Métodos de Desenvolvimento de Software (MDS) 2016/2017

Deployment Diagrams

Deployment

- The process of distributing artifacts over nodes, or artifact instances over node instances.

Deployment diagram

- A type of diagram used to model a system's hardware topology and software deployment
- Models the architecture of a given system in execution time
- Presents a static view of the configuration in execution time of the processing nodes and of the distribution of the components that are executed in these corresponding nodes
  - Examples of nodes: server, client, modem, printer, etc.
- The deployment diagrams show:
  - hardware,
  - software (installed in the hardware)
  - middleware (used to connect together the different machines involved)

Deployment Diagram

- Maps the software architecture to the hardware architecture
- Two types of installation diagrams:
  - **Descriptive**: contains nodes, relations between nodes and artifacts
    - Example of node: PC
    - Example of artifacts: jar archive
  - **Instantiated**: contains nodes instances, links between instance nodes and artifacts nodes (which can be anonymous)
    - Example of nodes: Miguel's PC
    - Example of artifacts: a particular jar archive

When do we model this diagram?

- First version during the design phase (descriptive) with the goal to help on the design process of the hardware architecture
- Refinements show one or more instantiation forms, using anonymous instances (instantiated diagrams)
- When the hardware details in the installation place are known, the instantiated diagram can discard anonymous instances and start to use the nodes ids and specific artifacts to be used

Summarizing the process...

- During the design, the emphasis is on the nodes or node instances and their corresponding links
- During the implementation phase, the emphasis is on attributing:
  - Artifact instances to node instances, instantiated diagrams
  - Artifacts to nodes in descriptive diagrams

Nodes and associations

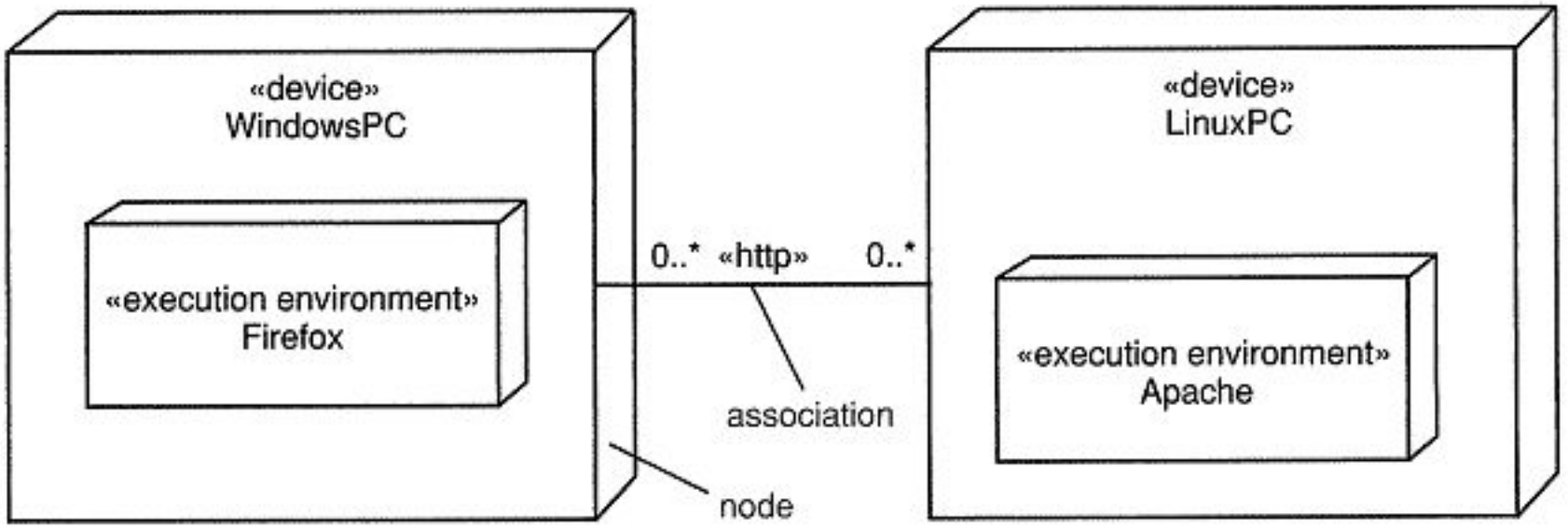
- The deployment diagrams include the several elements used in the component diagrams and also:
  - Nodes that represent either physical or virtual devices (for instance, one node can represent a mainframe)
    - The nodes are represented by 3-Dboxes and can be processors (like a server), or other devices (for instance a modem)
  - Associations (communication paths)
    - Are represented by simple lines and can be decorated with stereotypes to show the type of connection (example http)

Nodes

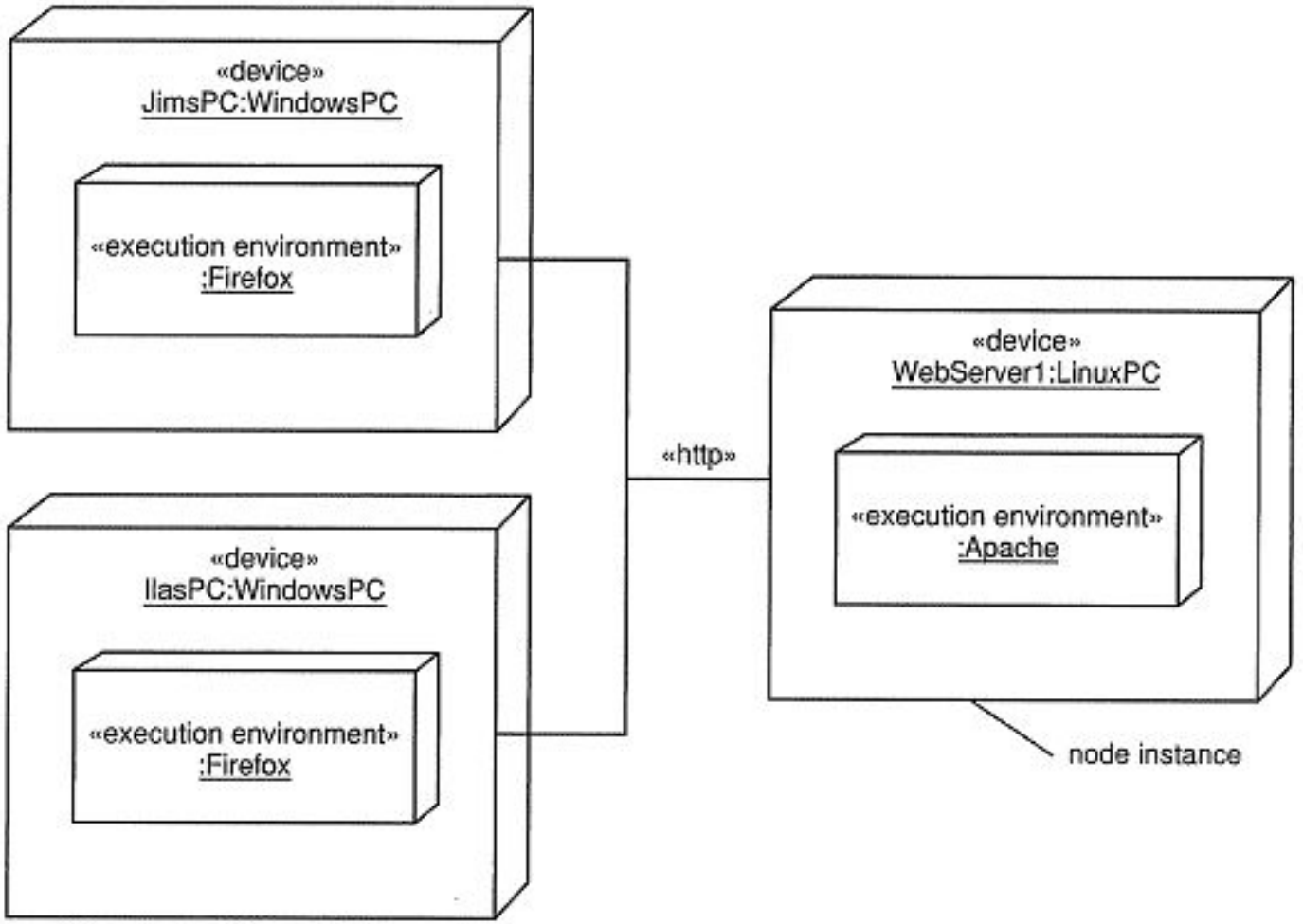
- Physical entities that can execute artifacts
- In general, have memory and, frequently, do processing

Nodes: standard stereotypes

- <<device>> physical device (e.g. PC)
- <<execution environment>> type of execution environment (e.g. Apache web server, Firefox)

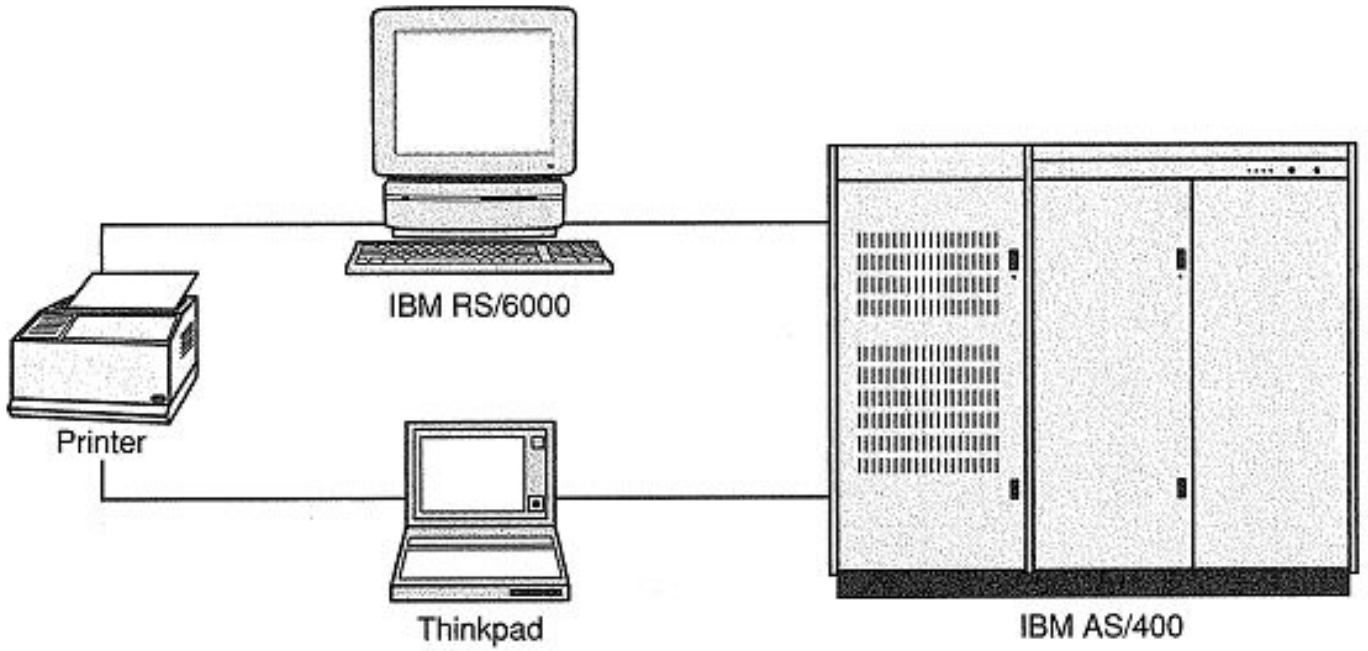


Nodes (instantiated version)



Where is the limits to stereotypes?

- We can create as many stereotypes as required and we can assign distinct images to each of them
  - Facility of reading ☺
  - Interoperability problems ☹



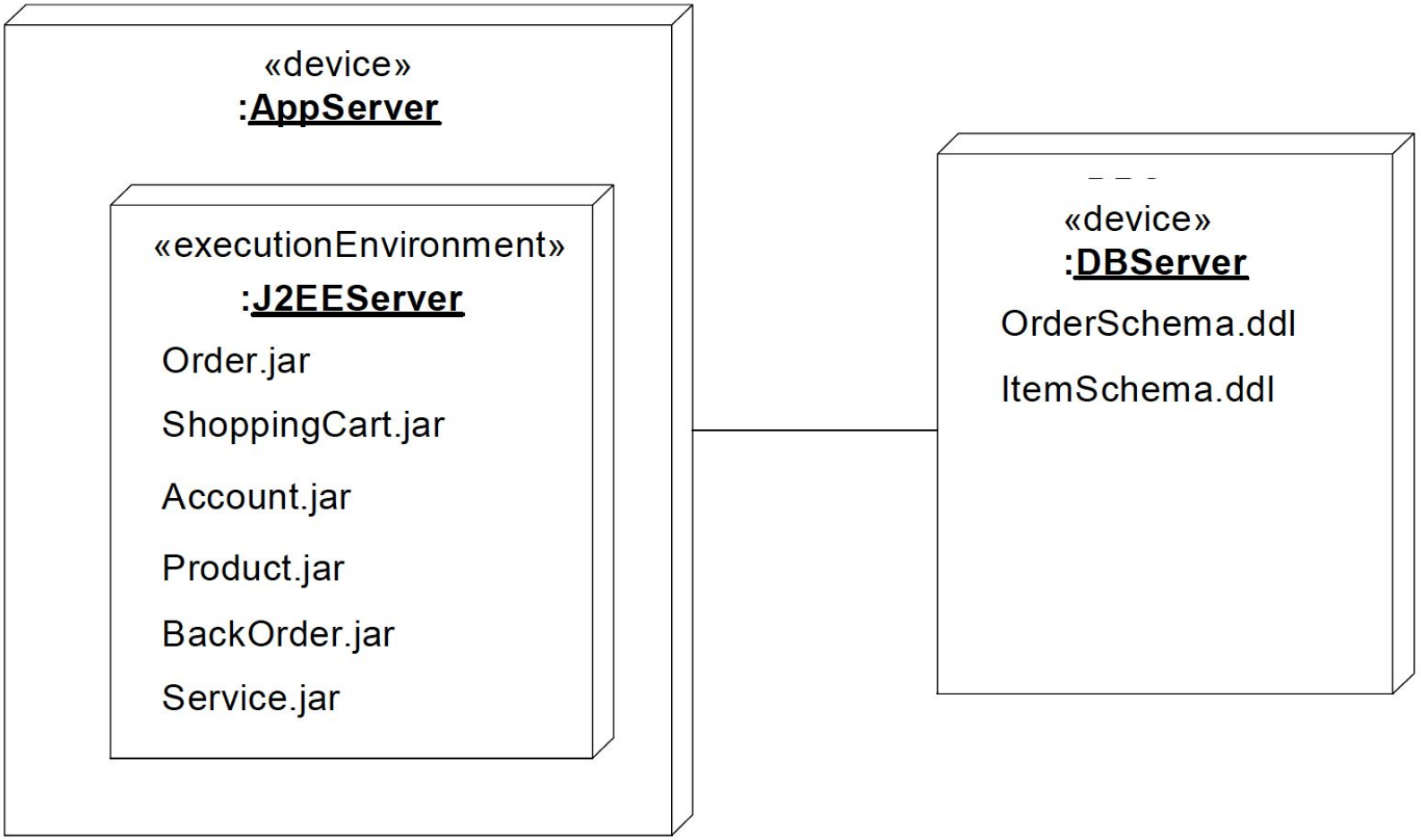
Artifacts

- Represent physical pieces of information related to the software development process.
- Placed in nodes
- Examples:
  - Source code files
  - Executable files
  - Scripts
  - Database tables
  - Documents
  - Other deliverables of the development process(e.g. Models)

Artifact Instances

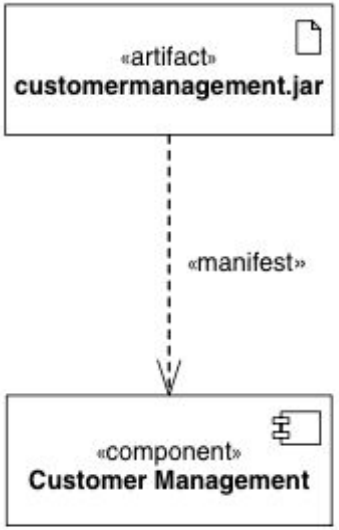
- Represent specific instances of artifacts (e.g. a physical dll on a node is an instance of an artifact)
- Are placed in instance nodes
- It is shown by underlining the name of the artifact

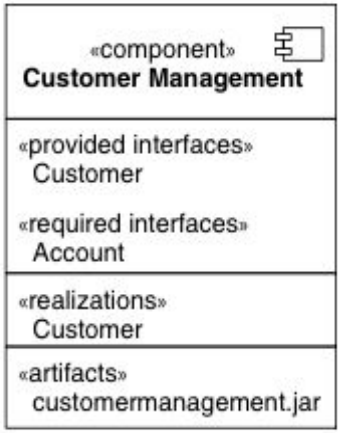
Installation diagram with artifact manifestation

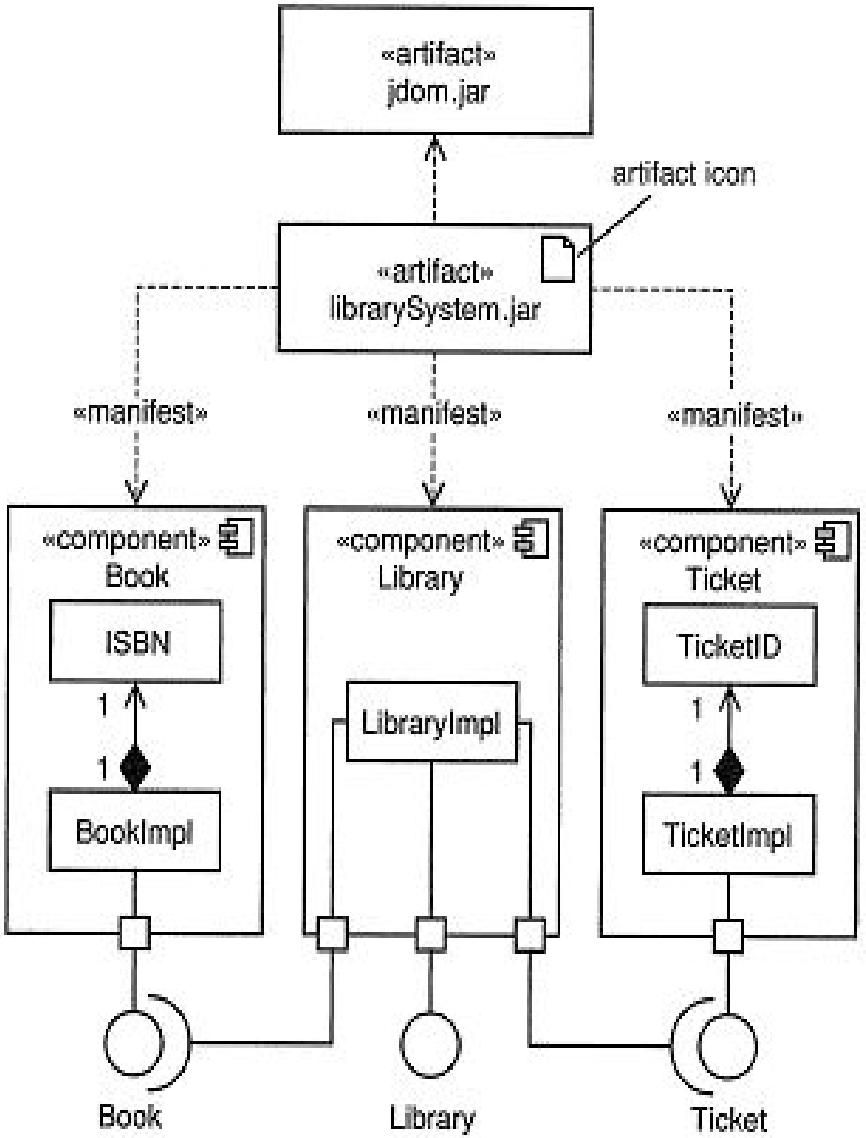


Artifact manifestations

- An artifact is a manifestation of another UML element
- In general, an artifact is a manifestation of one or more components

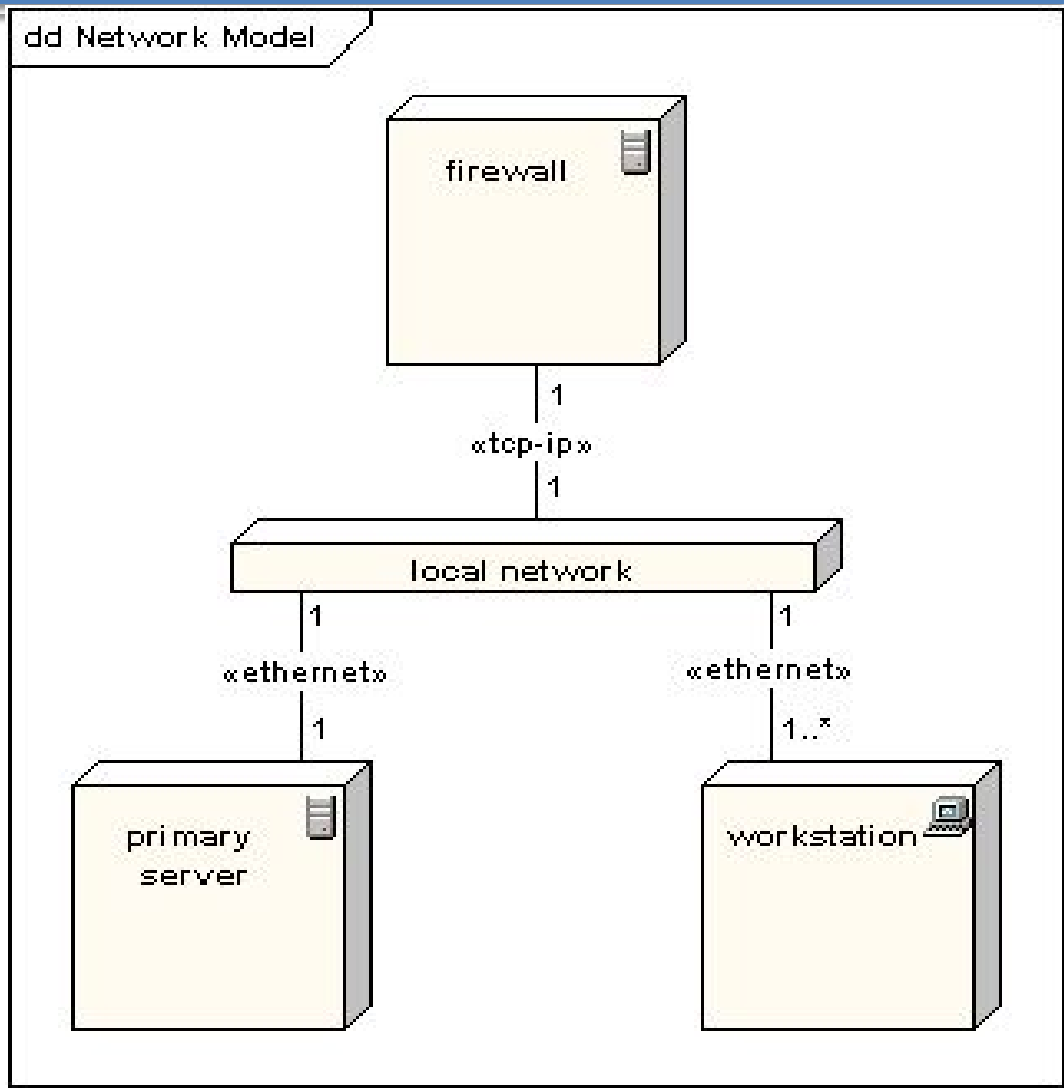




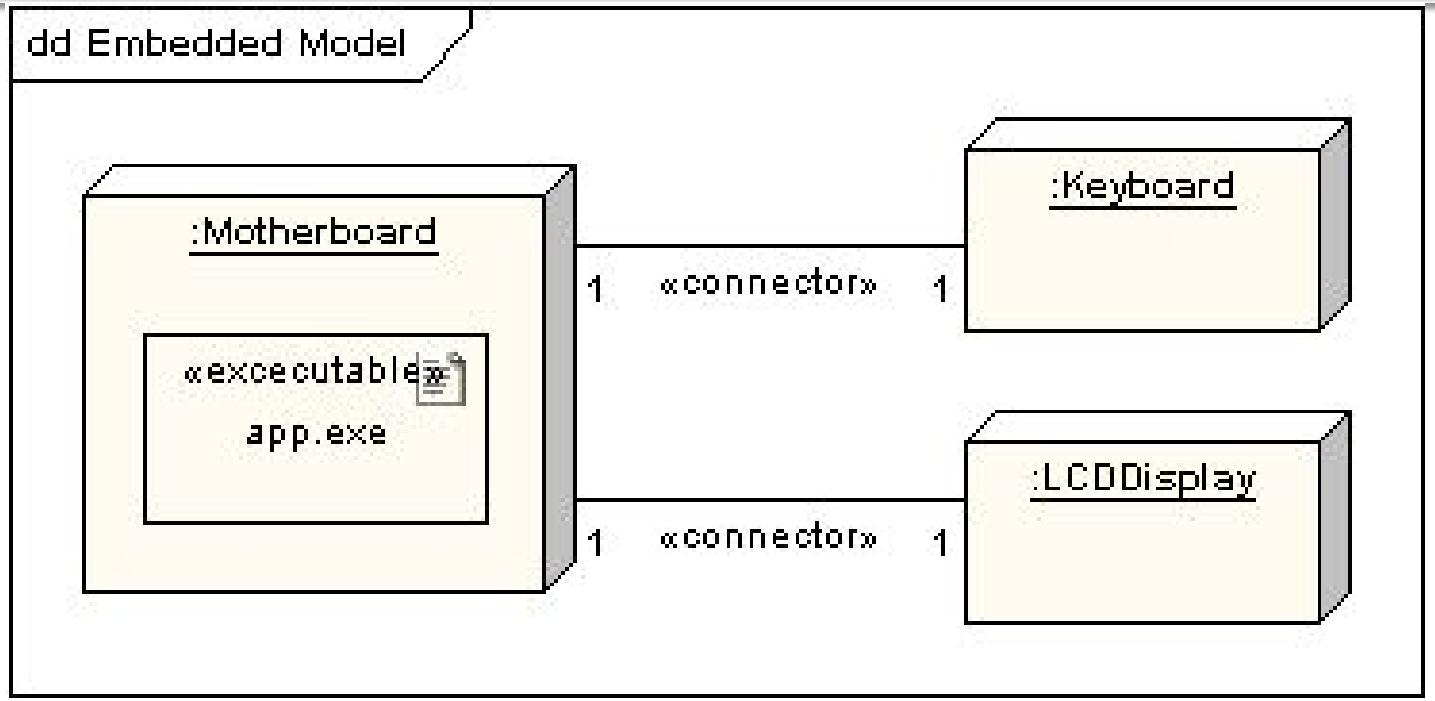


Deployment Diagram

- Network protocols as stereotypes
- Multiplicities

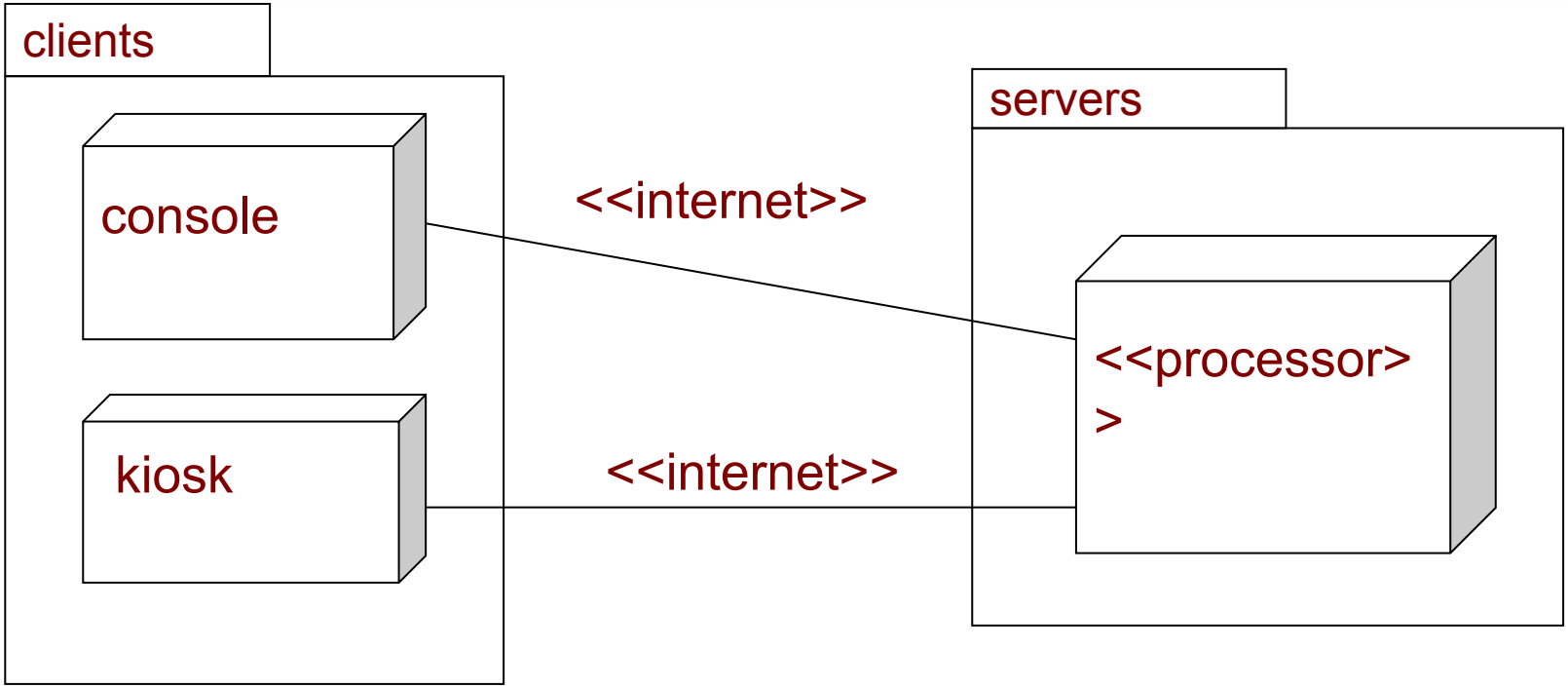


Deployment diagrams



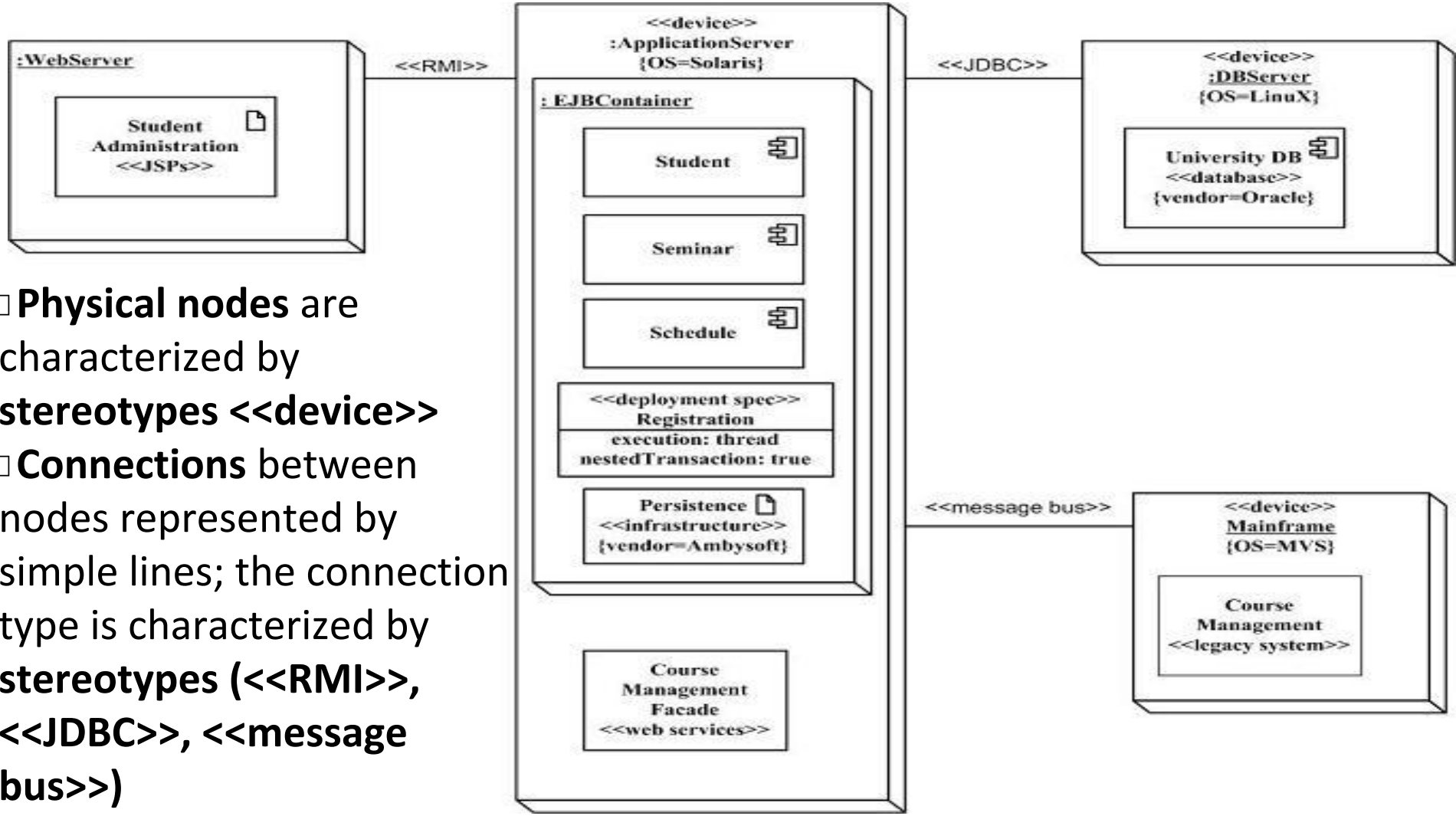
- Deployment diagrams to part of the embedded system, presenting an executable artifact as being contained by the motherboard node

Deployment Diagram



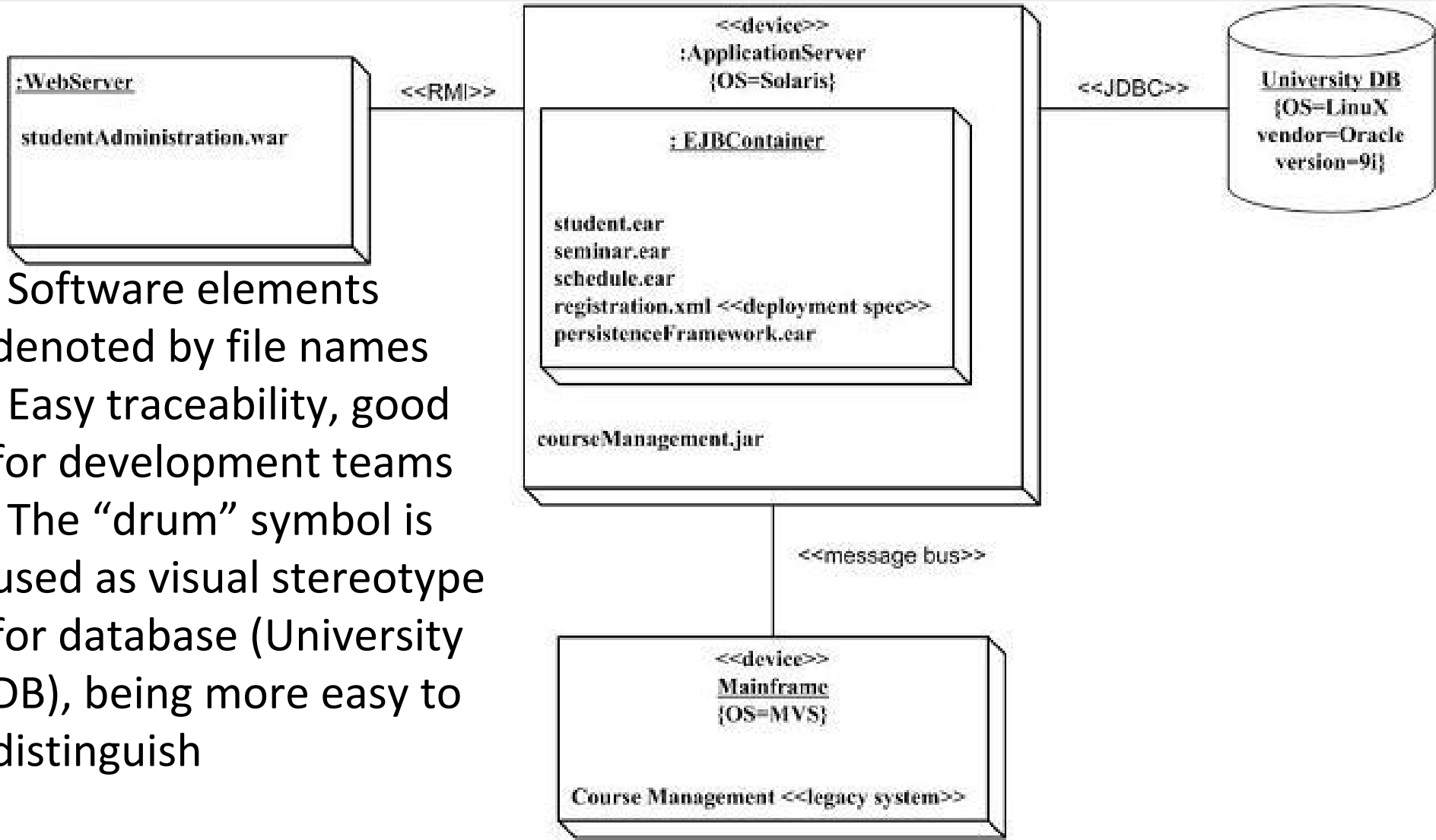
- The packages are used to structure different types of nodes

Deployment Diagrams



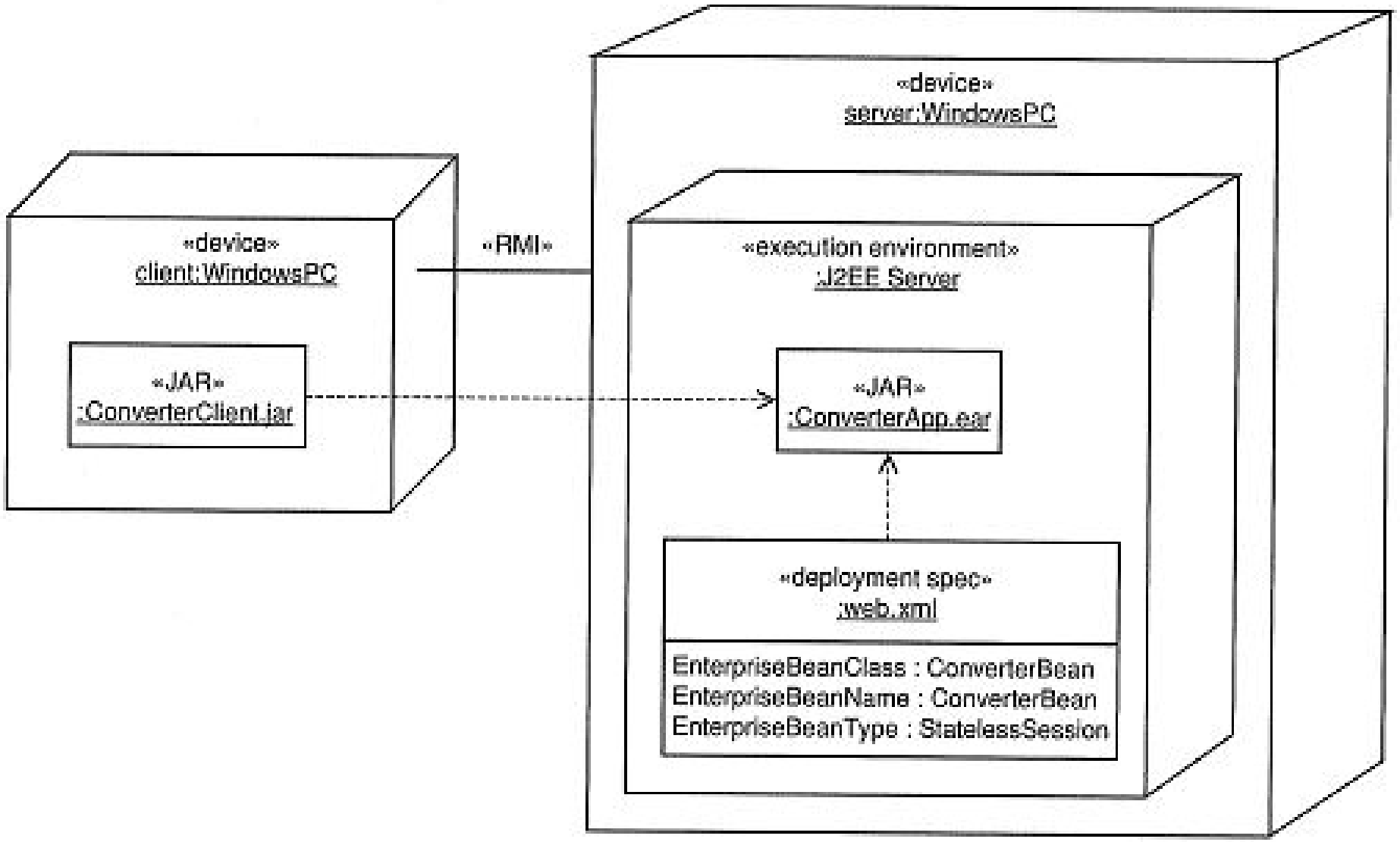
- **Physical nodes** are characterized by **stereotypes <<device>>**
- **Connections** between nodes represented by simple lines; the connection type is characterized by **stereotypes (<<RMI>>, <<JDBC>>, <<message bus>>)**

Deployment Diagram (more concise)



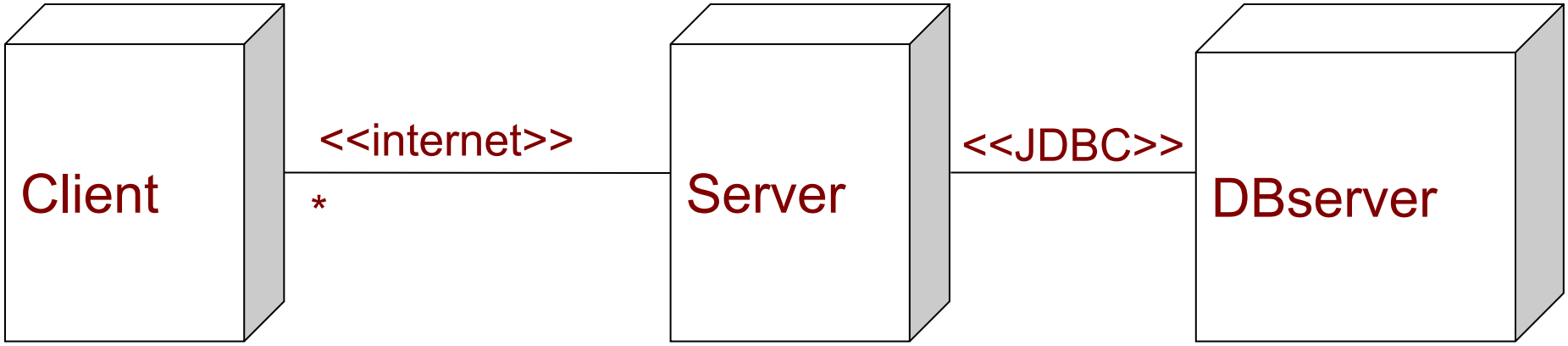
- Software elements denoted by file names
- Easy traceability, good for development teams
- The “drum” symbol is used as visual stereotype for database (University DB), being more easy to distinguish

Another example



Deployment Diagrams

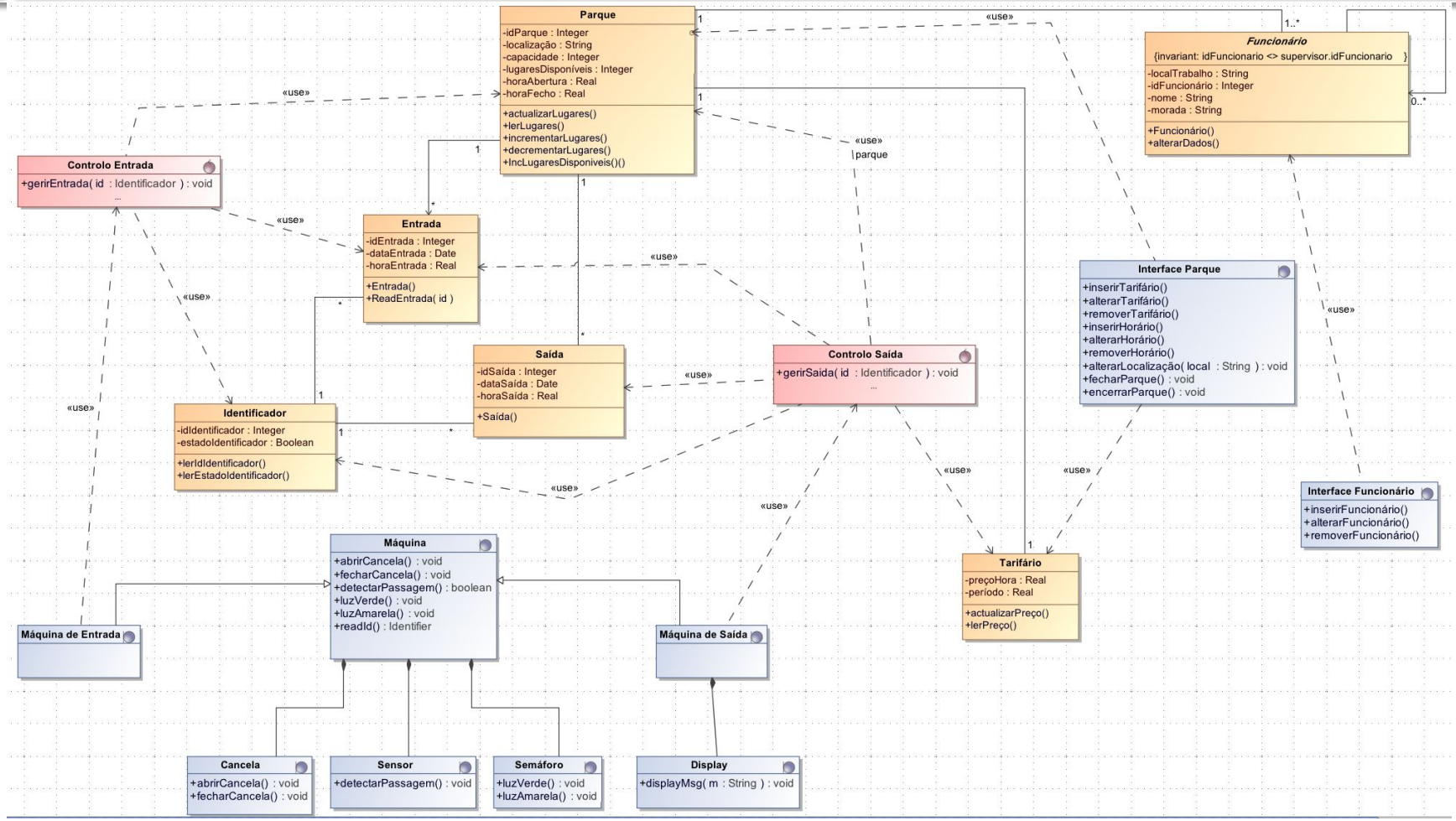
- Used to model:
  - Client/Server systems
  - Distributed systems
  - Embedded systems
  - ...



Bibliography:

UML 2 and the Unified Process, Arlow and Neustadt
The object primer: agile model-driven development with UML 2.0, Scott W. Ambler
UML 2.0 Superstructure
UML 2.0 Infrastructure

Está com coragem?



Proponha um diagrama de instalação para o modelo mostrado no slide anterior

- Sugestão:
  - No módulo anterior, produziu um diagrama de componentes para o mesmo exemplo. Parta desse diagrama para construir o seu diagrama de instalação

De quantos diagramas necessitamos?

- Depende:
  - Usamos diagramas para visualizar o sistema sob diferentes perspectivas.
  - É impossível compreender completamente um sistema a partir de apenas uma perspectiva.
  - Os diagramas são usados para comunicar.
- Há vários elementos do modelo que são usados em mais que um diagrama:
  - Por exemplo, uma classe pode aparecer em um ou mais diagramas de classe, pode ser representada num diagrama de estados, as suas instâncias podem aparecer em diagramas de sequência, etc.
  - Cada diagrama fornece uma nova perspectiva.

Absolutamente fundamental:

Bons diagramas contribuem para tornar o sistema que estamos a desenvolver **compreensível** e o seu desenvolvimento **gerível**.

A escolha de um conjunto de diagramas adequado obriga-o a perguntar **as questões mais pertinentes** sobre o sistema e ajuda a evidenciar **as implicações do modelo que criou**.

Referências

- UML 2 and the Unified Process, Arlow and Neustadt
- Agile Modeling
- IBM's Rational Library
- The object primer: agile model-driven development with UML 2.0, Scott W. Ambler
- UML 2.0 Superstructure
- UML 2.0 Infrastructure