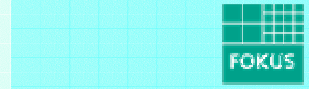


Fraunhofer Institute for Open Communication Systems

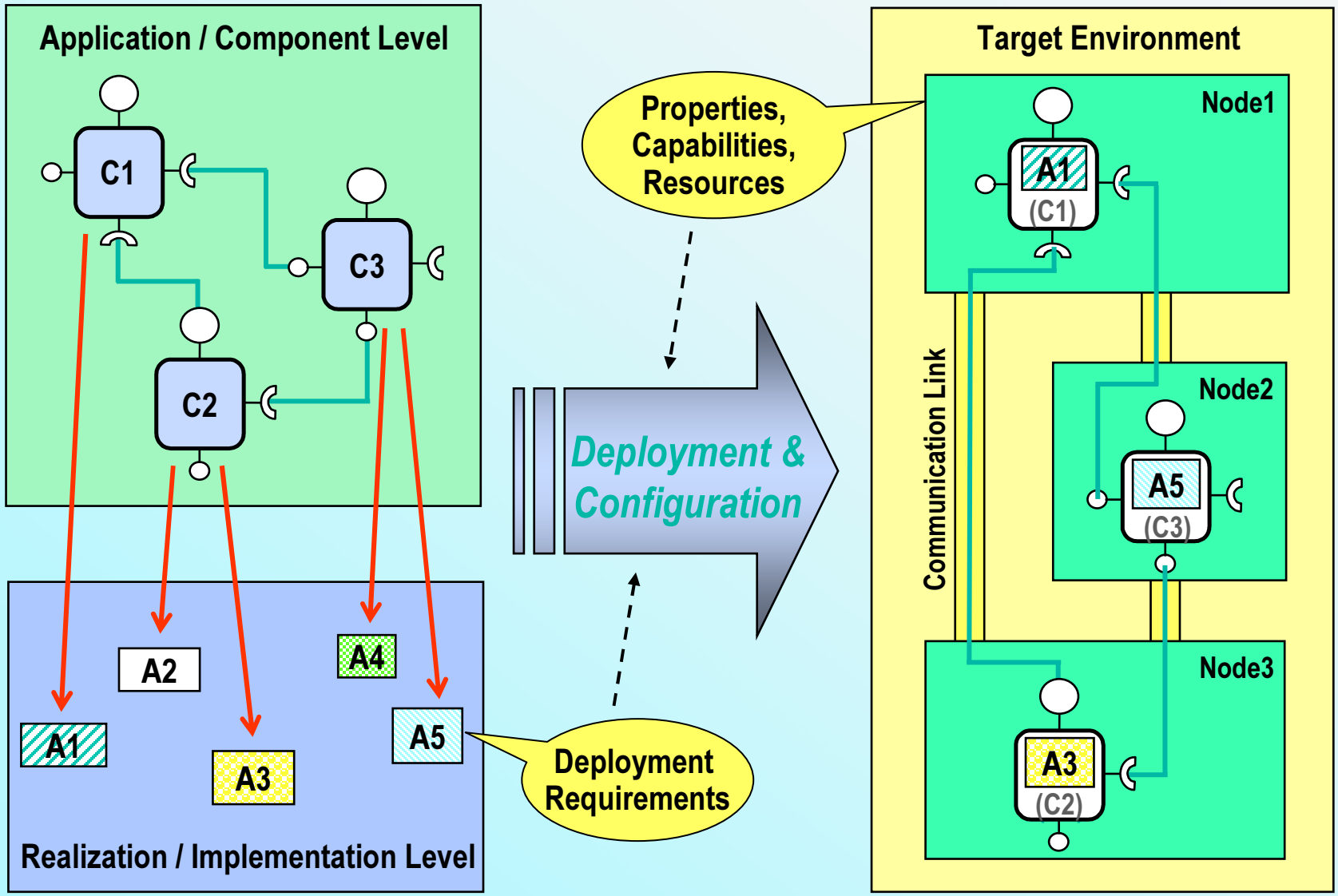
HUMBOLDT-UNIVERSITÄT ZU BERLIN

Deployment *and* Configuration *of Distributed Systems*

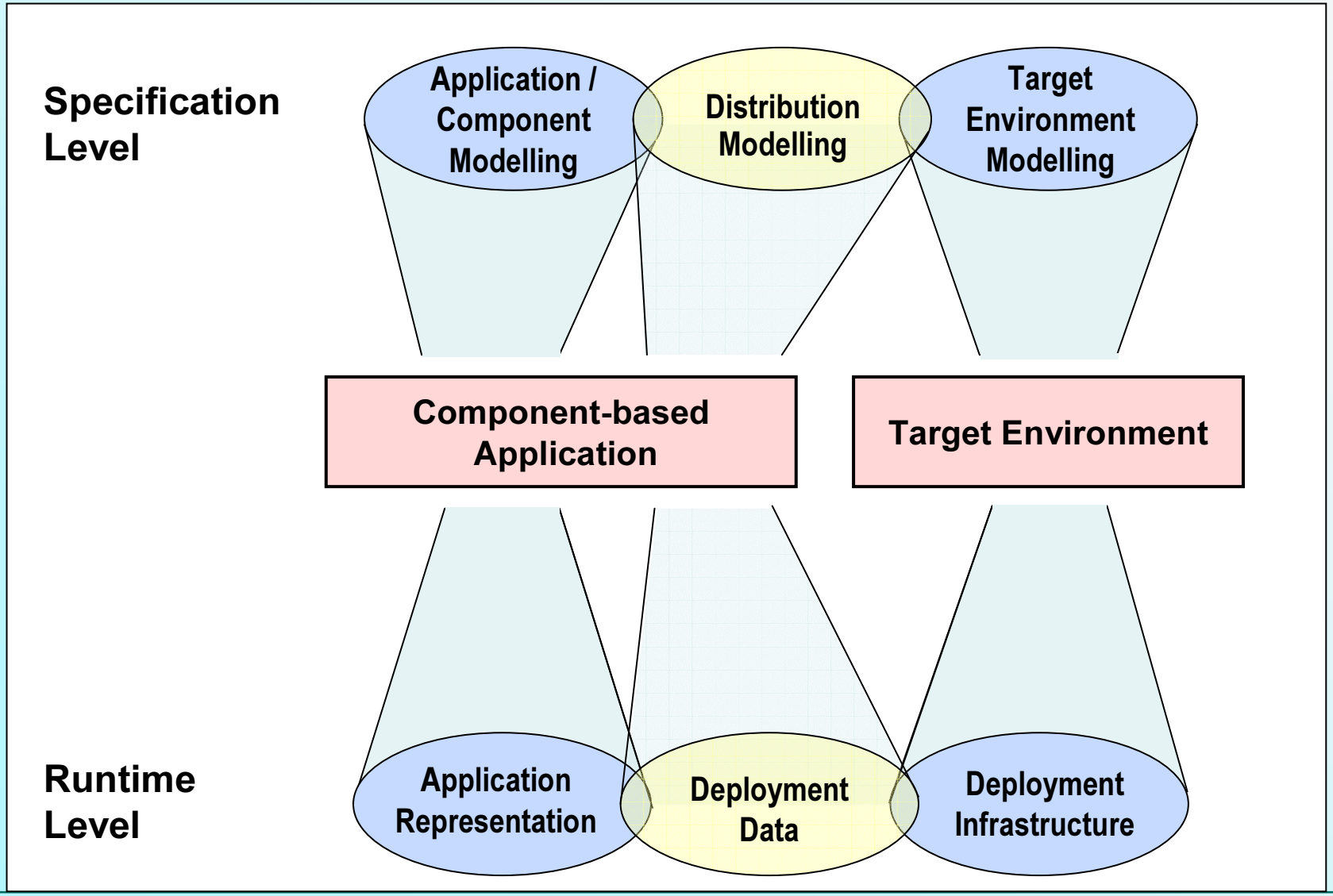
Andreas Hoffmann a.hoffmann@fokus.fraunhofer.de
Bertram Neubauer neubauer@informatik.hu-berlin.de

Table of Contents

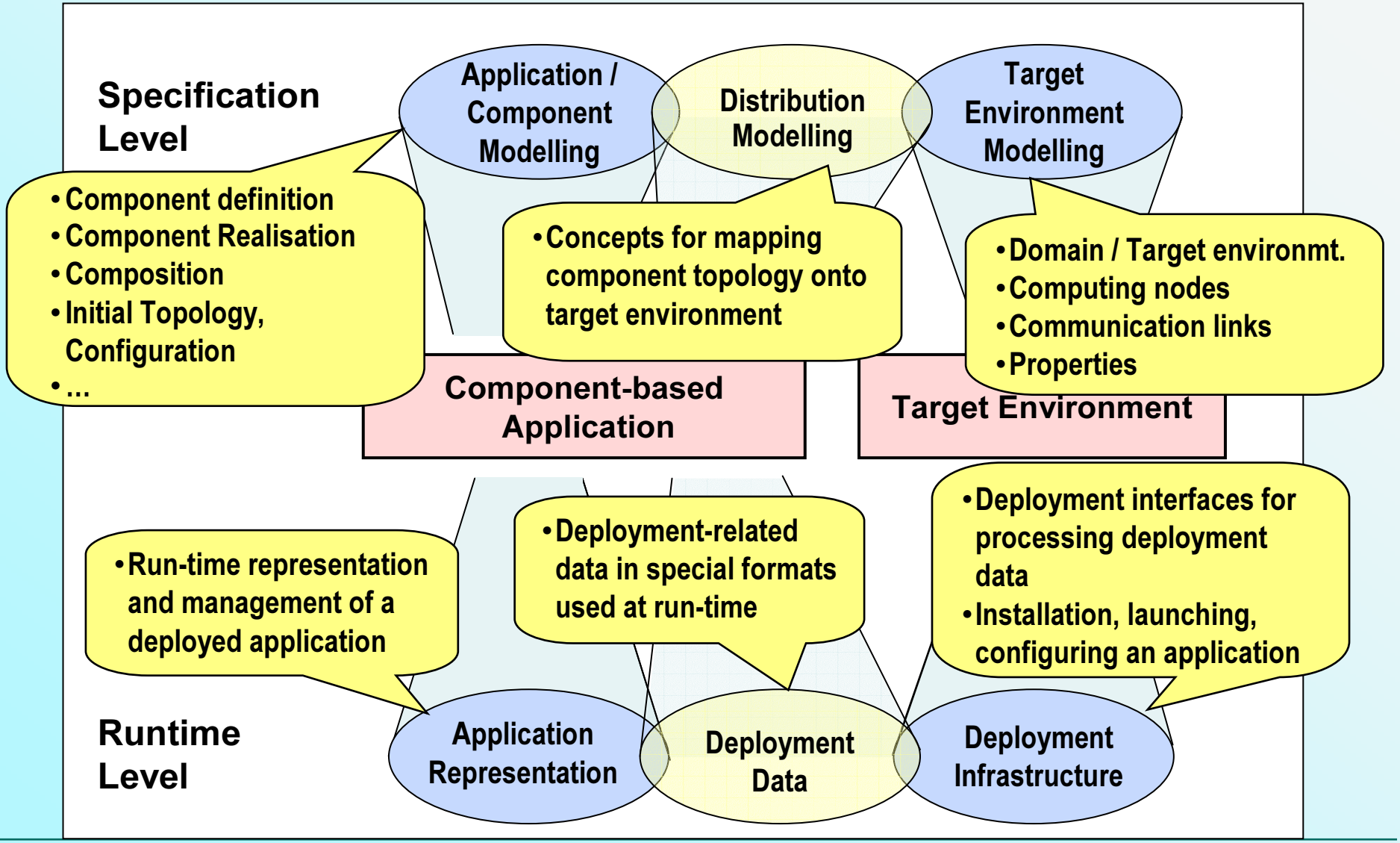
Introduction: Deployment & Configuration



Overview on Deployment Modelling



Overview on Deployment Modelling



Languages Investigated

- Only languages
  - suitable for modeling distributed systems at *platform-independent* (PIM) level
  - according to OMG's Model Driven Architecture (MDA) approach

  have been considered
- *Platform-specific* technologies such as CCM, EJB, .NET have not been considered
  - In general, mapping from PIM to different PSMs possible
- ***Investigation of the following languages***
  - extended Object Definition Language (ITU eODL)
  - Unified Modeling Language (UML 2.0)
  - OMG Deployment and Configuration (OMG DnC)
- All three languages have a MOF-based Metamodel

Overview on eODL

- Defined by ITU-T
- Language for platform-independent specifications
- Different viewpoints reflecting different aspects of a distributed system
  - *Computational VP*:
    - Modeling black-box components; no recursive composition
    - 3 types of ports
  - *Implementation VP*:
    - Component realisations
    - Deployment requirements
  - *Target Environment VP*: Modeling computing environment
  - *Deployment VP*: Deployment mapping
- No support for run-time modelling

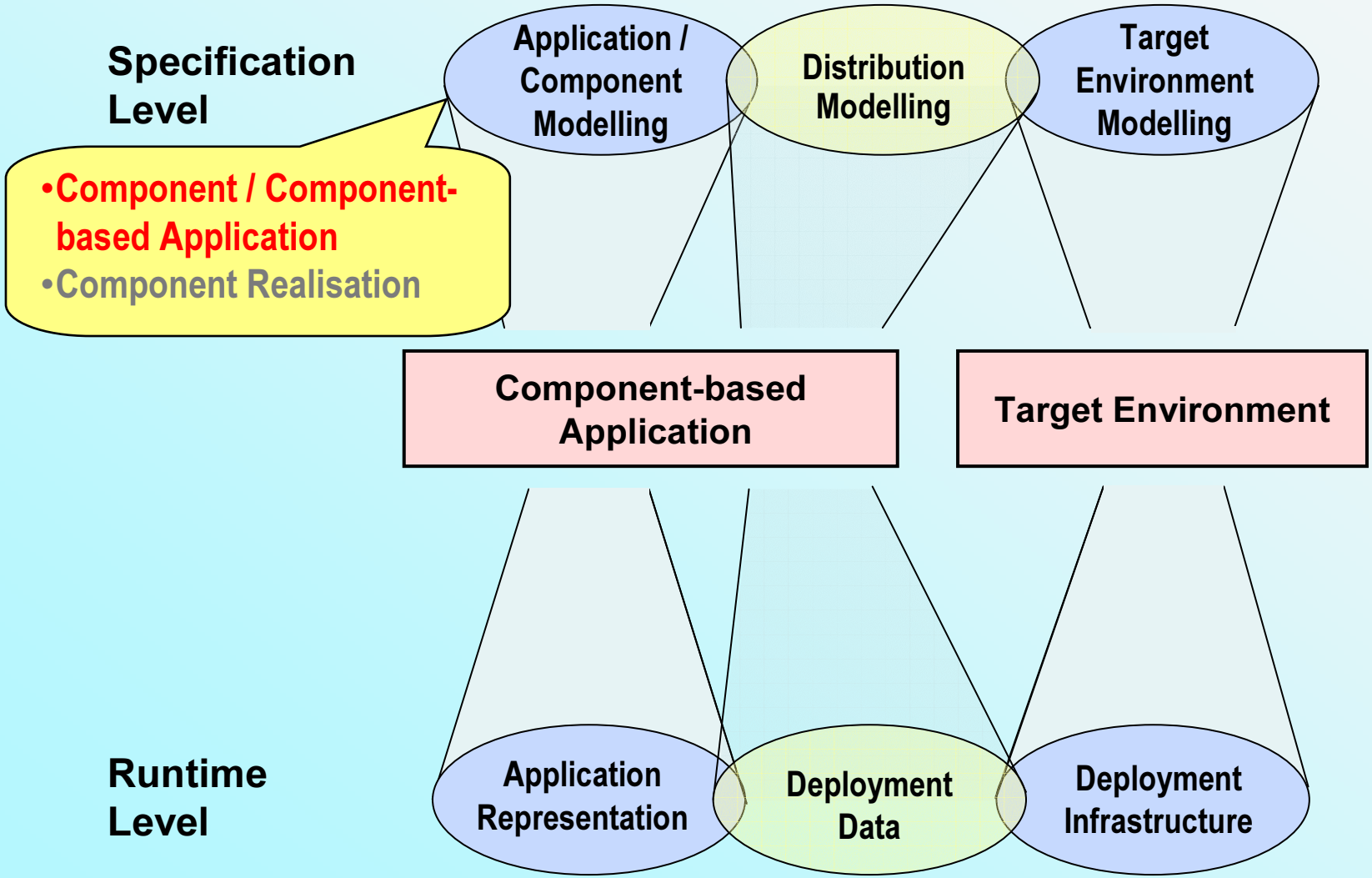
Overview on UML 2.0

- Standardised by OMG
- Language (family) for graphical modelling of arbitrary systems
- Recently adopted version UML 2.0 is major revision adding advanced modelling concepts,
  in particular for distributed systems
  - Component modelling:
    - Provided and used ports
    - Black-box and white-box components
  - Component realisation (implementation) by artifacts
  - Deployment concepts for
    - Target environment
    - Deployment mapping
- No support for run-time modelling, too

Overview on OMG DnC

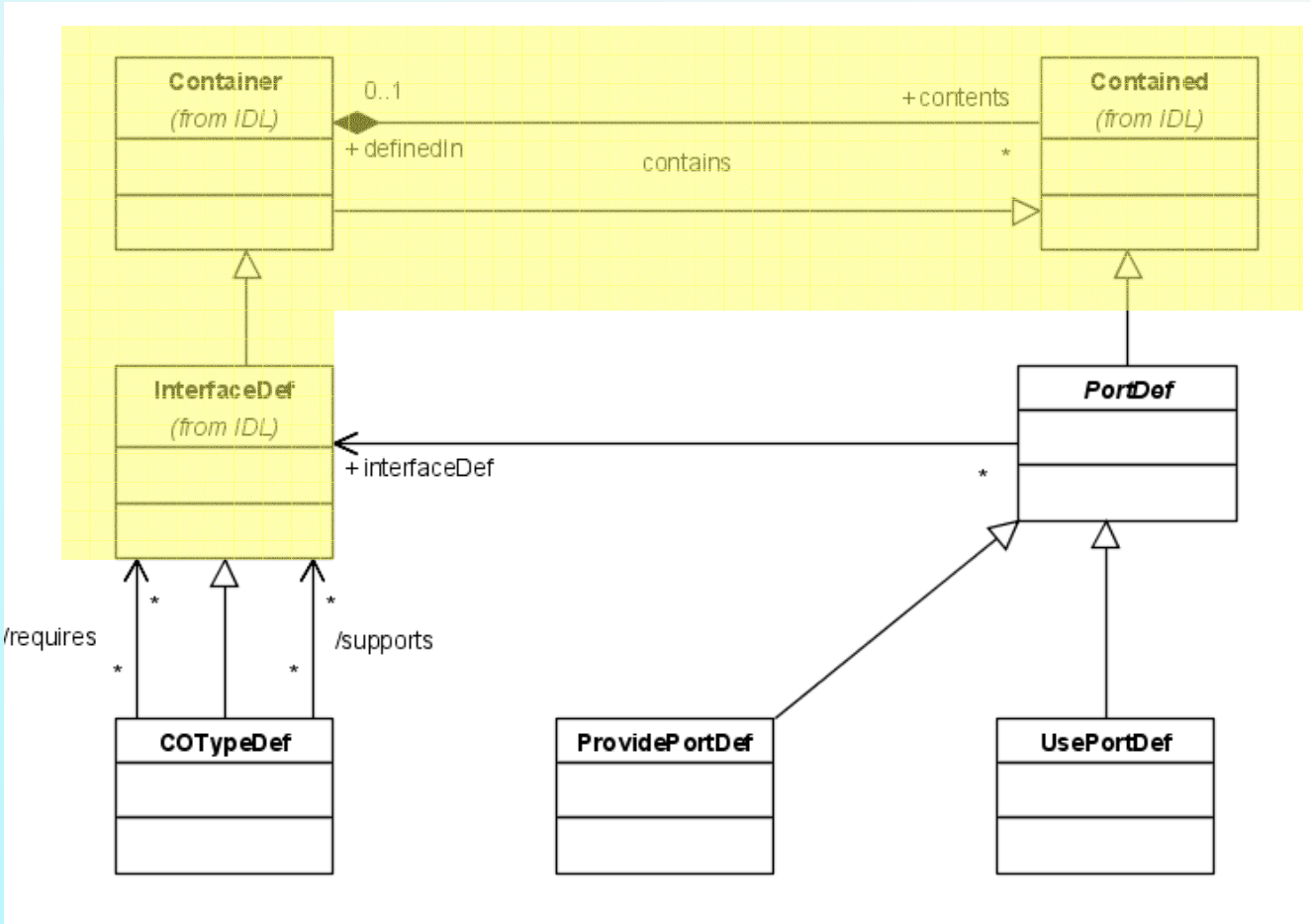
- In 2003 the OMG adopted the *Deployment and Configuration of Component-based Applications* specification
- Structured according the MDA approach
  - Core: MOF-compliant platform-independent model (PIM) for DnC
    - Basically aligned with respective UML 2.0 concepts (some open issues)
  - UML profile
    - For modelling components as well as the target environment
    - Is a concrete syntax for the abstract concepts defined by the PIM
  - Platform-specific model for the CORBA Component Model (CCM)
- Development phase out of scope of DnC
  - Starting point: complete (implemented) specification
  - DnC spec is applicable to wide range of different component-based methodologies
- Focus: Interoperable deployment machinery
  - Deployment architecture with well-defined interfaces and interchange formats
  - Both can be derived automatically from the PIM by proper mapping rules

Application / Component Modelling



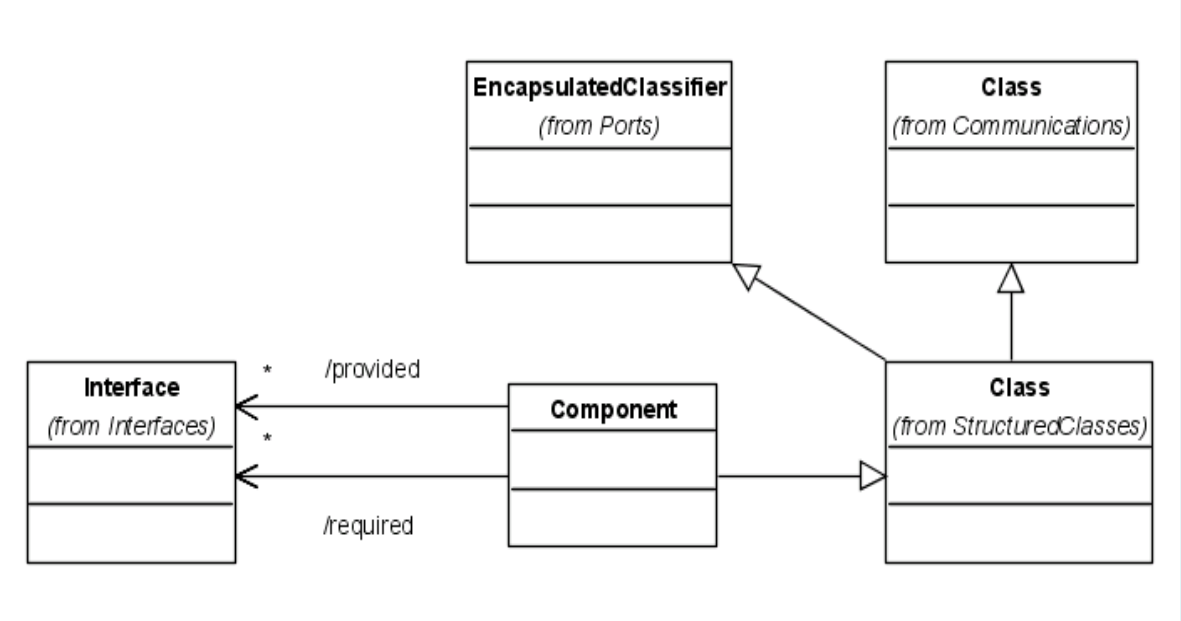
Component / Component-based Application in eODL

- General relationship between ***Container*** and ***Contained*** from IDL
- ***COTypeDef***
  - represents a *component*
  - is a Container potentially containing ports (***PortDef***)
- Ports may be used or provided
- An eODL component has no internal structure (black-box)
- ***AssemblyDef***
  - Assembly of components
  - Models an *application* or parts of it

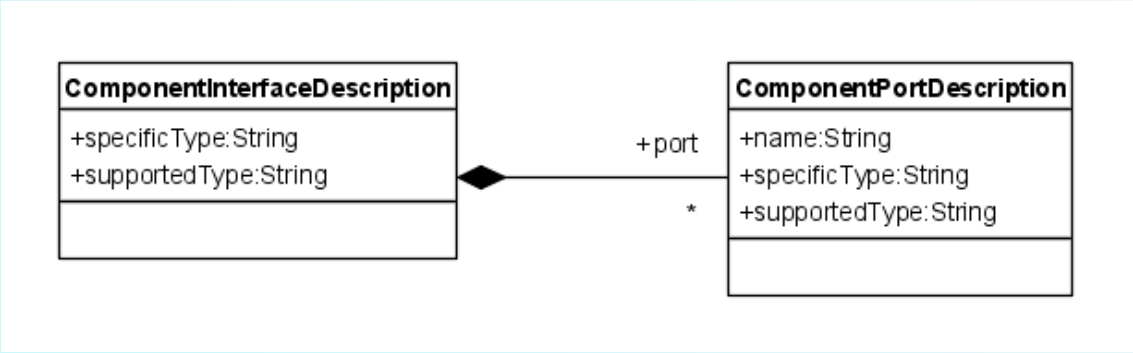


Component / Component-based Application in UML 2.0

- UML *Component* represented by the class ***Component***
- A ***Component***
  - may provide or require ***Interface***s
  - Ports and Connections inherited from ***EncapsulatedClassifier***
  - May be modelled as black-box or may have internal structure (white-box)
- A *component-based application* is modelled by a recursively decomposed component

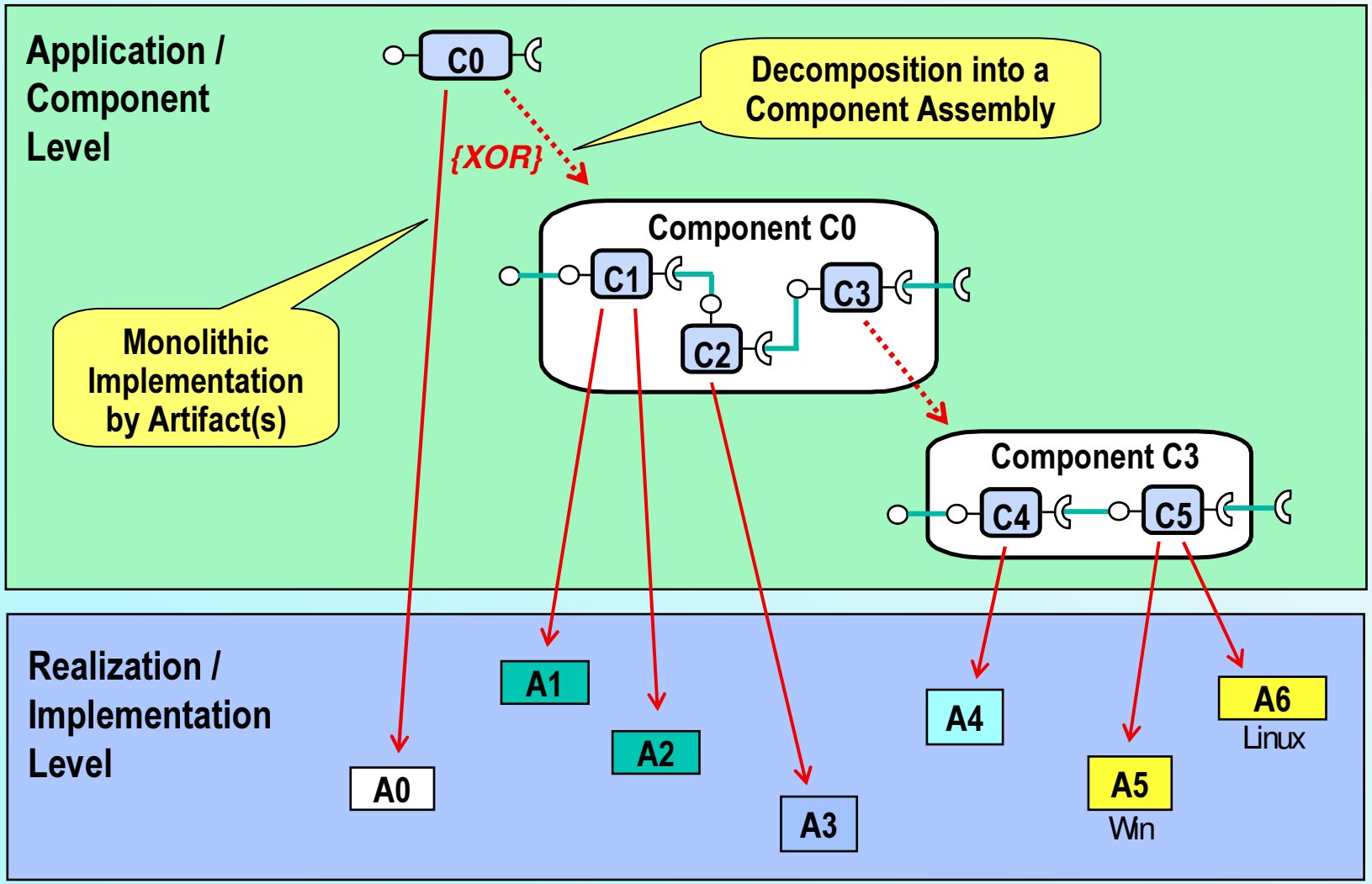


Component / Component-based Application in OMG DnC

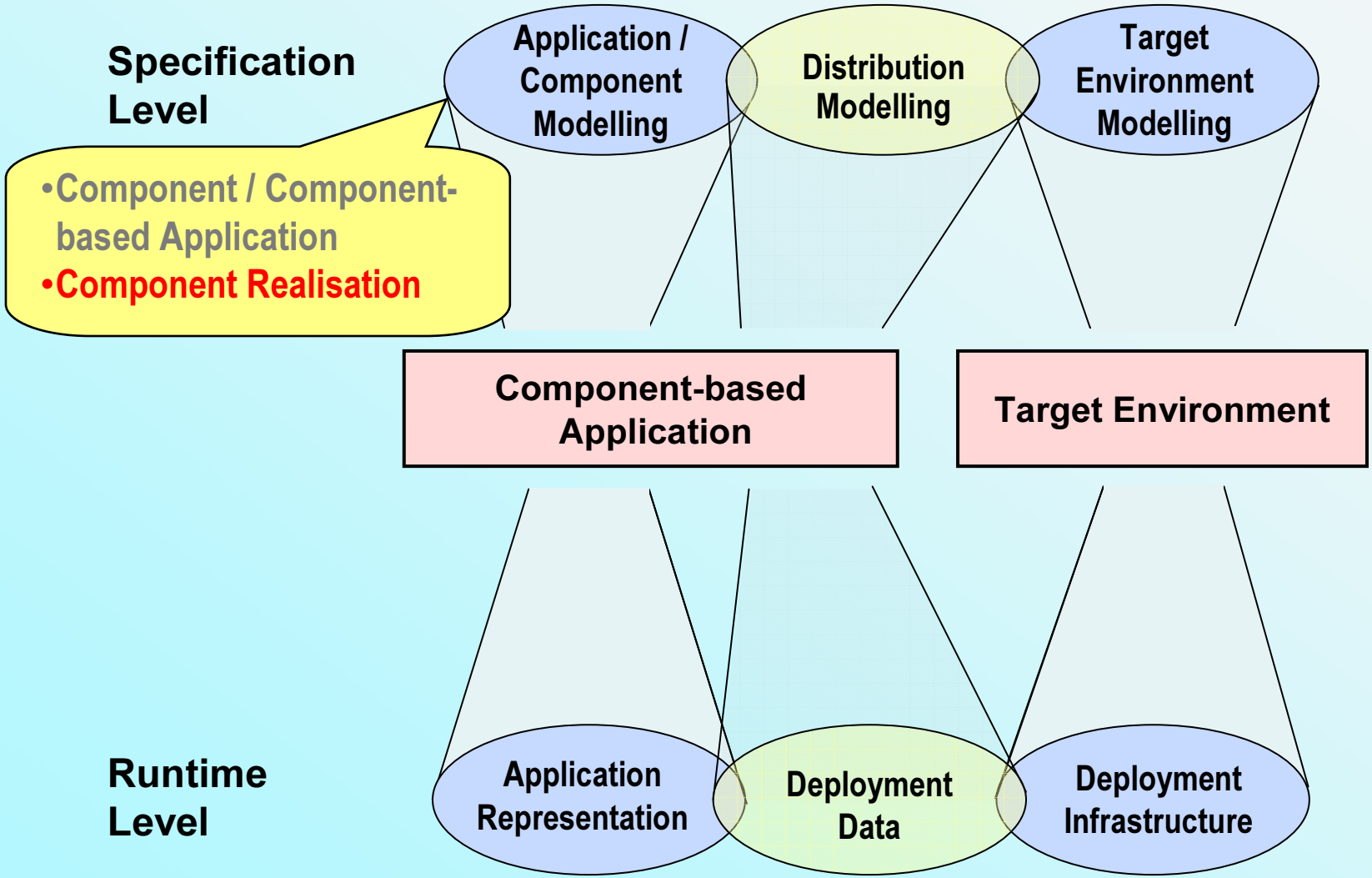


- DnC *Component*
  - Is represented by the class ***ComponentInterfaceDescription***
  - May have ports (ComponentPortDescription)
- Interfaces and types are simply referenced by identifiers of type string instead of associations
  - Reason: DnC may be applicated to wide range of component models
- A DnC Component is a black-box
  … but white-box view of components is also supported by decomposition concept

Component / Component-based Application in OMG DnC



Application / Component Modelling



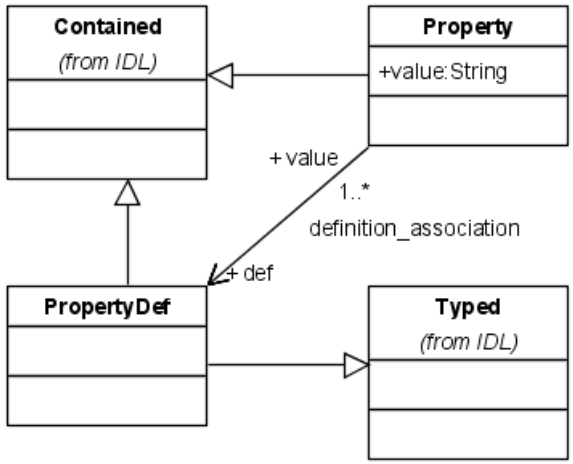
Component Realisation in eODL, UML 2.0 & OMG DnC

- Components have to be implemented before they can be deployed
- Model needs to reflect implementation-related information, such as name(s) of implementation file(s), their depedencies and requirements
- In general, one or more components are implemented by a software artifact or similar concept

- Unfortunately, different terminlology:
  - **eODL**: Components are realised by SoftwareComponentDef
  - **UML 2.0**: Components are manifested by an Artifact that represents an arbitrary file
  - **DnC**: Components are implemented by ComponentImplementation-Description which can be either a monolithic impementation of a component assembly
    Actual implementation files are modelled by ImplementationArtifact-Description

Properties / Requirements / Resources in eODL, UML 2.0 & OMG DnC

eODL

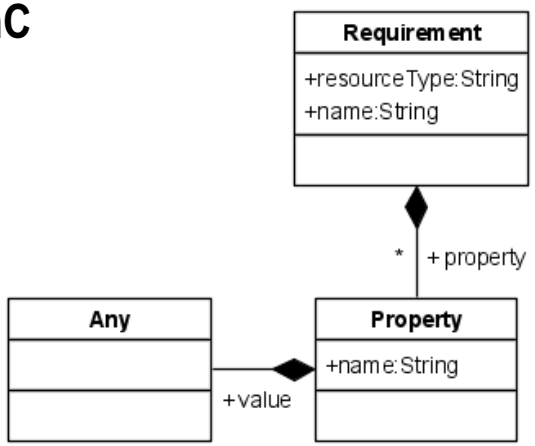


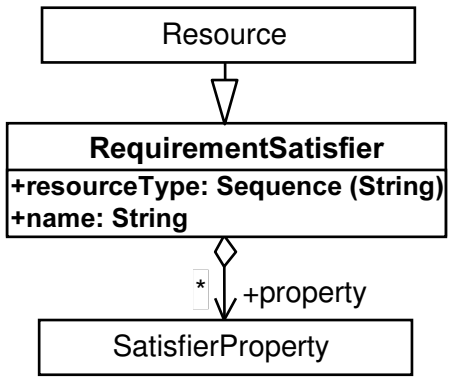
UML 2.0

- No dedicated concept available in UML 2.0
- General purpose annotation mechanism may be used instead
- example:

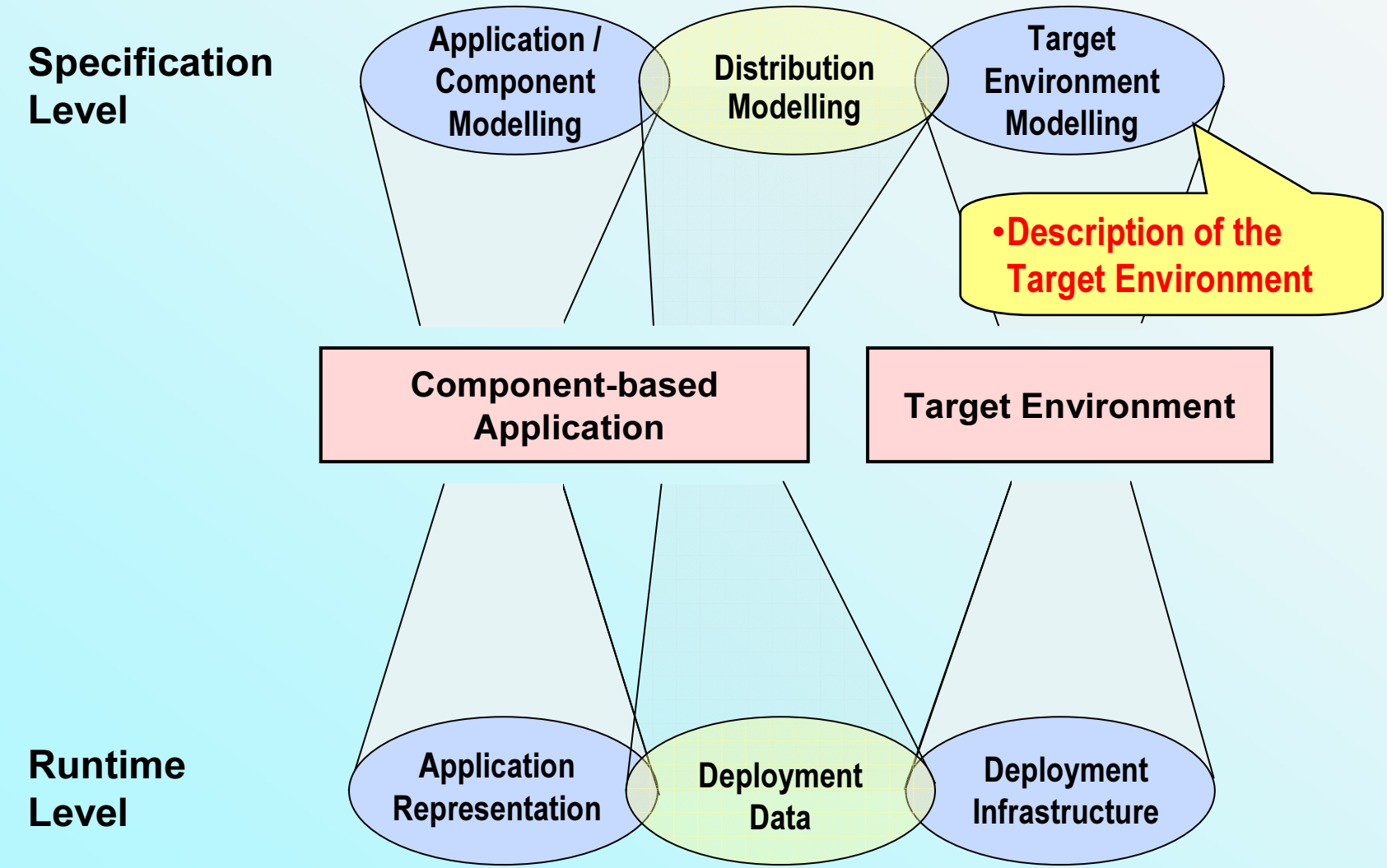
{ os = windows,
executionenvironment = java }

DnC

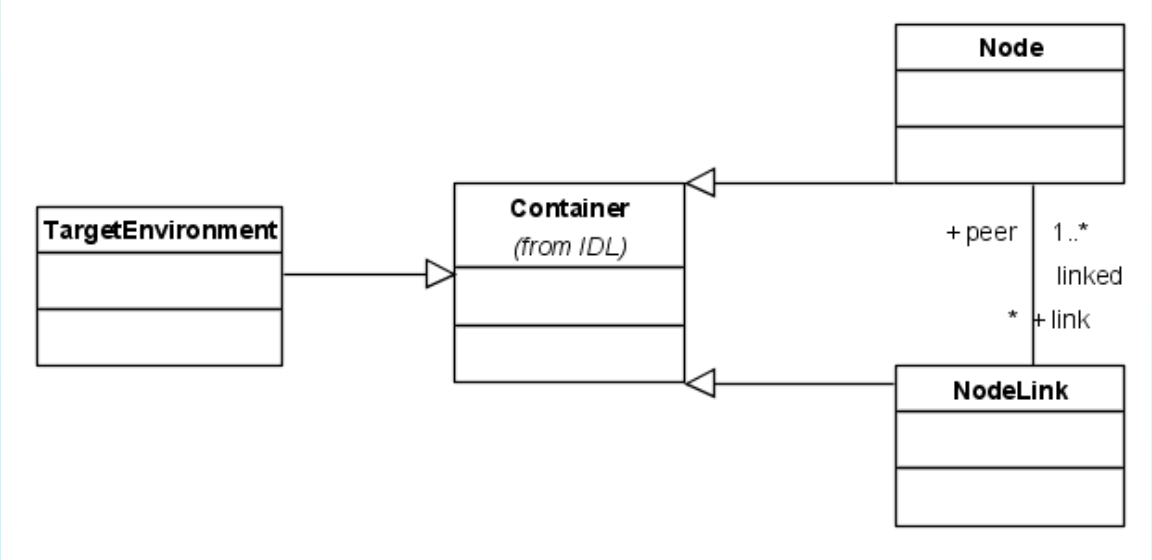




Target Environment Modelling

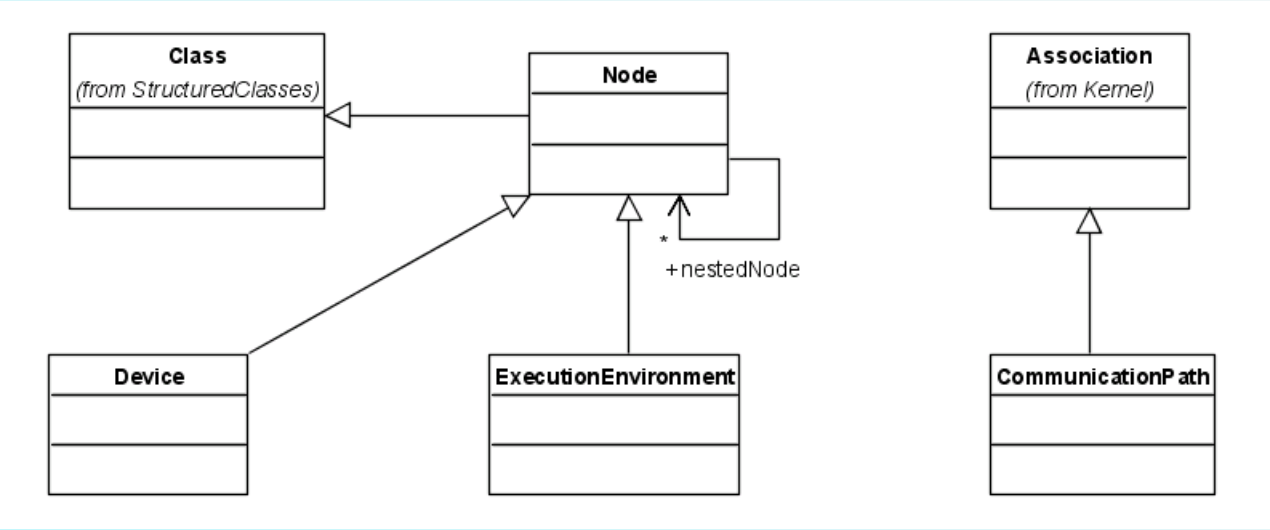


Target Environment Modelling in eODL



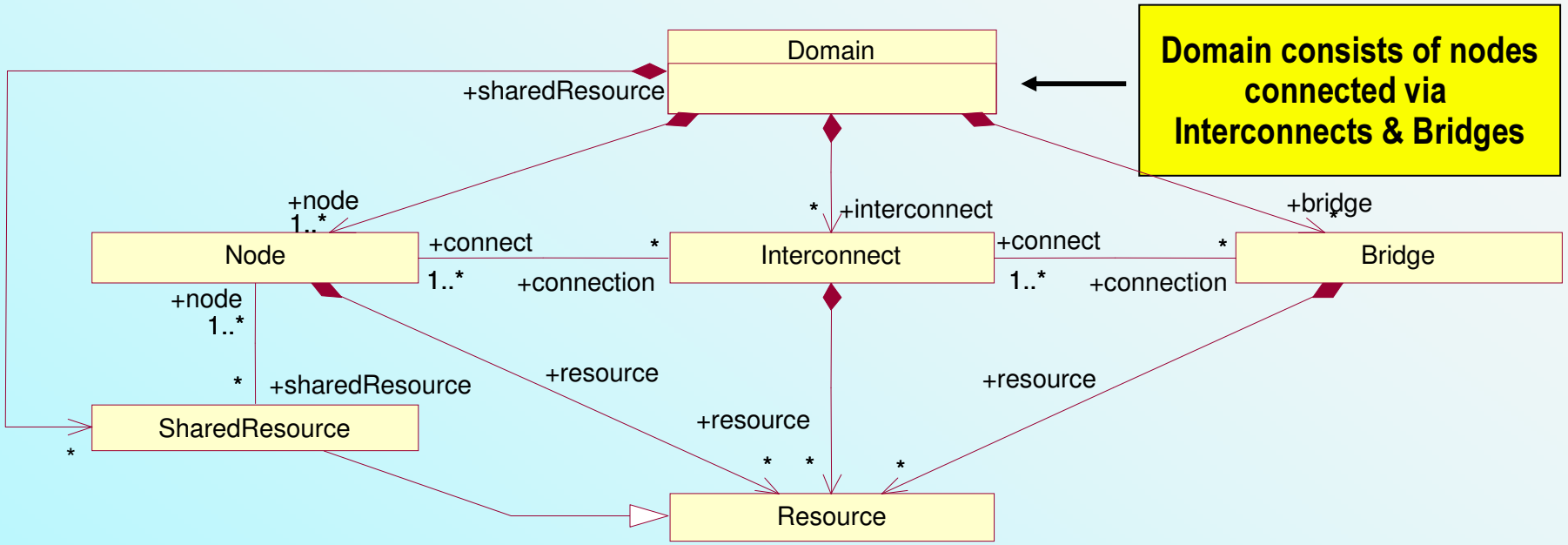
- Simple model of target environment
- A ***TargetEnvironment*** is made up from ***Node***s and ***NodeLink***s connecting nodes
- Nodes and NodeLinks may have typed property values attached
  - Achieved through inheritance

Target Environment Modelling in UML 2.0



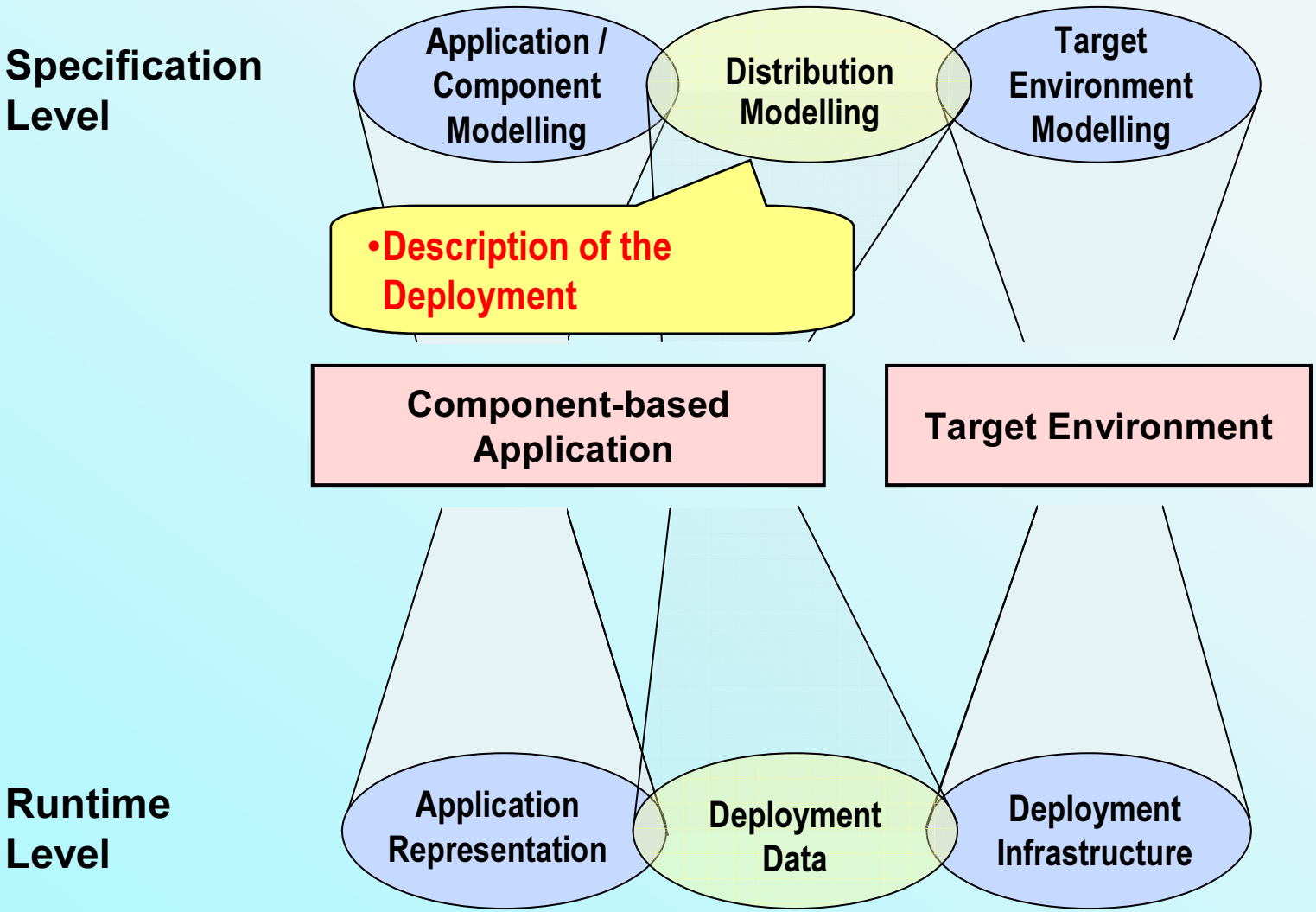
- No top-level element (e.g. target environment)
- ***Node***s have processing capabilities and
  - may be nested
  - Further concepts for substructuring: ***Device*** and ***ExecutionEnvironment***
- ***Node***s are connected by ***CommunicationPath***s
  - Achieved by inheritance
- No special mechanism for attaching properties to ***Node***s etc.
  - Only untyped (!) name-value pairs may be attached using the general-purpose annotation mechanism

Target Environment Modelling in OMG DnC

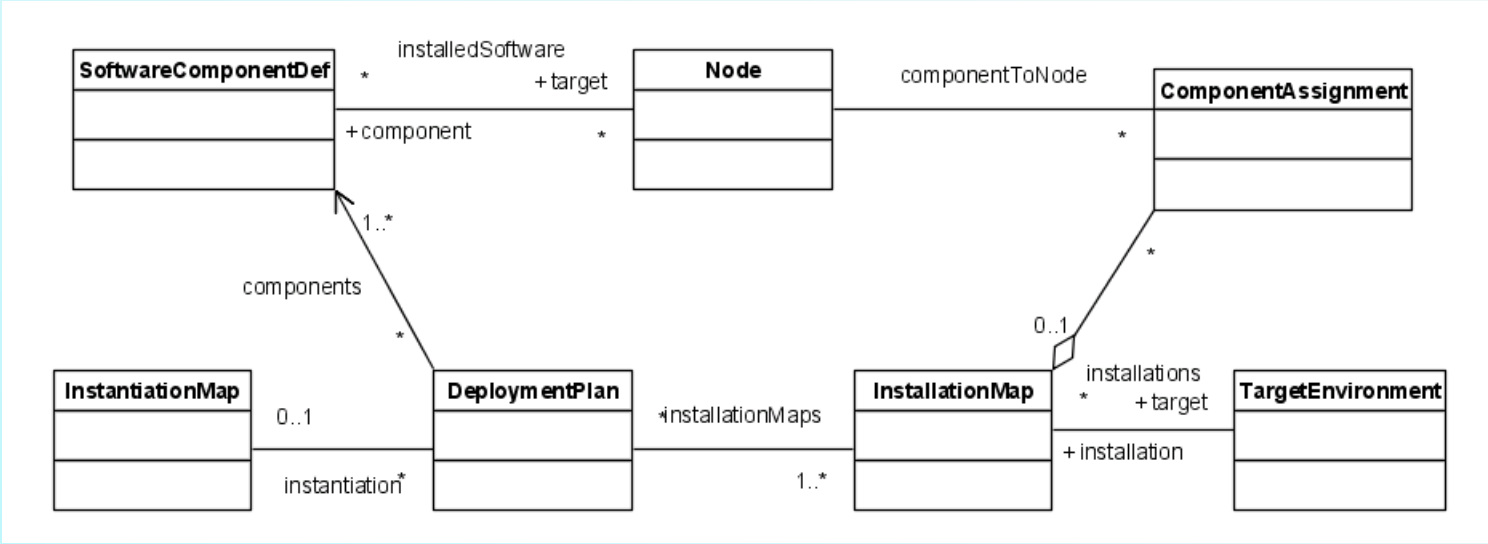


- ***Interconnect***s provide shared communication path between nodes
- ***Bridge***s connect interconnects and model routers and switches
- ***Node***, ***Interconnect*** and ***Bridge*** have resources (with special types)
  - Nodes: e.g. processors, hardware devices, memory, operating system
  - Interconnect: bandwidth, protocol
- Resources may be shared among Nodes
- Nodes are target for execution, Interconnects & Bridges are target for connections

Distribution Modelling

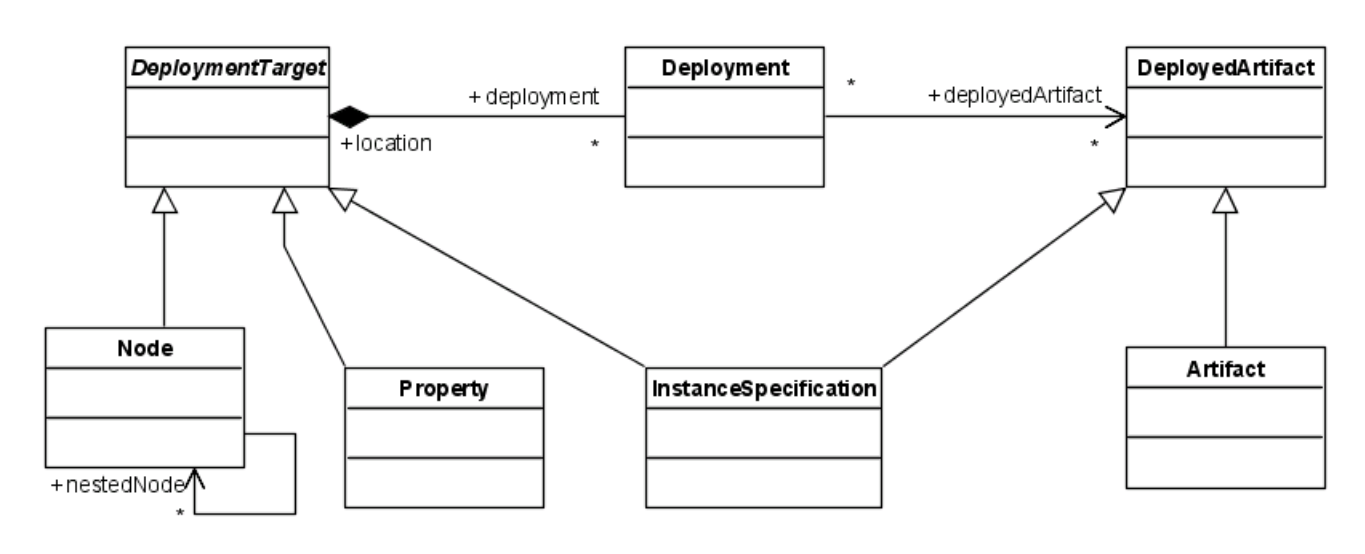


Distribution Modelling in eODL



- ***DeploymentPlan***
  - Mapping of an application onto a particular target environment
  - Consists of ***InstallationMap*** and ***InstantiationMap***
- ***InstallationMap*** specifies what components are to be installed at what node
- ***InstantiationMap*** specifies what/how many instances of components are to be instantiated at what node

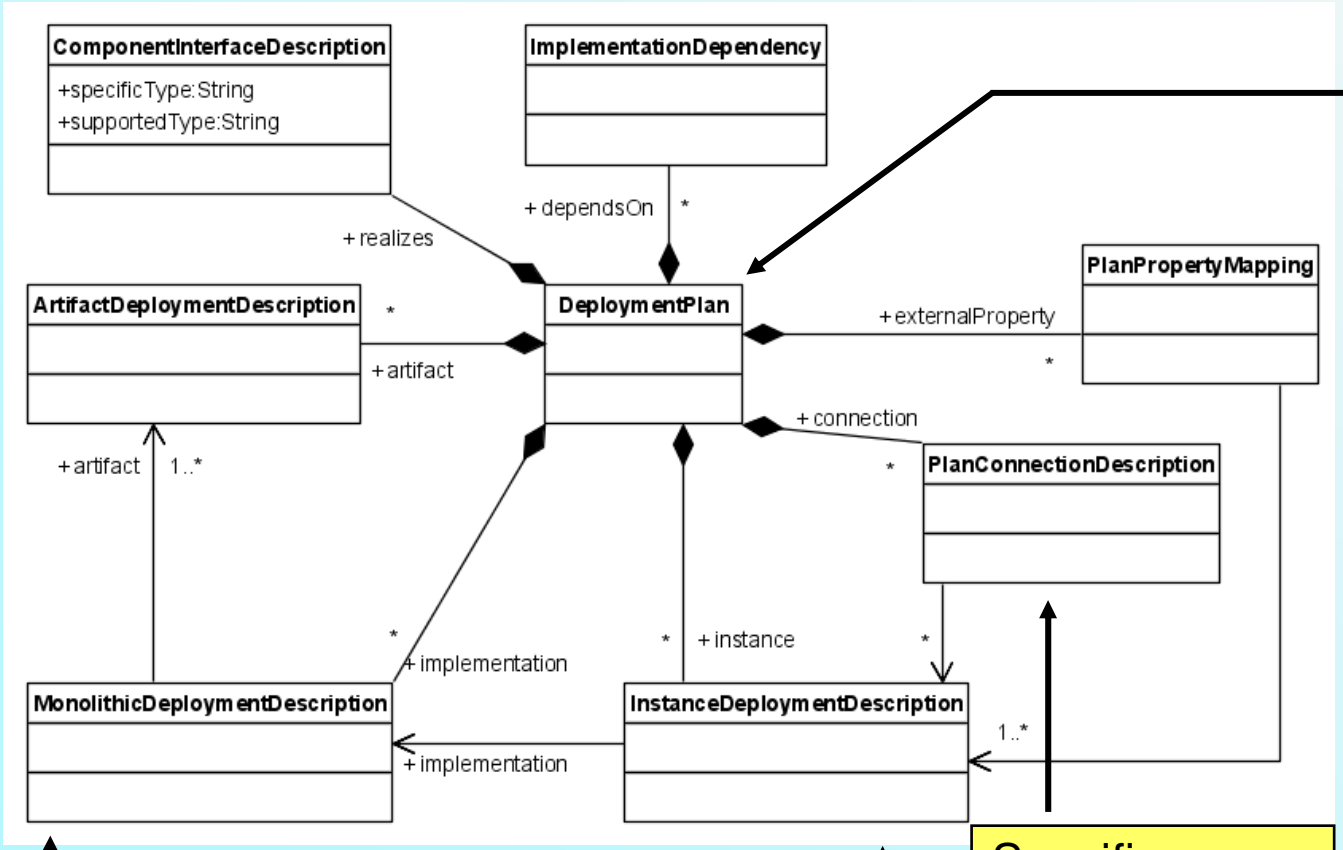
Distribution Modelling in UML 2.0



- No general concept of a deployment plan; quite simple model
- Just assignment of ***Artifact***s to ***DeploymentTarget***s, i.e. mainly nodes
  - A ***DeploymentTarget*** owns a set of ***Deployment***s reflecting the installation of ***Artifact***s or instances of ***Artifact***s (***InstanceSpecification***)
  - Again: no powerful matching mechanism between node properties and artifact requirements
- Deployment-related information such as ConfigValues may be specified using the ***DeploymentSpecification*** class not shown here

Distribution Modelling in OMG DnC

Component definition



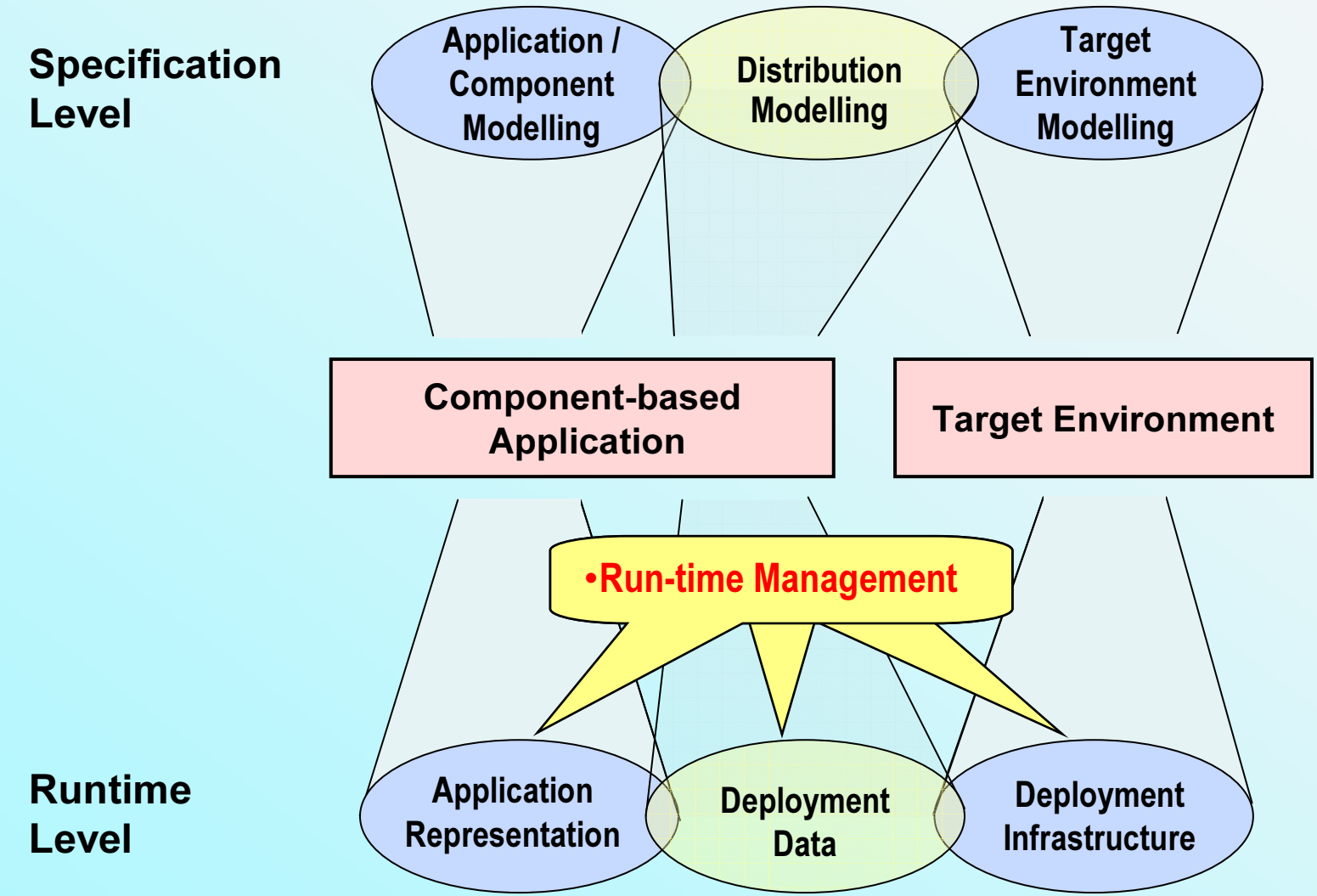
DeploymentPlan:

- Maps component implementations to nodes and connections between component instances to bridges and interconnects
- Records matching properties against resources

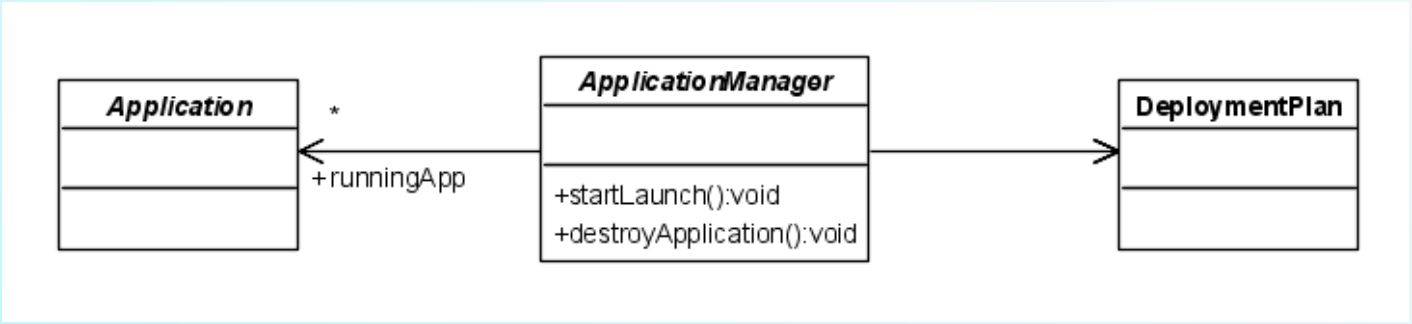
Specifies deployment of a particular component

Specifies Interconnected instances

Run-time Management



Run-Time Management in OMG DnC



- Only the DnC spec provides a model for executing a deployment specification
  - It defines a set of interfaces and data formats to be exchanged at those interface
  - Defines an vendor-independent interoperable deployment machinery
- One major concept is the ApplictionManager
- Further concepts are e.g. Target-, Node-, ExecutionManager

Summary on Compared Concepts

	eODL	UML 2.0	OMG DnC
Application / Component Modelling	COTypeDef	Component	ComponentInterfaceDescription
	PortDef	Port	ComponentPortDescription
	AssemblyDef	(nested) Component	AssemblyDescription
	SoftwareComponentDef	Artifact	ComponentImplementationDescription, MonolithicImplementationDescription, ComponentAssemblyDescription
	Property (name-type-value)	*(name-value annotation)*	typed Requirement
Target Environment Modelling	TargetEnvironment	-	Domain
	Node	Node	Node
	NodeLink	CommunicationPath	Interconnect, Bridge
	Property	*(name-value annotation)*	typed Resource and SharedResource
Distribution Modelling	DeploymentPlan	-	DeploymentPlan
	InstallationMap, ComponentAssignment	Deployment	ArtifactDeploymentDescription
	InstantiationMap	InstanceSpecification	InstanceDeploymentDescription
	<Properties of SoftwareComp. meet TargetEnv. Properties>	-	<typed Resources meet typed Requirements>
Run-time Management	-	-	ApplicationManager, Application (Target-,Node-,ExecutionManager)

Conclusion

- Three languages suitable for deployment specifications have been investigated
- They use different modelling elements and have different main focus
- However, there is a set of common concepts that could be the basis for an alignment
- All languages investigated have a MOF-based metamodel
- On this basis mappings between all three languages as well as to other platform-dependent languages can be defined