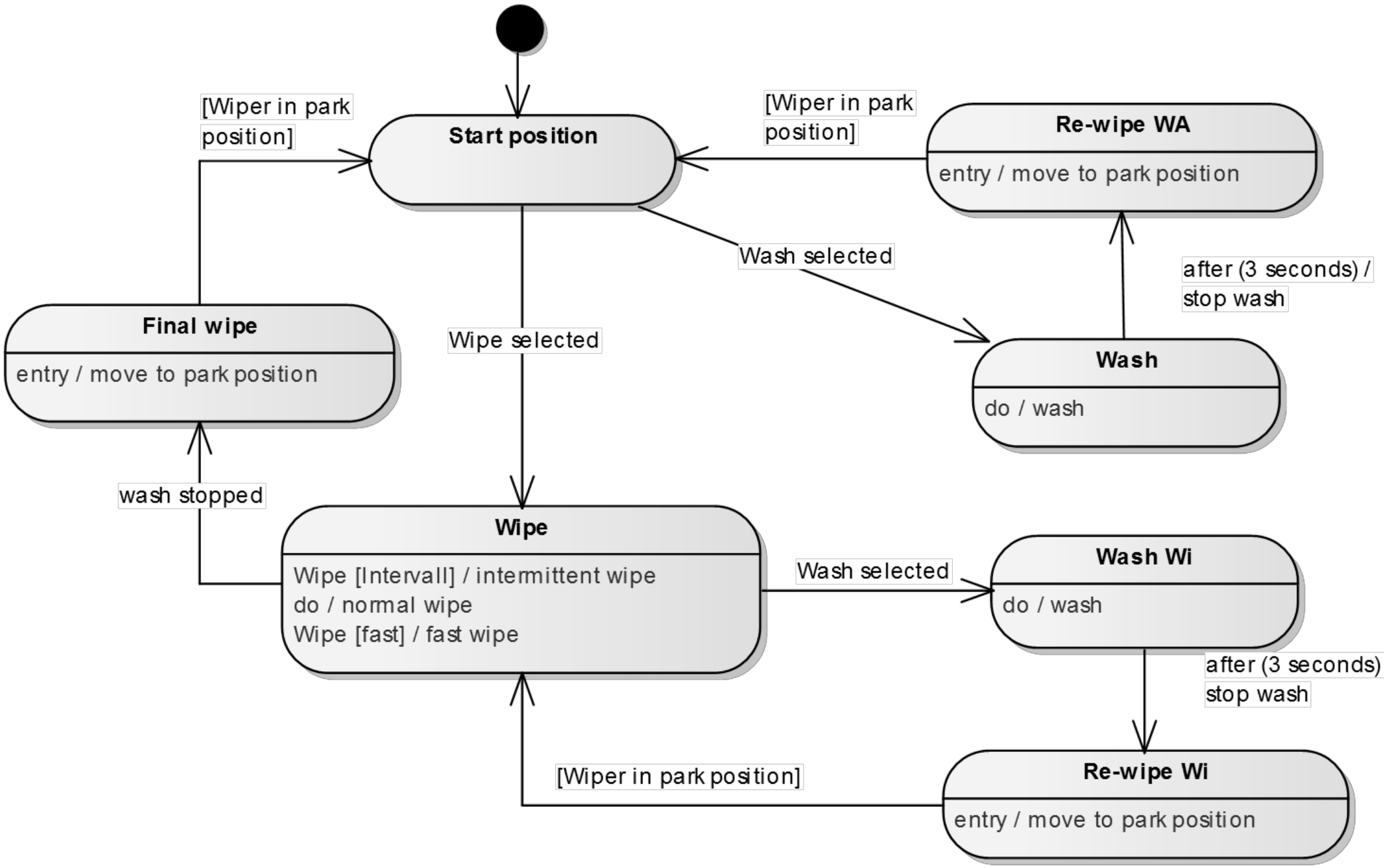
Source: Handbook of Requirements Modeling IREB Standard

Basic modeling elements of state diagrams

Notation	Name
	Simple state
	Transition
	Initial state
	Final state

Source: Handbook of Requirements Modeling IREB Standard

A state machine for a windshield wiper system



Source: Handbook of Requirements Modeling IREB Standard

States

A state should always have a name. In addition, you can specify in this state which functions are called. In UML, the following types of function calls are defined in a state. The italic identifiers are defined in the UML keywords with specific semantics. The identifier "function" re- fers to the function that is executed.

- Entry behavior: entry / Function: When entering a state, the function is executed. This cannot be interrupted.
- Exit behavior: exit / Function: When leaving a state, the function is executed. This can- not be interrupted.
- State function: do / Function: While the object of observation is in the state, the function is executed. This can be interrupted by a trigger, which leads to a state change.
- Triggered Function: trigger [guard] / Function: Upon arrival of the trigger and if the guard is true, the function is performed without leaving the state.
- Delay: trigger [guard] / defer: if an event in the deferred event list of the current state occurs, the event will be kept for future processing until a state is entered that does not list the event in its deferred event list.

Source: Handbook of Requirements Modeling IREB Standard

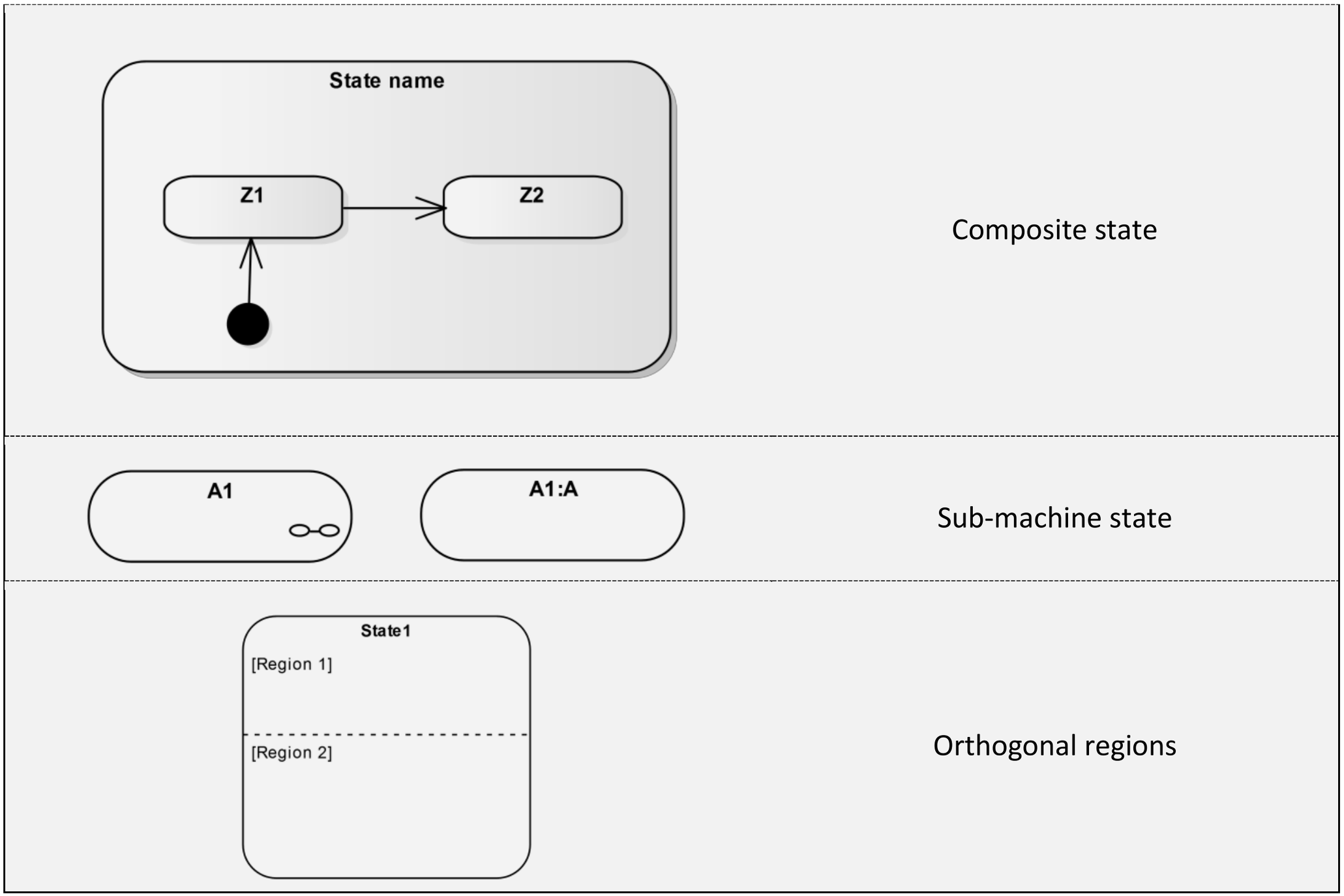
States cont'd

The following rules apply:

- A state is entered when a transition is passed that leads to this state as the end point.
- A state is abandoned (exit) when a transition is passed through that leads away from the state.
- A state becomes active as soon as it is entered. On leaving a state the state becomes inactive.
- Immediately after entering a state the entry behavior (here: Function 1) is executed. On leaving a state the last thing to happen is the execution of the exit behaviour.
- The state behavior of a state (Do-behavior) is the function (here: Function 3) that is started directly after ending the entry behaviour.
- A state can be abandoned (exit) through a transition only after the entry behavior (here: Function 1) has been fully executed.
- The initiation of Function 4 by a trigger under an optional guard condition does not lead to an external state change even if the behaviour of a function is part of the list of deferred behaviors of the state.

Source: Handbook of Requirements Modeling IREB Standard

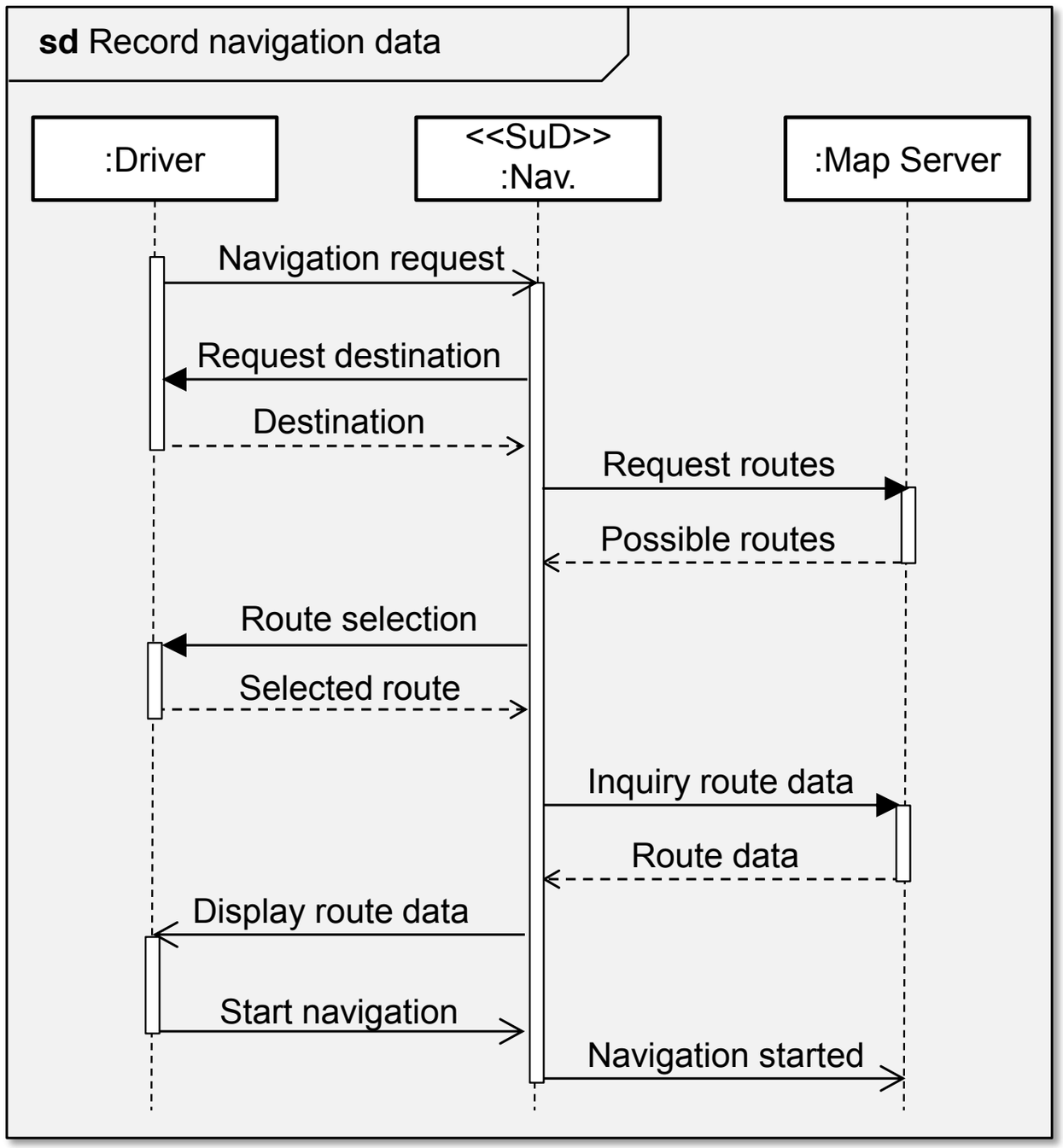
More modeling elements of state diagrams

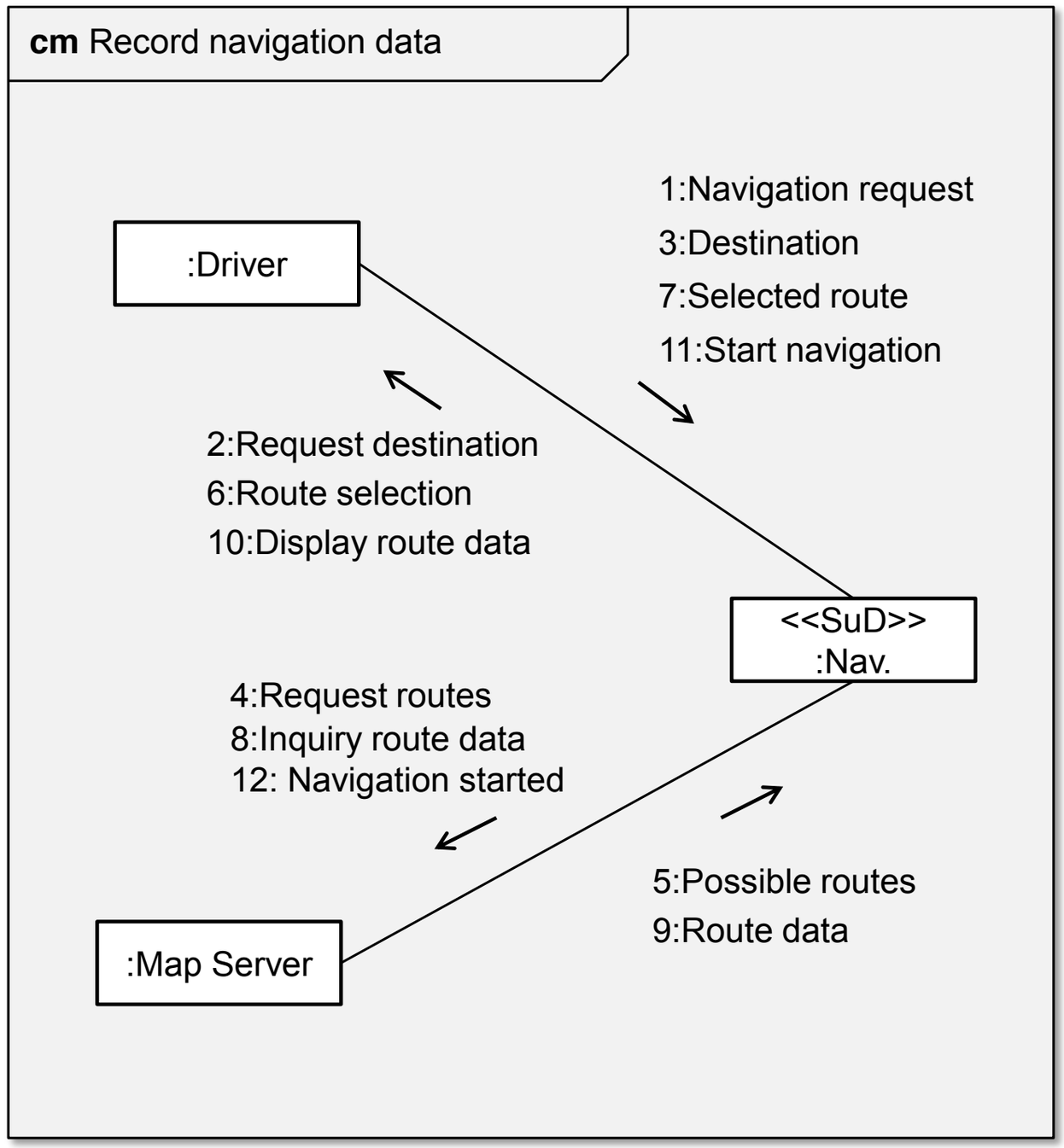


Scenario modeling

Source: Handbook of Requirements Modeling IREB Standard

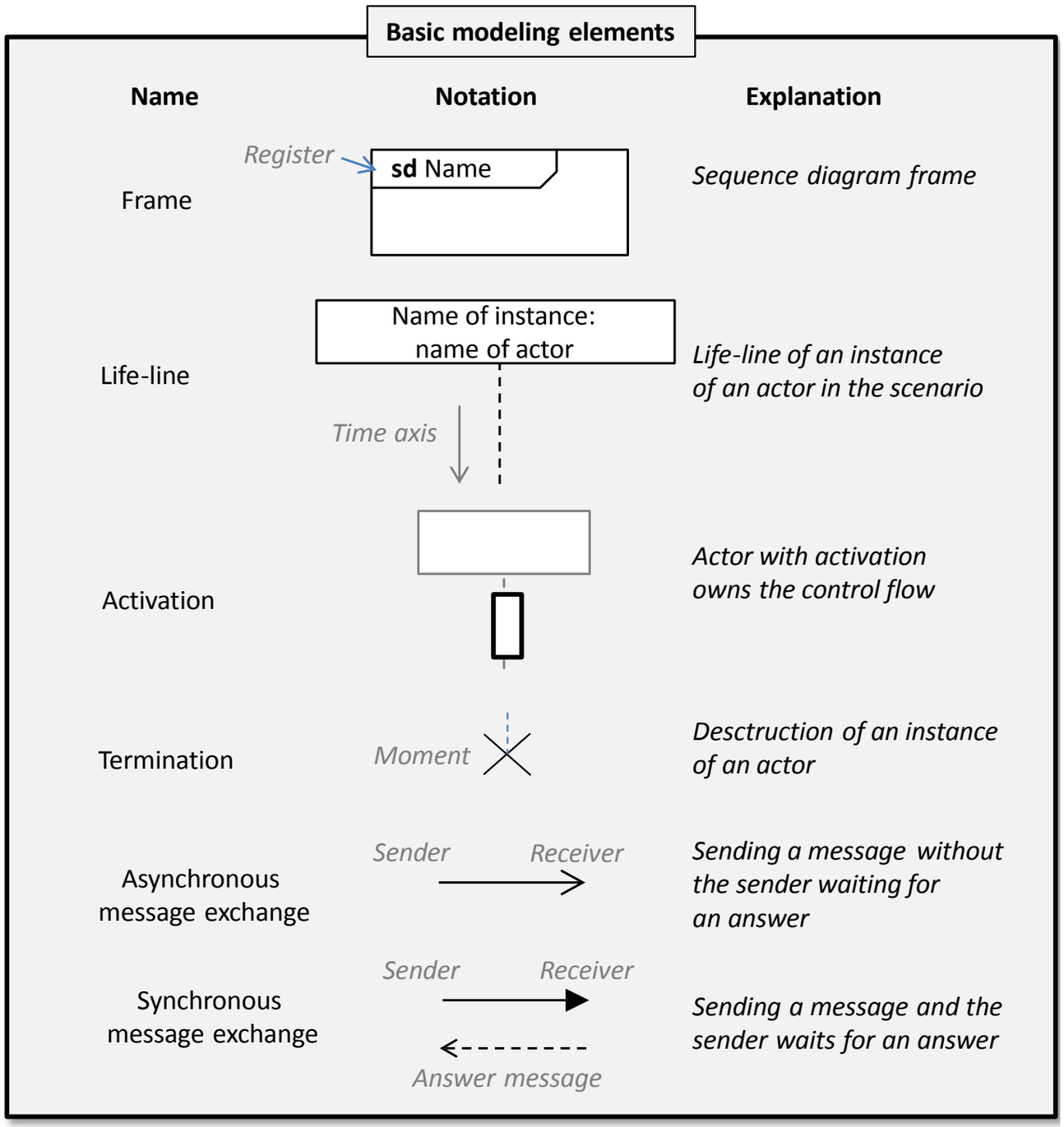
A navigation scenario modeled with sequence versus communication diagram





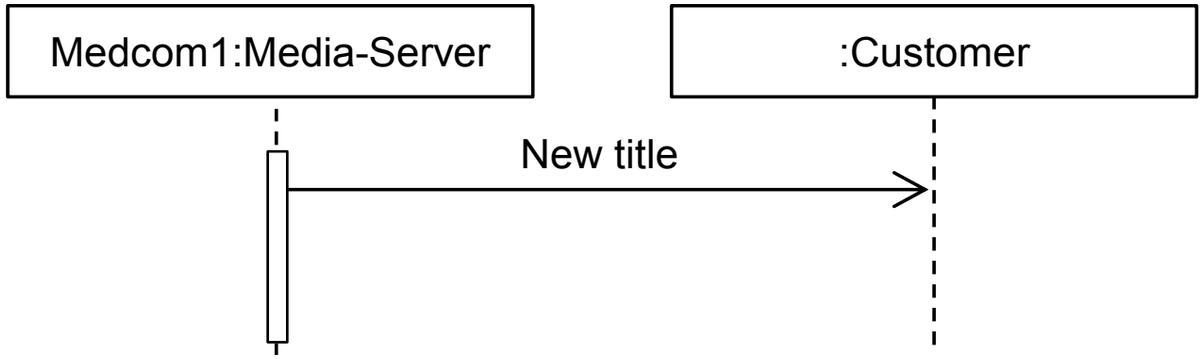
Source: Handbook of Requirements Modeling IREB Standard

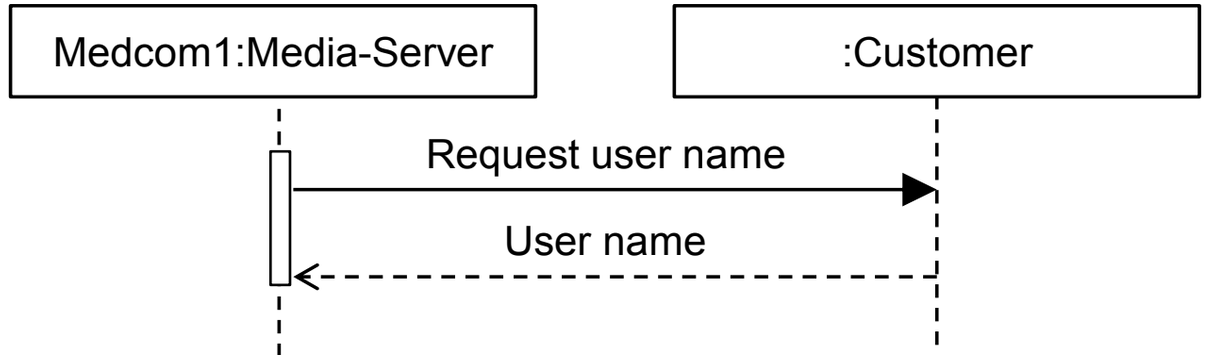
Modeling elements of scenario modeling with sequence diagrams



Source: Handbook of Requirements Modeling IREB Standard

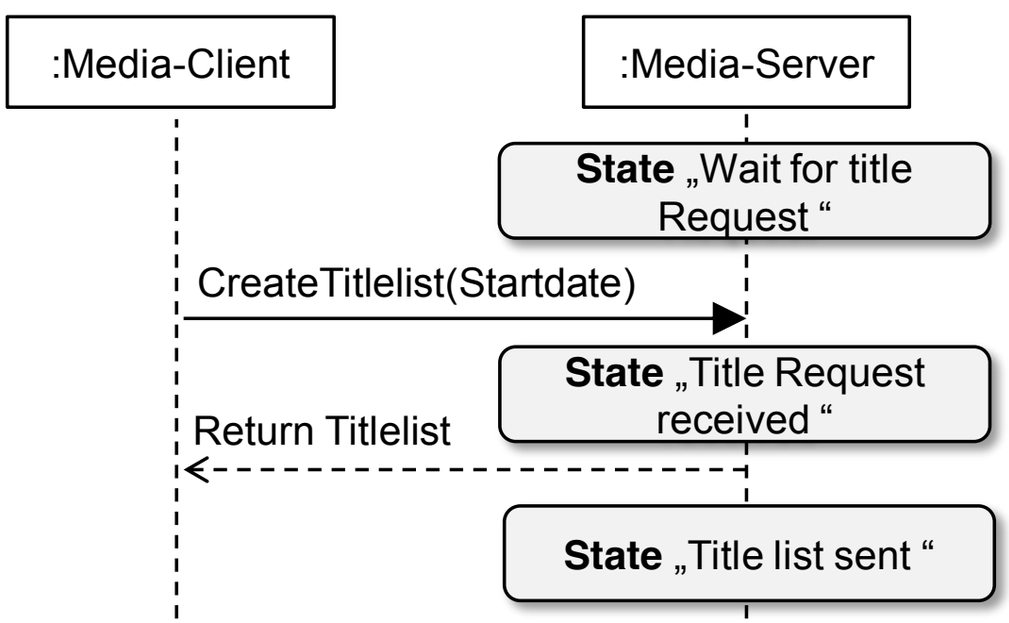
Modeling asynchronous and synchronous messages



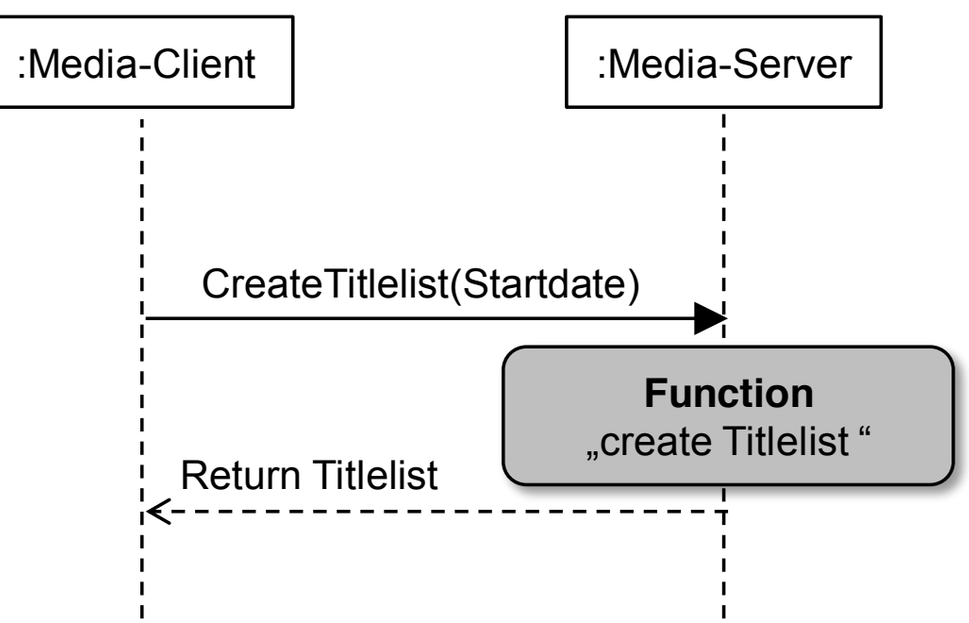


Source: Handbook of Requirements Modeling IREB Standard

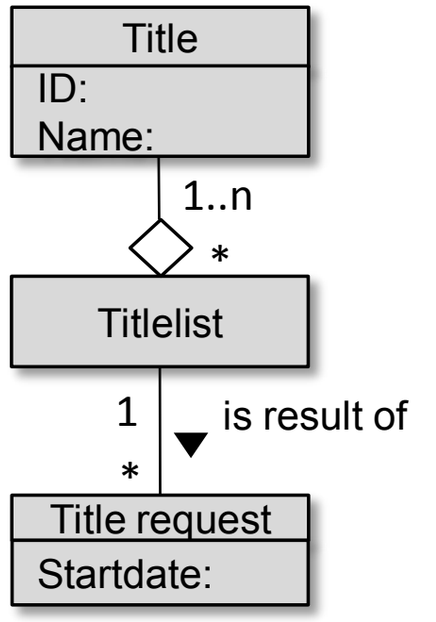
Relationship of messages with the rest



(a) States of :Media-Server



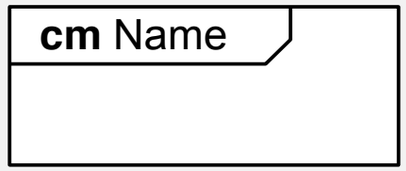
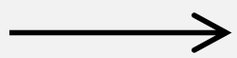
(b) Functions of :Media-Server



(c) Information structures

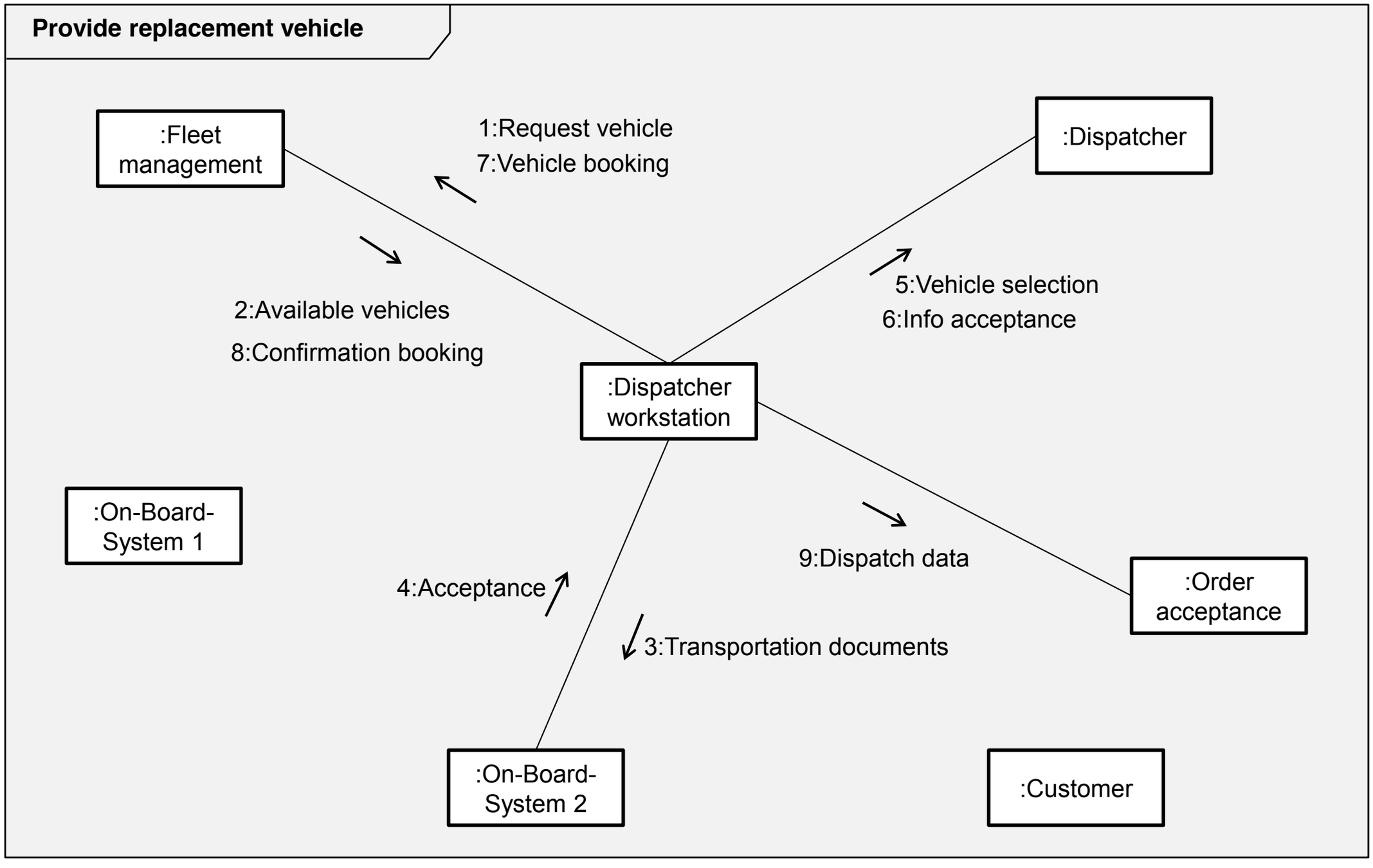
Source: Handbook of Requirements Modeling IREB Standard

Modeling elements of scenario modeling with communication diagrams

Name	Notation	Explanation
Frame		*Frame of the communication diagram*
Lifeline		*Lifeline of an actor in the scenario*
Message exchange		*Models a generic message exchange between actors*
Direction of communication		*Models the direction of a message exchange*
Message signature	**Sequence number: message**	*Each message in a scenario is provided with a sequence number corresponding to the order of occurrance a message*

Source: Handbook of Requirements Modeling IREB Standard

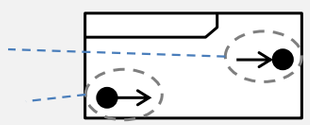
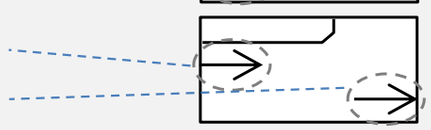
Scenario modeling with a communication diagram



Source: Handbook of Requirements Modeling IREB Standard

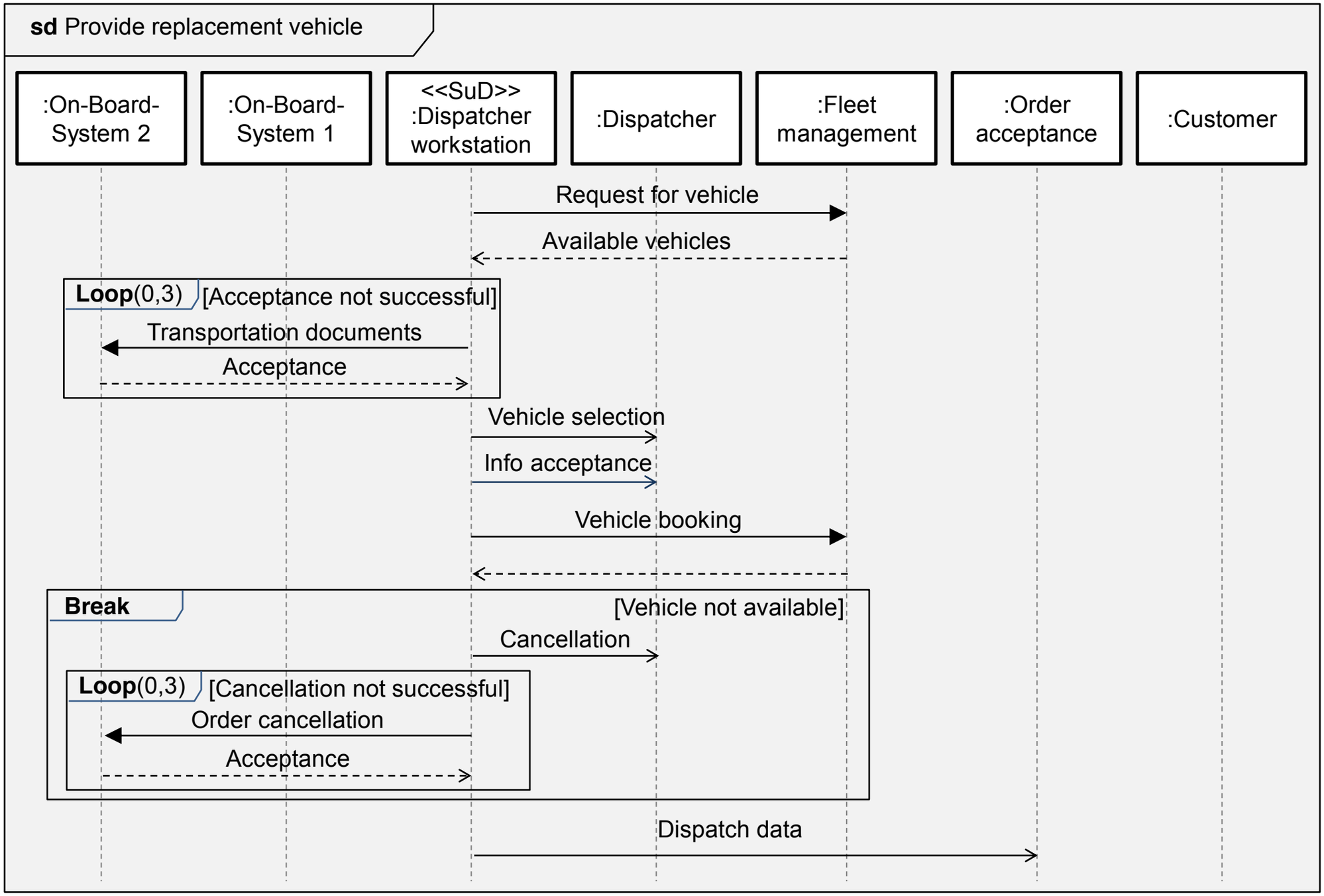
Modeling elements of scenario modeling with sequence diagrams

Advanced modeling elements

Name	Notation	Explanation
Combined fragments		
Alternatives	*Register* → **alt** [condition] / [¬ condition] ← *Interaction frame*	*Modeling alternative interaction, Depending on conditions*
Optional	**opt** [condition]	*Modeling of an optional interaction , depending on a condition*
Reference	**ref** Name	*Modeling of a reference to an interaction of an other sequence diagram*
Repetition	**loop(0,m)** [condition]	*Repetition of the interaction, m times or up to m times, depending on the condition*
Termination	**break** [condition]	*Modeling of an interaction that will be executed on occurance of a termination condition*
Advanced message types		
- Lost message - Found message		*Message of which the source / receiver is unknown*
- Incoming message - Outgoing message		*Externally incoming, or externally outgoing message*

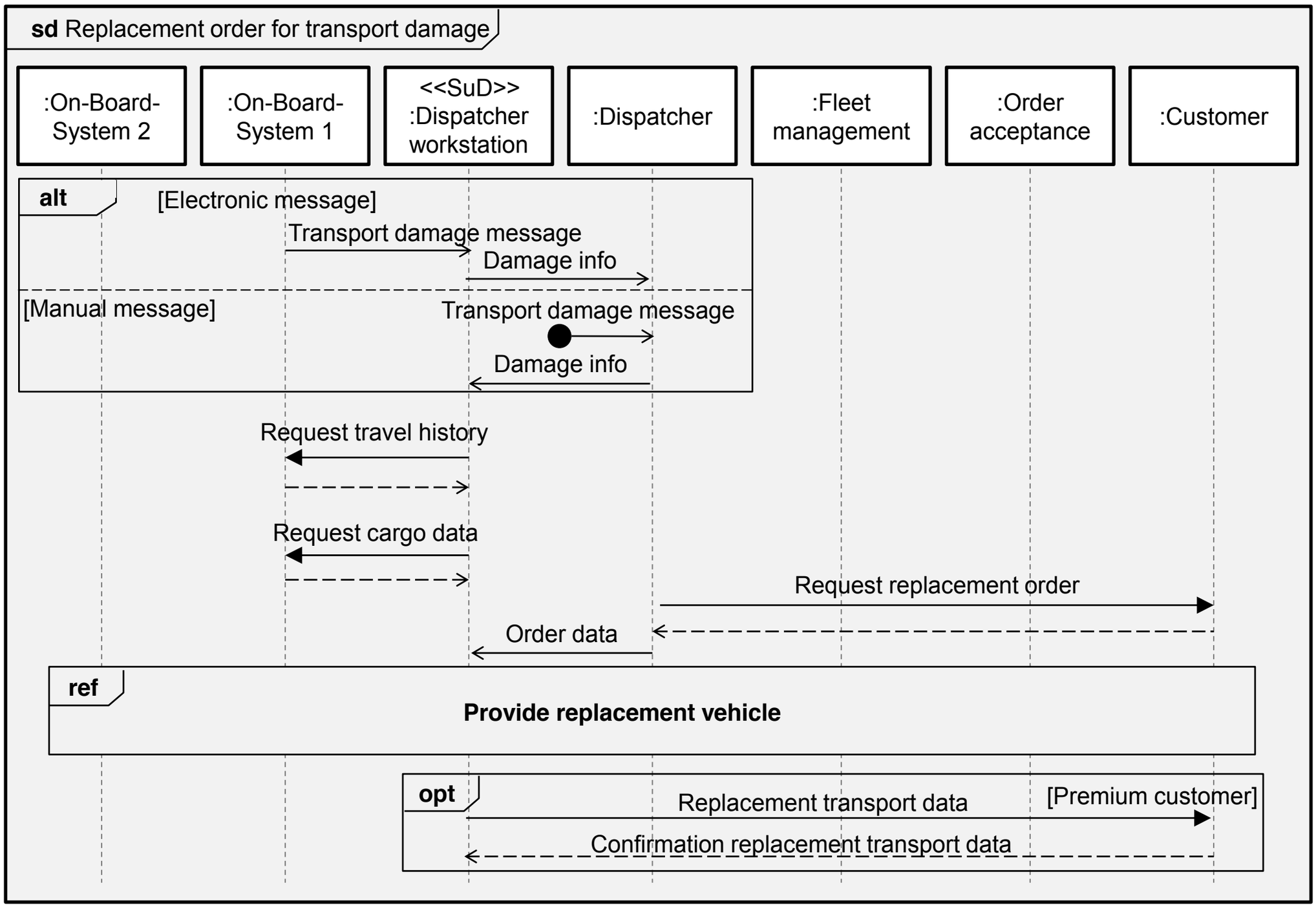
Source: Handbook of Requirements Modeling IREB Standard

Scenario modeling with a sequence diagram



Source: Handbook of Requirements Modeling IREB Standard

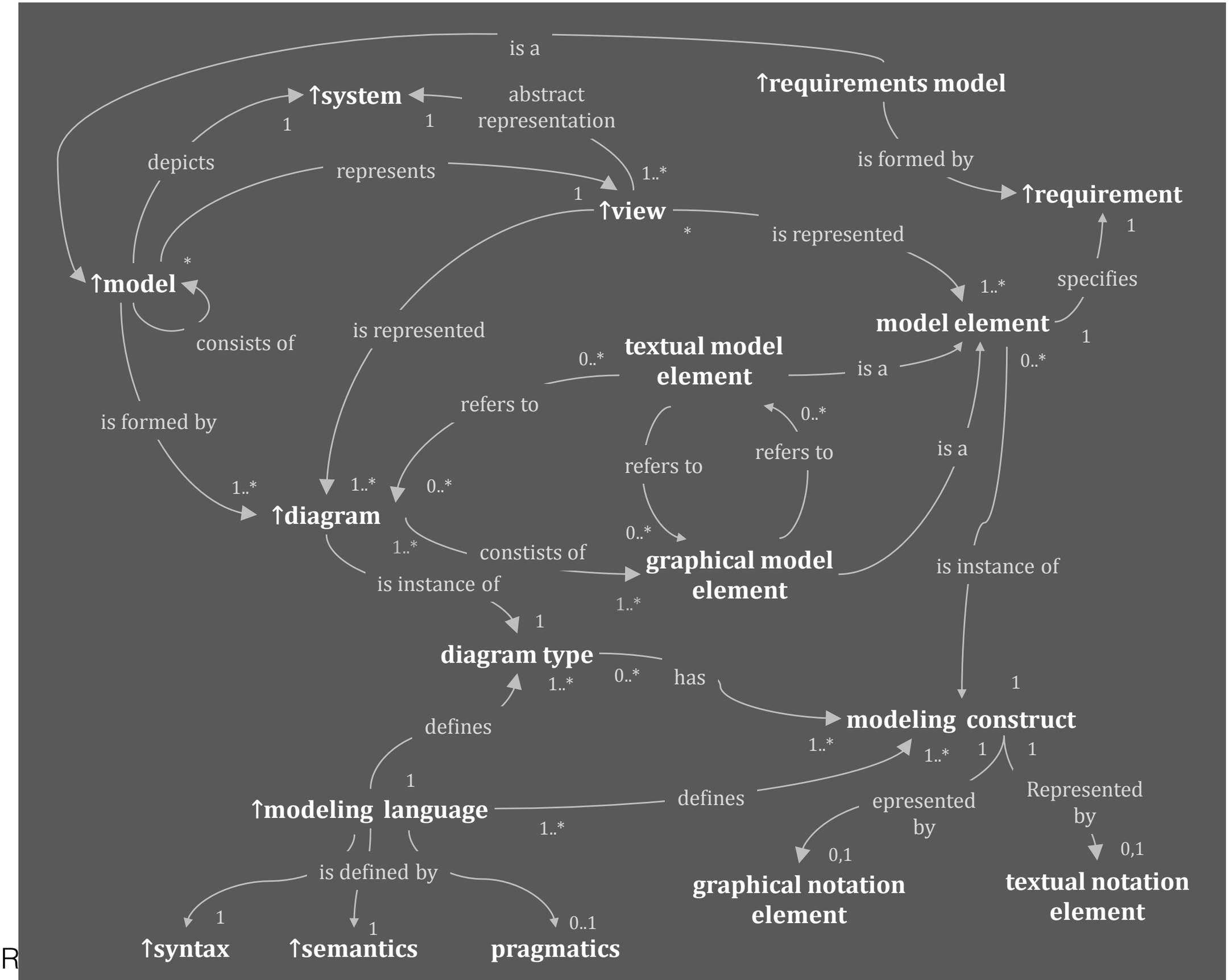
Scenario modeling with a sequence diagram



Summary

Source: Handbook of Requirements Modeling IREB Standard

Conceptual network of the core terminology in requirements modeling



Source: Handbook of Requirements Modeling IREB Standard

Glossar

- Action
- Activity
- Activity diagram
- Actor
- Aggregation
- Alternative scenario
- Association
- Attribute
- Basic scenario
- Class
- Class diagram
- Communication diagram
- Composition
- Context diagram
- Context view
- Control flow
- Data flow
- Data flow diagram
- Data type
- Diagram
- Diagram type
- Event
- Exception scenario
- Function
- Generalization
- Instance scenario
- Interaction
- Interaction-based view
- Model
- Model element
- Modeling construct
- Modeling language
- Object
- Operational context
- Pragmatic quality
- Pragmatics
- Process flow
- Requirements view
- Requirements model
- Role
- Scenario
- Semantics
- Semantic quality
- Sequence diagram
- Signal
- State
- State diagram
- State machine
- State machine diagram
- Statechart
- Syntactic quality
- Syntax
- System
- System boundary
- System context
- System environment
- System under development
- System under study
- Transition
- Trigger
- Type scenario
- Use case
- Use case diagram
- View