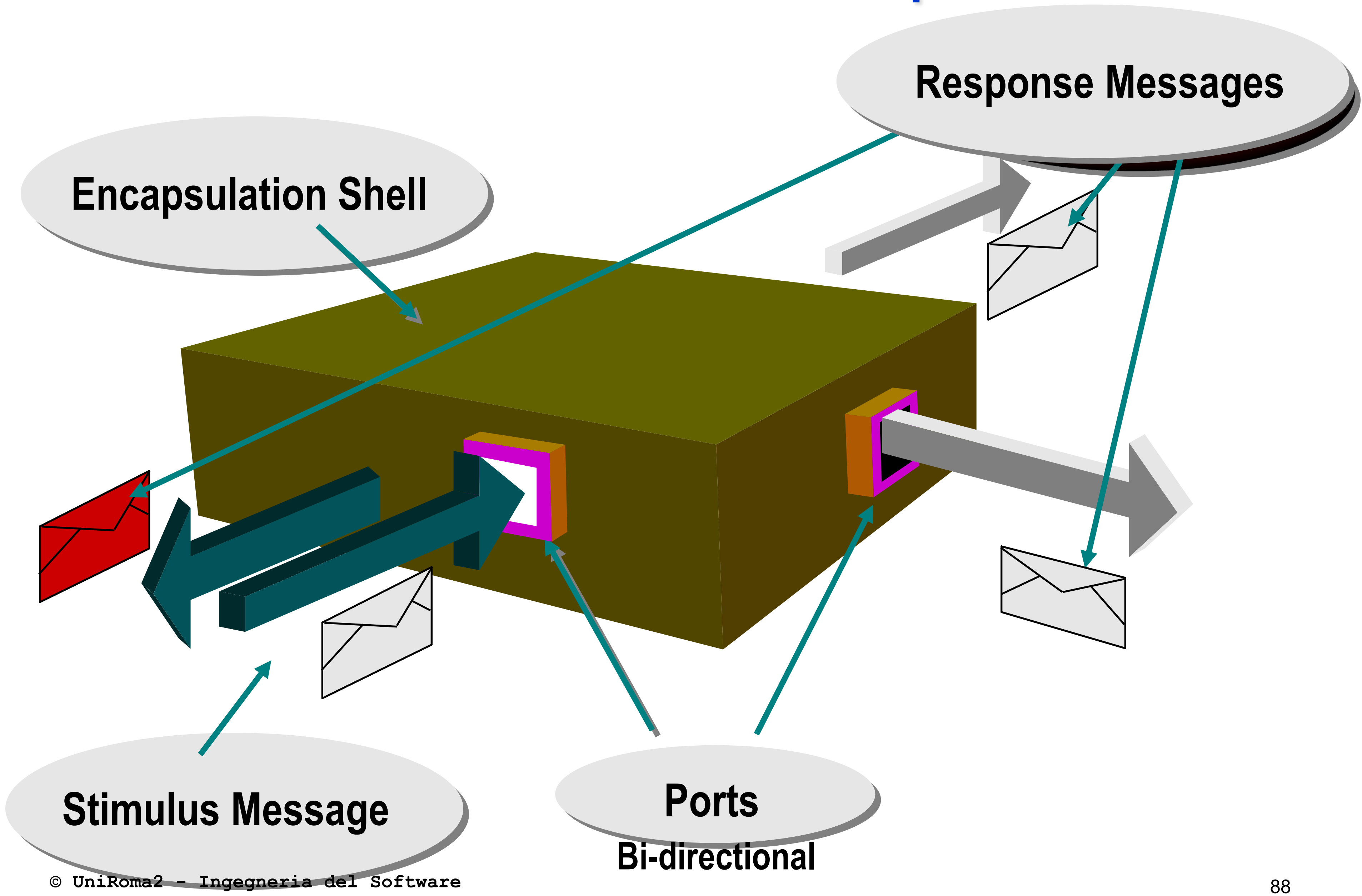
UML Structured Class

- A structured class contains *roles* or *parts* that form its structure and realize its behavior
  - Describes the internal implementation structure
- The *parts* themselves may also be structured classes
  - Allows hierarchical structure to permit a clear expression of multilevel models
- A *connector* is used to represent an association in a particular context
  - Represents communications paths among parts

Structured Class Usage

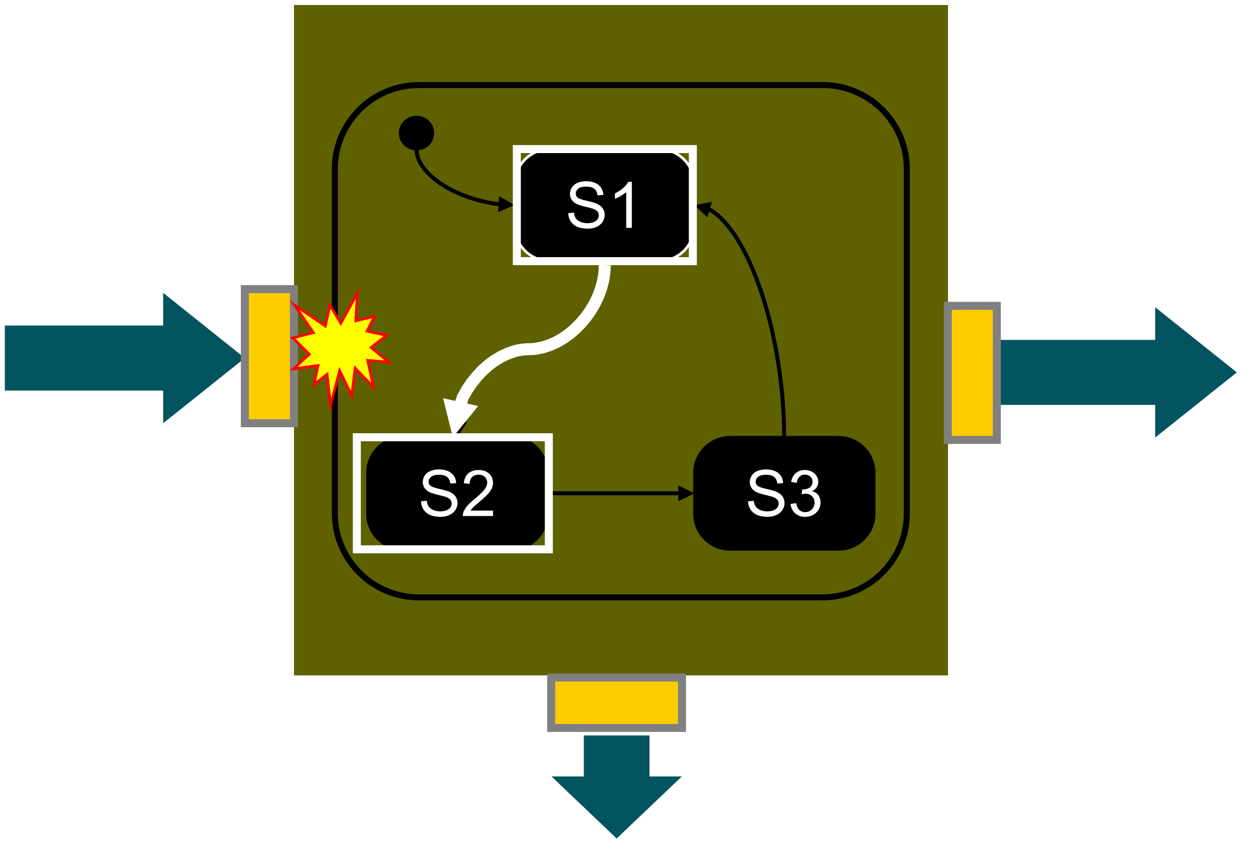
- Can be used as the primary building blocks of an application
  - Provides graphical representation of design elements
  - Can *hide implementation details*
    - Powerful abstraction tool - same construct applies to multiple semantic levels
    - Clear communication and understanding of system architecture
  - *Strict encapsulation of behavior*
    - Interactions restricted to message-based communications passed through external interfaces (ports)

Structured Class: Conceptual View



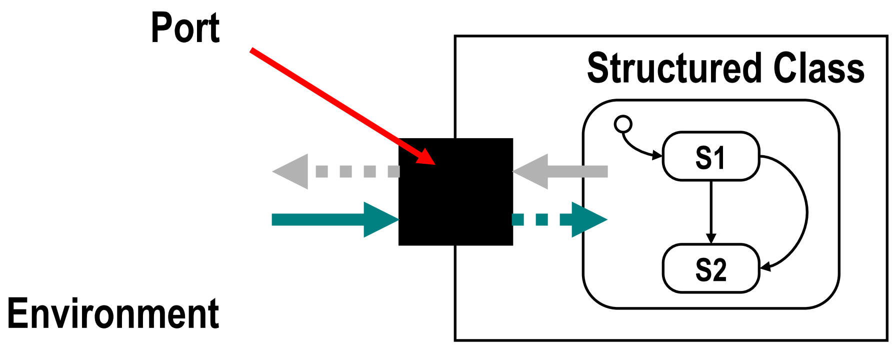
Structured Class: Behavior

- Optional hierarchical state machine
  - State based signal handler

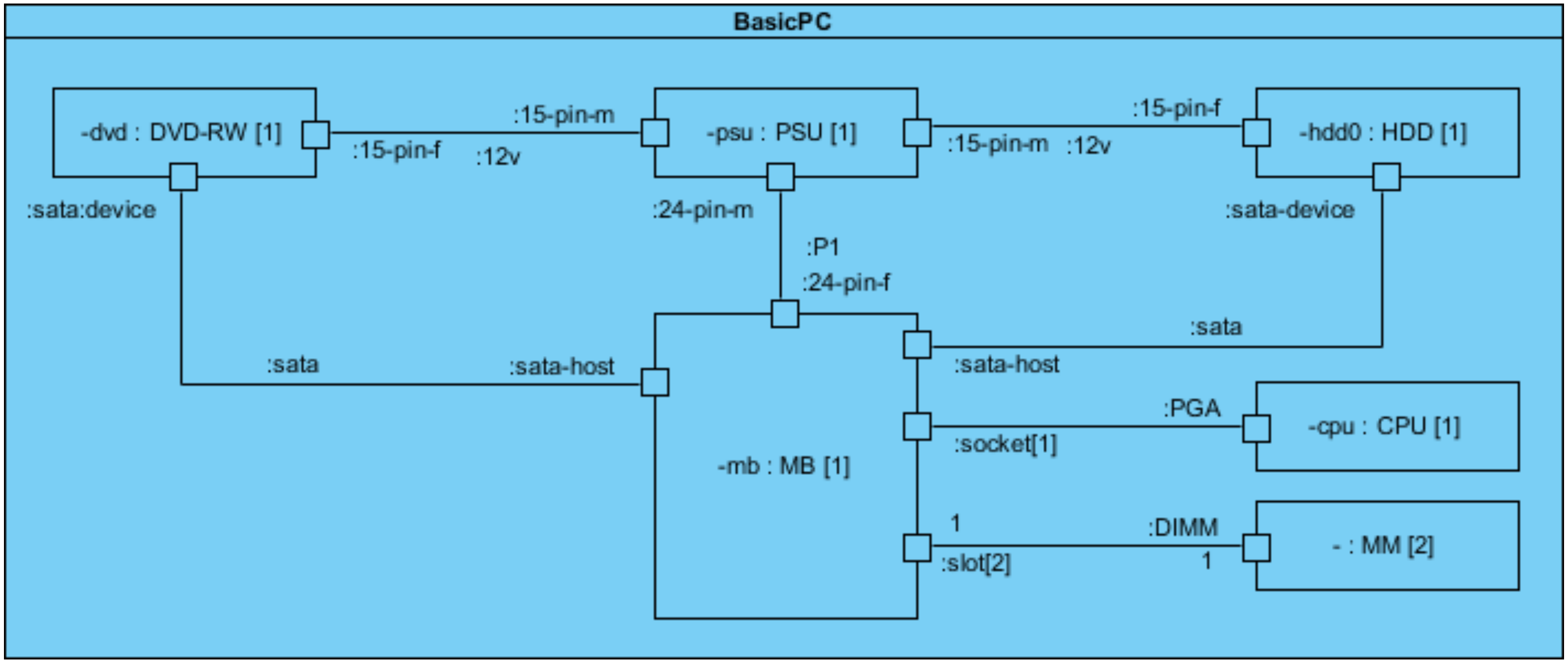


Structured Class: Autonomous Design Unit

- Strict encapsulation ensures that the implementation is independent from the environment
  - Ports can play a bi-directional mediation role
    - Environment only sees the port of the structured class
    - The internal behavior is built to the "specification" provided by its interface
  - Structured classes can be independently designed, and unit tested



Example: UML Composite Structure Diagram

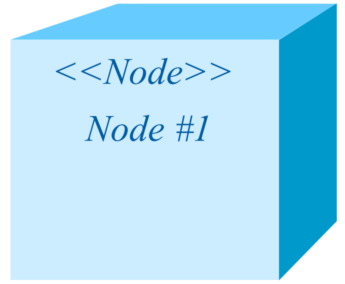


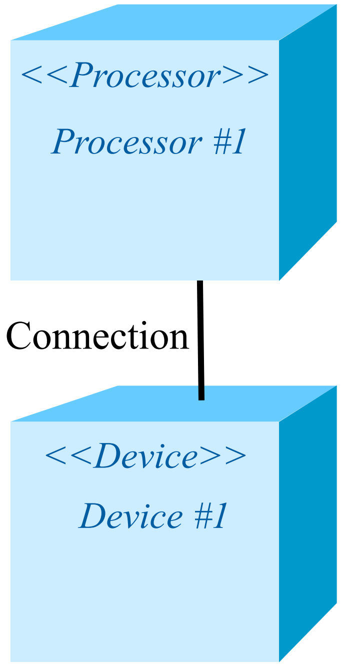
Platform Configuration

- A platform configuration describes the hardware/software solution that defines how the functionality of the system can be distributed across physical nodes
  - Explain the relationship between model elements and their implementation, as well as their deployment
- It is obtained by:
  - defining the platform configuration by use of a *deployment diagram*
  - allocating system elements (artifacts) to nodes of the deployment diagrams

Deployment Model Modeling Elements

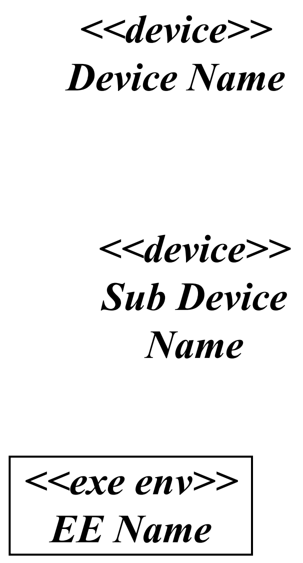
- Node
  - Physical run-time computational resource
  - Processor node - Executes system software
  - Device node
    - Support device
    - Typically controlled by a processor
- Connection
  - Communication mechanism
  - Physical medium
  - Software protocol





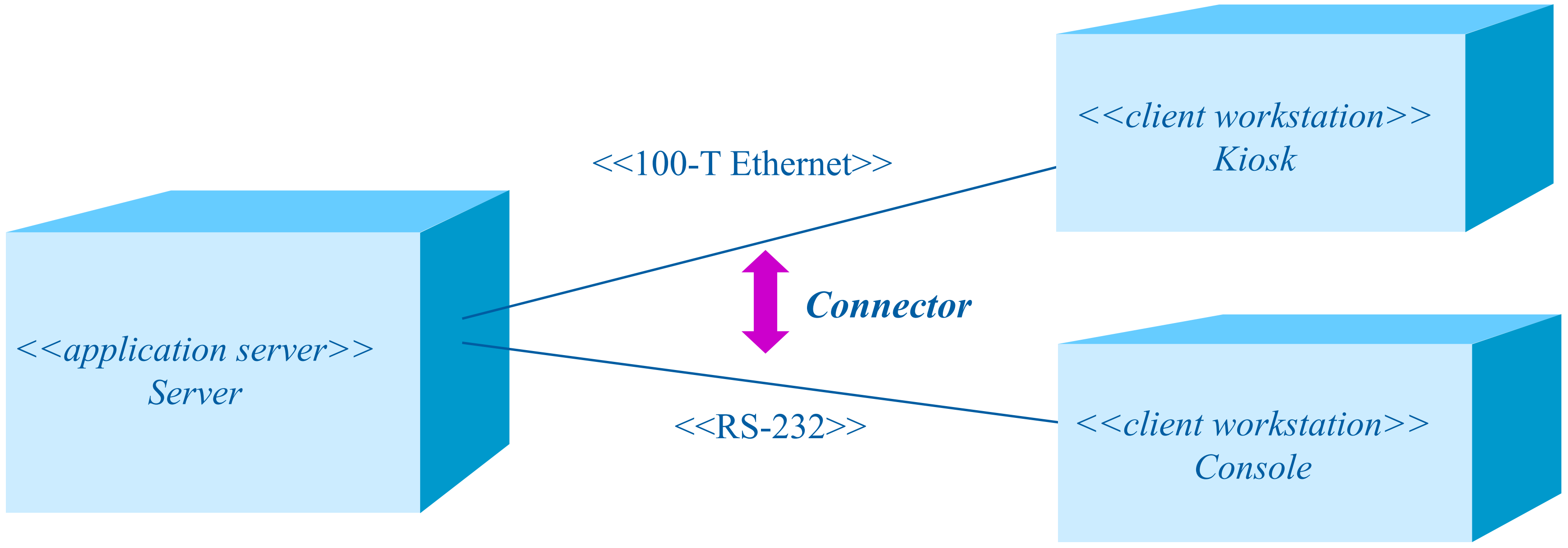
What Is a Node?

- Represents a run-time computational resource, and generally has at least memory and often processing capability.
- Types:
  - *Device* - Physical computational resource with processing capability. Devices may be nested
  - *Execution Environment* - Represents particular execution platforms

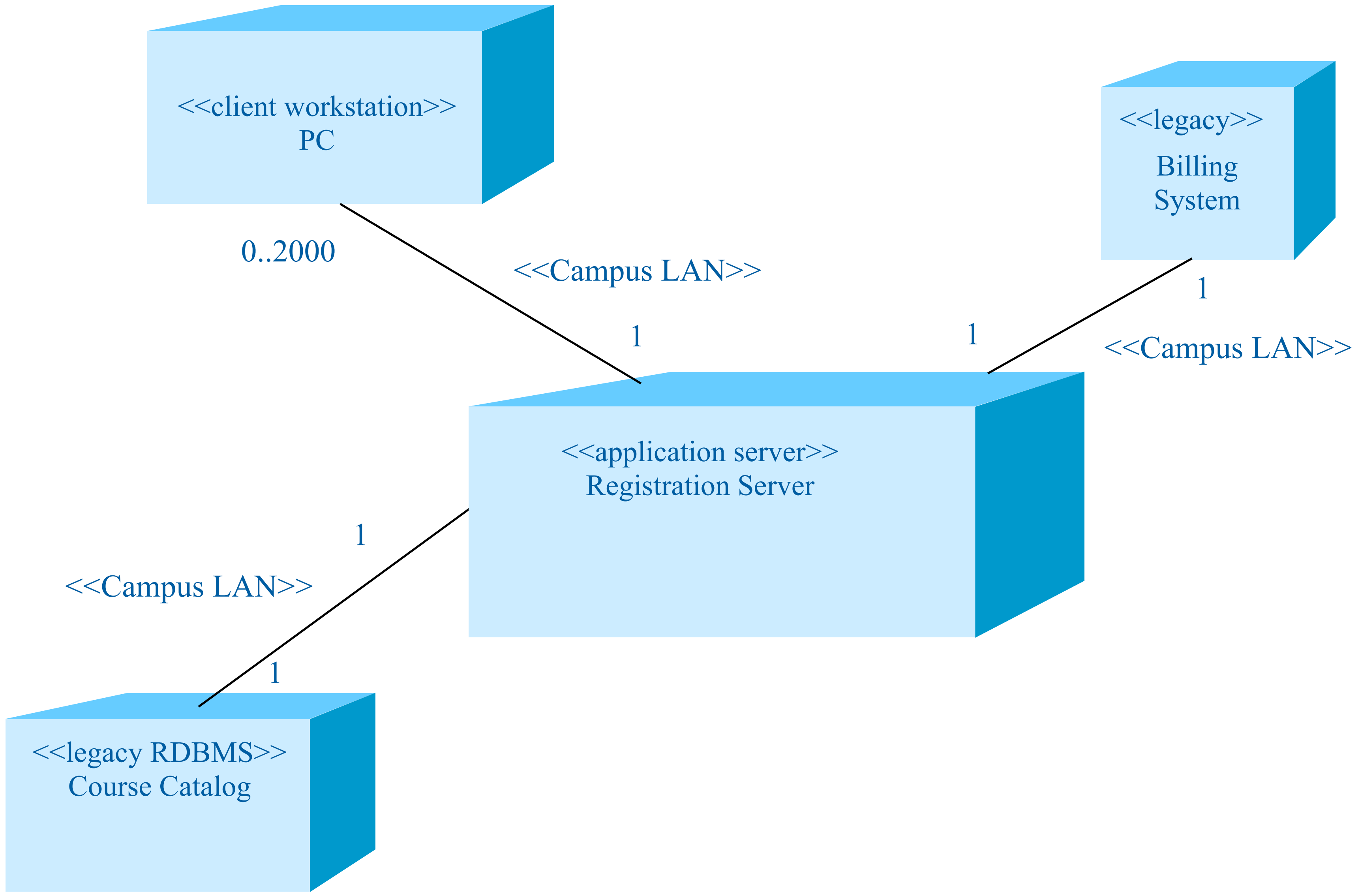


What Is a Connector?

- A connector represents a communication mechanism described by:
  - Physical medium
  - Software protocol



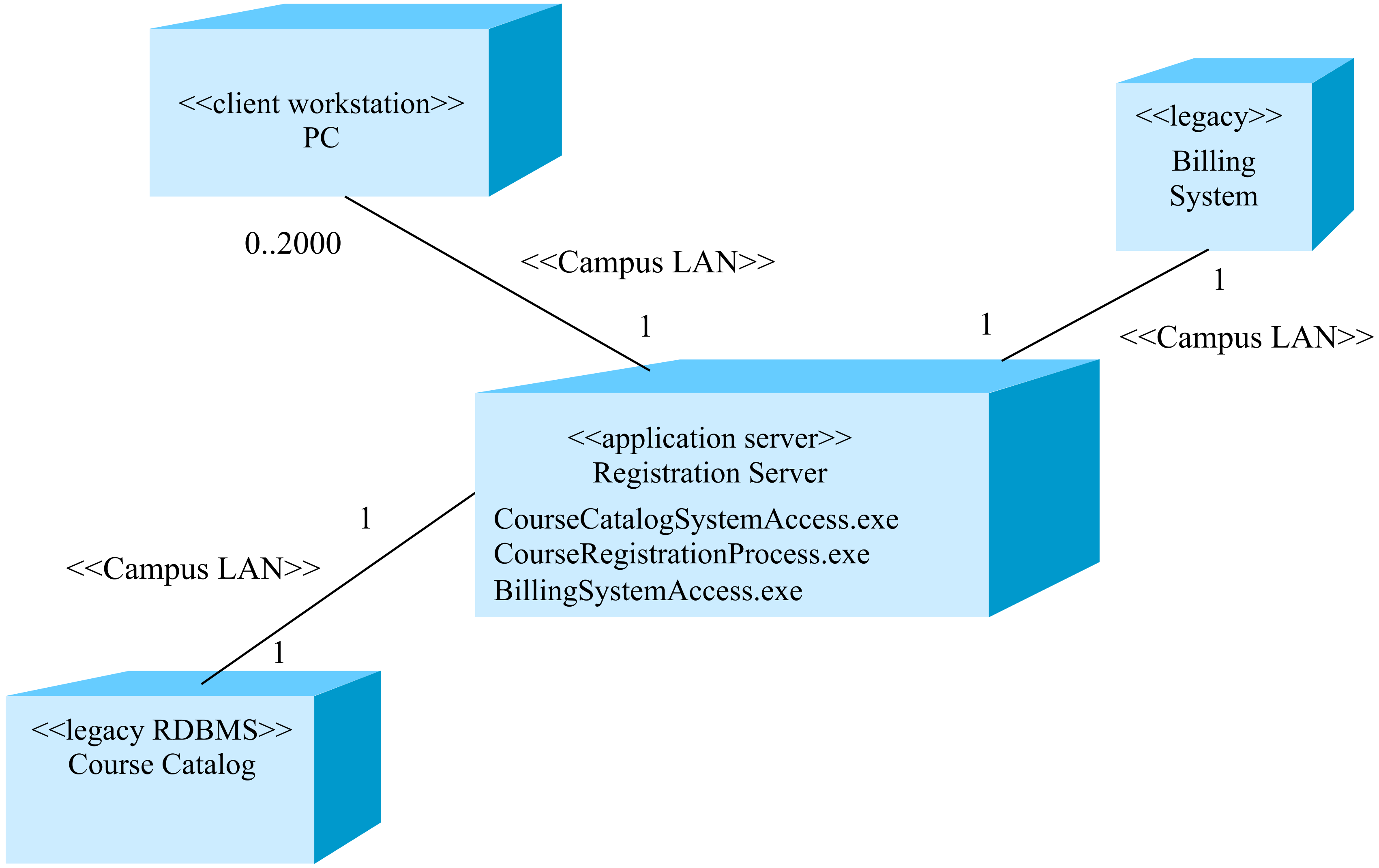
Example: Deployment Diagram



Process-to-Node Allocation Considerations

- Distribution patterns
- Response time and system throughput
- Minimization of cross-network traffic
- Node capacity
- Communication medium bandwidth
- Availability of hardware and communication links
- Rerouting requirements

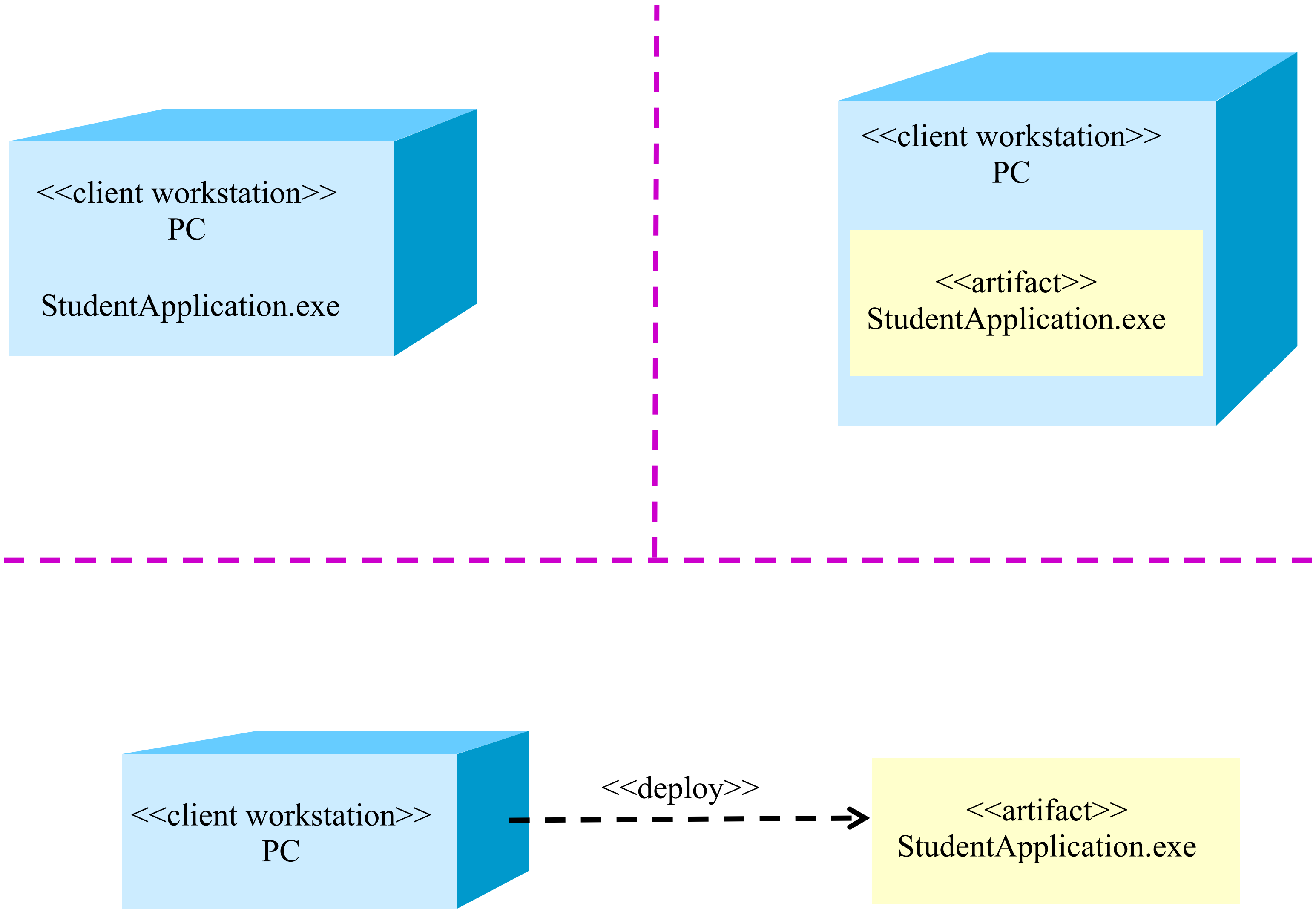
Example: Deployment Diagram with Processes



What is Deployment?

- Deployment is the assignment, or mapping, of software artifacts to physical nodes during execution
  - Artifacts are the entities that are deployed onto physical nodes
    - Processes are assigned to computers
- Artifacts model physical entities
  - Files, executables, database tables, web pages, and so on.
- Nodes model computational resources
  - Computers, storage units

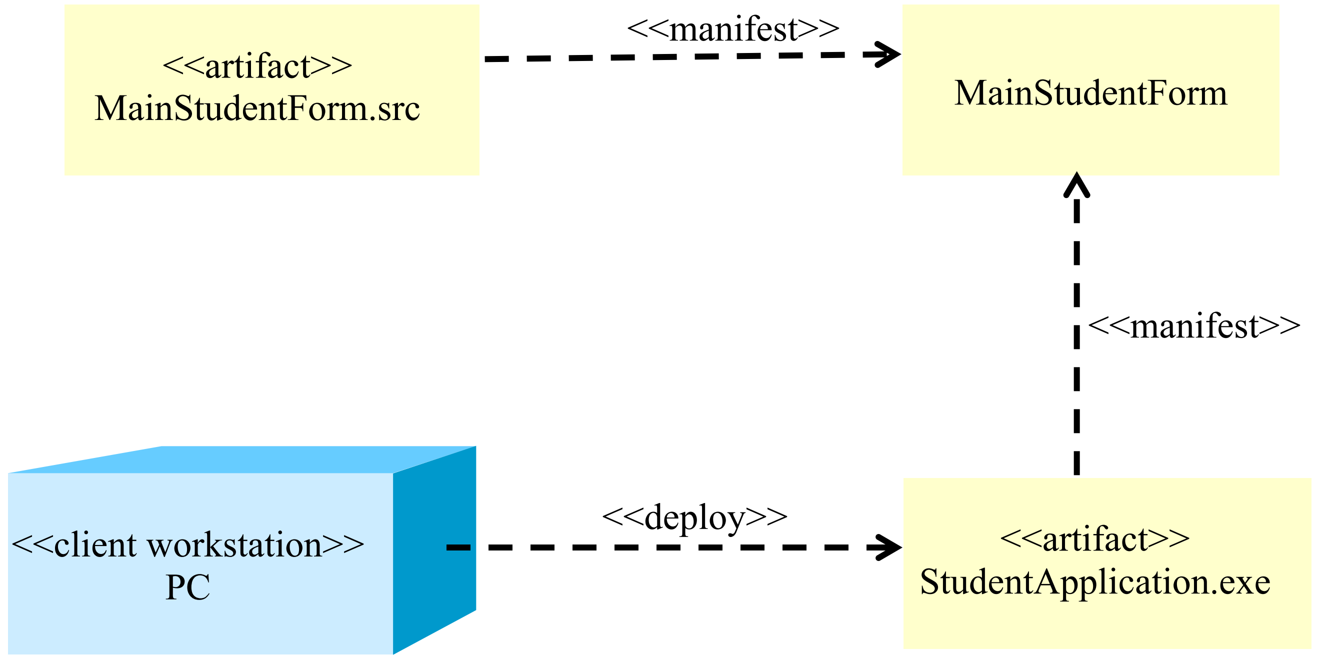
Example: Deploying Artifacts to Nodes



What is Manifestation?

- The physical implementation of a model element as an artifact.
  - A relationship between the model element and the artifact that implements it
  - Model elements are typically implemented as a set of artifacts.
  - Examples of Model elements are source files, executable files, documentation file

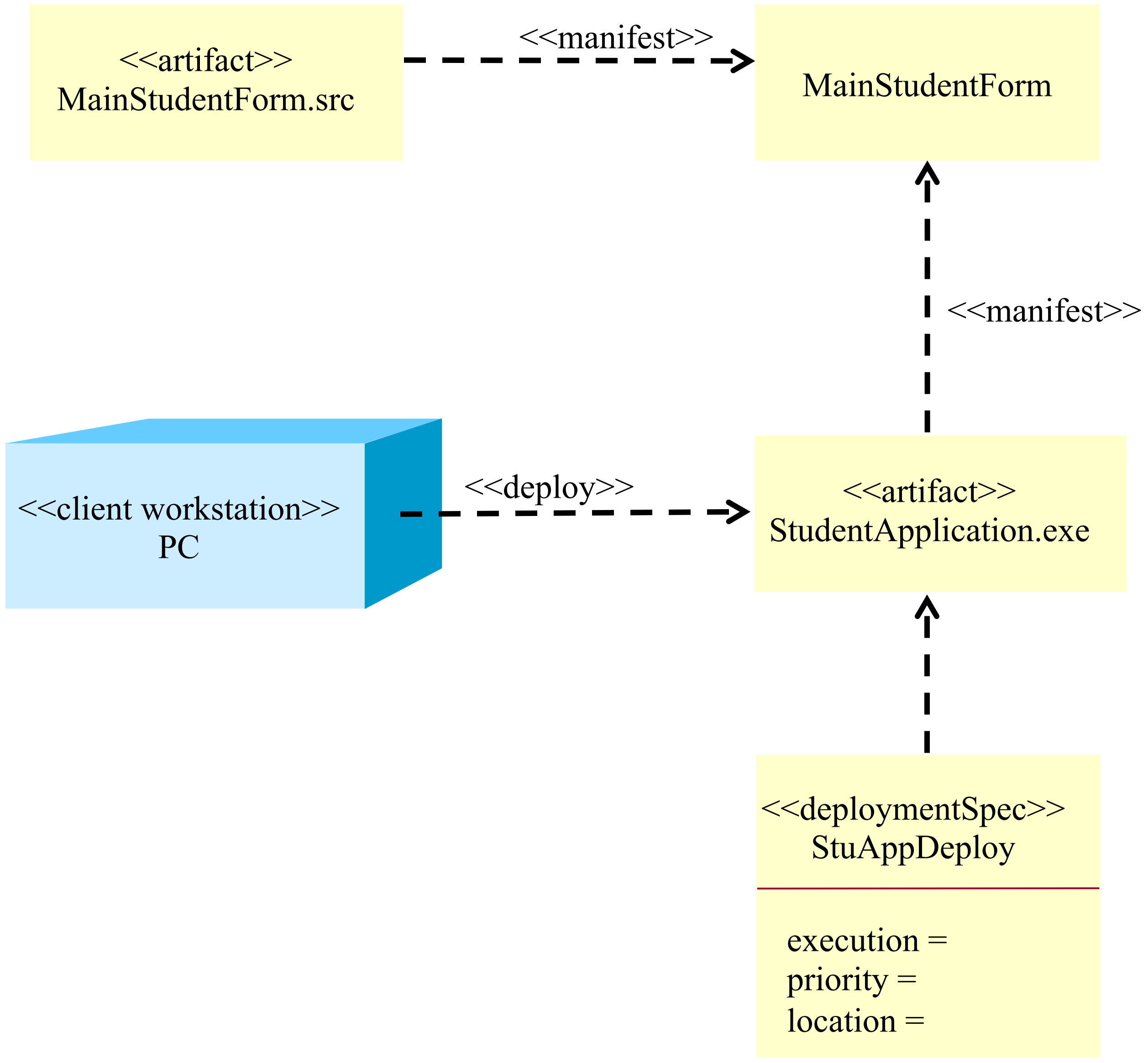
Example: Manifestation



What is a Deployment Specification?

- A detailed specification of the parameters of the deployment of an artifact to a node
  - May define values that parameterize the execution

Example: Deployment Specification


<<artifact>>
MainStudentForm.src
<<manifest>>
MainStudentForm
<<manifest>>
<<client workstation>>
PC
<<deploy>>
<<artifact>>
StudentApplication.exe
<<deploymentSpec>>
StuAppDeploy
execution =
priority =
location =