Model Driven Software Development

Rogardt Heldal

From requirements to code

Requirement/top level architecture
- Specification - quite informal

Can use models communications

Software architecture
- Precise structure of the system

Can use models analysis

Software design
- Behavior

Can use models Simulations

Requirement/top level architecture
- Specification - quite informal

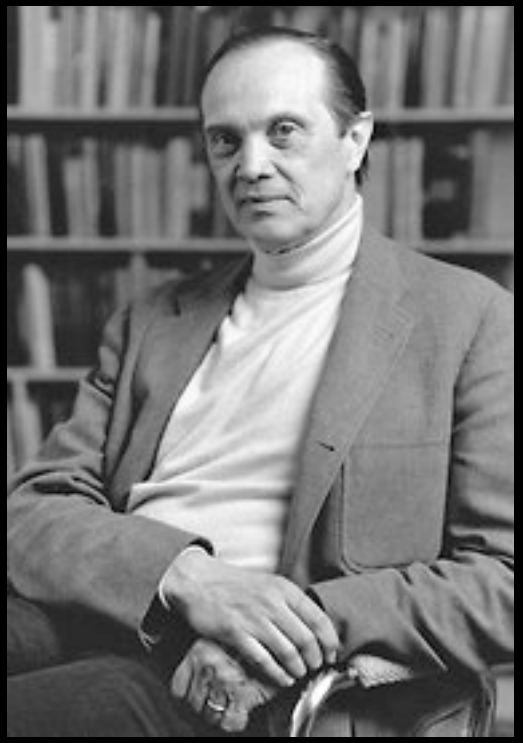
Software architecture
- Precise structure of the system

Software design
- Behavior

Requirement/top level architecture
- Specification - quite informal

- Often over-specified models

Miller's Law



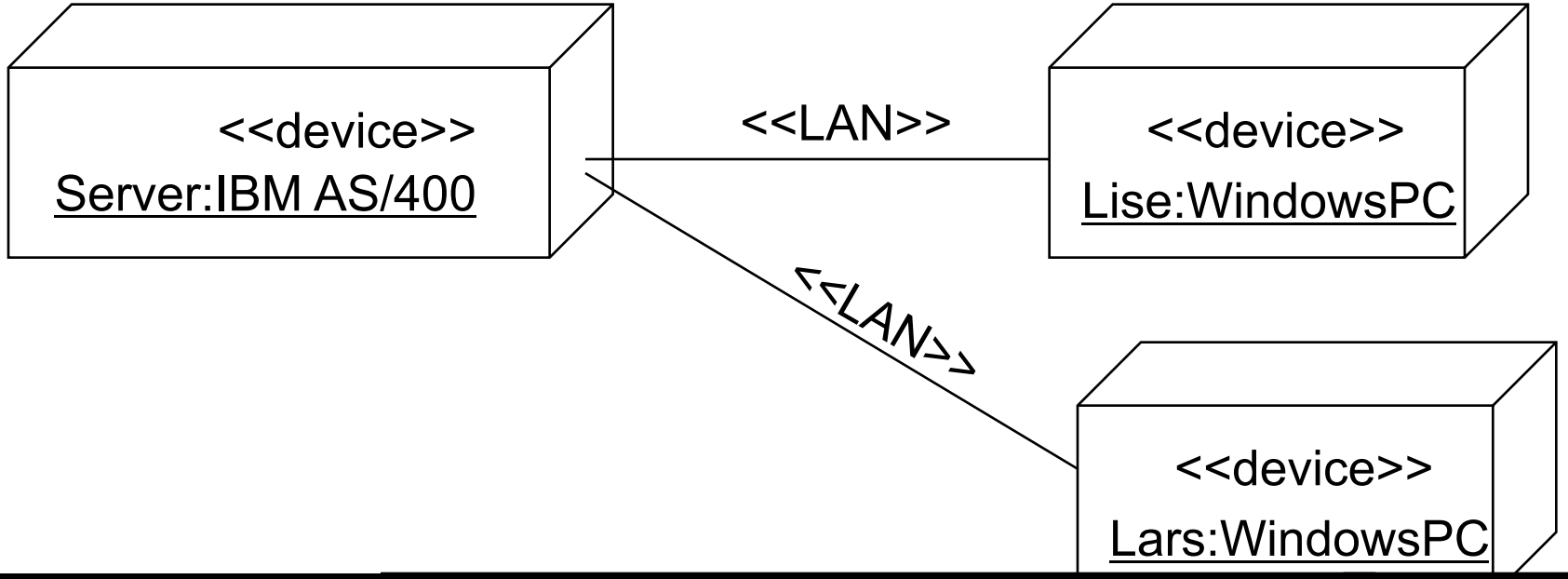
George A. Miller

The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capac for Processing Information

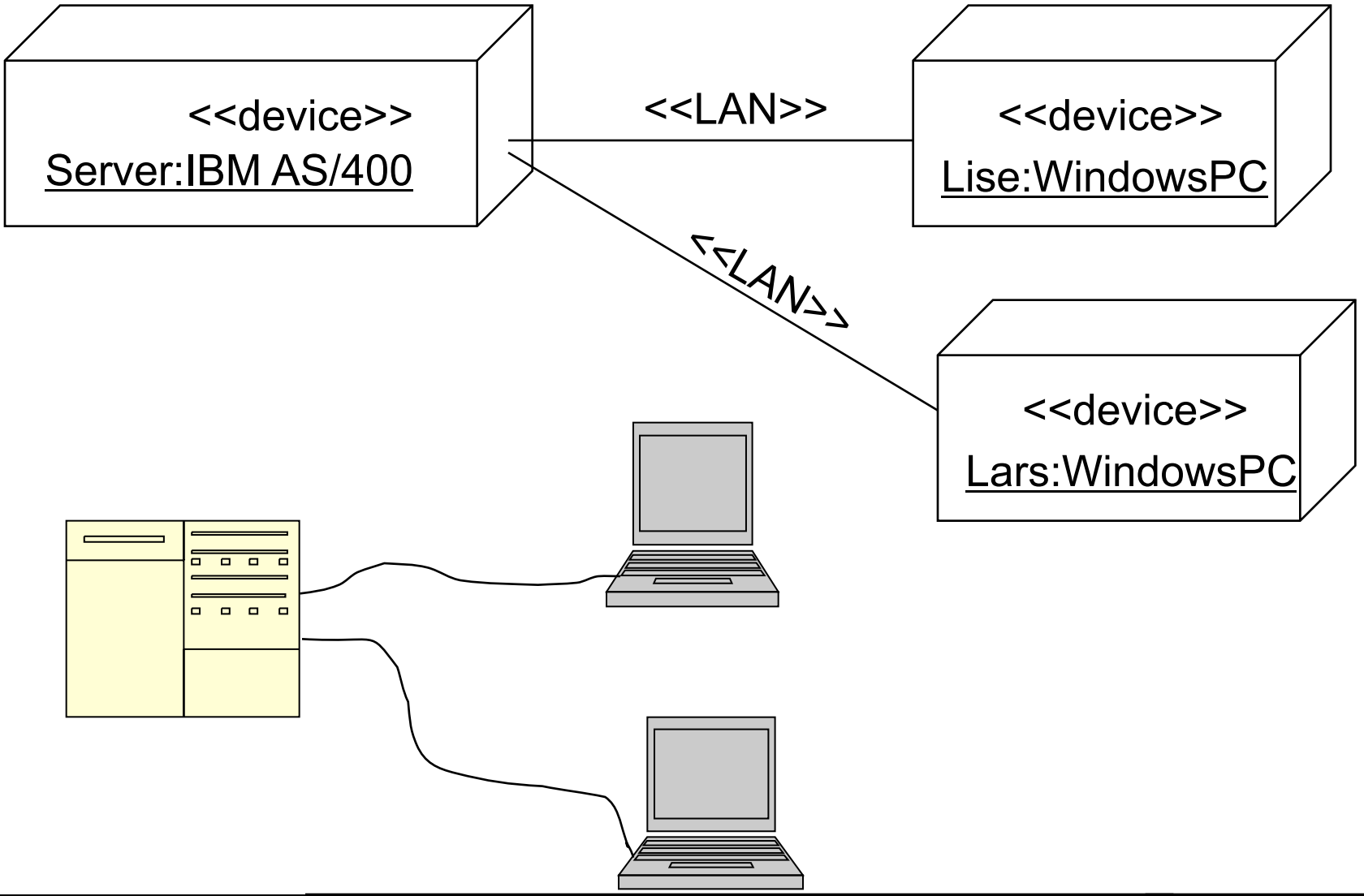
7 +/- 2

Deployment model

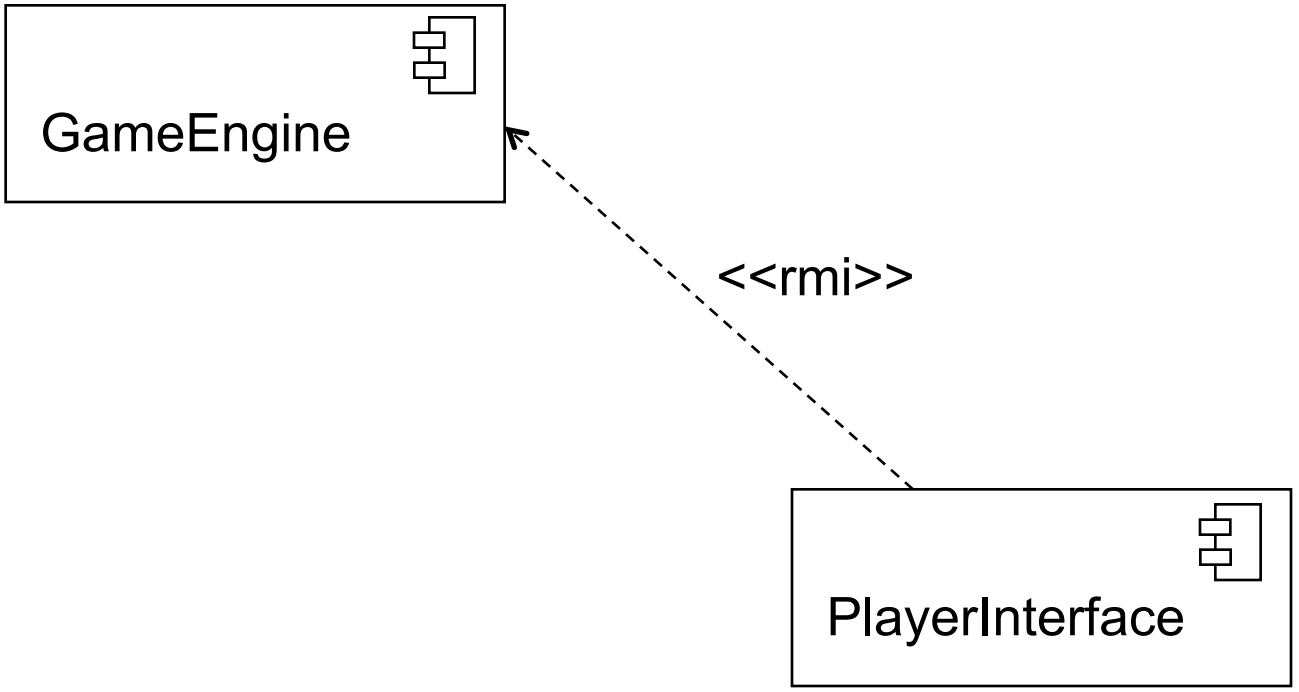
Instance:



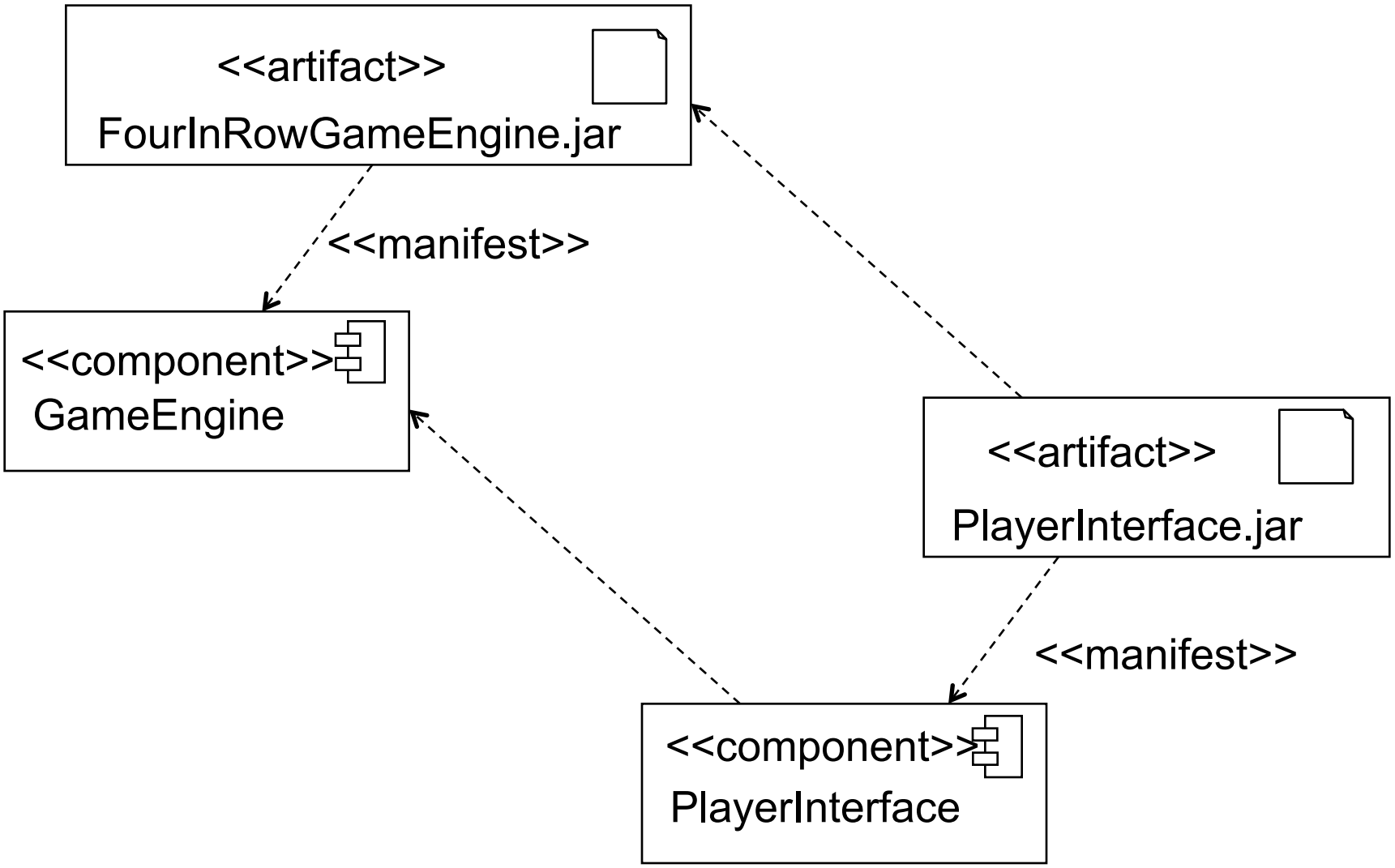
Real world


<<device>>
Server:IBM AS/400
<<LAN>>
<<device>>
Lise:WindowsPC
<<LAN>>
<<device>>
Lars:WindowsPC

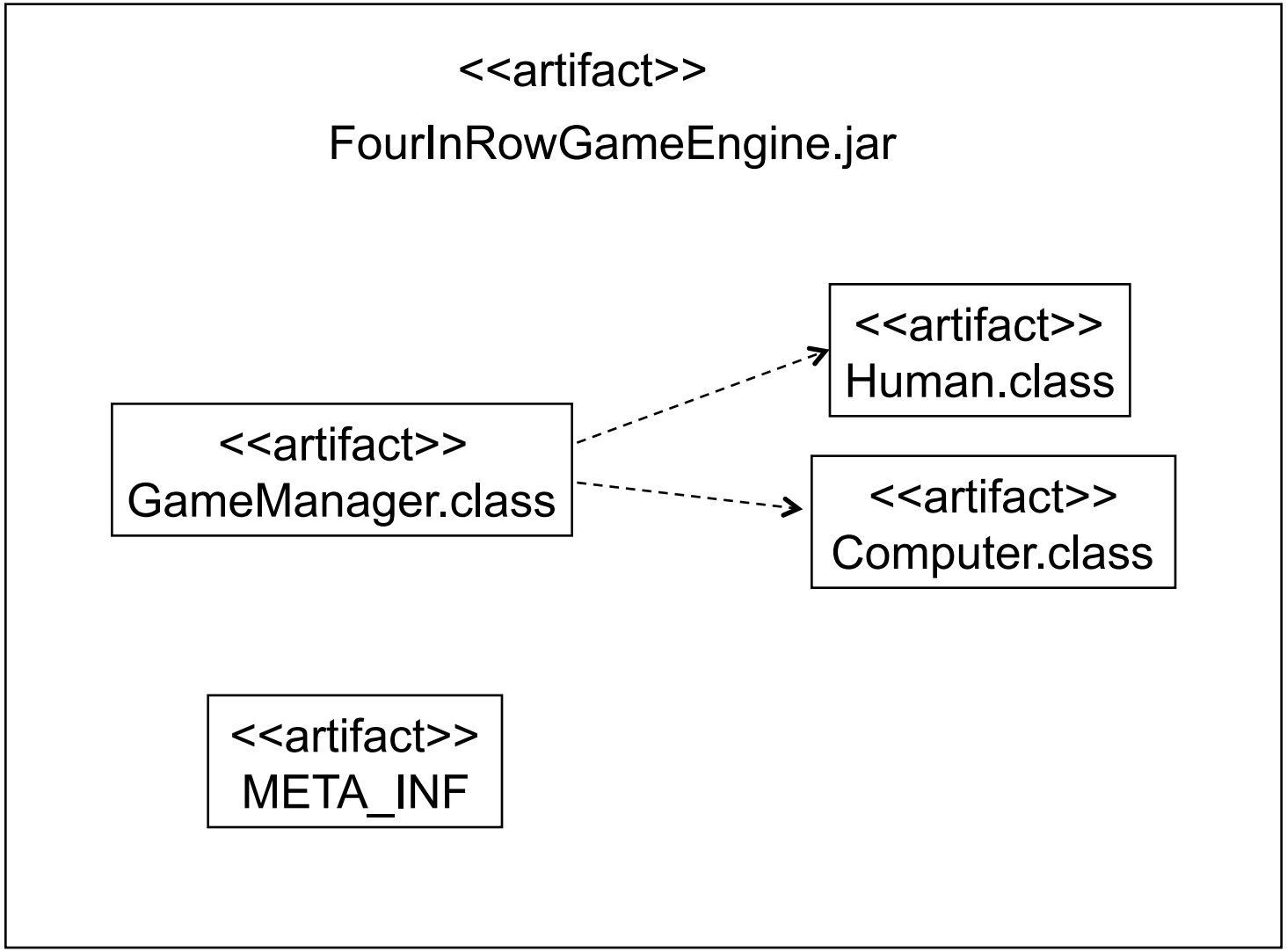
Example components



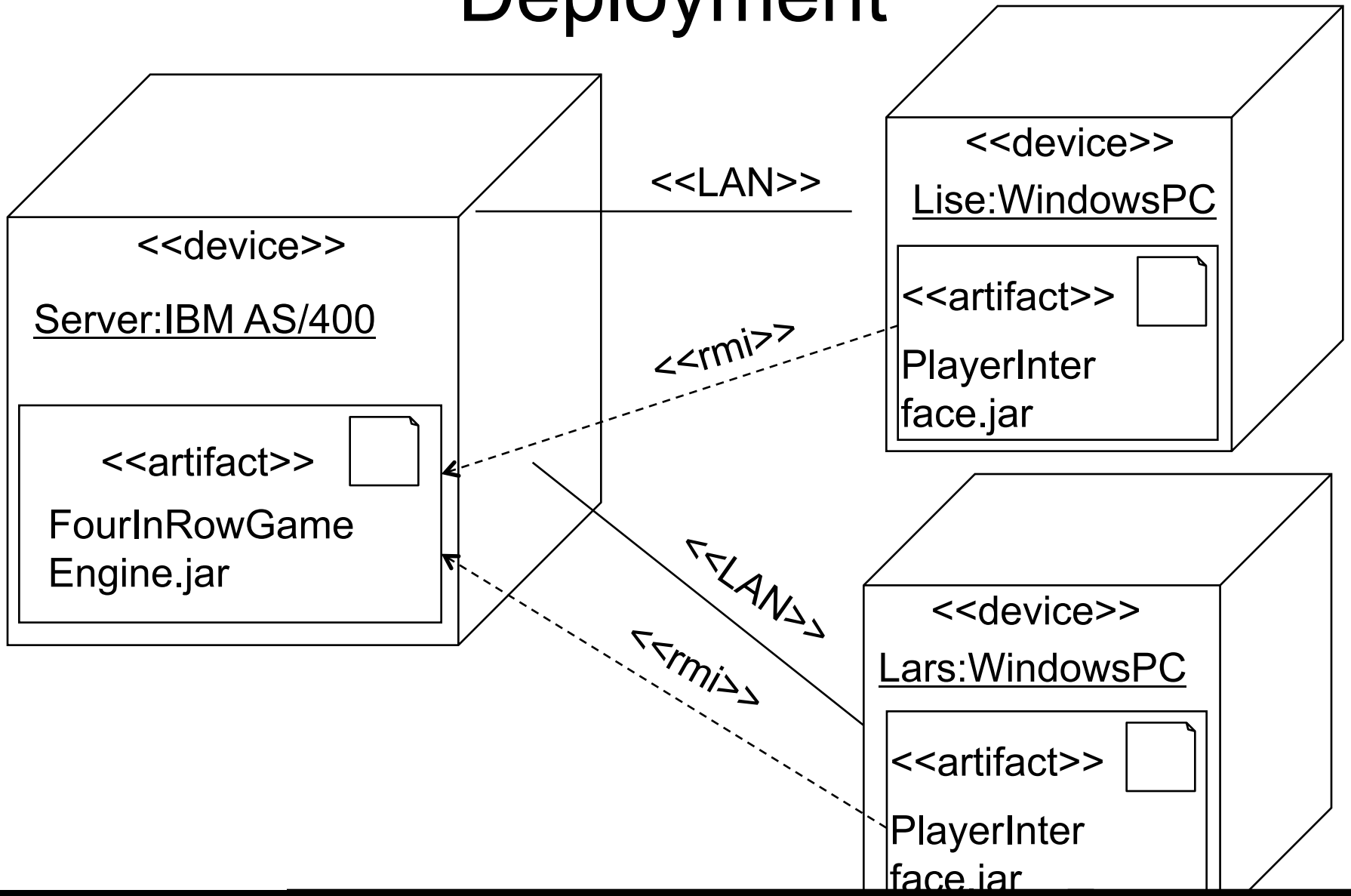
Artifact


<<artifact>>
FourInRowGameEngine.jar
<<manifest>>
<<component>>
GameEngine
<<artifact>>
PlayerInterface.jar
<<manifest>>
<<component>>
PlayerInterface

FourInRowGameEngine.jar



Deployment


<<device>>
Server:IBM AS/400
<<artifact>>
FourInRowGame
Engine.jar
<<LAN>>
<<device>>
Lise:WindowsPC
<<artifact>>
PlayerInter
face.jar
<<rmi>>
<<LAN>>
<<device>>
Lars:WindowsPC
<<artifact>>
PlayerInter
face.jar
<<rmi>>

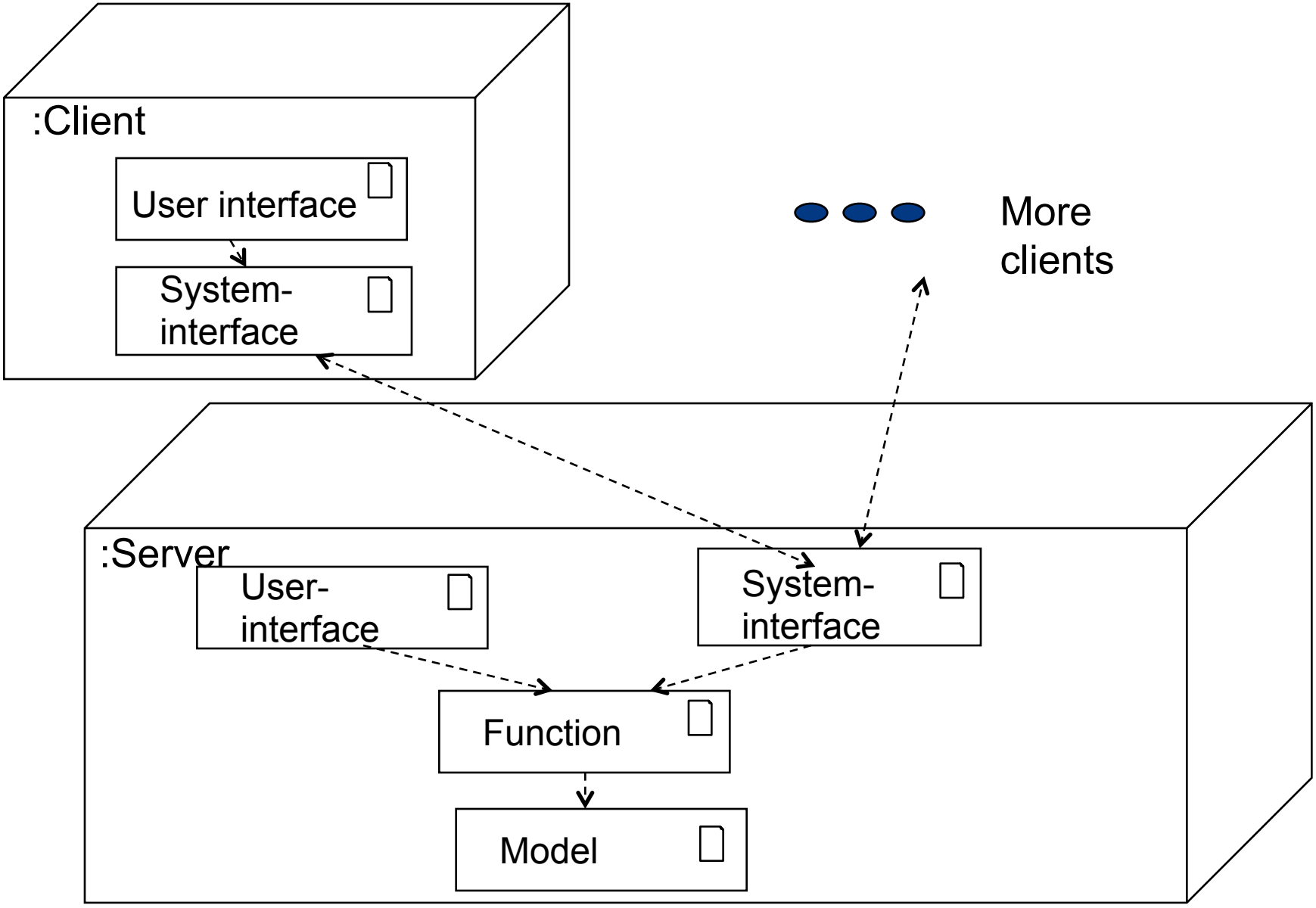
Artifact

- Artifact are deployed on nodes. Some examples of artifacts are:
  - Scripts
  - Source files
  - Database tables
  - Documents
  - Components

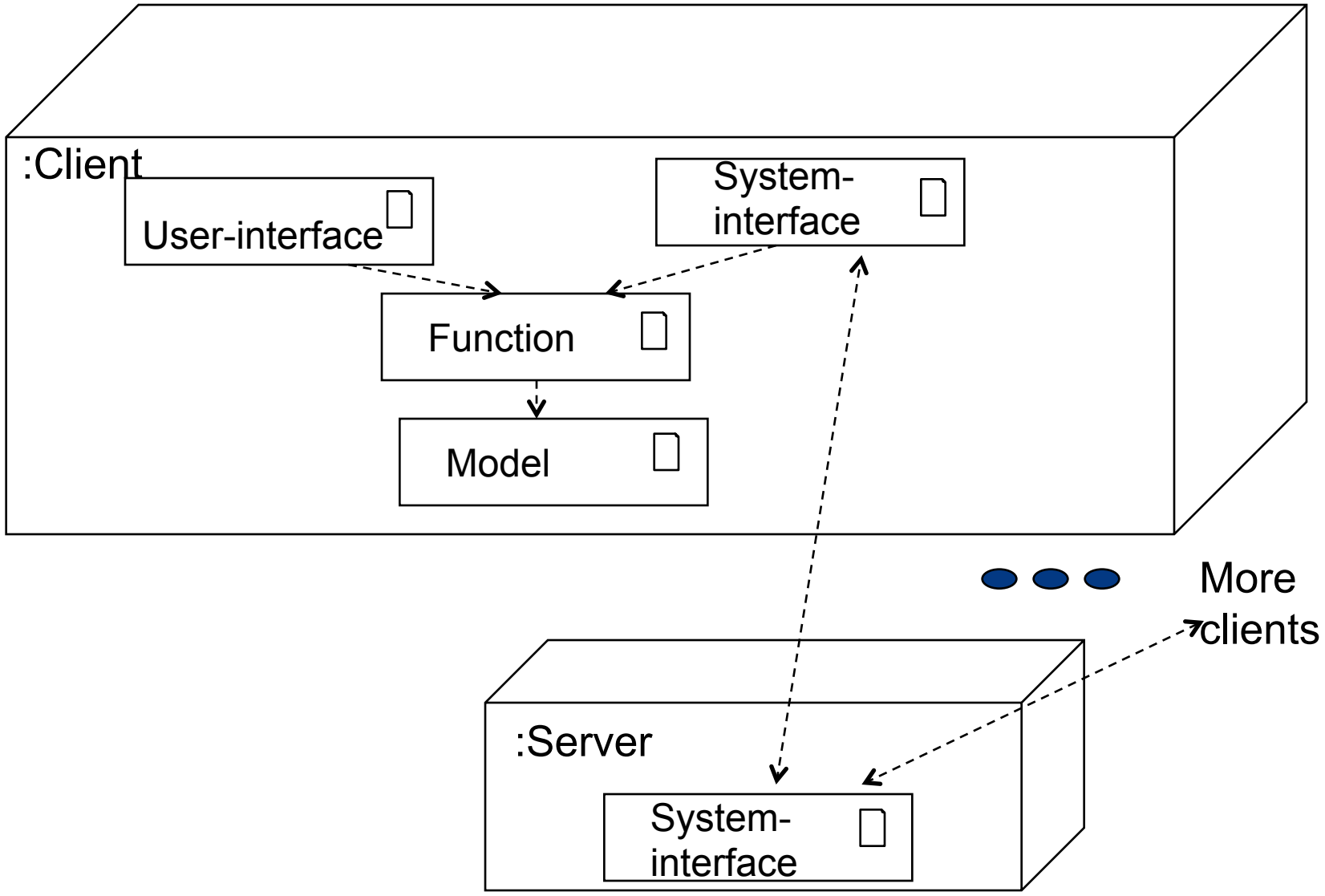
Patterns for splitting up the work on hardware

- The centralised pattern
- The distributed pattern
- The decentralised pattern

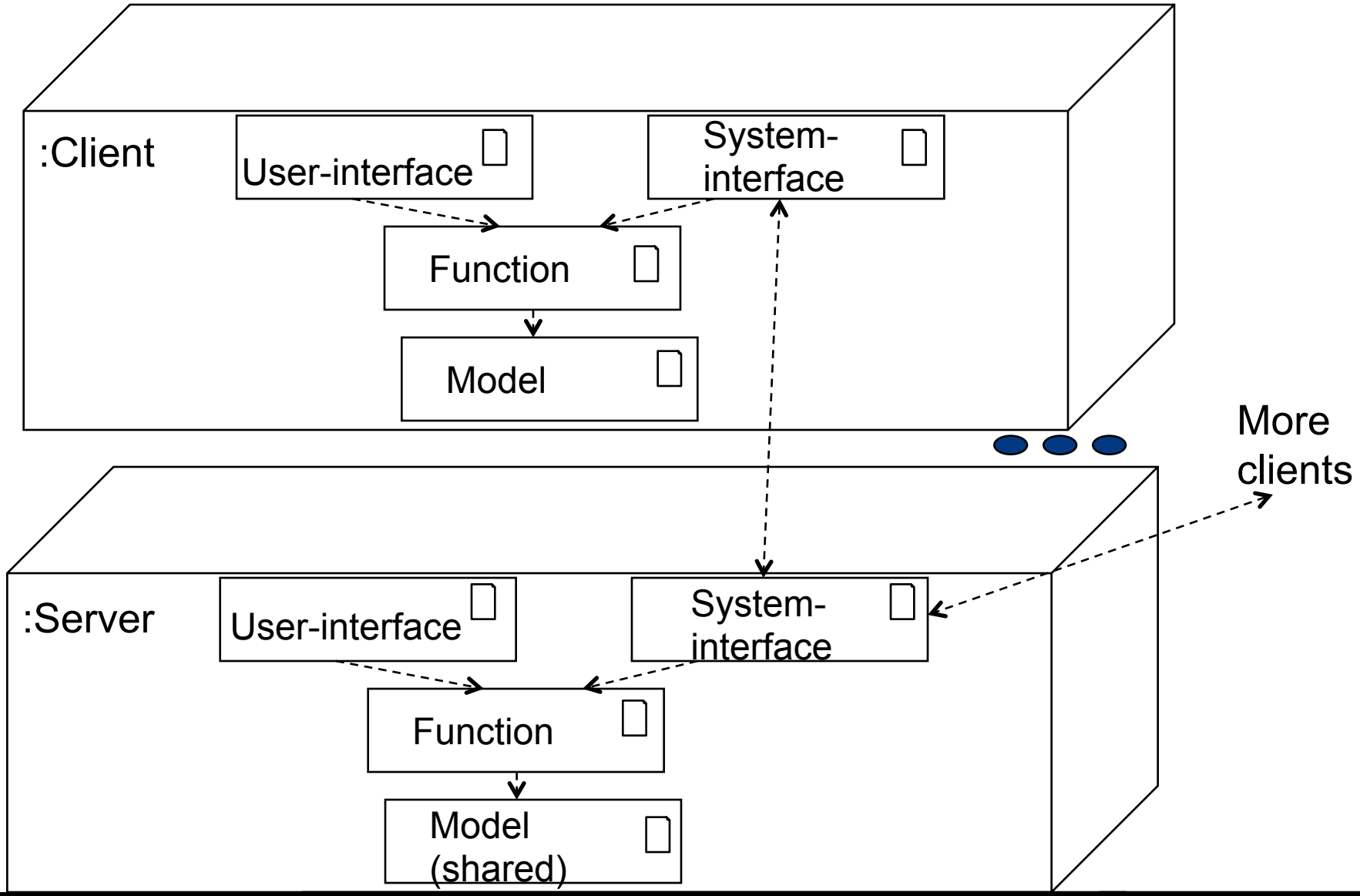
The centralised pattern



The distributed pattern



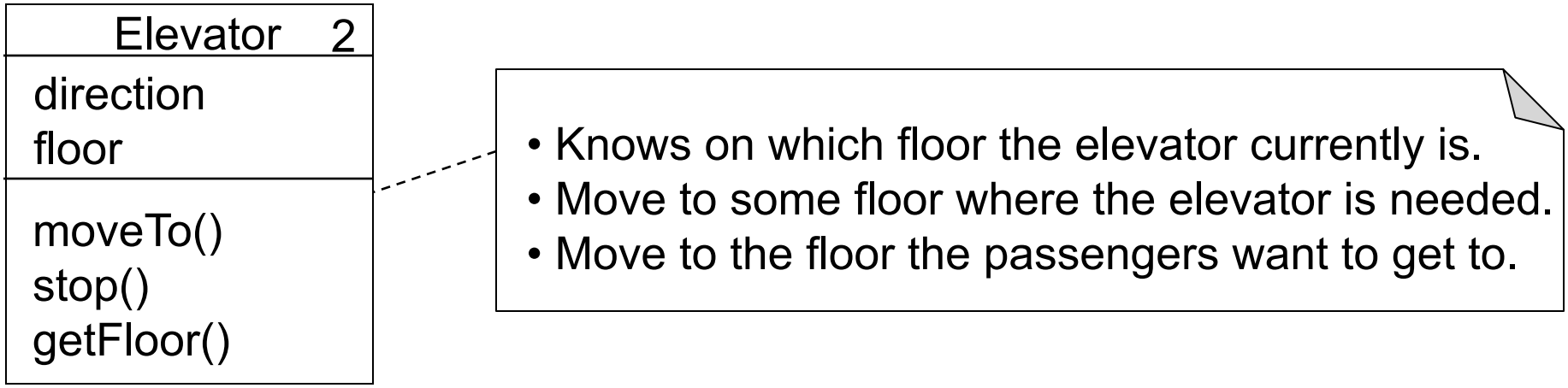
The decentralised pattern



Common terms

- **Comment:** comment to an element, e.g., specification in natural language or a constraint
- **Stereotype:** a keyword that can be added to a UML element to alter its meaning, e.g., <<interface>>
- **Tagged value**: additional properties for a UML element, e.g., an author-tag for a class
- **Constraint**: restriction of the usage of a UML element. Here, we consider constraints written in the formal language OCL

Comment



- Too many notes tend to make a diagram hard to read. In tools, there are often also other ways to attach information to UML elements.

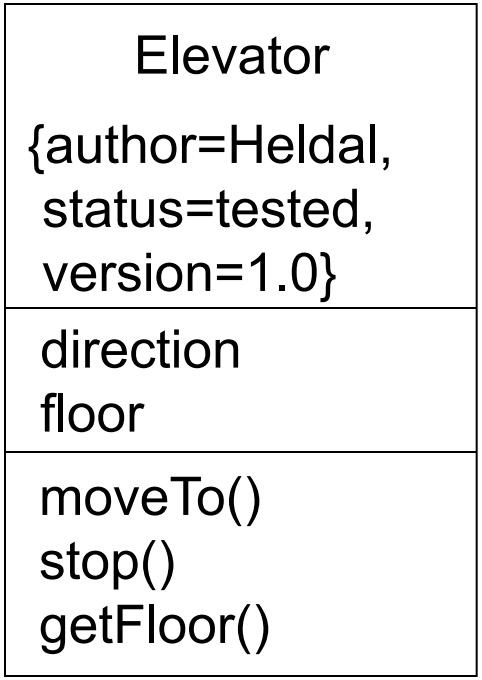
UML extension

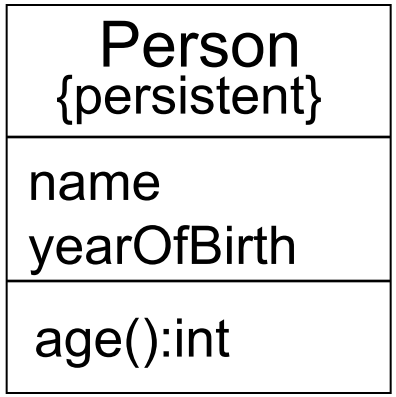
- Often UML as it is given is not powerful or precise enough to express things.
- Two ways of extending UML:
  - Change the meta-model
  - Using the standard UML extension mechanism
    - This is the recommended way

Profiles

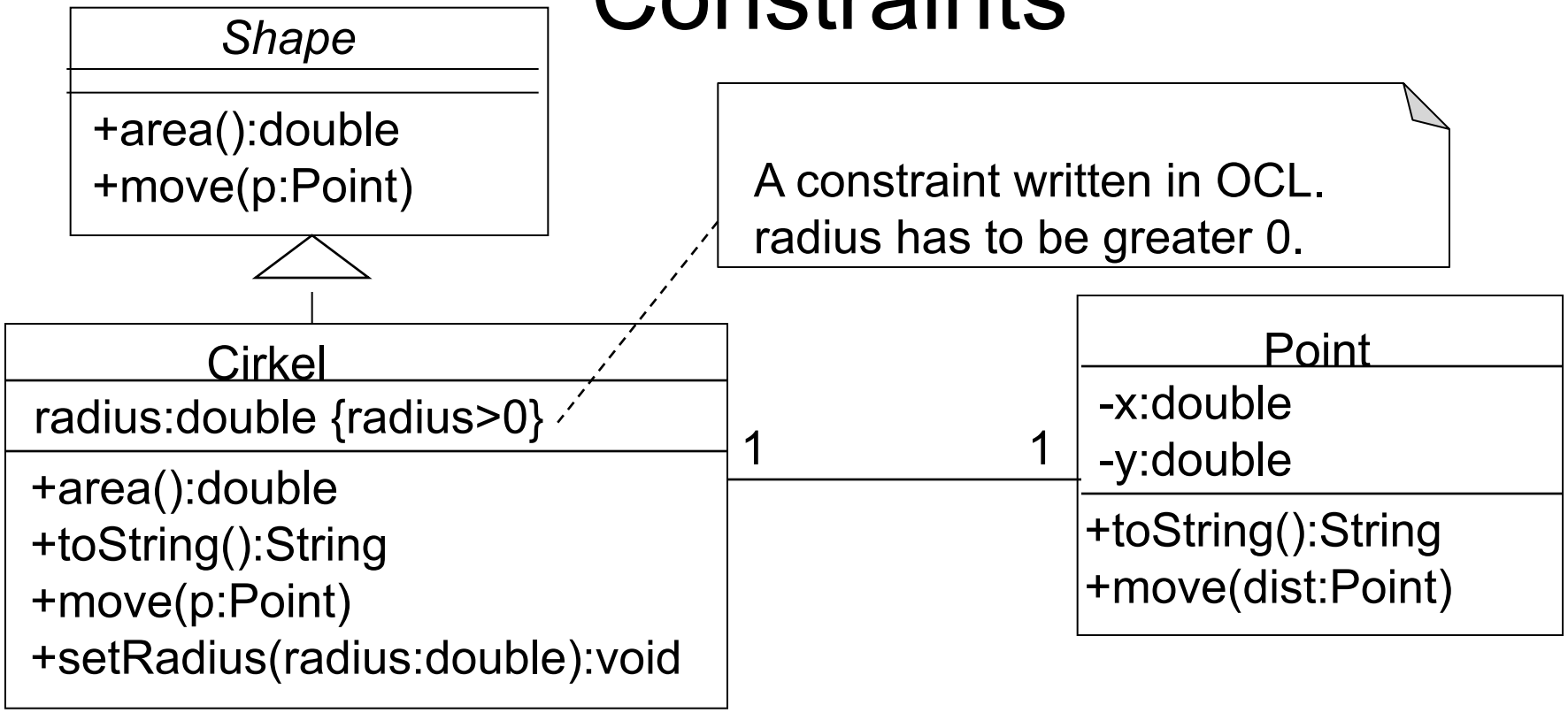
- One can make own profiles or use already existing profiles, e.g.
  - Profile for realtime
  - Profile for testing

Tagged value





Constraints



Constraints can be written in natural language or using a Formal language like OCL. Advantage of a formal language is that there are no ambiguities.