

47006- ANÁLISE E MODELAÇÃO DE SISTEMAS

Behavioral models (sequence, state)

Ilídio Oliveira

v2020/11/13, TP11

Learning objectives for this lecture

Understand the role of behavior modeling in the SDLC

Understand the rules and style guidelines for sequence, communication and state diagrams

Understand the complementarity between sequence and communication diagrams

Map sequence diagrams in code and reverse

Explain the relationship between function, structural and behavior models

The target of behavioral modeling

Behavioral models describe the dynamic aspects of an information system.

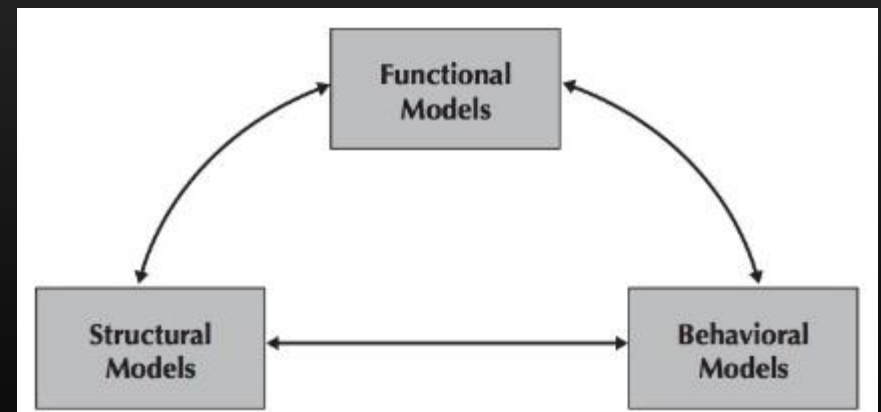
During analysis: behavioral models describe what the **internal logic of the processes** is without specifying how the processes are to be implemented. (in the design and implementation phases, the detailed design is fully specified)

Behavioral modeling is also **use-case driven**. You specify use cases realizations.

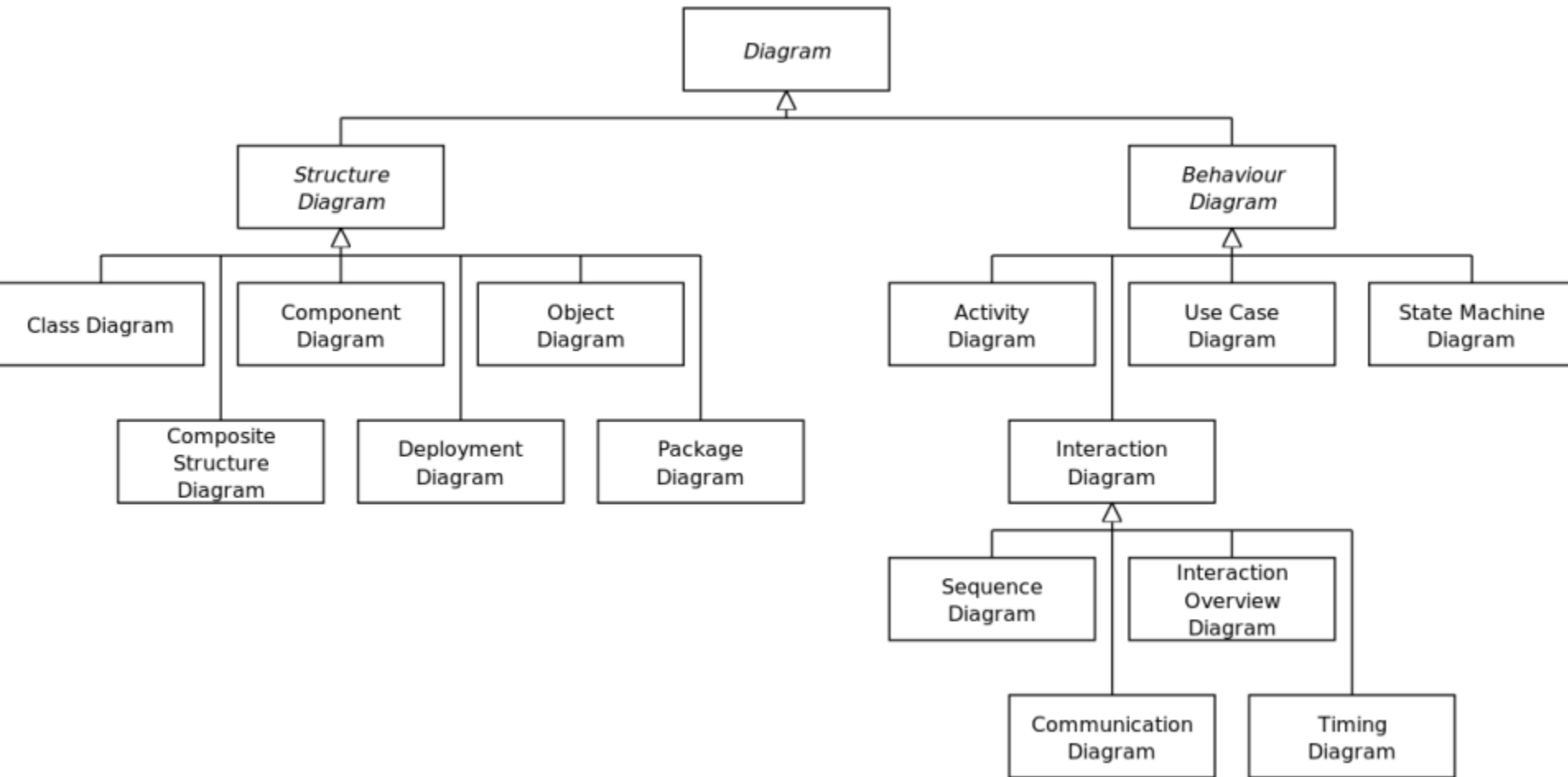
One of the primary purposes is to **show how the underlying objects in a problem domain will work together to form a collaboration** to support each of the use cases' scenarios.

Structural models → the objects and the relationships

Behavioral models → internal view of the process that a use case describes.



Diagramas da UML 2.x



Behavioral model types

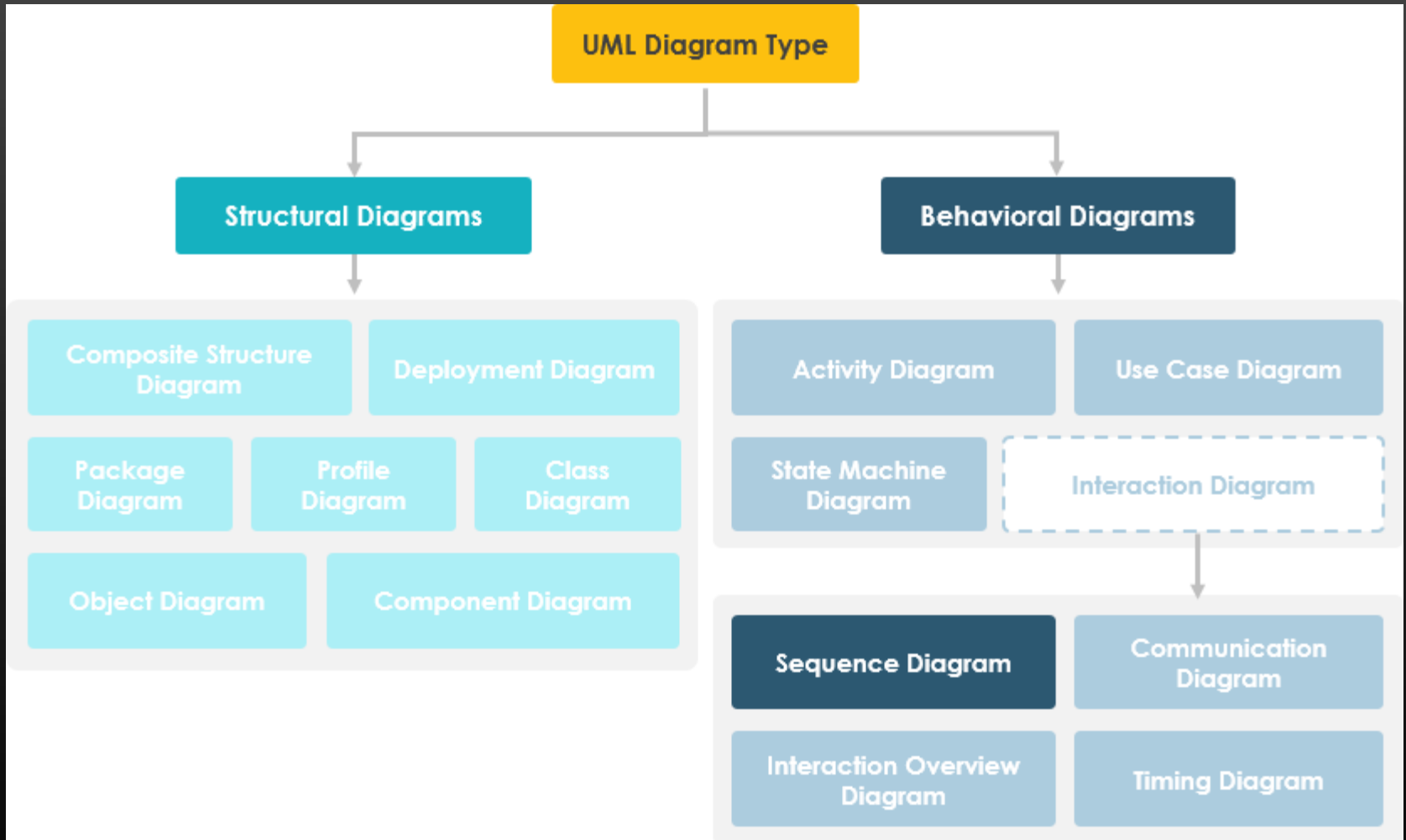
Representation of the details of a process

- Interaction diagrams (Sequence & Communication)
- Shows how objects collaborate to provide the functionality defined in the use cases.

Representations of changes in the data (state)

- Behavioral state machines

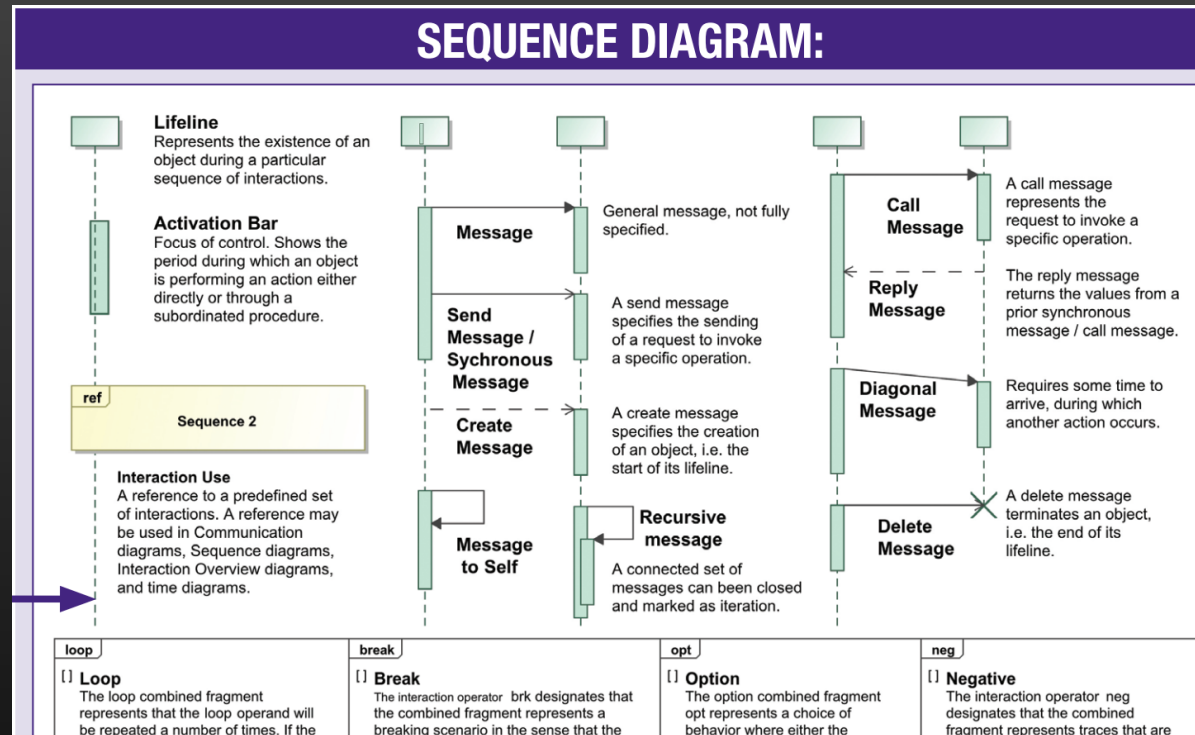
UML diagrams



Sequence diagram

Illustrate the objects in a collaboration and the **messages** that pass between them over **time**.

A sequence diagram is a dynamic model that shows the explicit sequence of messages that are passed between objects in a defined interaction.



NoMagic's [UML Reference card](#)

Focus on **object-level collaboration**

Class diagrams

The modeling focus of class diagrams is at the class level.

Each object has attributes that describe information (state) about the object.

Interaction diagrams

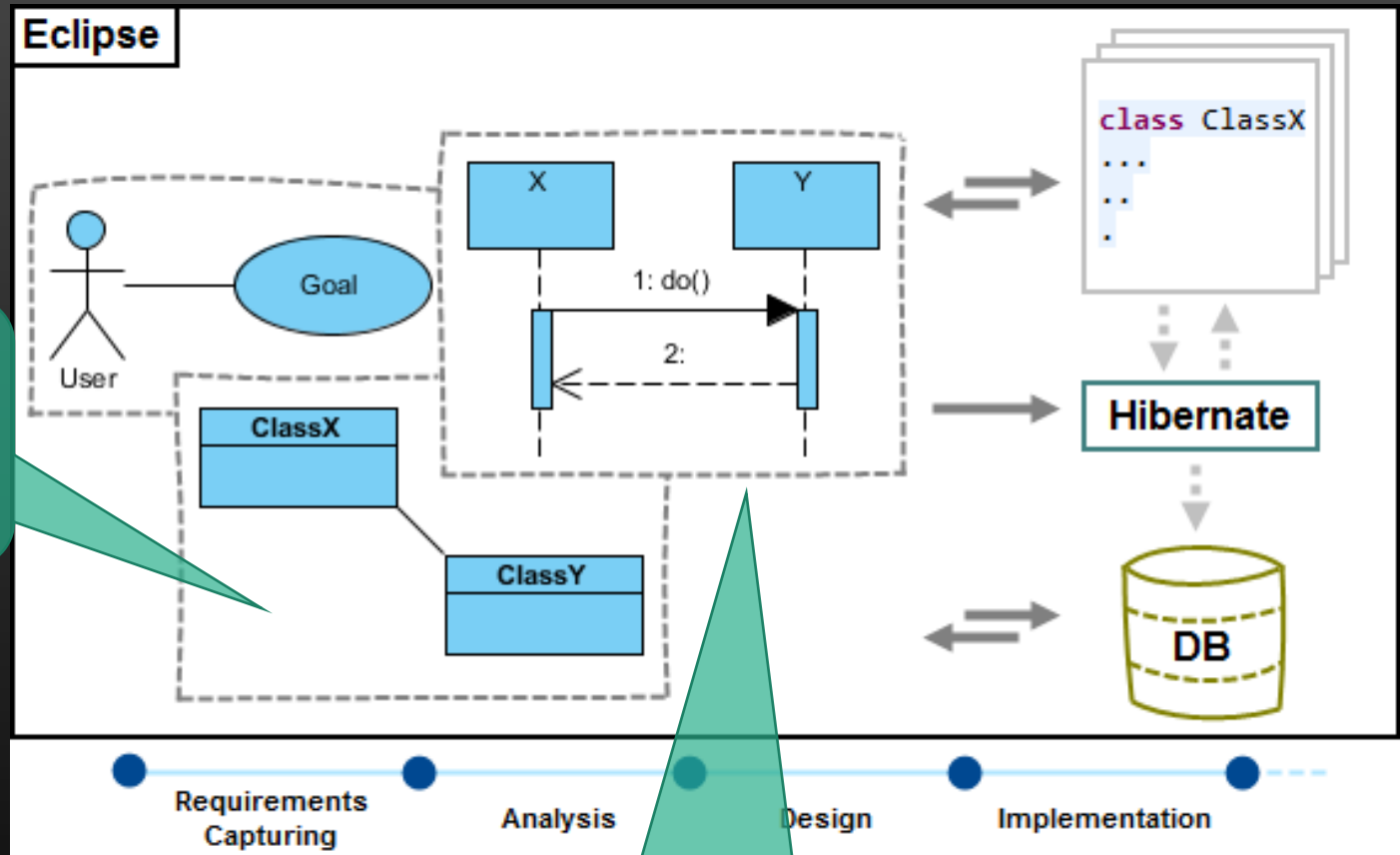
The interaction diagrams focus on the **object level**

Each object also has behaviors. The behaviors are described by operations. An operation is an action that an object can perform.

Each object also can **send and receive messages**. Messages are information sent to objects to tell an object to execute one of its behaviors. Essentially, a message is a function or a *call* from one object to another object.

Nível de abstração: análise ou implementação?

As Classes definem características dos objetos de um tipo (*molde*).
As classes podem representar entidades do domínio ou da implementação.



O D. de Sequência mostra como é que os objetos (que seguem uma Classe) colaboram com outros, para realizar uma atividade. Também aqui, essa atividade pode ser no nível do "domínio" ou no nível do código.

participant1 : ParticipantClass

participant2

Time



participant1:ClasseParticipante

participant2:

1: mensagem(argumentos)

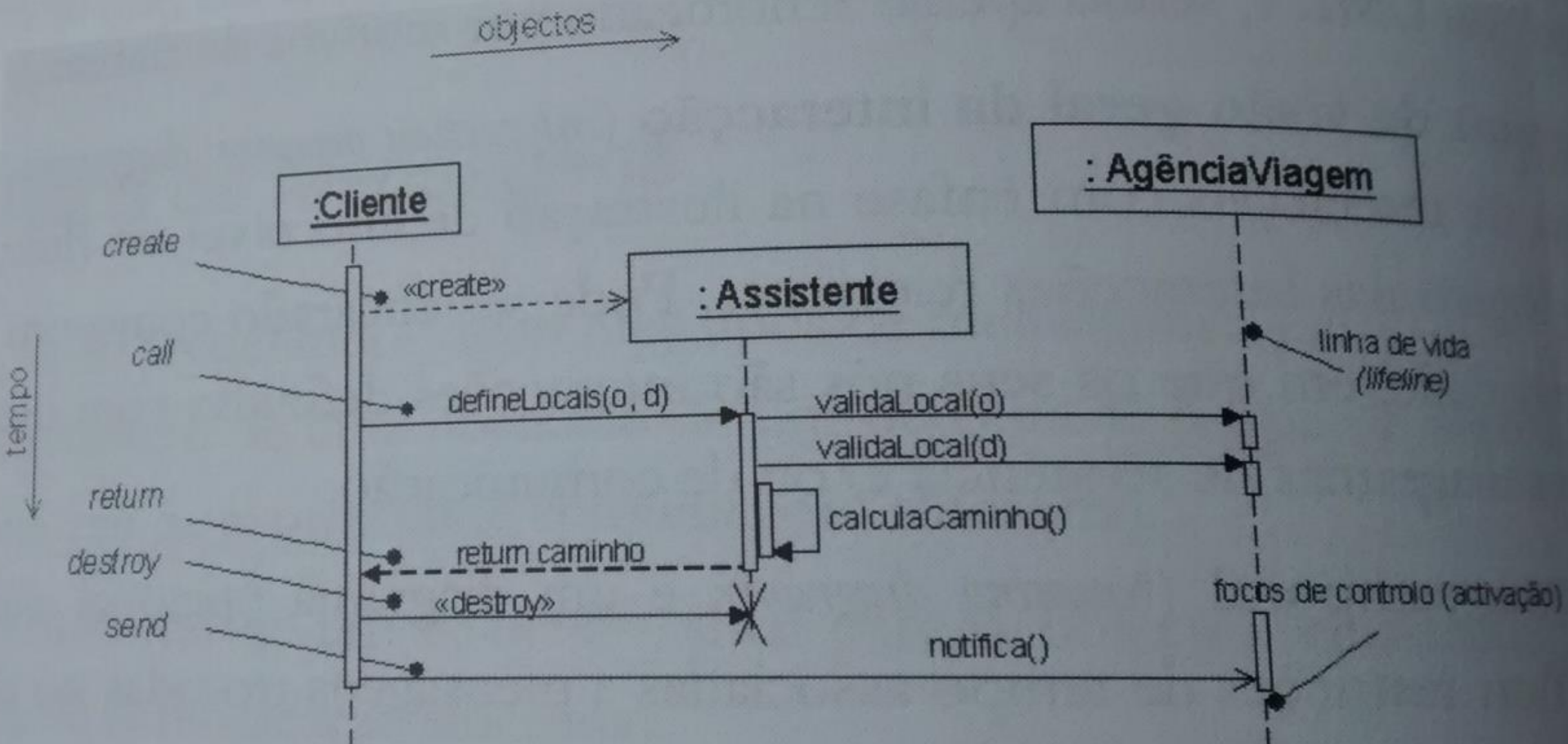
O invocador da mensagem (solicita colaboração)


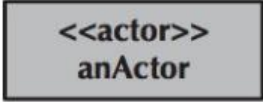


O recetor da mensagem (presta colaboração/serviço)

Mensagem, com parâmetros (assinatura da mensagem)

Retorno (opcional)

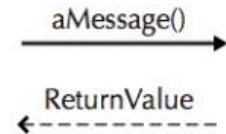
Colaboração entre objetos por mensagens (síncronas)



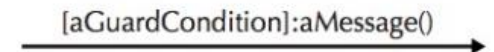
Term and Definition	Symbol
<p>An actor:</p> <ul style="list-style-type: none"> ■ Is a person or system that derives benefit from and is external to the system. ■ Participates in a sequence by sending and/or receiving messages. ■ Is placed across the top of the diagram. ■ Is depicted either as a stick figure (default) or, if a nonhuman actor is involved, as a rectangle with <<actor>> in it (alternative). 	 <p>anActor</p> 
<p>An object:</p> <ul style="list-style-type: none"> ■ Participates in a sequence by sending and/or receiving messages. ■ Is placed across the top of the diagram. 	
<p>A lifeline:</p> <ul style="list-style-type: none"> ■ Denotes the life of an object during a sequence. ■ Contains an X at the point at which the class no longer interacts. 	

A message:

- Conveys information from one object to another one.
- A operation call is labeled with the message being sent and a solid arrow, whereas a return is labeled with the value being returned and shown as a dashed arrow.

**A guard condition:**

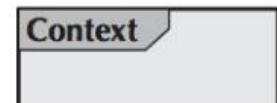
- Represents a test that must be met for the message to be sent.

**For object destruction:**

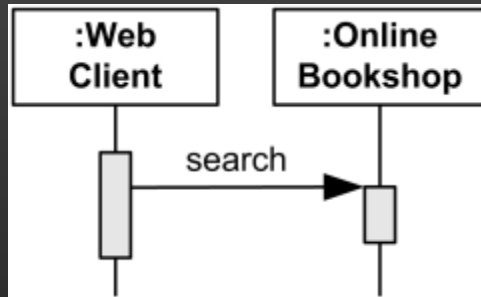
- An X is placed at the end of an object's lifeline to show that it is going out of existence.

X**A frame:**

- Indicates the context of the sequence diagram.



Semântica da invocação

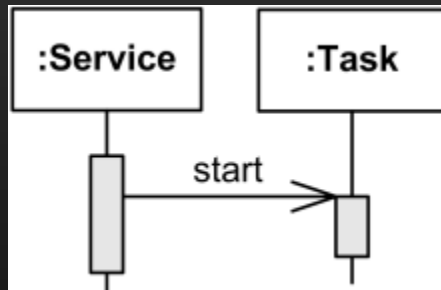


síncrona

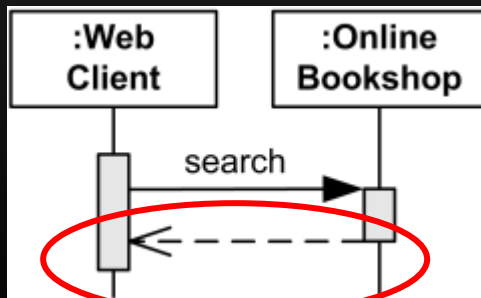
...

```
result = B.search();
```

...

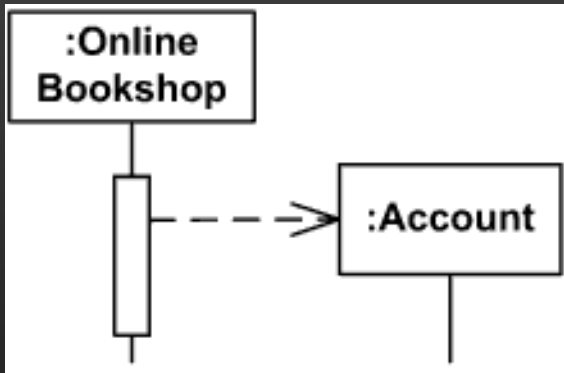


assíncrona



retorno

Modelar a criação/destruição do participante

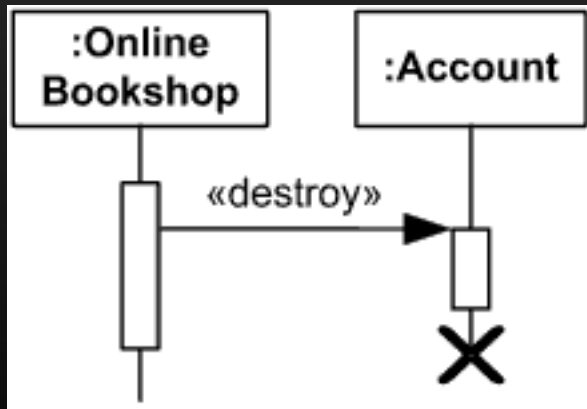


create

...

```
Account a= new Account()
```

...



destroy

...

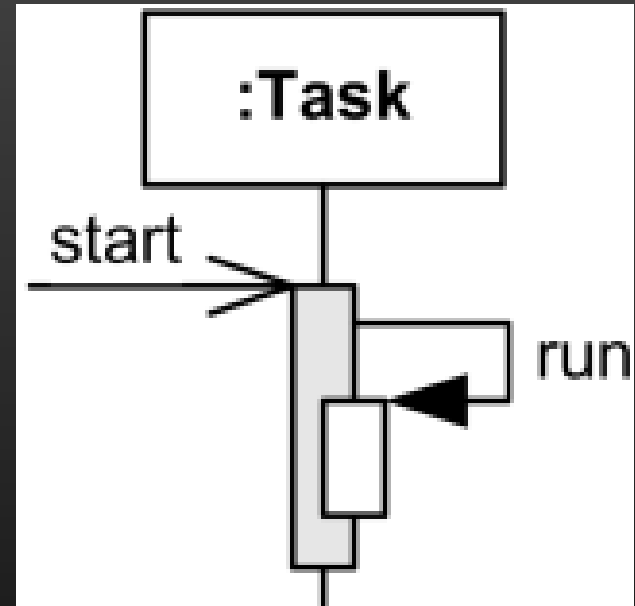
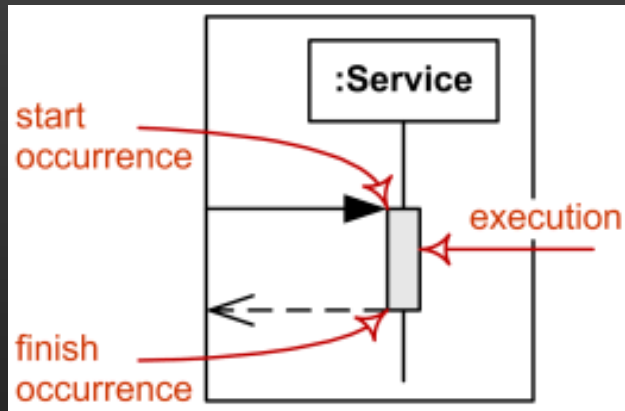
```
a=null; // in java
```

```
[a release] // objective C
```

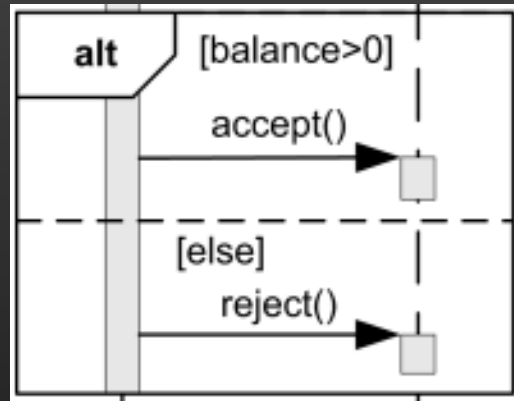
```
free a; // C
```

From <http://www.uml-diagrams.org/sequence-diagrams.html>

Ativação



Fragmentos para mostrar alternativas

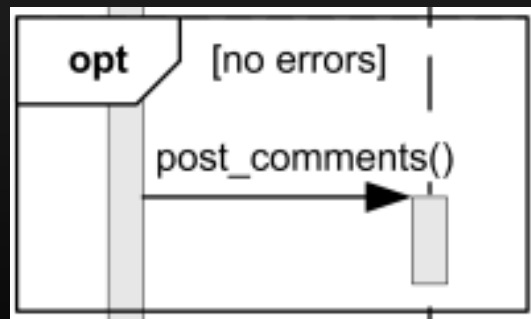


If (balance>0)

otherObj.Accept()

Else

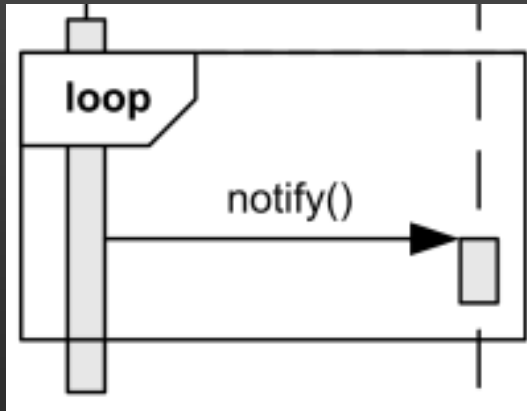
otherObj.Reject()



If (no errors)

otherObj.Post_comments()

Fragmentos para mostrar loops



```
i=0;
```

```
While( i<10 )
```

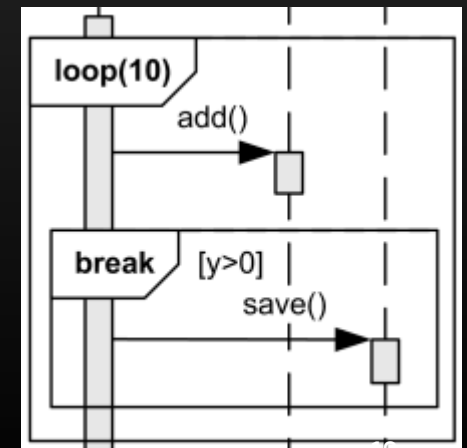
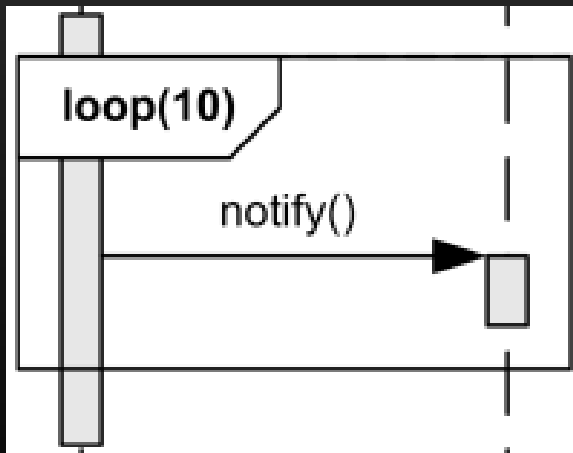
```
{
```

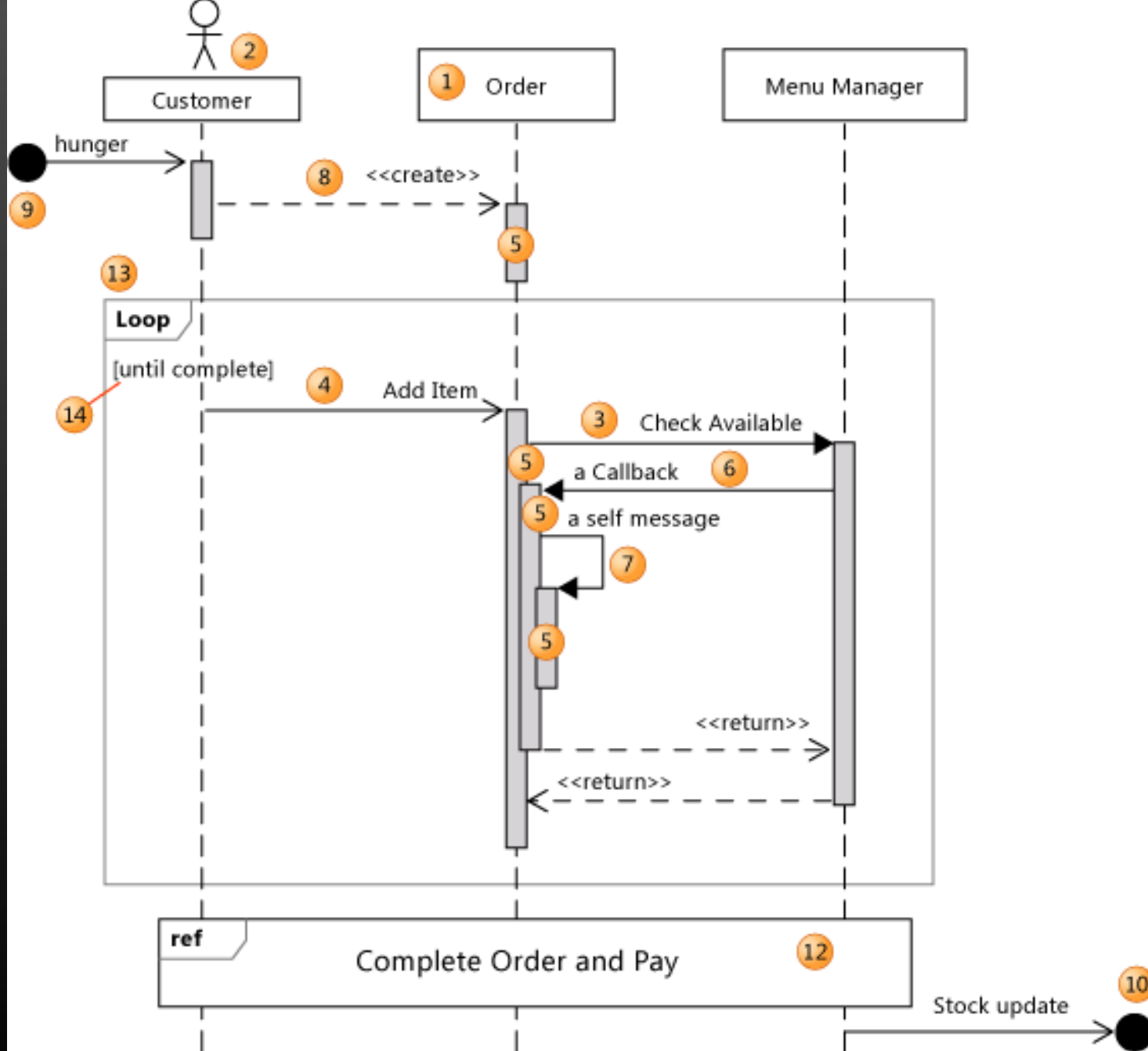
```
    otherObj.Add()
```

```
    if ( y>0 ) break;
```

```
    i++
```

```
}
```





Building Sequence Diagrams

1. Set the context
2. Identify actors and objects that interact in the use-case scenario
3. Set the lifeline for each object
4. **Add messages by drawing arrows**
 - Shows how they are passed from one object to another
 - Include any parameters in parentheses
 - Obvious return values are excluded
5. **Add execution occurrence to each object's lifeline**
6. **Validate the sequence diagram**
7. Ensures that it depicts all of the steps in the process

Quatro tipos de diagramas de iteração disponíveis

D. Sequência

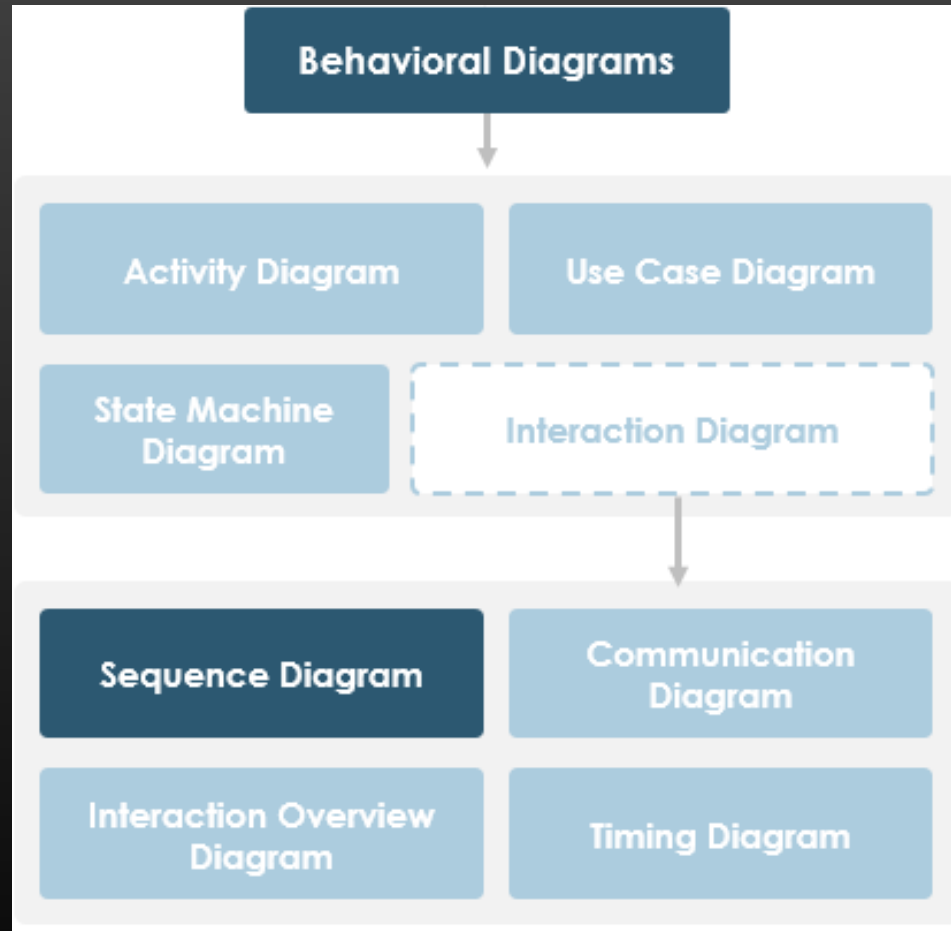
→ Formato alinhado

D. Comunicação

Formato grafo

Diagrama temporal (*timming*)

Diagrama de visão geral da interação (*interaction overview*)



Quatro tipos de diagramas de iteração disponíveis

D. Sequência

Formato alinhado

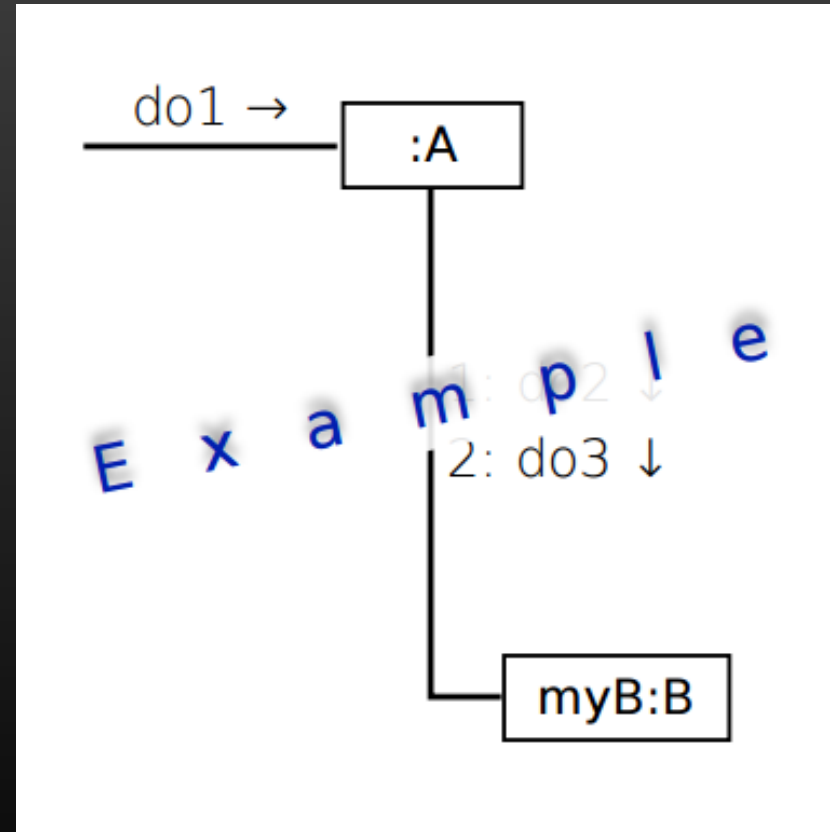


D. Comunicação

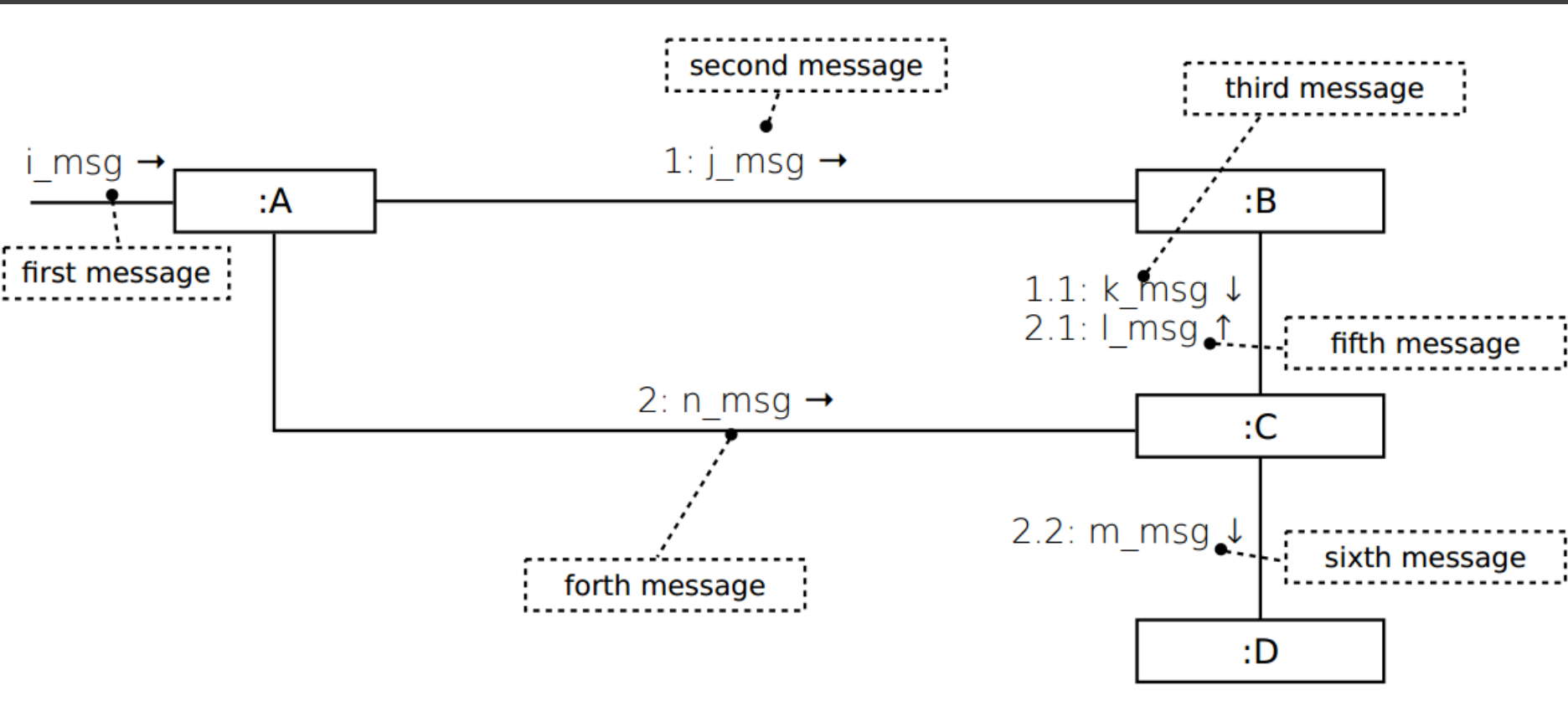
Formato grafo

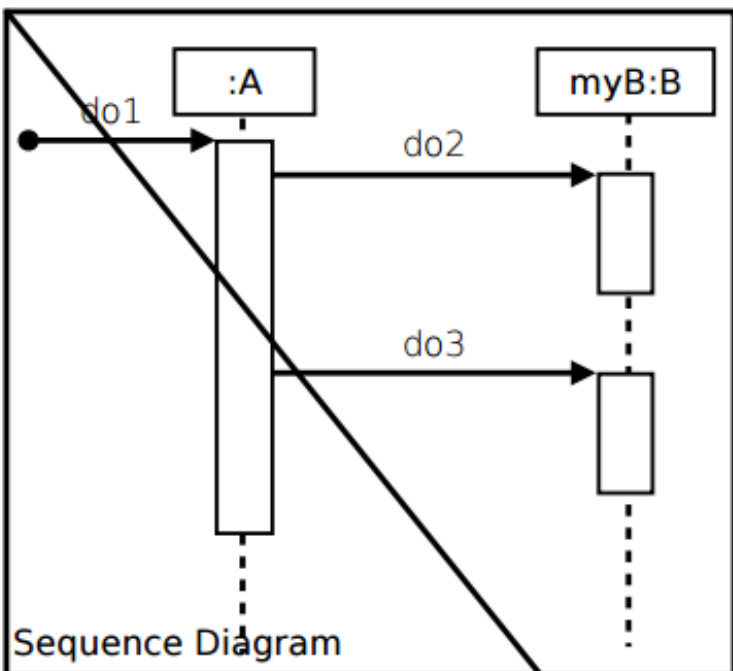
Diagrama temporal (*timming*)

Diagrama de visão geral da interação (*interaction overview*)



Diagramas de comunicação





Os diagramas de podem ser usados para visualizar a colaboração entre objetos em Java.

```
public class A {  
    private B myB = ...;  
  
    public void do1() {  
        myB.do2();  
        myB.do3();  
    }  
}
```

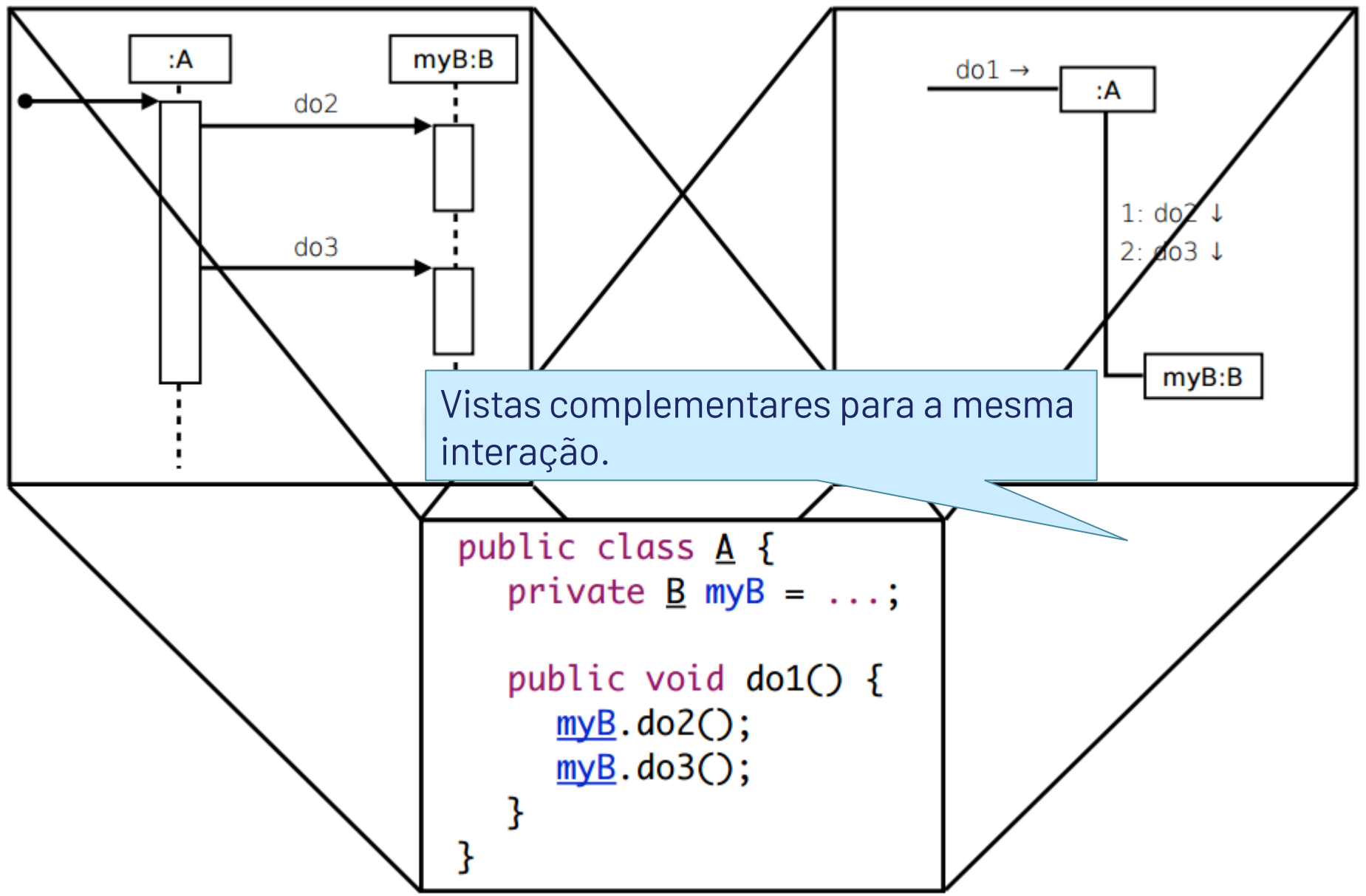



Figure Sequence Diagram for a Scenario of the Make Patient Appt Use Case

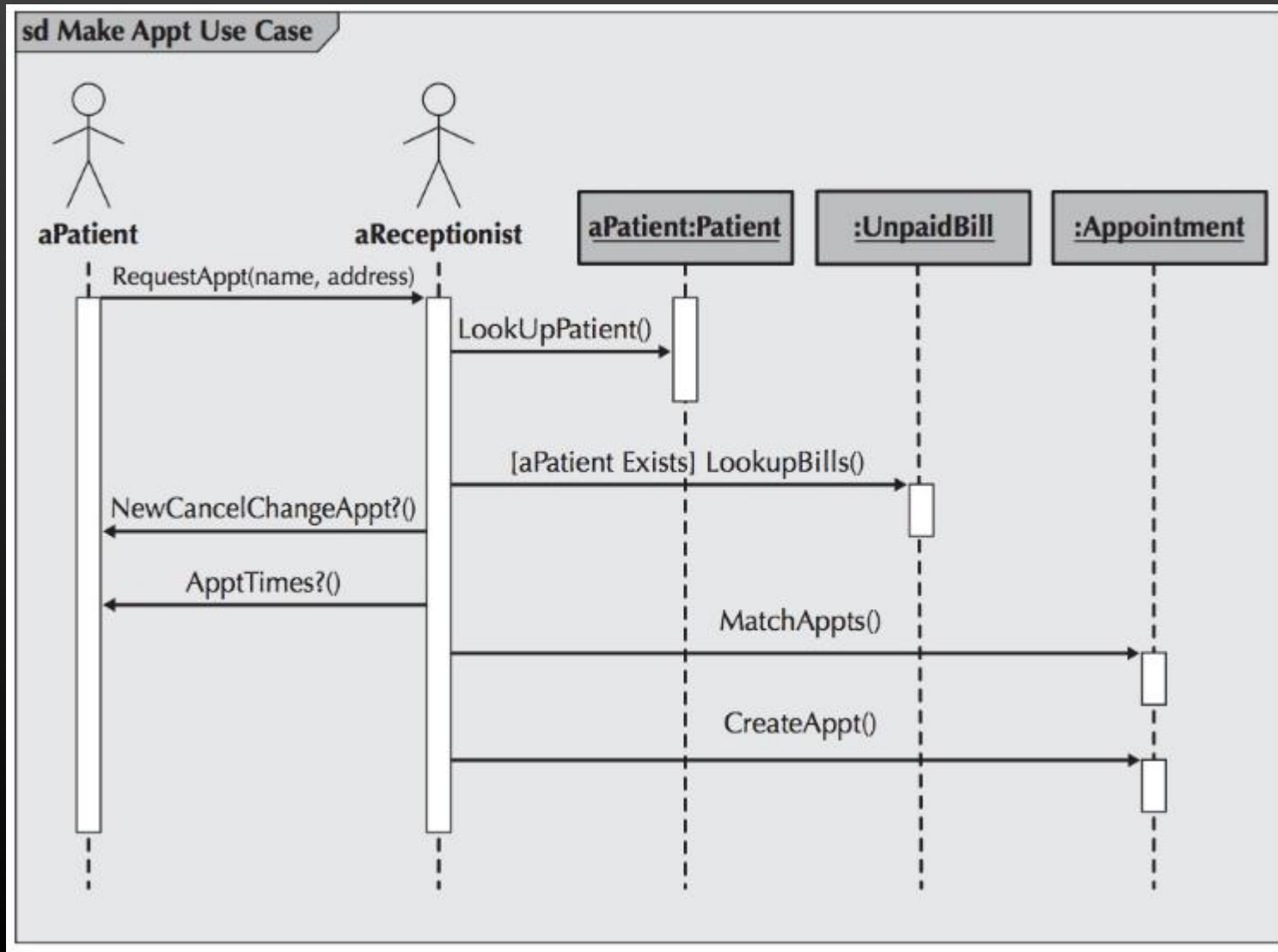
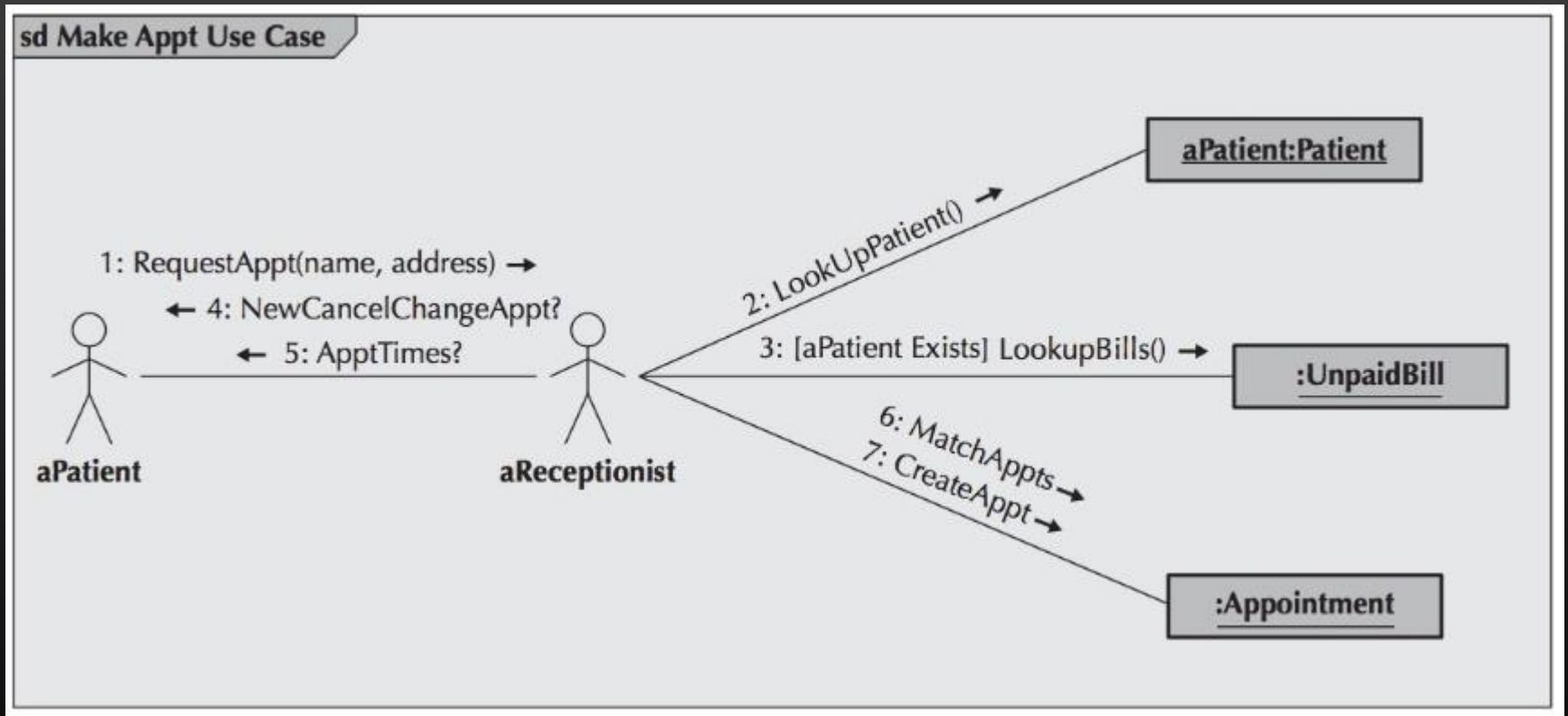


Figure Communication Diagram for a Scenario of the Make Patient Appt Use Case



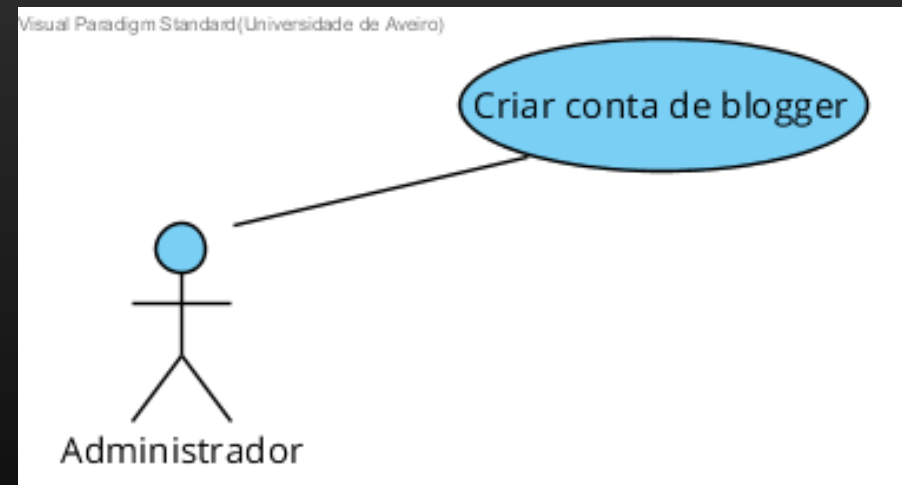
Diagramas de sequência *vs* diagramas de comunicação

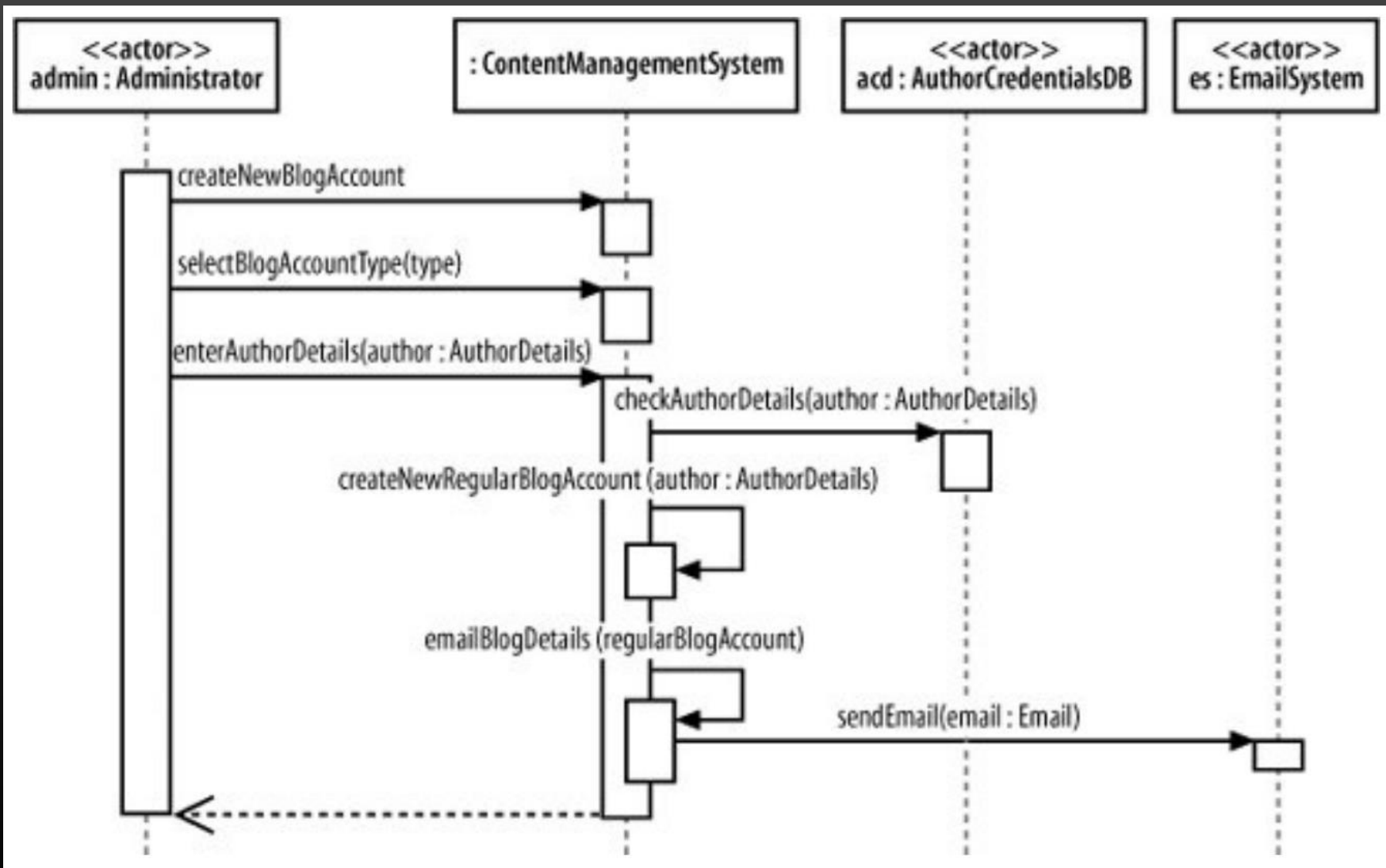
	Benefícios	Limitações
Diagrama de sequência	<ul style="list-style-type: none">• Mostra claramente a sequência/ordem temporal das mensagens• Possibilidades de notação alargadas	<ul style="list-style-type: none">• Cresce para a direita à medida que se acrescentam objetos
Diagrama de comunicação	<ul style="list-style-type: none">• Mais fácil de desenhar (objetos podem ser adicionado em qq parte)	<ul style="list-style-type: none">• Menos expressivo (ordem temporal)• Menos opções de notação• Pouco suportado nas ferramentas UML

Um caso de utilização é realizado pela colaboração entre atores e sistema

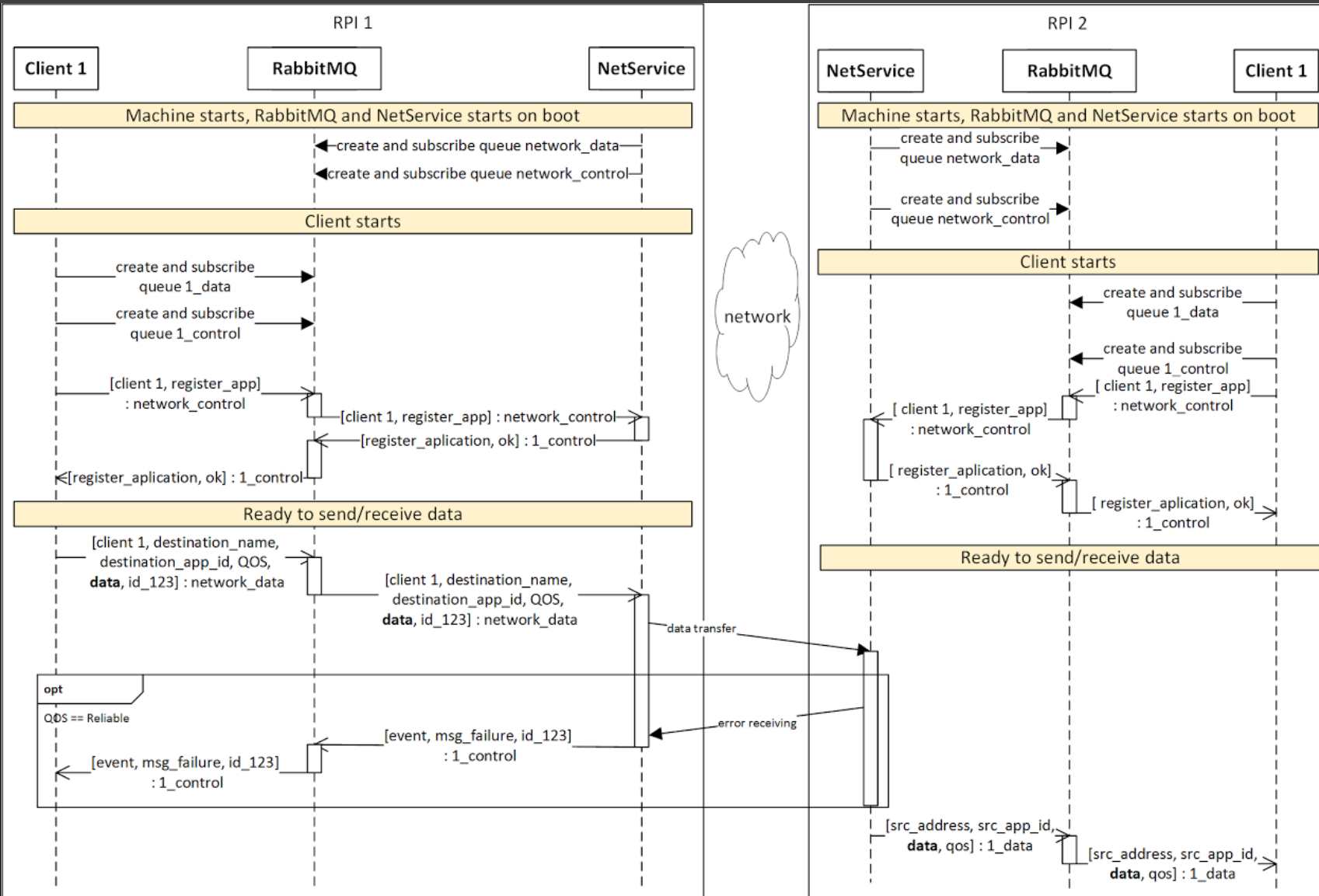
FLUXO TÍPICO

1. O Administrador pede ao sistema (CMS) para criar uma nova conta de *blogger*
2. O Administrador escolhe o tipo Normal
3. O Administrador fornece os detalhes do autor do blog
4. O CMS verifica os detalhes (se já existe) na base de Dados de Autores
5. O CMS cria a nova conta normal.
6. Um sumário dos detalhes da nova conta são enviados por email ao autor.





Outro exemplo de aplicação: **documentar um protocolo**



Behavioral State Machines

Objects may change state in response to an event

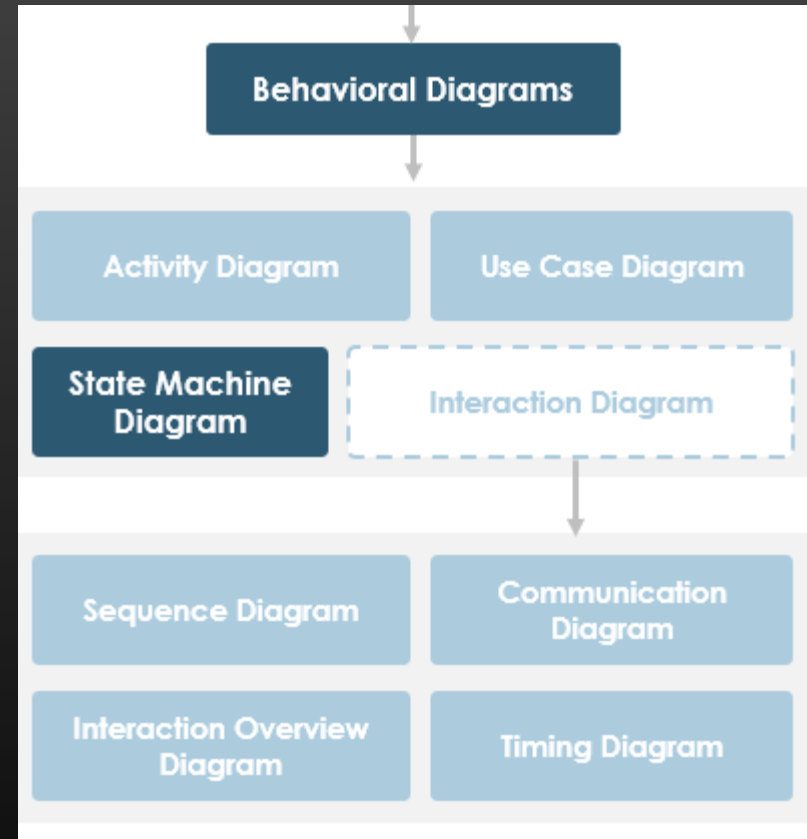
Different states are captured in this model

- Shows the different states through which a single object passes during its life
- May include the object's responses and actions

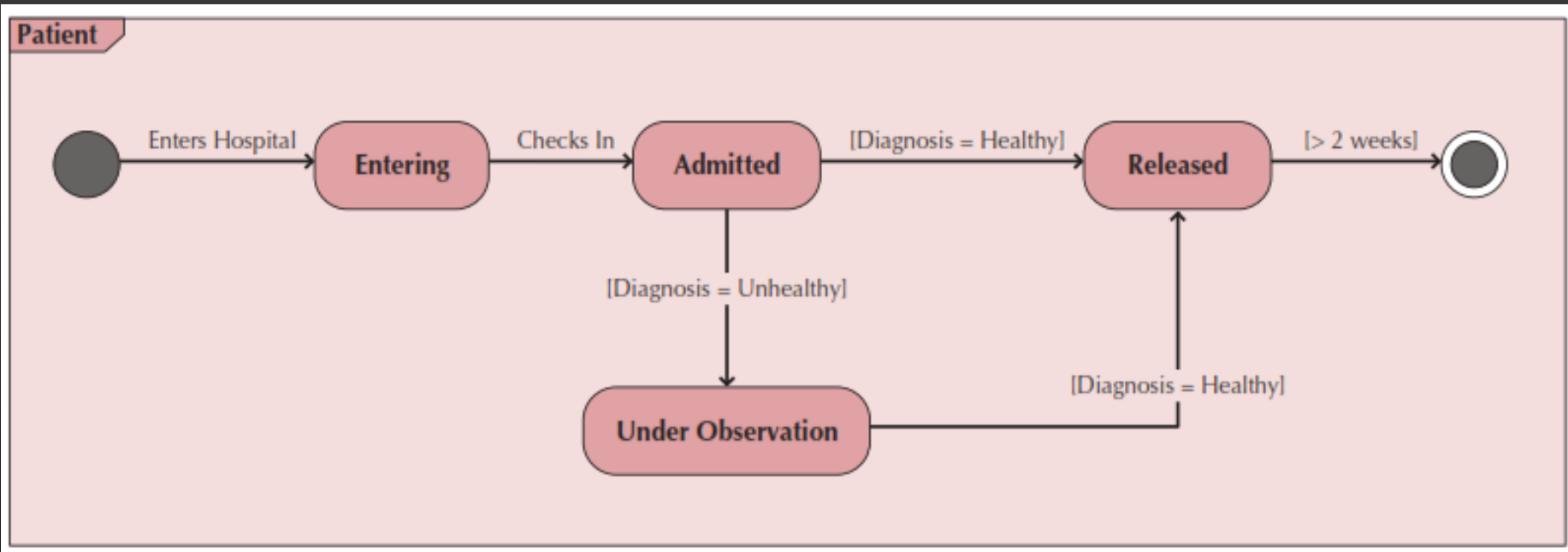
Example: patient states

- New patient—has not yet been seen
- Current patient—is now receiving treatment
- Former patient—no longer being seen or treated






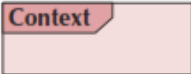
Typically used **only for complex objects**

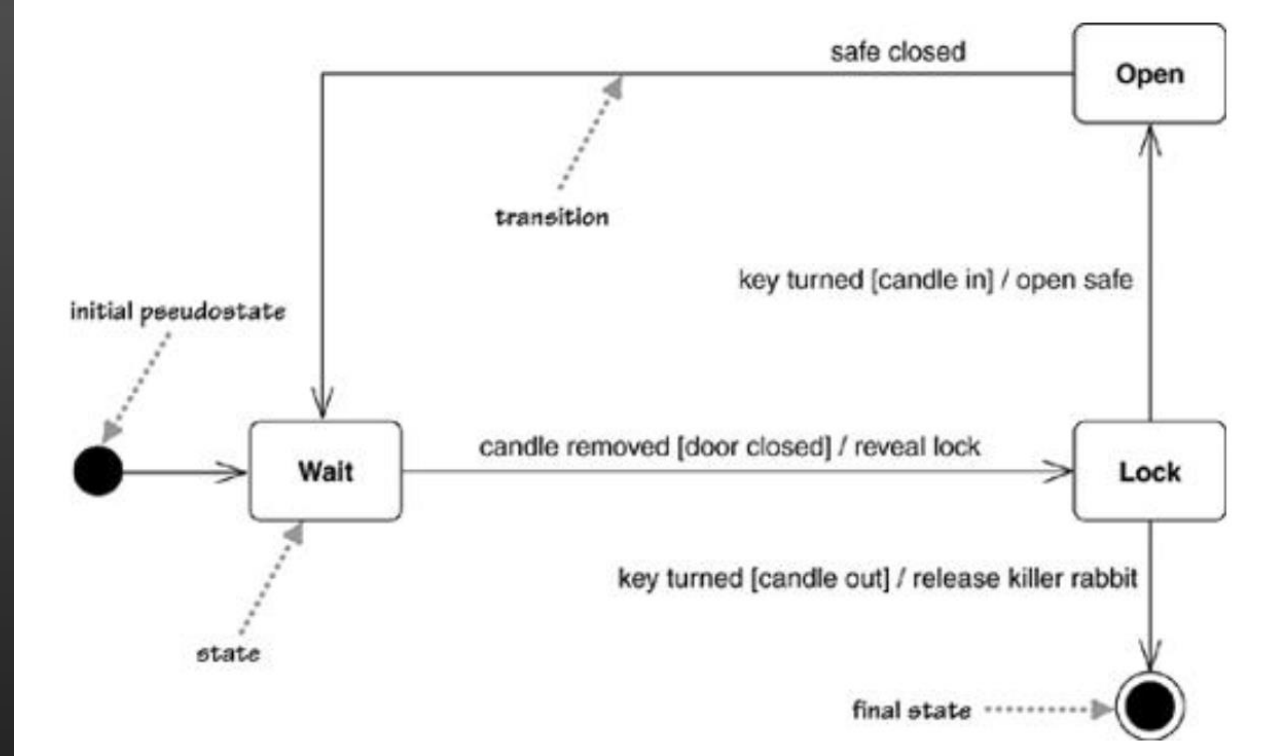


Sample State Machine



State Machine Syntax

Term and Definition	Symbol
A state: <ul style="list-style-type: none">■ Is shown as a rectangle with rounded corners.■ Has a name that represents the state of an object.	
An initial state: <ul style="list-style-type: none">■ Is shown as a small, filled-in circle.■ Represents the point at which an object begins to exist.	
A final state: <ul style="list-style-type: none">■ Is shown as a circle surrounding a small, filled-in circle (bull's-eye).■ Represents the completion of activity.	
An event: <ul style="list-style-type: none">■ Is a noteworthy occurrence that triggers a change in state.■ Can be a designated condition becoming true, the receipt of an explicit signal from one object to another, or the passage of a designated period of time.■ Is used to label a transition.	
A transition: <ul style="list-style-type: none">■ Indicates that an object in the first state will enter the second state.■ Is triggered by the occurrence of the event labeling the transition.■ Is shown as a solid arrow from one state to another, labeled by the event name.	
A frame: <ul style="list-style-type: none">■ Indicates the context of the behavioral state machine.	



The transition indicates a movement from one state to another.

Each transition has a label that comes in three parts: trigger-signature [guard]/activity. All the parts are optional.

The trigger-signature is usually a single event that triggers a potential change of state. The guard, if present, is a Boolean condition that must be true for the transition to be taken. The activity is some behavior that's executed during the transition.

Notação básica dos D. Estado

Estados → caixas

Condição em que se encontra o objeto

Transições → setas

Evolução de um estado para outro

Triggers → etiquetas

Acontecimentos relevantes que causam transições de estado

Condições de acesso

Marcador início/fim

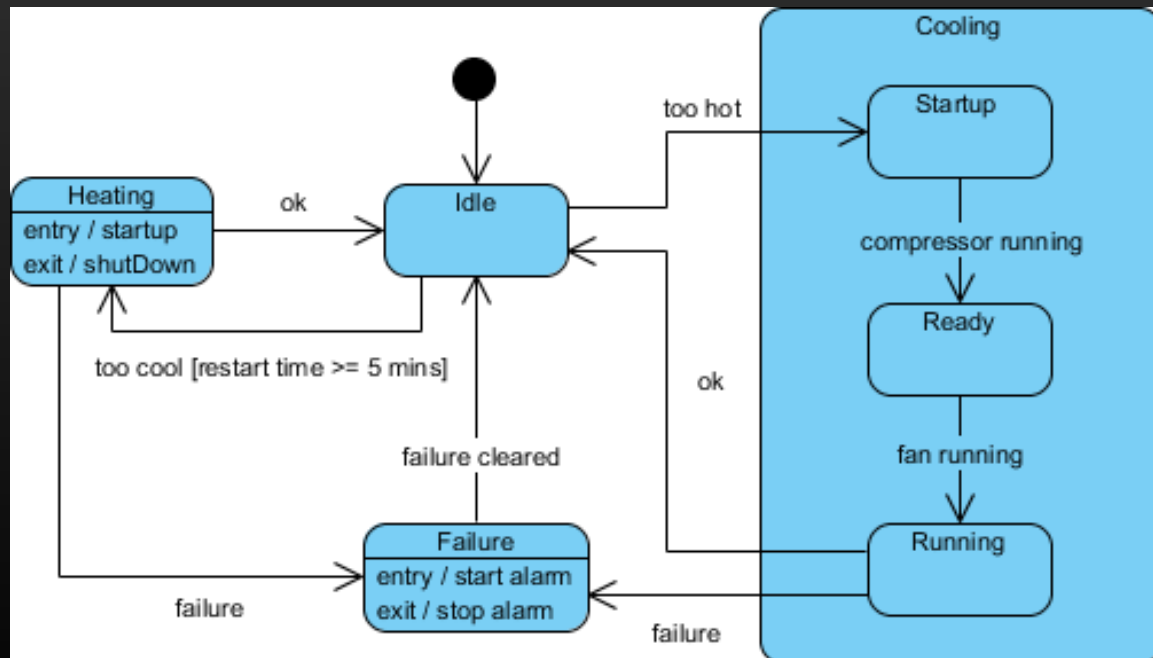
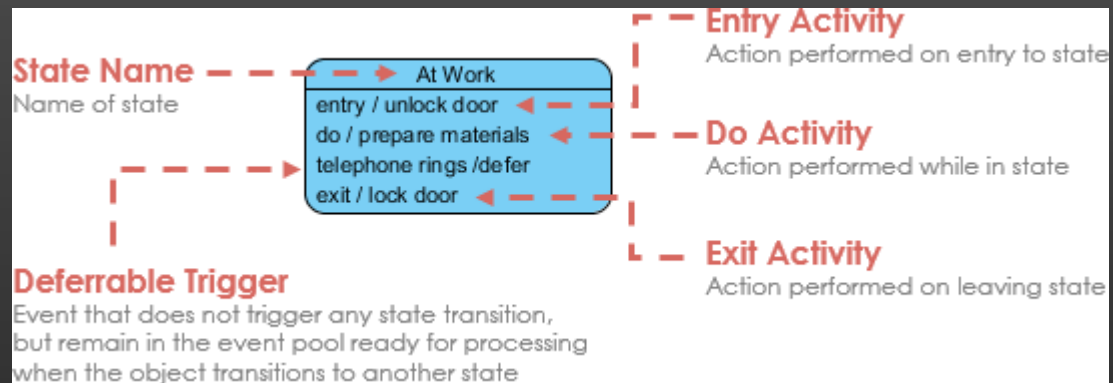
"An event is a significant or noteworthy occurrence e.g. *a telephone receiver is taken off the hook.*

A state is the condition of an object at a moment in time e.g. *a telephone is in the state of being "idle" after the receiver is place on the hook and until it is taken off the hook.*

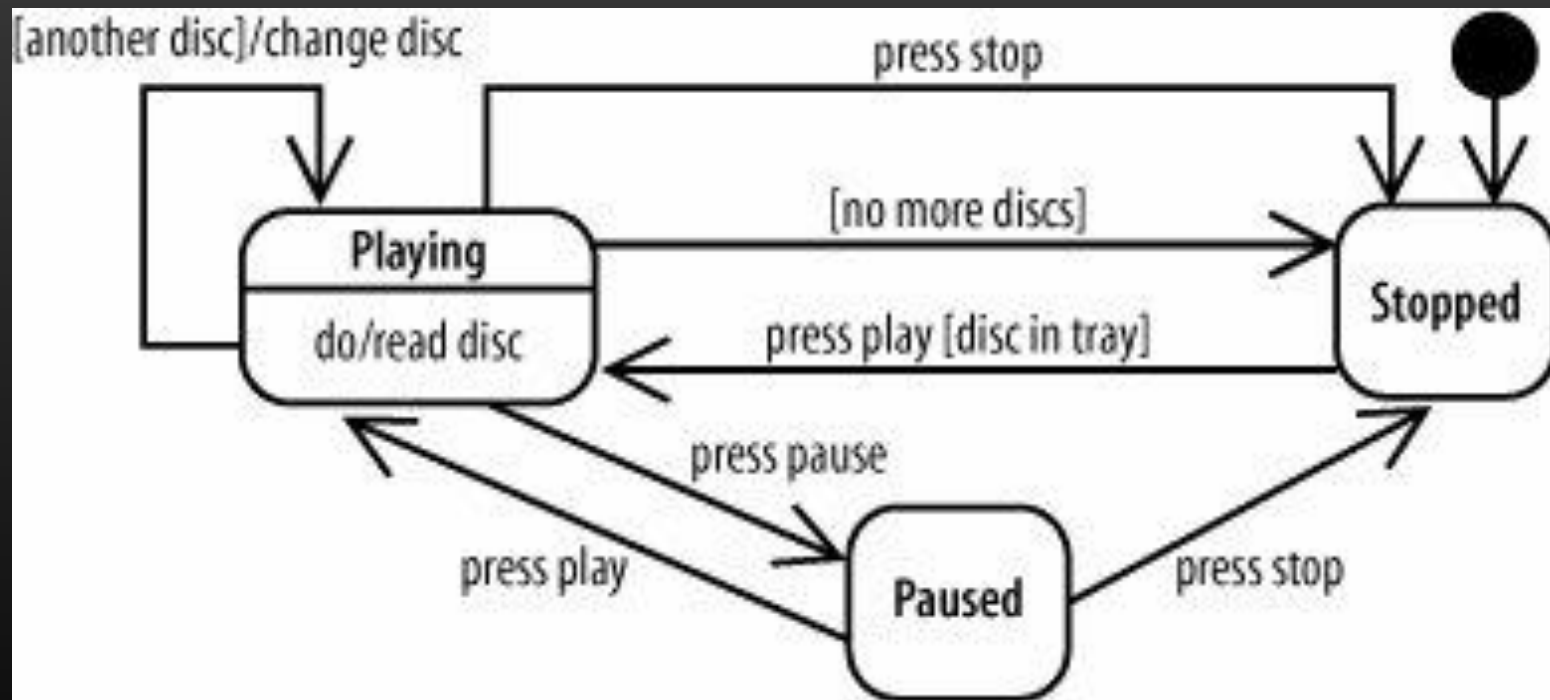
A transition is a relationship between two states that indicates when an event occurs e.g. *when the event "off hook" occurs, transition the telephone from "idle" to "active" state.*

Internal Activities

States can react to events without transition, using internal activities: putting the event, guard, and activity inside the state box itself



Exemplo



Quando usar?

Objetos com comportamento dependente do estado

Exemplos:

Controlador de um dispositivo físico (hw)

Lógica de objetos de negócio (Venda, Reserva,...)

Protocolos de comunicação

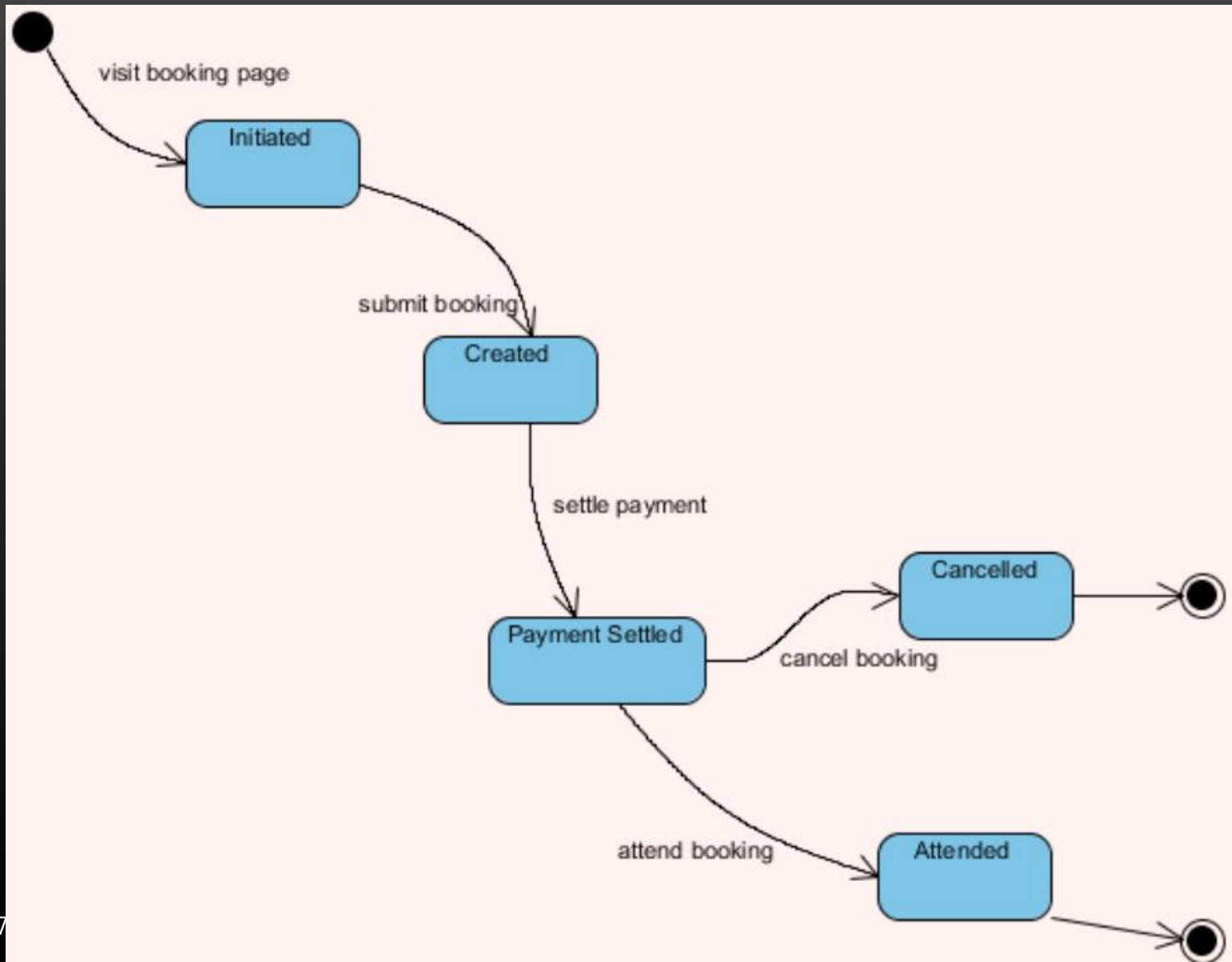
Coordenadores do fluxo da interface gráfica

No modelo do domínio:

Caraterizar os estados de uma classe complexa

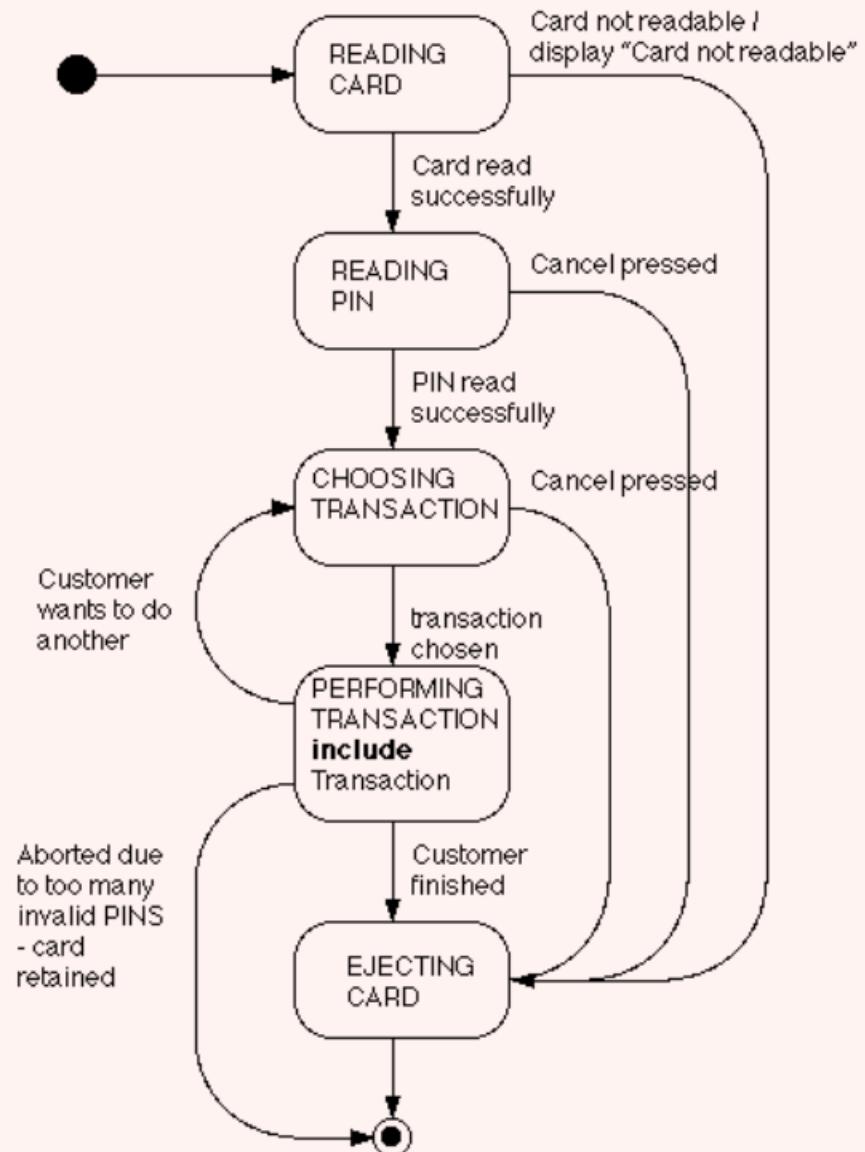
e.g.: reserva, inscrição,...

Modeling the states of an appointment



ATM controller

State-Chart for One Session



Readings & references

Core readings	Suggested readings
<ul style="list-style-type: none">• [Dennis15] – Chap. 6• What is Sequence Diagram?, VisualParadigm docs• What is Communication Diagram?, VisualParadigm docs• What is State Machine Diagram?, VisualParadigm docs	<ul style="list-style-type: none">• [Larman04] – Chap. 10, Chap. 15.