



**SOFTWARE
ARCHITECTURE**

2025-26

Jose E. Labra
Pablo González
Diego Martín
Celia Melendi

Lab 2

Documentation
Overview of UML
PlantUML
Introduction to Arc42

Architecture is more than code

The code doesn't tell the whole story

Questions the code doesn't answer

- How the software fits into existing system landscape?

- Why were the technologies chosen?

- What's the overall structure of the system?

- Where are the components deployed at runtime?

- How do the components communicate?

- How and where to add new functionality?

- What common patterns and principles are used?

- How the interfaces with other systems work?

- How security/scalability/... has been achieved?

- ...

Note:
This slide appears also in
the theory classes

Goal of documentation

Main goal: communicate the structure

Understand the *big picture*

Create a **shared vision**: team and stakeholders

Common vocabulary

Describe what the software is and how is being built

Focus for **technical** conversations about new features

Provide a **map** to navigate the source code

Justify design decisions

Help new developers that join the team

Note:
This slide appears also in
the theory classes

Documentation requirements

Understandable by different stakeholders

Technical and non-technical stakeholders

Reflect the reality

Be careful of the model-code gap

Move fast and adapt to changes

Adapt to agile projects

Evolutionary architecture

Note:
This slide appears also in
the theory classes

Rules for good documentation

Write documentation from reader's point of view

- Find who will be the readers and their expectations

Avoid unnecessary repetition (DRY principle)

Avoid ambiguity

- Explain the notation (or use a standard one)

- For diagrams, use legends

Use a standard organization or template

- Add TBD/To do when necessary

- Organize for easy of reference/links

Record rationale

Keep documentation current

Note:
This slide appears also in
the theory classes

Problem vs Solution space

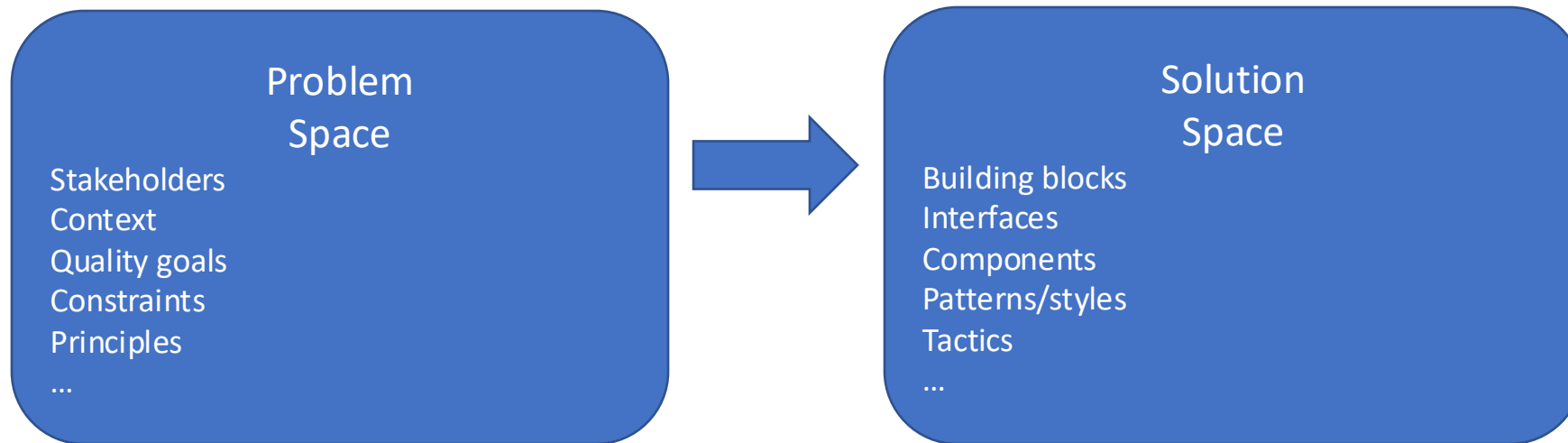
Software architecture = path from problem to solution

Understand the problem

Design a solution

Rationale for the solutions proposed

Record different design alternatives



Note:
This slide appears also in
the theory classes

the theory classes
also in

UML

Unified Modeling Language

Before UML there were several proposals

UML notation unifies them

Proposed by OMG (Object Management Group)

Current version UML 2.5.1 (2017)

Model = abstraction of a problem

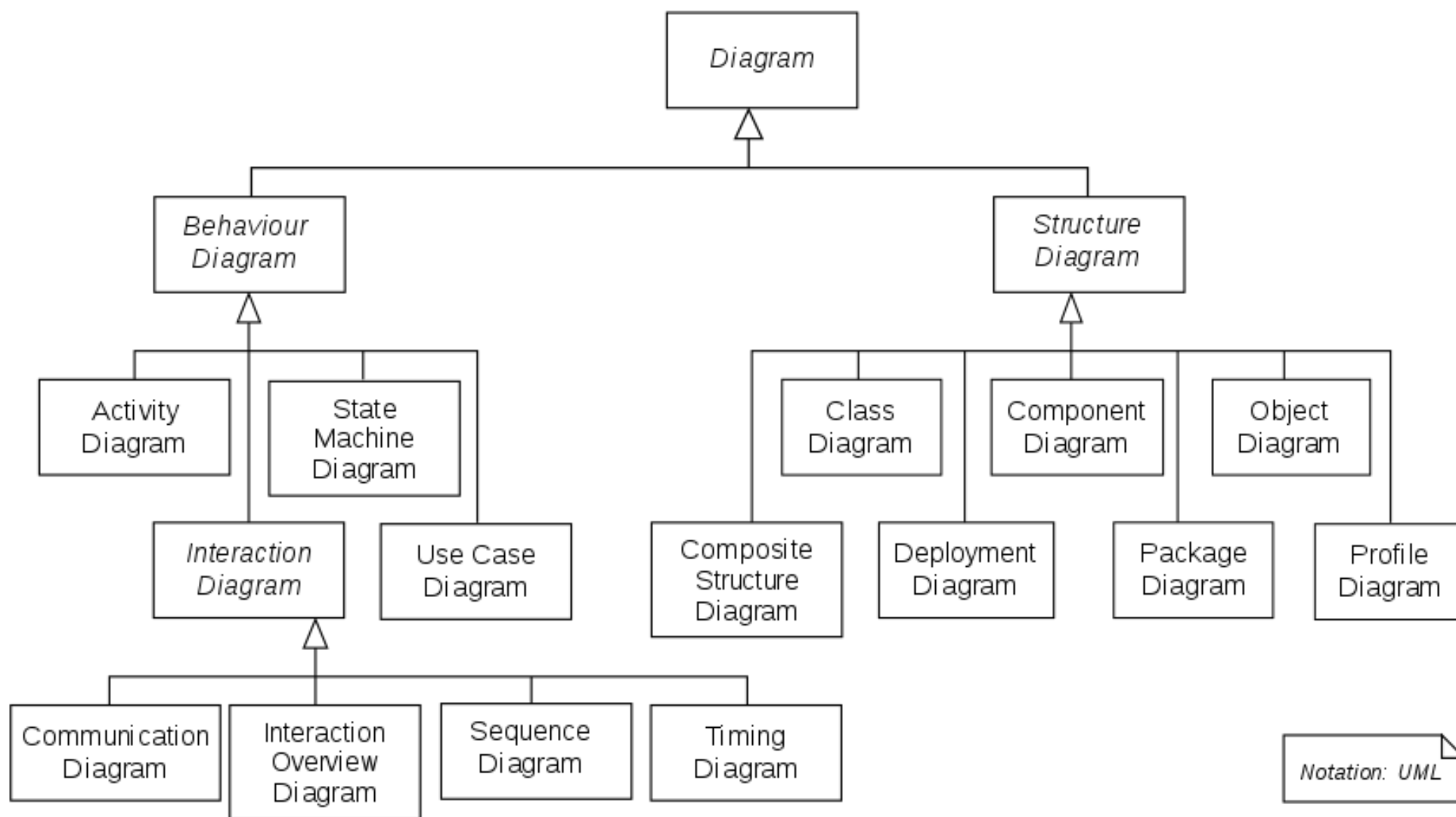
It can have different diagrams

Diagram = partial graphic representation of a model

OCL = Object Constraint Language

Constraints between objects using formal language

14 UML Diagram types



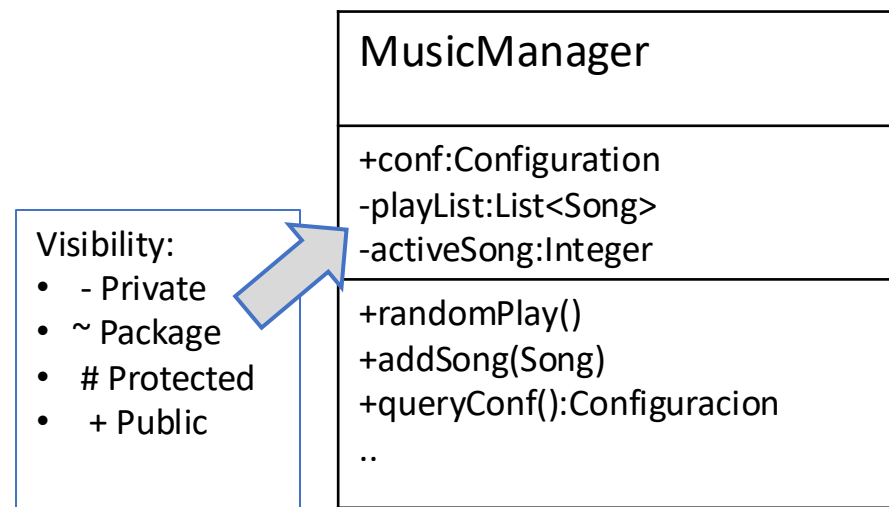
Notation: UML

Class diagrams

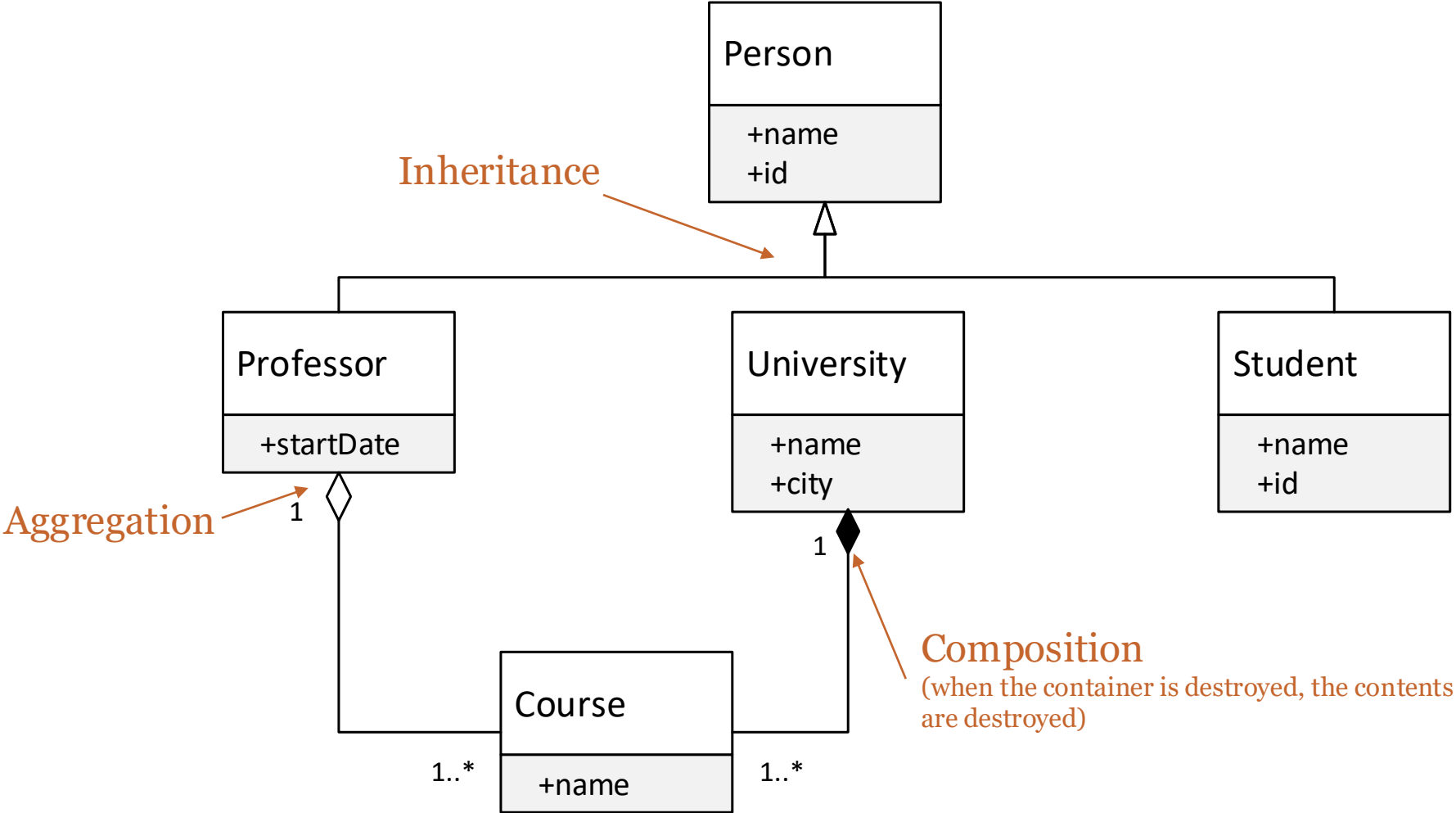
Models the static part of the project, without taking into account the time aspect

Explains the relationships between the different classes.

Arc42: 8-Concepts

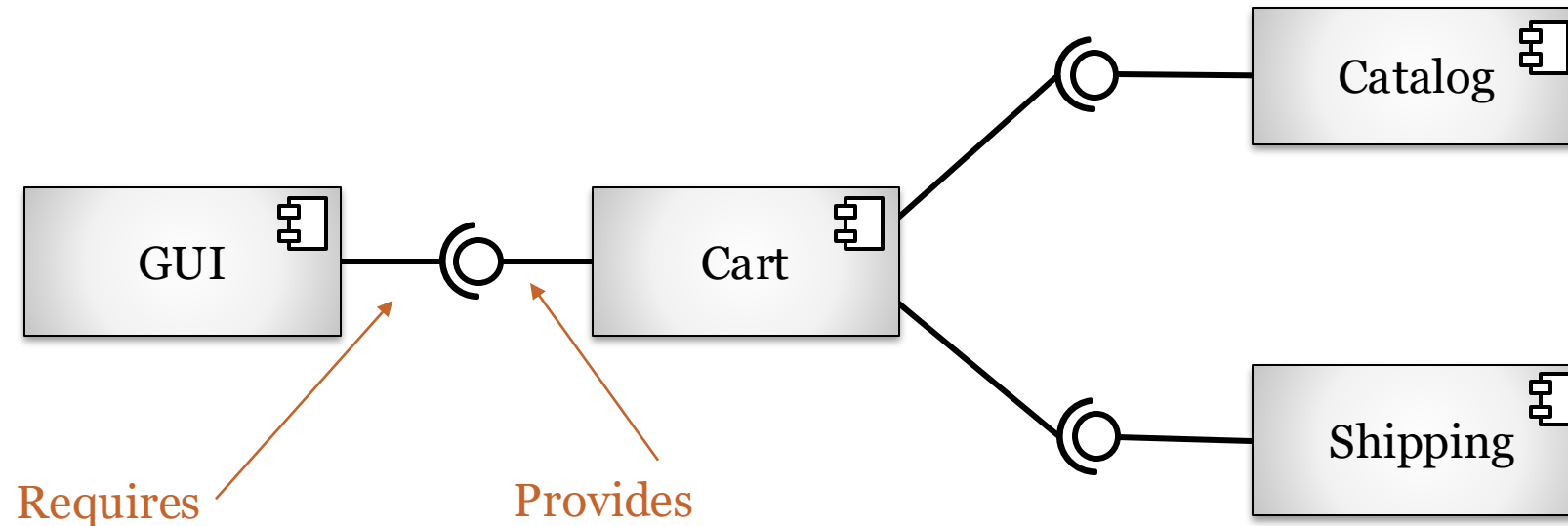


Example



UML Component diagram

Component diagram represents components relationships
Useful for Complex Systems with many components
Interface is usually represented with lollipop notation

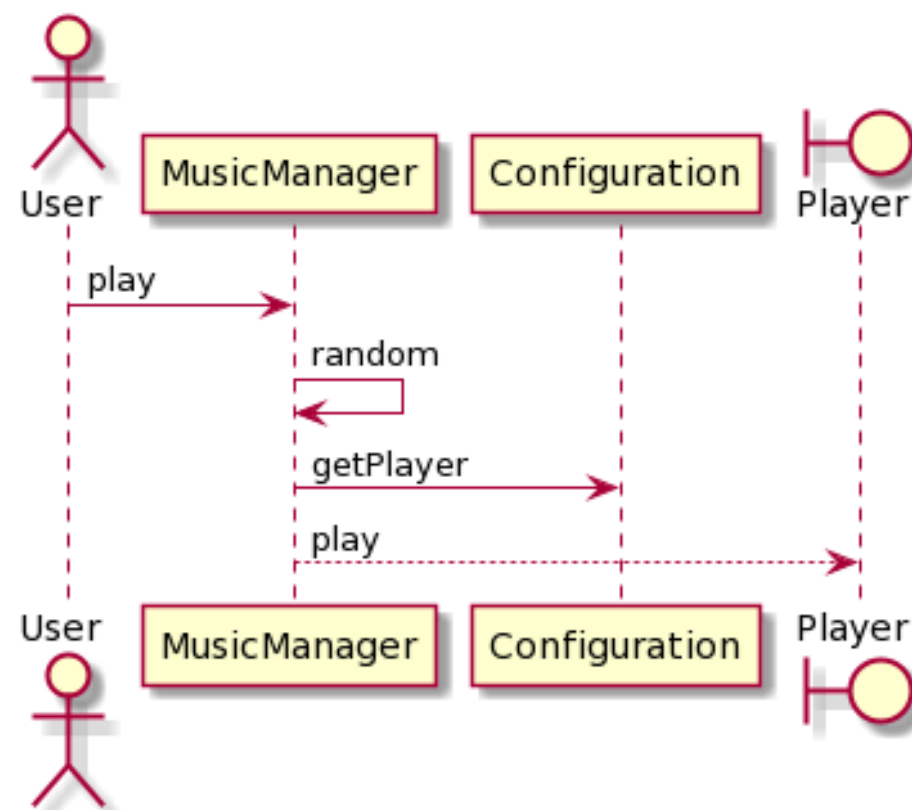


Sequence diagram

Models communication between some objects at a given time

Objects can send two types of messages: synchronous or asynchronous

Arc42:6-RuntimeView



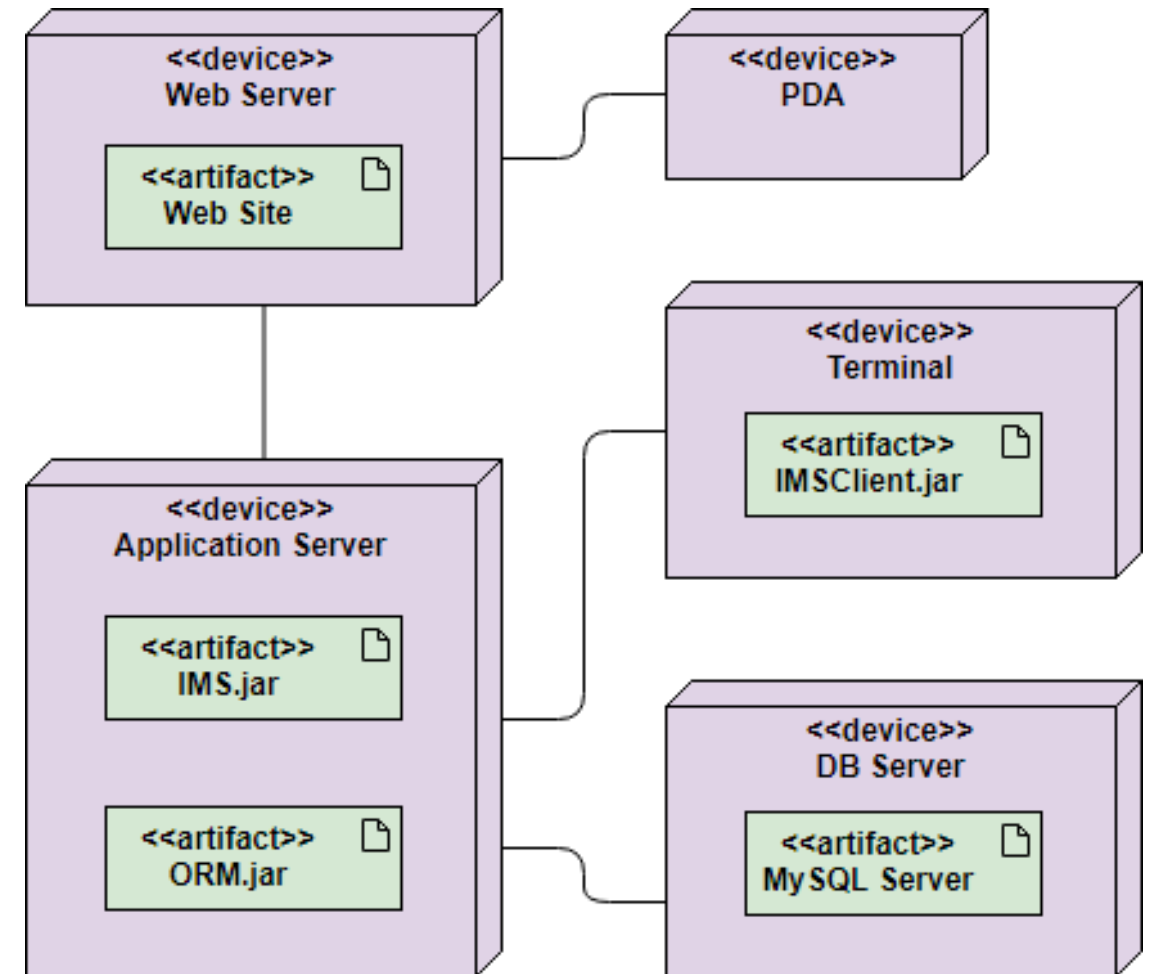
Deployment diagrams

Represents the final location of the components in an app

Elements:

Nodes , Components, relationships

Arc42: 07.DeploymentView



Text-based tools

PlantUML

```
@startuml
component
actor User
participant MusicManager
participant Configuration
boundary Player
User -> MusicManager: play
MusicManager --> MusicManager: random
MusicManager --> Configuration : getPlayer
MusicManager --> Player : play
@enduml
```

Mermaid

```
stateDiagram-v2
    [*] --> Still
    Still --> [*]
    Still --> Moving
    Moving --> Still
    Moving --> Crash
    Crash --> [*]
```

Drawing tools

Powerpoint

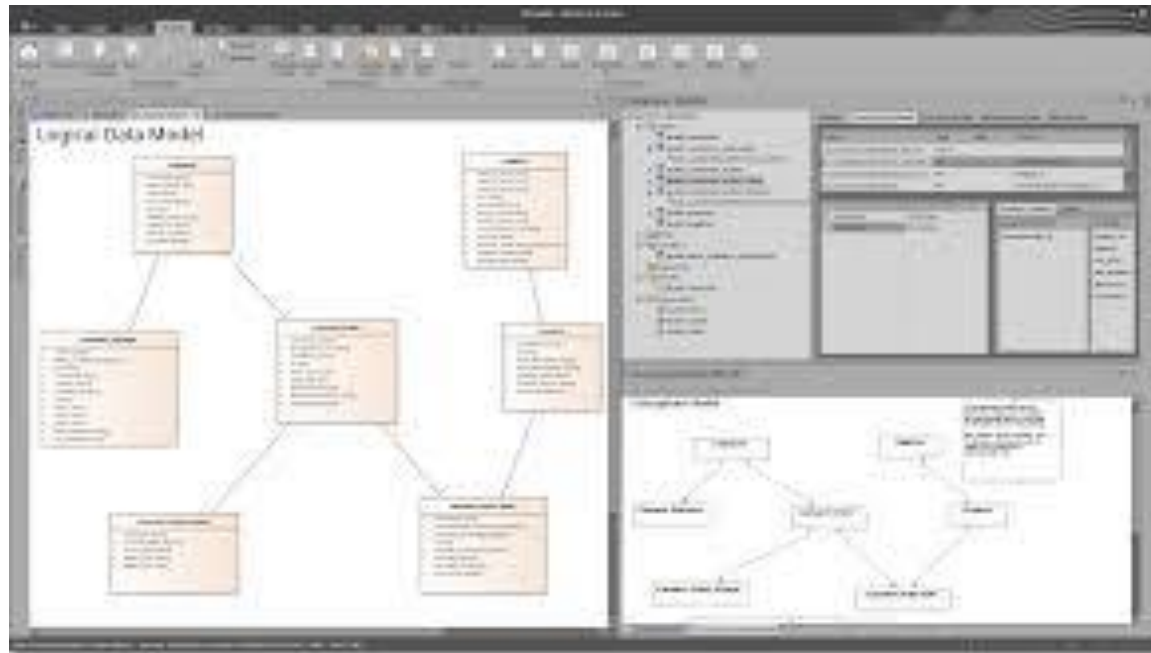
MsVisio

UMLet (<https://www.umlet.com/>)

CASE tools

EnterpriseArchitect

- Reverse Engineering with Java/C++
- Oracle connection for relational databases
- Word, HTML templates



MagicDraw

- Java based
- UML diagrams
- Reverse Engineering Java , C++

Visual Paradigm

- Commercial (student license)

Modelio

- Open source
- Java based
- Reverse Engineering Java , C++

Diagramming the architecture

Video:

<https://www.youtube.com/watch?v=wgpSdpny-0c>

Checklist:

<https://c4model.com/diagrams/checklist>

Arc42 templates

Arc42

<https://arc42.org/>

WIQ already follows the template:

https://arquisoft.github.io/yovi_0/

Generation of docs (locally):

```
$ cd docs  
$ npm install      (only first time)  
$ npm run build
```

GitHub Pages

GitHub supports creating websites

Useful for personal – project/repository

Branch **gh-pages**

GitHub Pages - examples

Organization level

Repository:

<https://github.com/Arquisoft/Arquisoft.github.io>

Deployed:

<https://arquisoft.github.io/>

It can be very useful for personal web pages

<http://pglez82.github.io>

Documentation deployment

Documentation is deployed using GitHub Pages

GitHub Pages allows users to publish a simple website directly on GitHub

Generated website will be pushed to the branch **gh-pages**

npm package **gh-pages** pushes doc website to gh-pages branch

Everything is automatized with the following command:

```
$ npm run deploy
```

