Projektowanie złożonych systemów telekomunikacyjnych

UML/ Aleksander Miera

Agenda

1. Introduction to modeling
2. UML
3. Diagrams
    1. Structural
    2. Behavioral
4. Tools and further reading

Introduction

Introduction

What is a model?

Introduction

What is a model?

- Simplification of reality

Introduction

What is a model?

- Simplification of reality
- Which allows to better comprehend a complex system

Introduction

What is a model?

- Simplification of reality
- Which allows to better comprehend a complex system
- Which could not be easily understood and reasoned about as a whole

Introduction

Why use models?

- To visualise the system as it is or the way it is intended to be

Introduction

Why use models?

- To visualise the system as it is or the way it is intended to be
- To specify the structure or behaviour of a system

Introduction

Why use models?

- To visualise the system as it is or the way it is intended to be
- To specify the structure or behaviour of a system
- To document design decisions

Introduction

Why use models?

- To visualise the system as it is or the way it is intended to be
- To specify the structure or behaviour of a system
- To document design decisions
- As a guideline when implementing the system

Introduction

The principles of modeling (1)

Introduction

The principles of modeling (1)

1. The choice of what models to create has a profound influence on how a problem is attacked and how a solution is shaped.

Introduction

The principles of modeling (1)

1. The choice of what models to create has a profound influence on how a problem is attacked and how a solution is shaped.
2. Every model may be expressed at different levels of precision

Introduction

The principles of modeling (1)

1. The choice of what models to create has a profound influence on how a problem is attacked and how a solution is shaped.
2. Every model may be expressed at different levels of precision
3. The best models are connected to reality.

Introduction

The principles of modeling (1)

1. The choice of what models to create has a profound influence on how a problem is attacked and how a solution is shaped.
2. Every model may be expressed at different levels of precision
3. The best models are connected to reality.
4. No single model or view is sufficient. Every nontrivial system is best approached through a small set of nearly independent models with multiple viewpoints.

UML – What it is and what is not

UML

What is it and what is not

- Pictorial language for visualizing

UML

What is it and what is not

- Pictorial language for visualizing
- Focused on representation of the system

UML

What is it and what is not

- Pictorial language for visualizing
- Focused on representation of the system
- Not a programming language, but can be generated programatically

UML

What is it and what is not

- Pictorial language for visualizing
- Focused on representation of the system
- Not a programming language, but can be generated programatically
- ...or can be transpiled into executable/compilable code

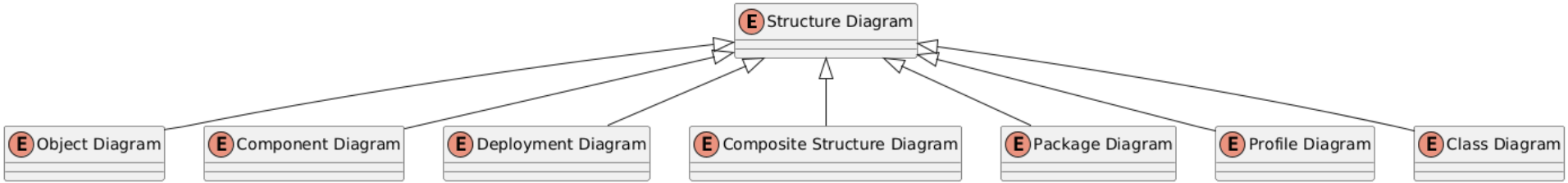
UML

What is it and what is not

- Pictorial language for visualizing
- Focused on representation of the system
- Not a programming language, but can be generated programatically
- ...or can be transpiled into executable/compilable code
- Technology-agnostic (mostly)

Diagrams

Structure (static) diagrams



Diagrams

Structural

- Classes/Interfaces

Diagrams

Structural

- Classes/Interfaces
- Attributes

Diagrams

Structural

- Classes/Interfaces
- Attributes
- Operations/methods

Diagrams

Structural

- Classes/Interfaces
- Attributes
- Operations/methods
- Class hierarchy/relationships

Diagrams

Structural

Class

- + for public
- # for protected
- - for private
- ~ for pakcage

```
class "Application" {
  #id
  -Running
  ~isRunning()
  +stop()
}
```

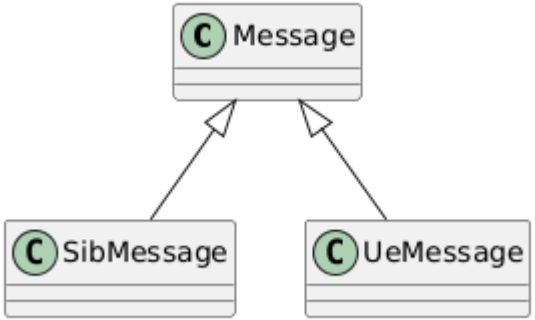
Diagrams

Structural

Derivation/generalization

- Class hierarchy
- Describing abstractions
- Is-a relationship
- Note: upcasting/downcasting

```
class "Message"
class "SibMessage"
class "UeMessage"
"Message" <|-- "SibMessage"
"Message" <|-- "UeMessage"
```



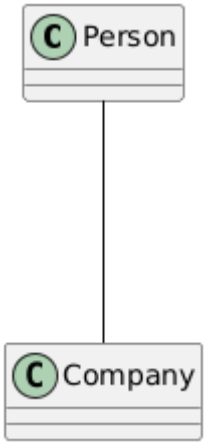
Diagrams

Structural

Association

- Some relationship between classes
- Classes remain independent

```
class "Person"
class "Company"
"Person" --- "Company"
```



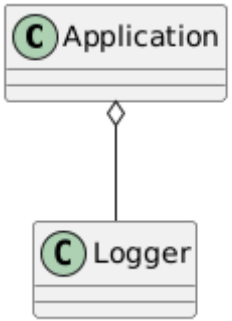
Diagrams

Structural

Aggregation

- Variant of "has-a" relationship
- Entities can exists independently

```
class "Application"
class "Logger"
"Application" o-- "Logger"
```



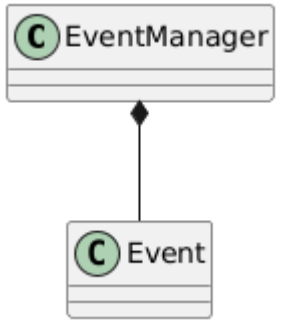
Diagrams

Structural

```
class "EventManager"
class "Event"
"EventManager" *-- "Event"
```

Composition

- Strong aggregation
- Models ownership: if the owner is removed, so are the aggregated entities



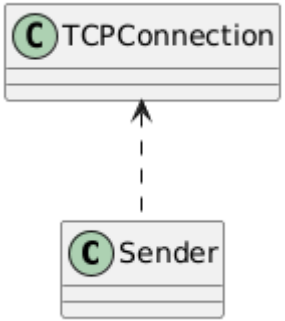
Diagrams

Structural

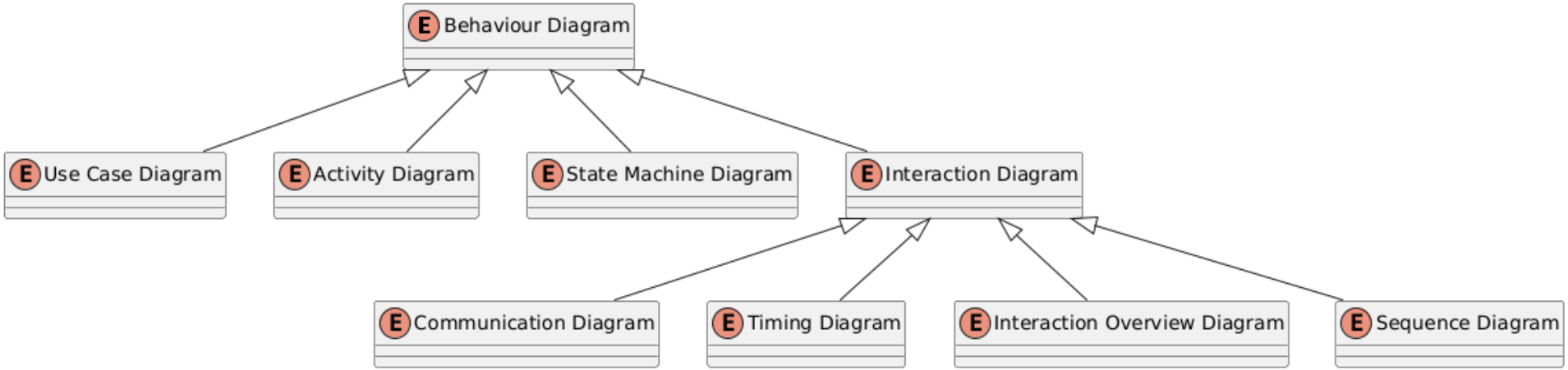
Dependency

- Component X is dependent on component Y is some way, e.g. MessageSender uses TCPConnection

```
class "Sender"
class "TCPConnection"
"TCPConnection" <.. "Sender"
```



Behaviour (dynamic) diagrams

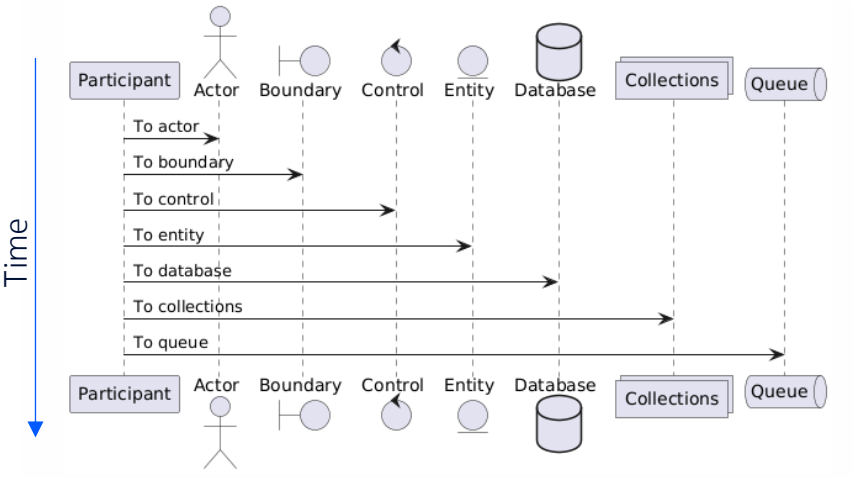


Diagrams

Behavioral

Sequence

- Interaction between actors over time in a sequential manner



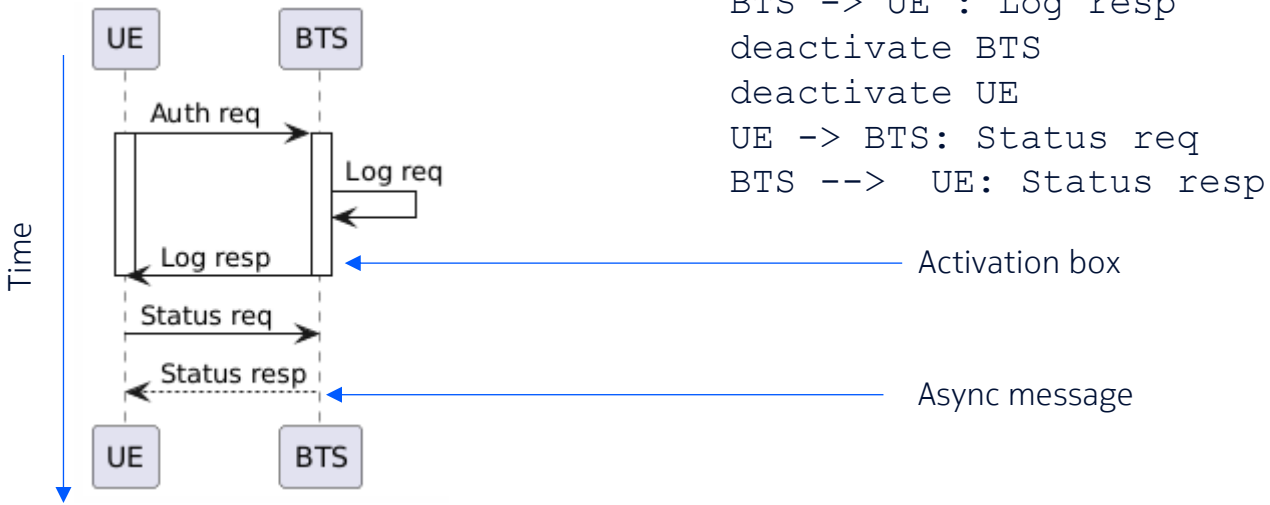

```
participant Participant as Foo
actor       Actor       as Foo1
boundary    Boundary    as Foo2
control     Control     as Foo3
entity      Entity      as Foo4
database    Database    as Foo5
collections Collections as Foo6
queue       Queue       as Foo7
Foo -> Foo1 : To actor
Foo -> Foo2 : To boundary
Foo -> Foo3 : To control
Foo -> Foo4 : To entity
Foo -> Foo5 : To database
Foo -> Foo6 : To collections
Foo -> Foo7: To queue
```

Diagrams

Behavioral

Sequence, continued
Notice activation box on BTS's end

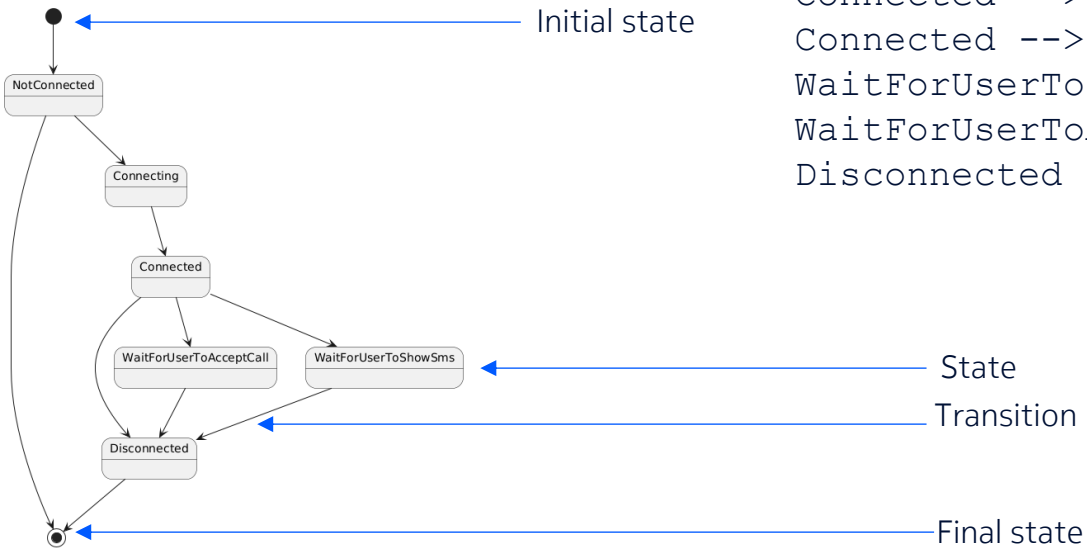
```
participant UE
participant BTS
UE -> BTS : Auth req
activate BTS
activate UE
BTS -> BTS : Log req
BTS -> UE : Log resp
deactivate BTS
deactivate UE
UE -> BTS: Status req
BTS -->  UE: Status resp
```



Diagrams

Behavioral

State machine: FSM transition diagram

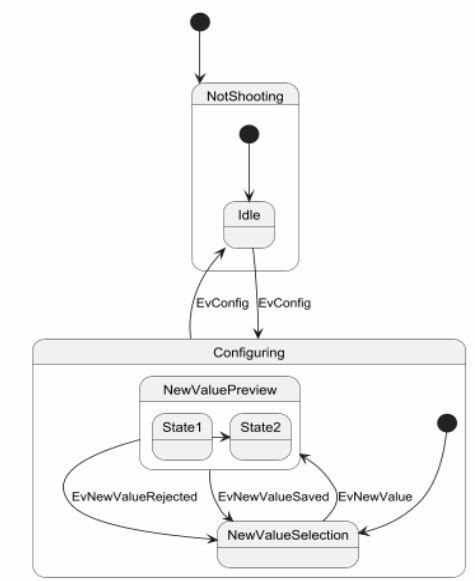



```
[*] --> NotConnected
NotConnected --> [*]
NotConnected --> Connecting
Connecting --> Connected
Connected --> Disconnected
Connected --> WaitForUserToAcceptCall
Connected --> WaitForUserToShowSms
WaitForUserToShowSms --> Disconnected
WaitForUserToAcceptCall --> Disconnected
Disconnected --> [*]
```

Diagrams

Behavioral

Hierarchical state machines/substates




```
[*] --> NotShooting
state NotShooting {
  [*] --> Idle
  Idle --> Configuring : EvConfig
  Configuring --> Idle : EvConfig
}
state Configuring {
  [*] --> NewValueSelection
  NewValueSelection --> NewValuePreview : EvNewValue
  NewValuePreview --> NewValueSelection : EvNewValueRejected
  NewValuePreview --> NewValueSelection : EvNewValueSaved
  state NewValuePreview {
     State1 -> State2
  }
}
```

Diagrams

Behavioral

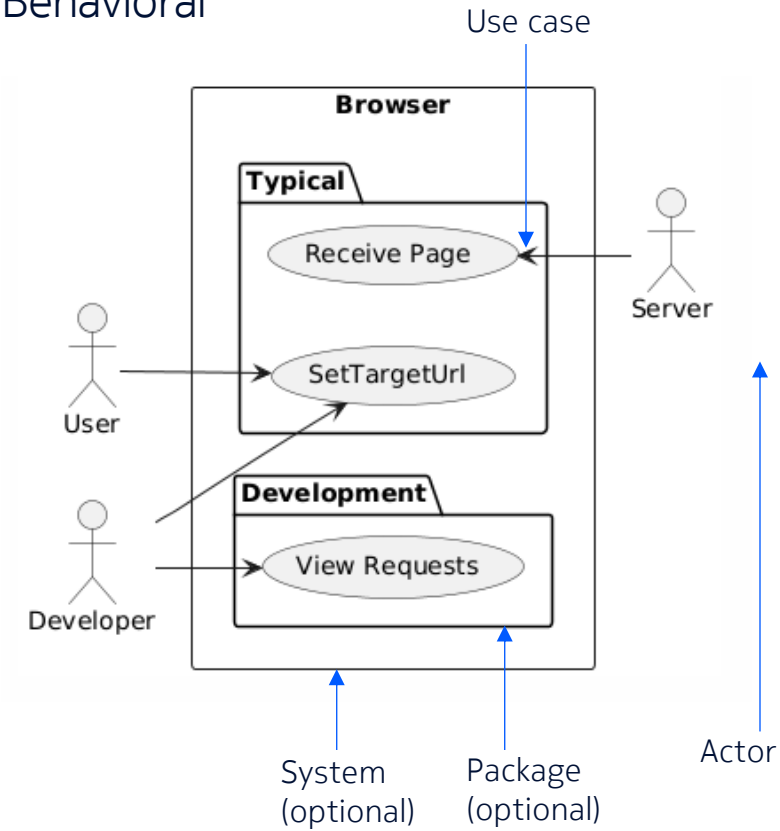
Use case

- High level overview of interactions in the system
- Allows to identify roles
- ...and the way the interact with the system

```
[*] --> NotConnected
NotConnected --> [*]
NotConnected --> Connecting
Connecting --> Connected
Connected --> Disconnected
Connected --> WaitForUserToAcceptCall
Connected --> WaitForUserToShowSms
WaitForUserToShowSms --> Disconnected
WaitForUserToAcceptCall --> Disconnected
Disconnected --> [*]
```

Diagrams

Behavioral




```
left to right direction
actor "User" as u
actor "Server" as s
actor "Developer" as d
rectangle Browser {
package Typical {
  usecase "SetTargetUrl" as UC1
  usecase "Receive Page" as UC2
}
Package Development {
  usecase "View Requests" as UC3
}
}
u --> UC1
UC2 <-- s
d --> UC1
d --> UC3
```

Further reading, tools etc.

If you are interested in expanding knowledge

- (1) *The Unified Modeling Language User Guide, 2nd Edition,* Booch G., Rumbaugh J., Jacobson I., Addison-Wesley Professional, May 2005 https://learning.oreilly.com/library/view/the-unified-modeling/0321267974/
- https://plantuml.com/ UML diagram generation tool

Questions?


NOKIA