Fraunhofer Institute for Open Communication Systems

Deployment and Configuration

of Component-based Distributed Applications

SBC Workshop Tutorial

Andreas Hoffmann
a.hoffmann@fokus.fraunhofer.de

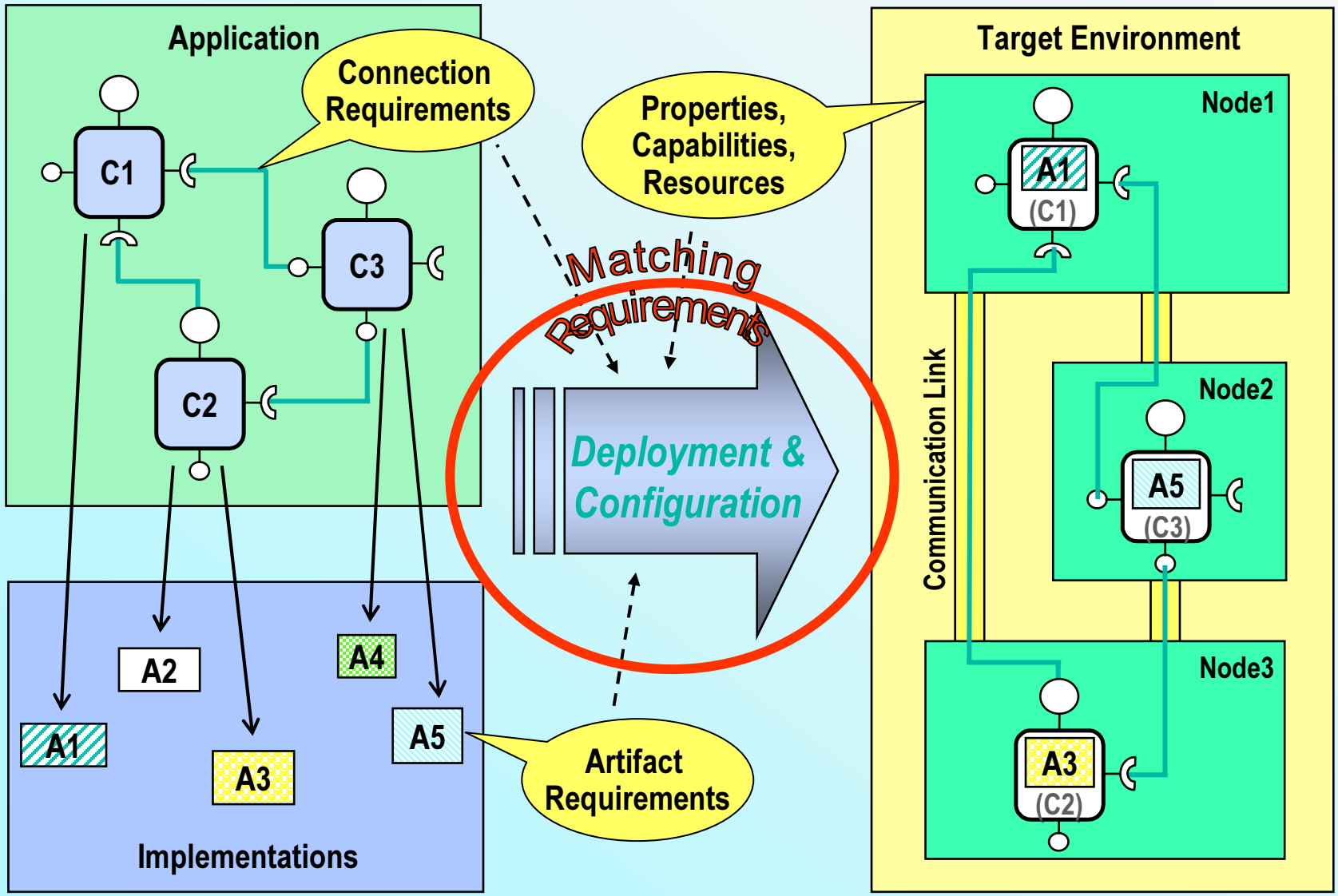
Table of Contents

Table of Contents

What is Deployment?

- **Deployment and configuration of distributed applications:**
  - Essential part of the overall system life-cycle
  - Process of installing and customizing a distributed application in a target environment and take it into operation
- **Deployment process includes wide range of deployment tasks:**
  - assembling & packaging,
  - target node assignment, code upload, installation,
  - instantiation, configuration, connection set-up
- **In order to *automate* the component-based**

Deployment & Configuration Overview



General Deployment Tool Chain

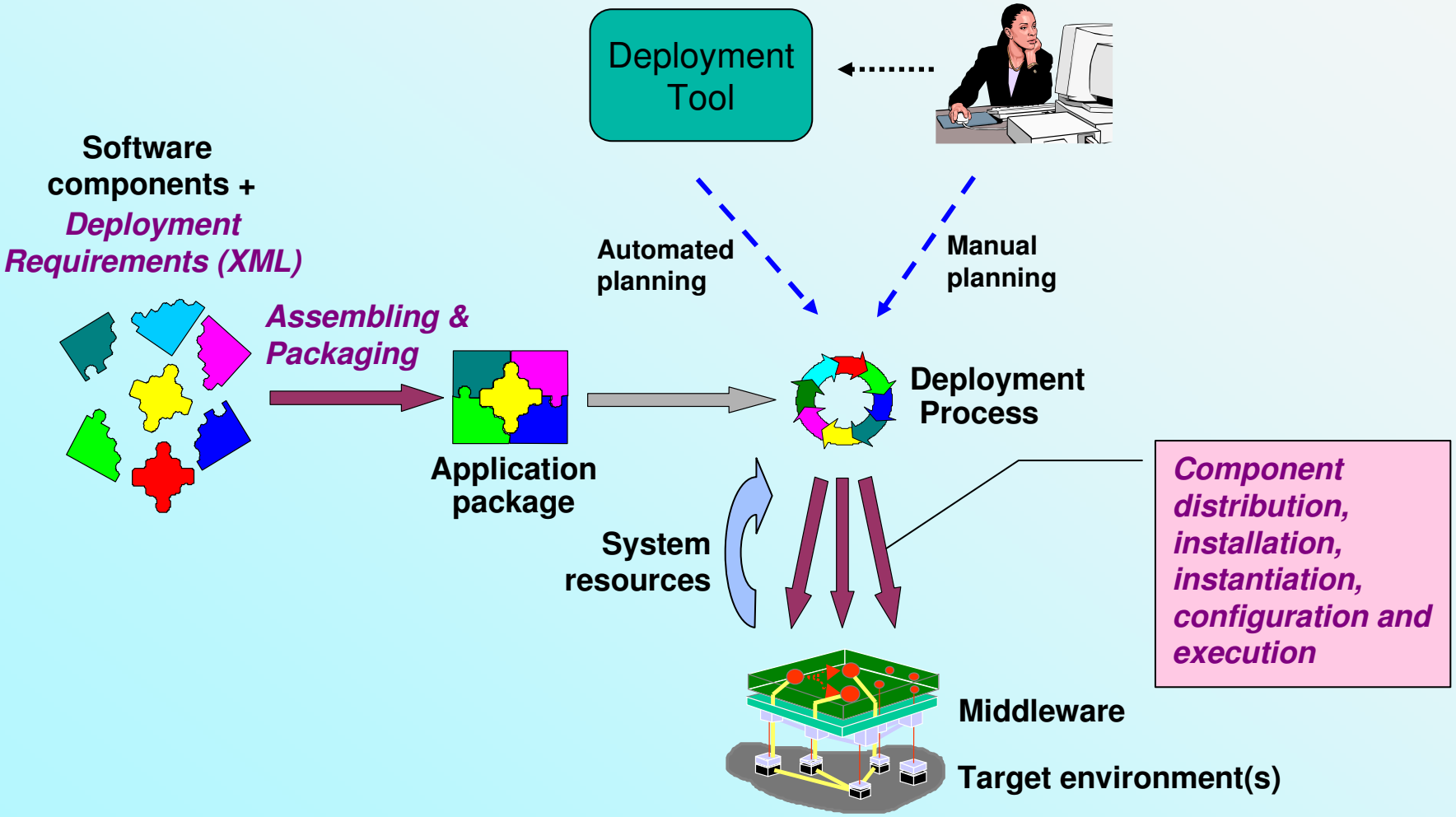


Table of Contents

Overview on OMG D+C

- In 2003 the OMG adopted the ***Deployment and Configuration of Component-based Applications*** specification
- Focus: Interoperable deployment machinery
  - Deployment architecture with well-defined interfaces and interchange formats
- Development phase out of scope of D+C
  - Starting point: complete (implemented) specification
  - D+C spec is applicable to wide range of different component-based methodologies
  - Precondition: Compatible component definition

The D+C Specification…

… Standardizes a common deployment model, interfaces and interchange formats for vendor-independent *automated* deployment of distributed component-based applications

… Enables *Interoperability* between vendors

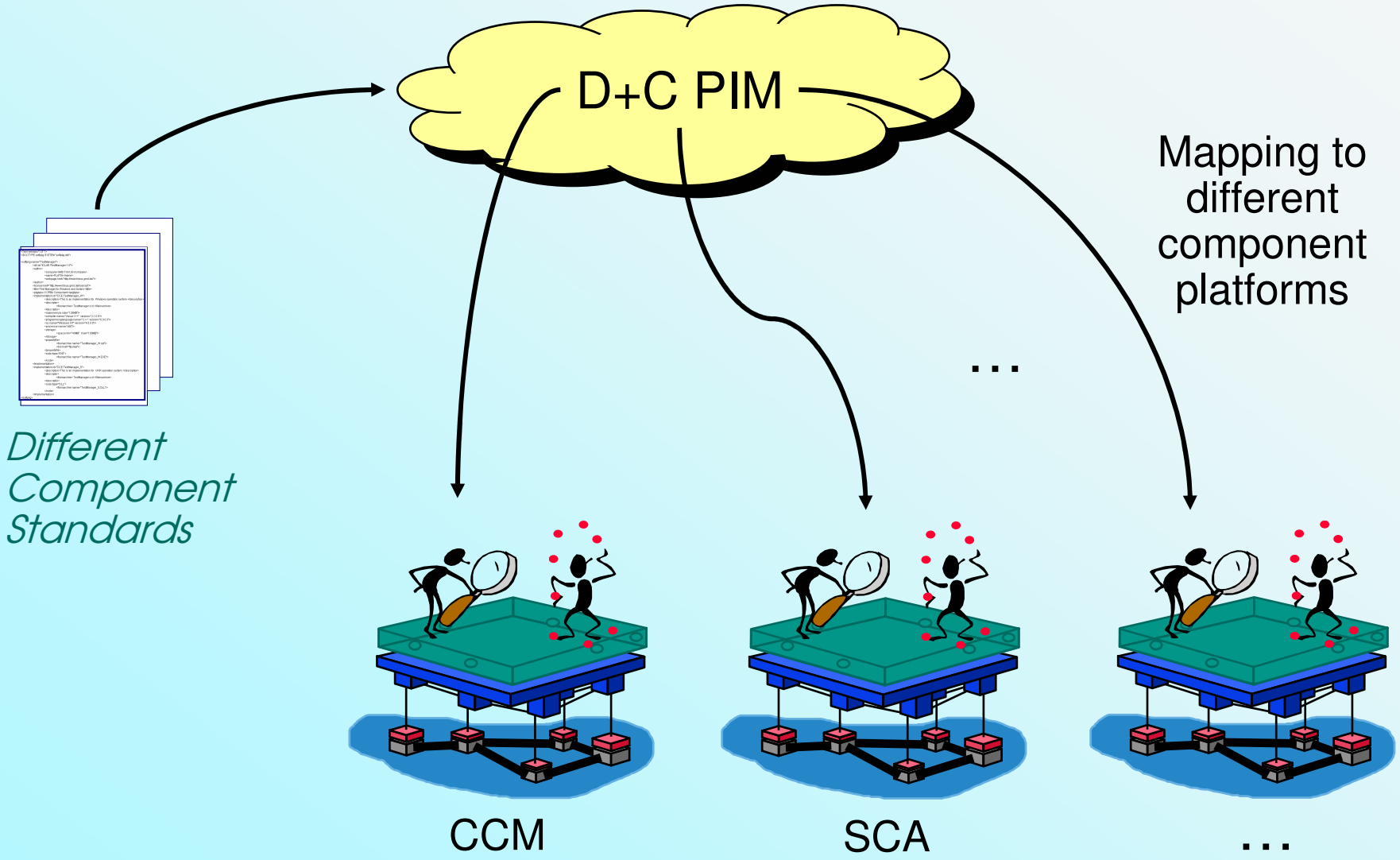
… Is compatibility with OMG standards

- MOF 1.4, XMI 2.0
- CCM (adding extensions)
- Alignment with UML2.0
  => refinement of deployment & infrastructure concepts

… Overcomes CCM deployment limitations

- Hierarchical components and assemblies with external interfaces
- Heterogeneous multi-vendor assemblies
- Add target environment description
- Define some deployment infrastructure interfaces for interoperability

… and is Technology Independent



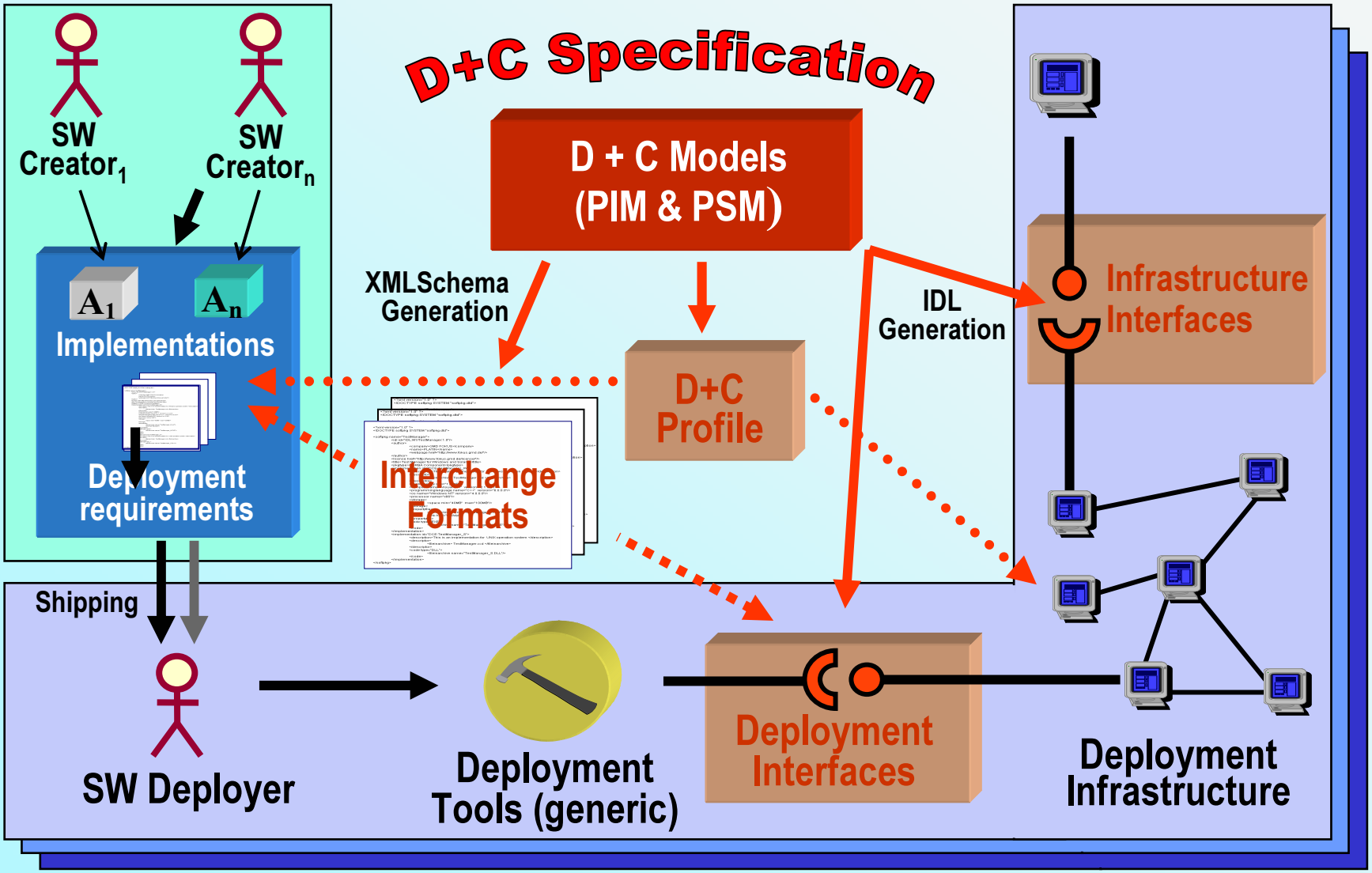
Relationship to MDA

Submission conforms to MDA.

It is composed of three main levels of models:

- **D+C PIM** (core of the D+C spec)
  - Defining general concepts for Deployment & Configuration
  - Independent of distributed component middleware technology (e.g.CORBA or J2EE), information formatting technology (e.g. XML DTD and XML), and programming languages (e.g. C++ and Java)
- **D+C PSM** for CCM (integral part of the D+C spec)
  - Constitute a realization of the D+C PIM on a concrete platform
  - A PIM-to-PSM transformation is explicitly defined for this PSM
  - Mappings to CORBA IDL and XML schemata at PSM level
- **D+C Tool-Support Profile (for UML 1.4)**
  - Closely related to the D+C PIM
  - This profile defines a concrete UML-based syntax for the abstract language defined by the D+C PIM for specifying the deployment and configuration of distributed components
  - Enables use of generic UML tools
  - Use of defined stereotypes enables the automatic generation of D+C classes and descriptors from D+C UML models

D+C Specification Overview

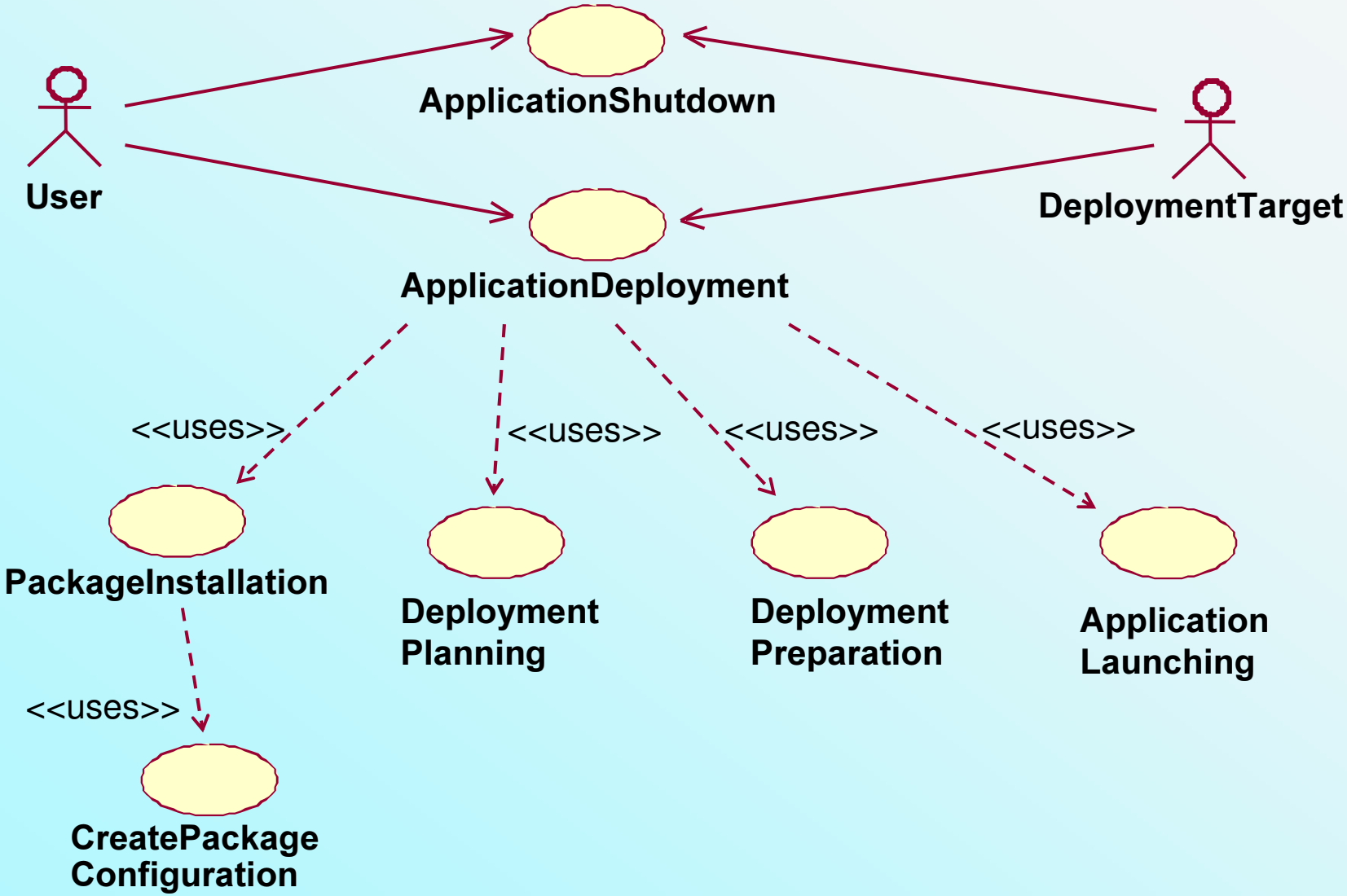


In more Detail…

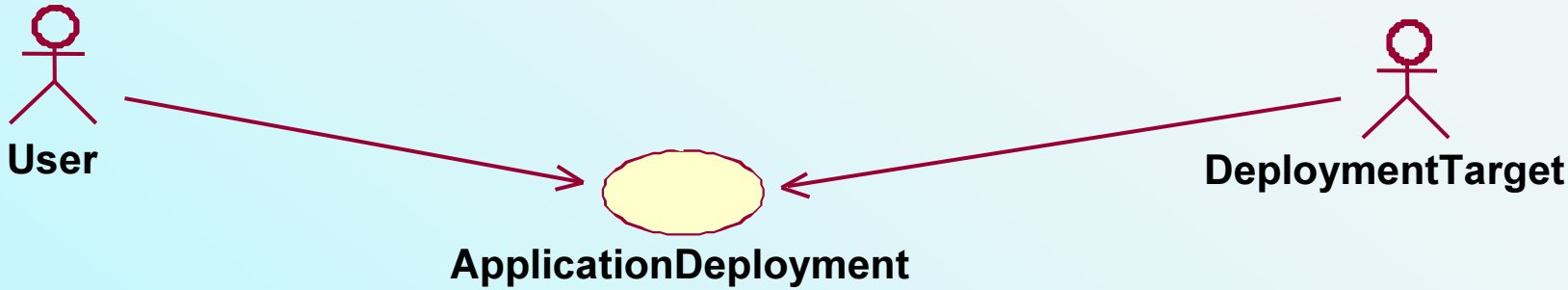
- **The D+C submission provides means for:**
  - Describing the **deployment requirements** of the software
  - Packaging the software and associated **metadata** for delivery between the software producer and the deployer
  - **Configuring** the software *before* deployment decisions are made
  - Describing the facilities of the **distributed target execution infrastructure**
  - **Deployment planning**, i.e. deciding how the software will be deployed onto the targeted distributed execution infrastructure
  - **Performing** the actual preparation of the application for execution, e.g., uploading parts of the software to their location of execution
  - **Launching**, monitoring, and **terminating** the application

Table of Contents

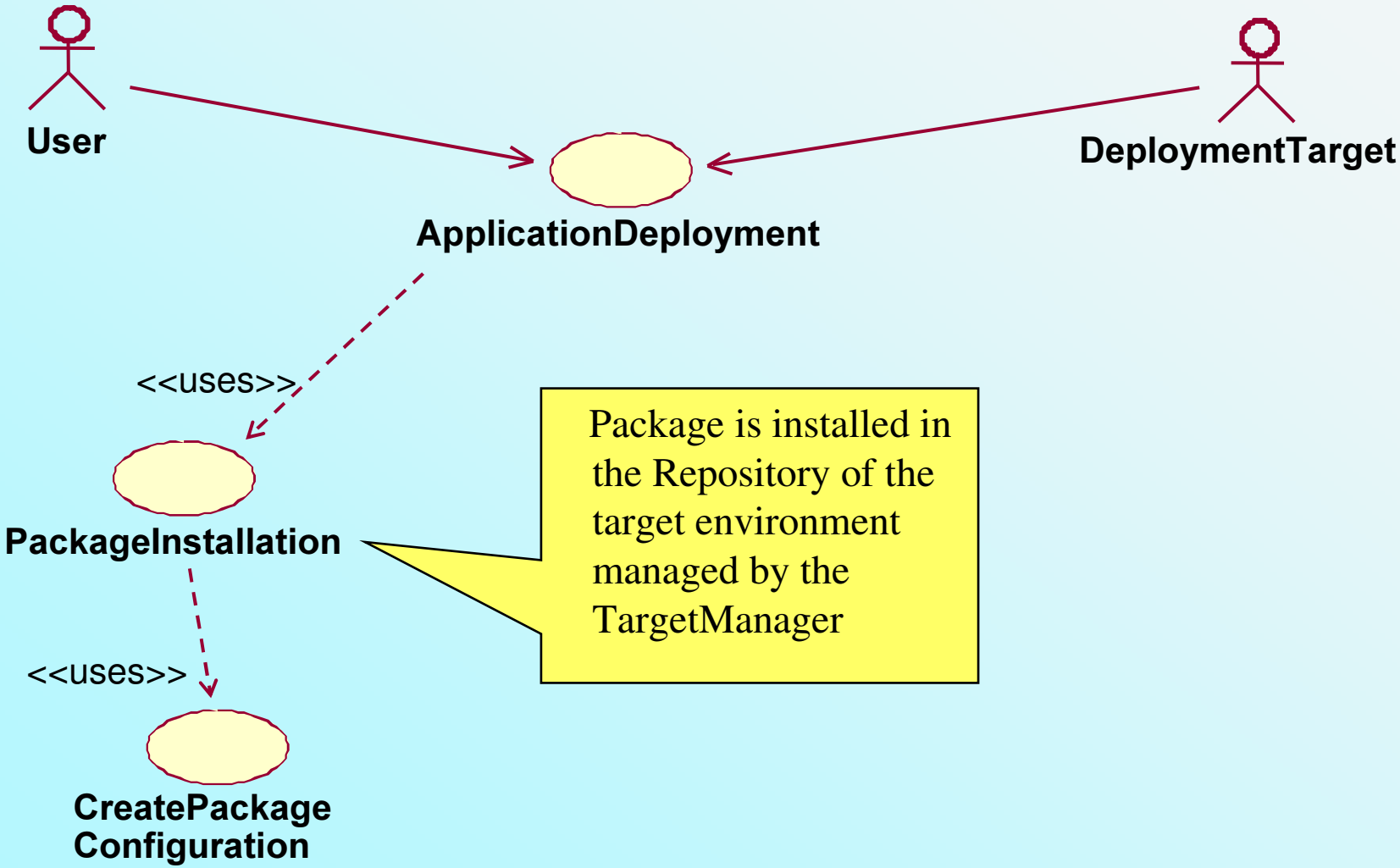
Deployment Use Cases (Phases)



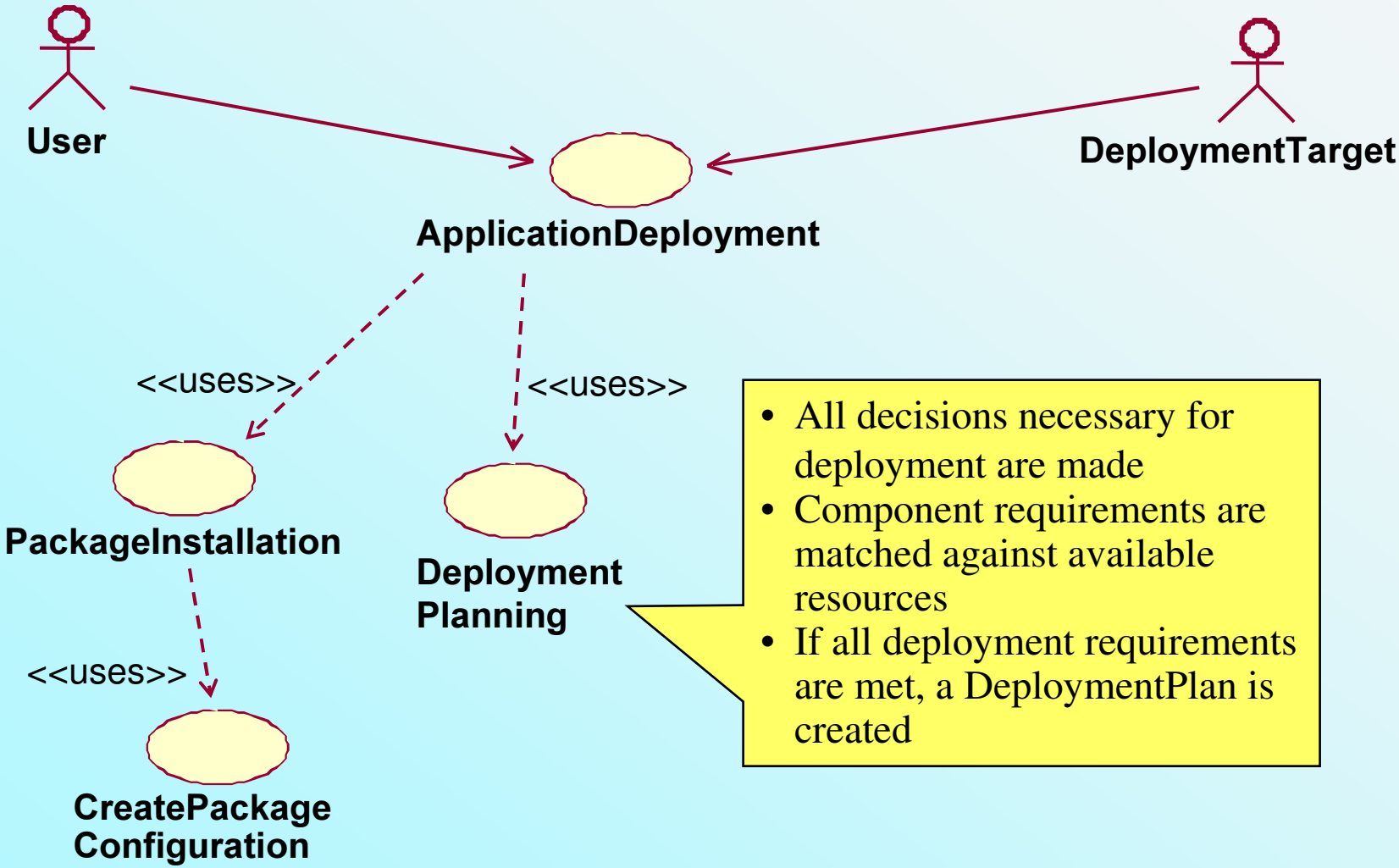
Application Deployment



Package Installation

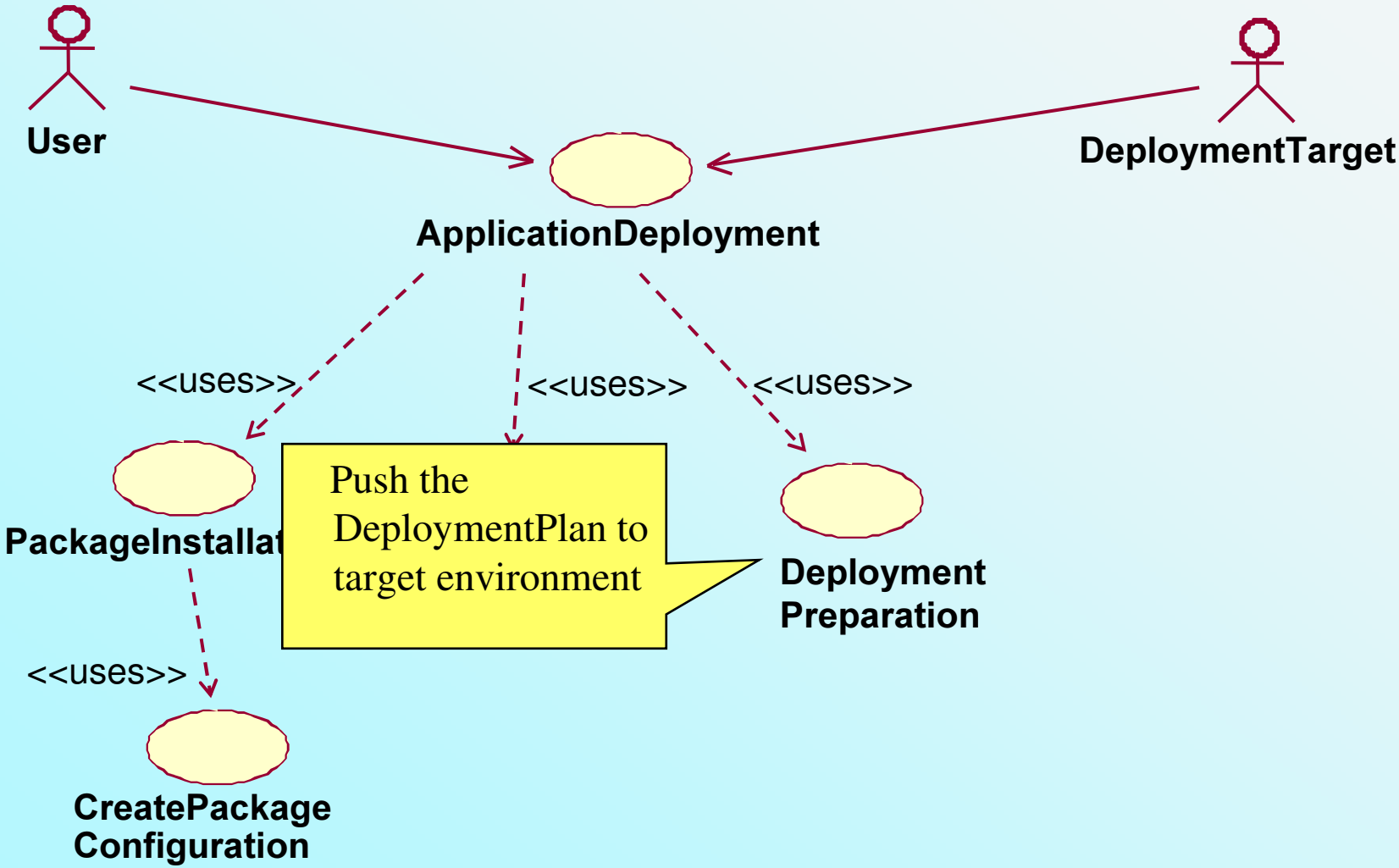


Deployment Planning

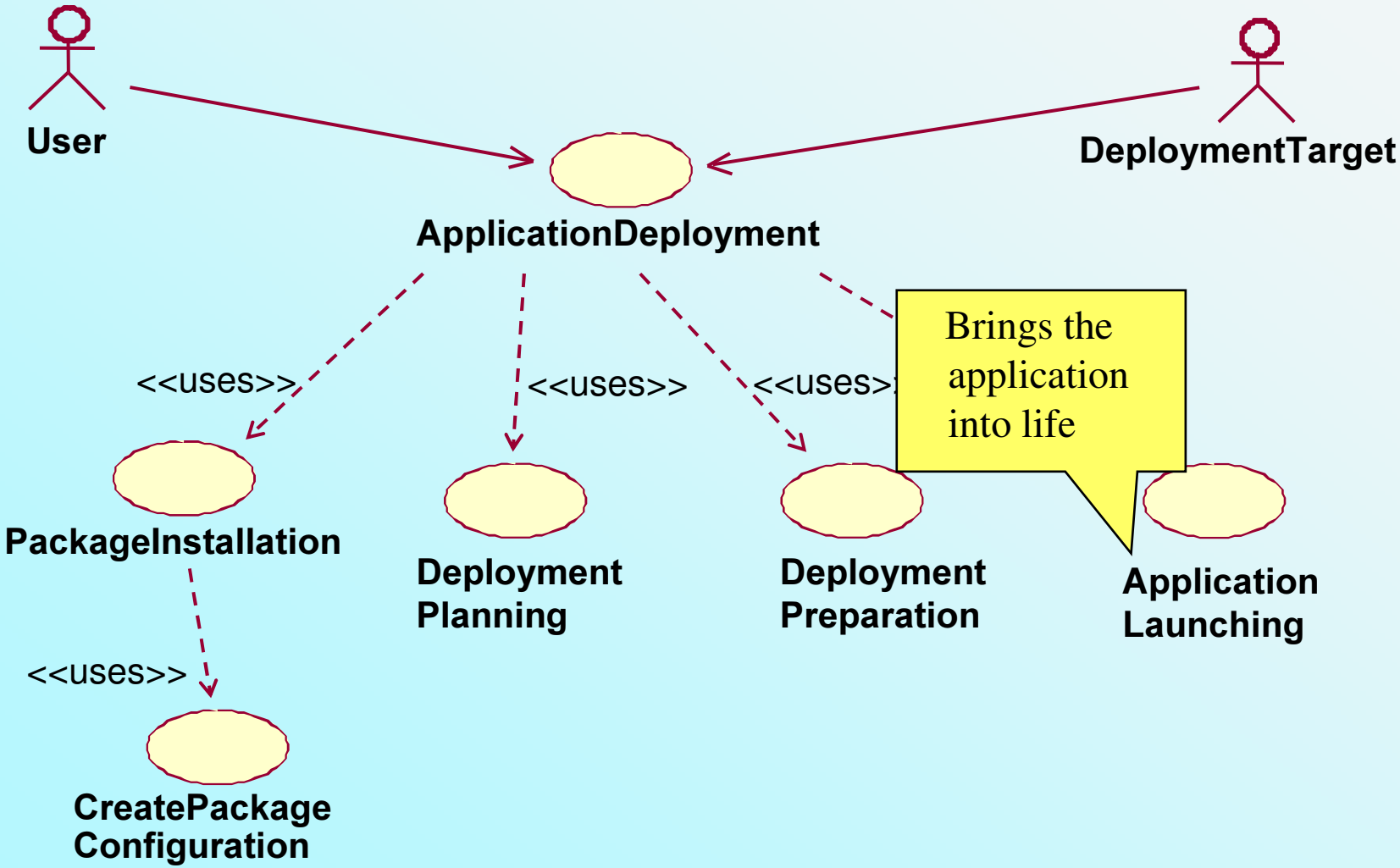


- All decisions necessary for deployment are made
- Component requirements are matched against available resources
- If all deployment requirements are met, a DeploymentPlan is created

Deployment Preparation

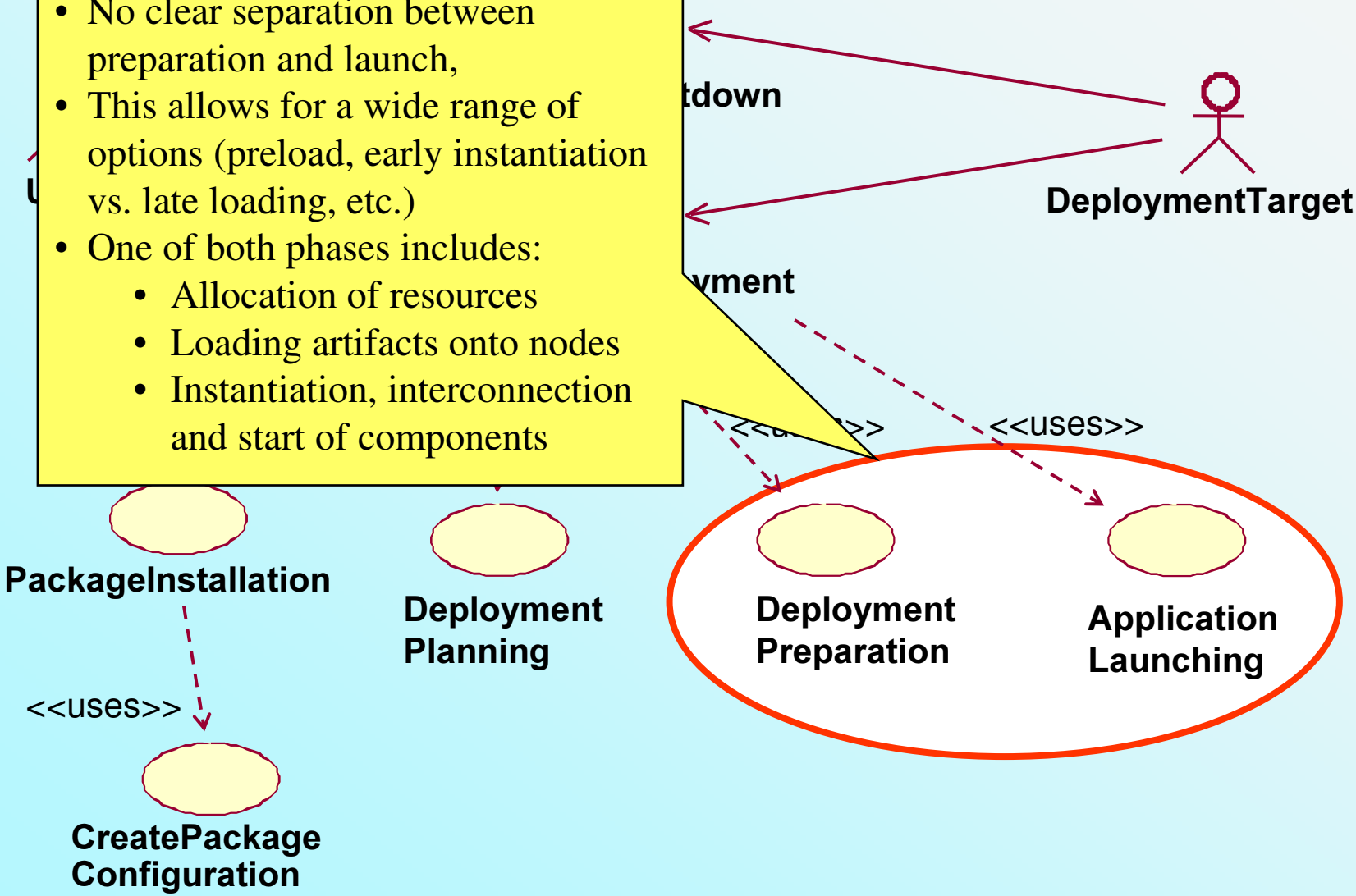


Application Launching



Deployment Use Cases

- No clear separation between preparation and launch,
- This allows for a wide range of options (preload, early instantiation vs. late loading, etc.)
- One of both phases includes:
  - Allocation of resources
  - Loading artifacts onto nodes
  - Instantiation, interconnection and start of components



Application Shutdown

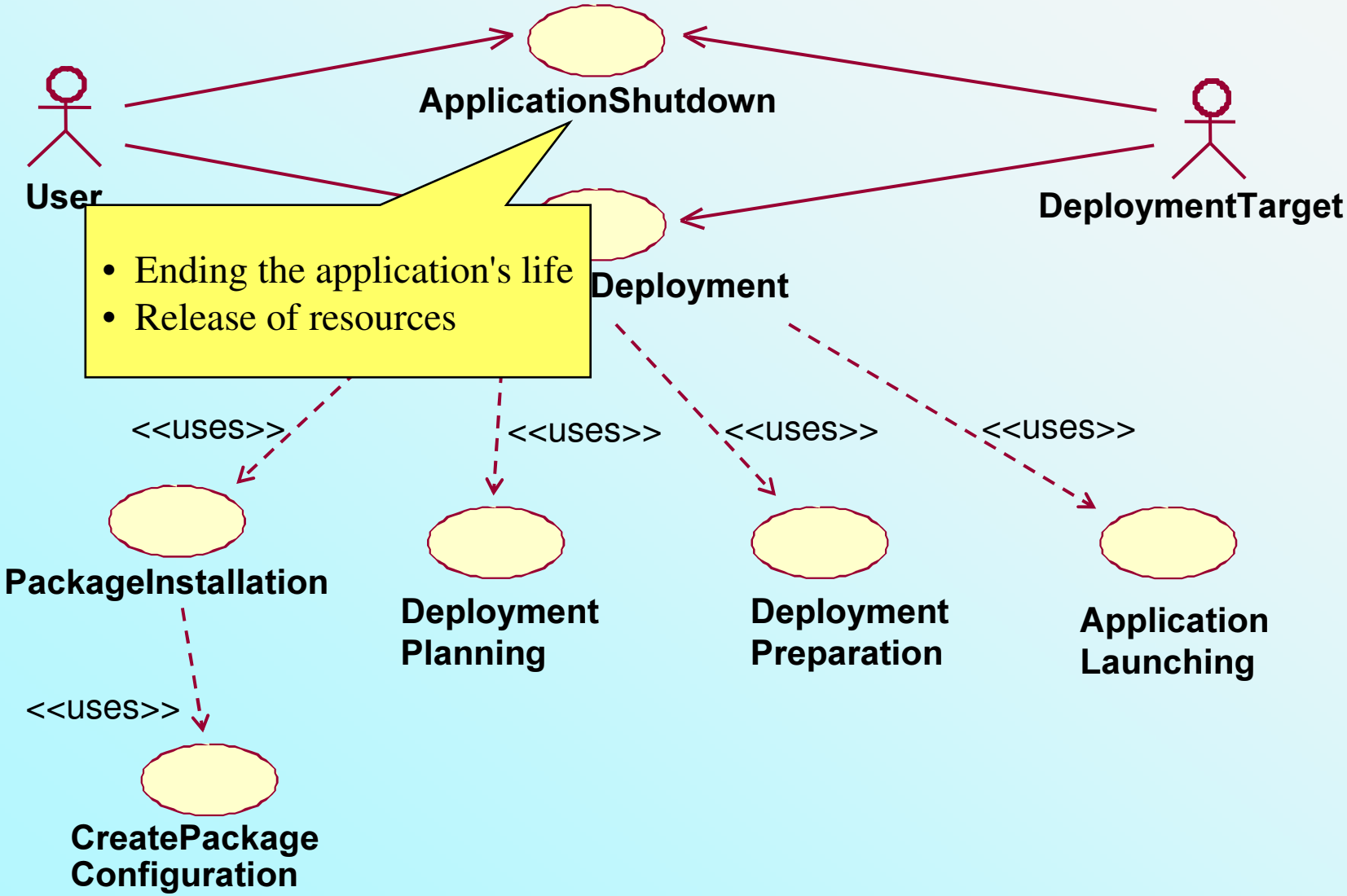
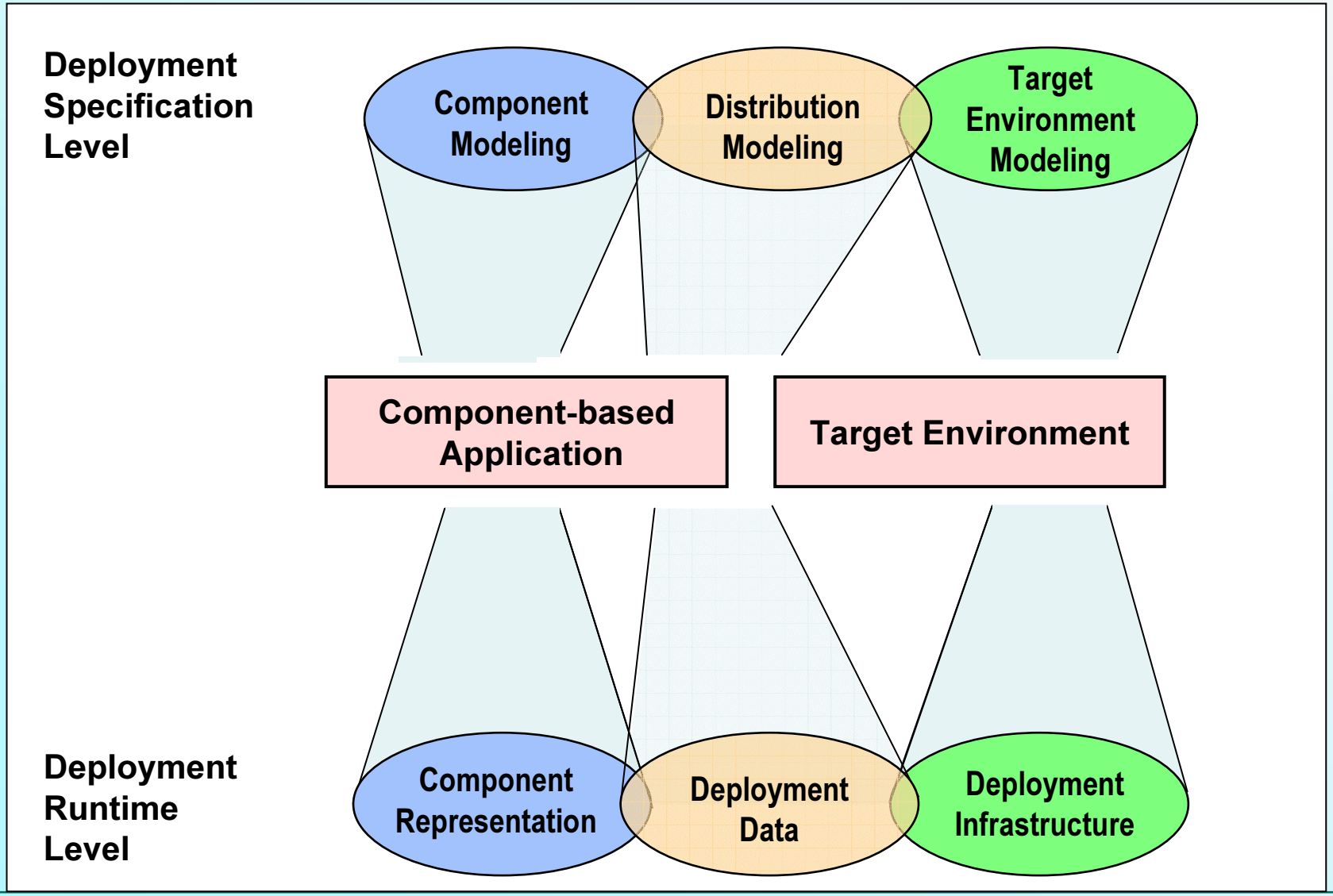
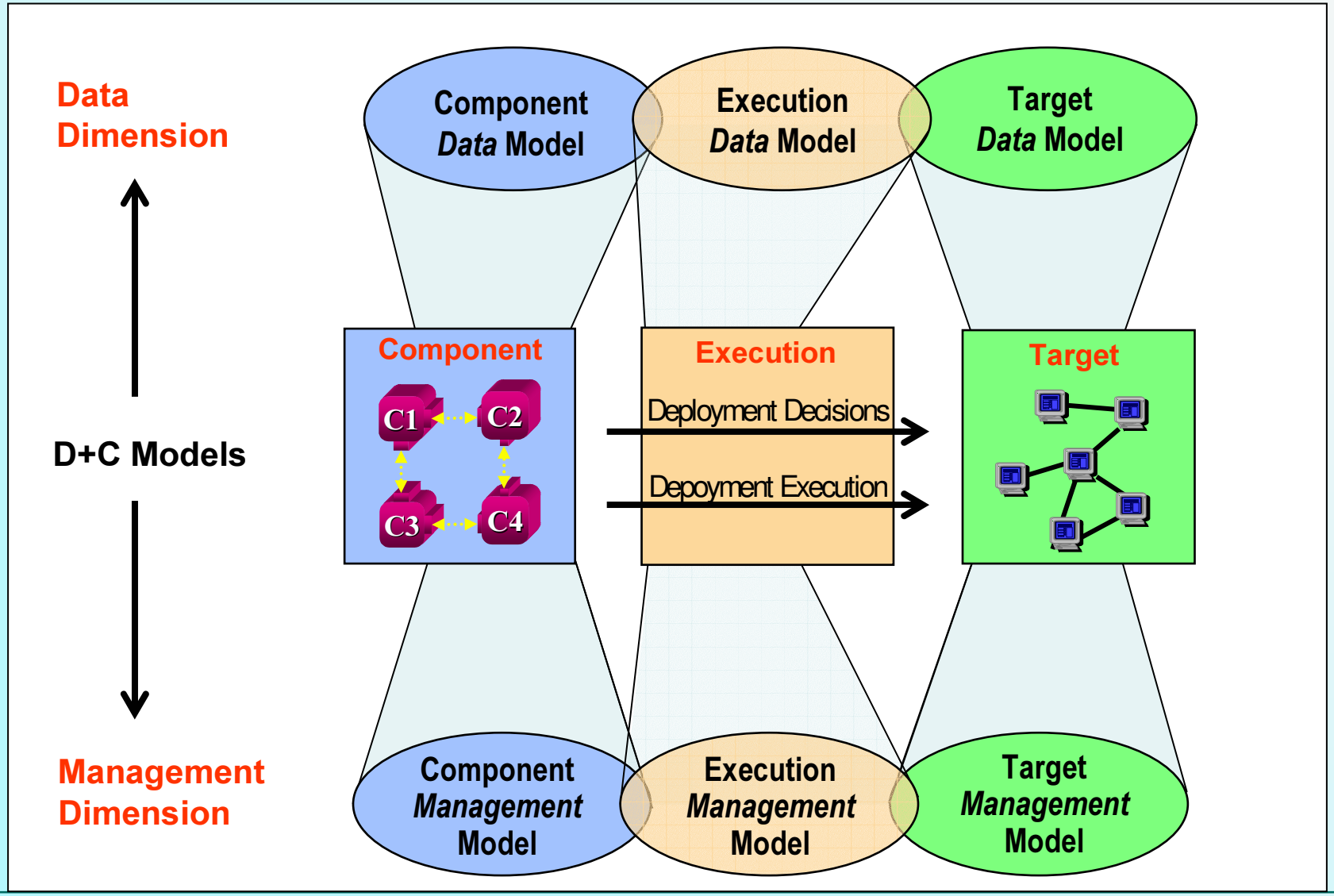


Table of Contents

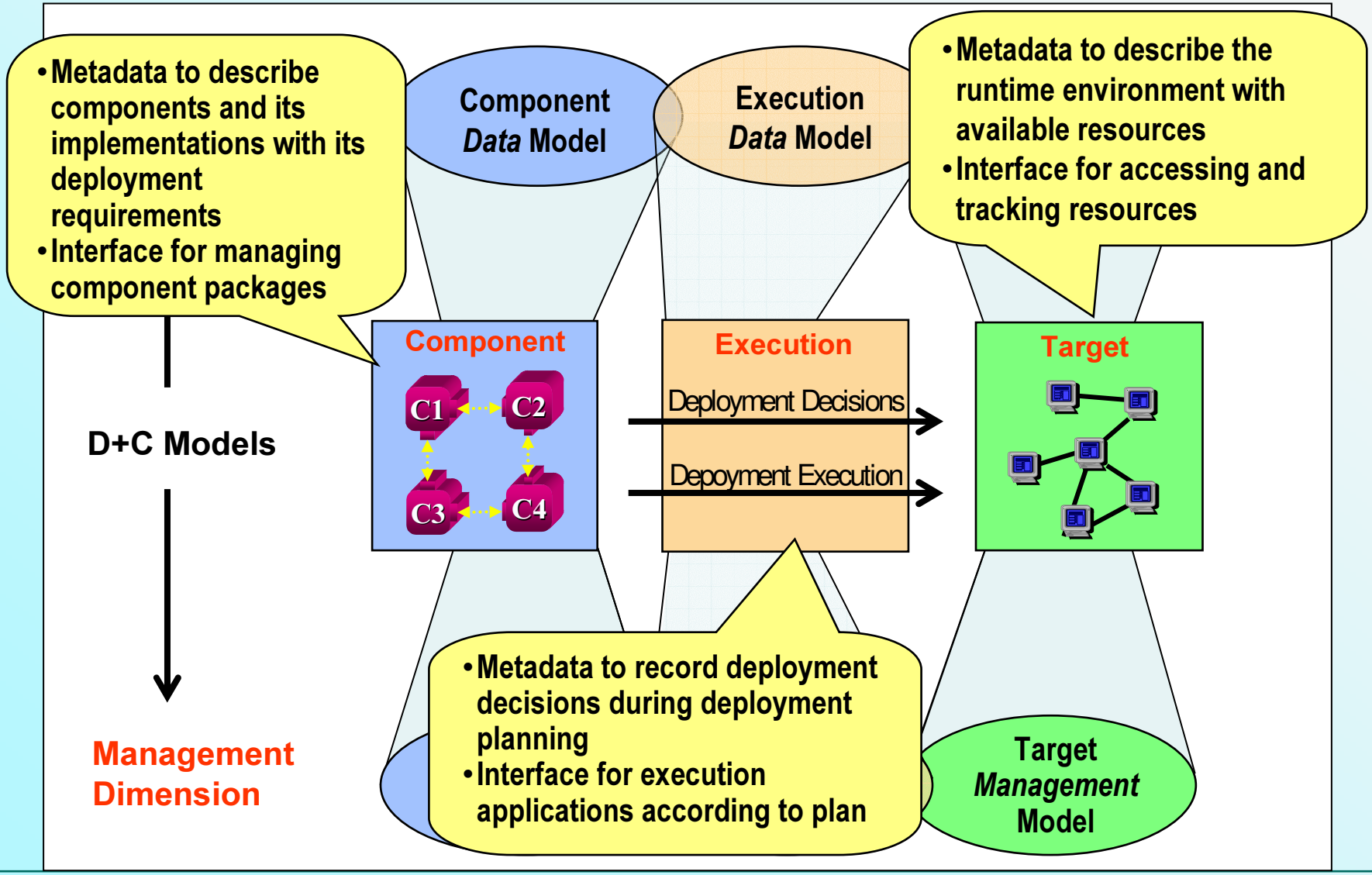
Overview on Deployment Modeling



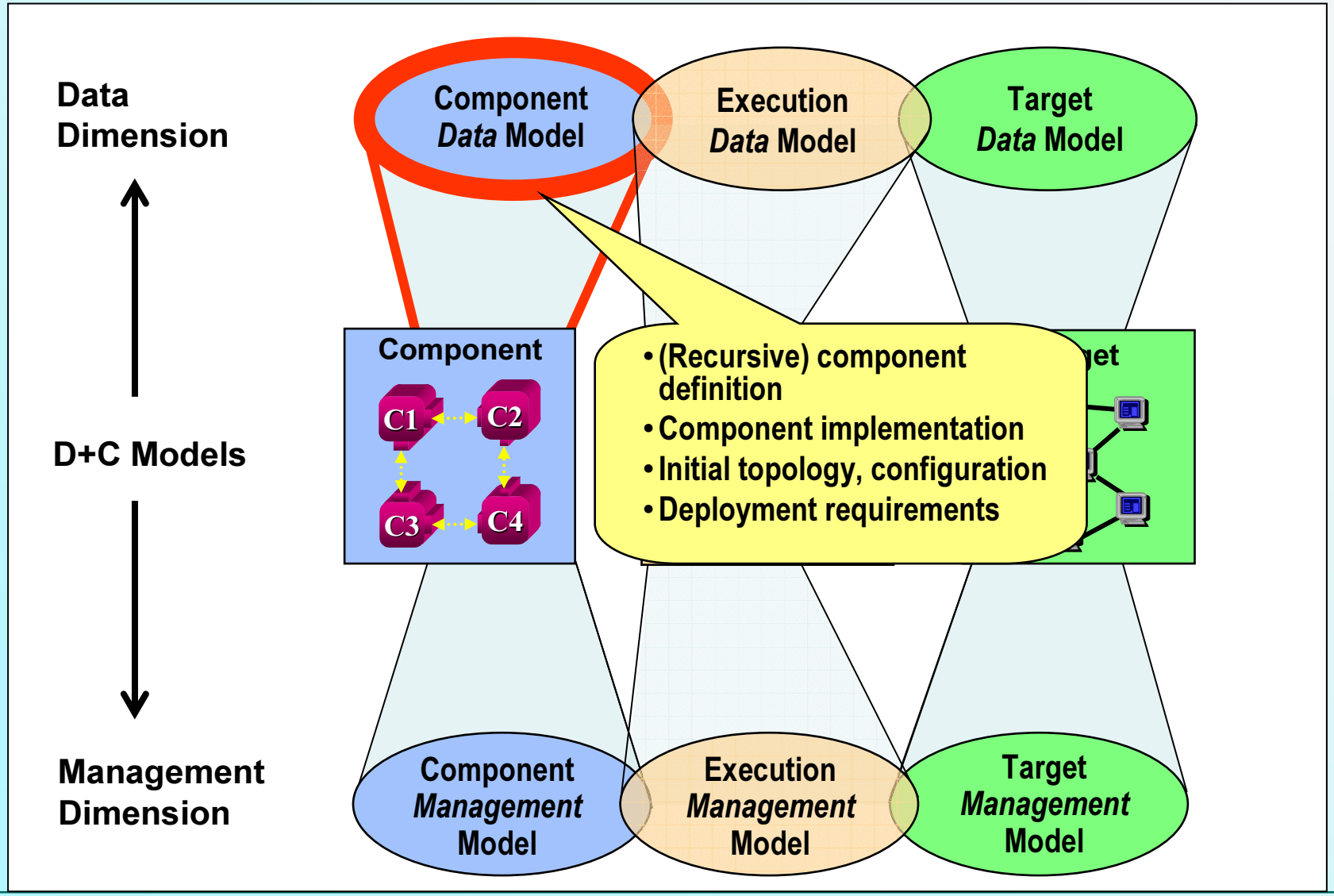
Segmentation of D+C Models



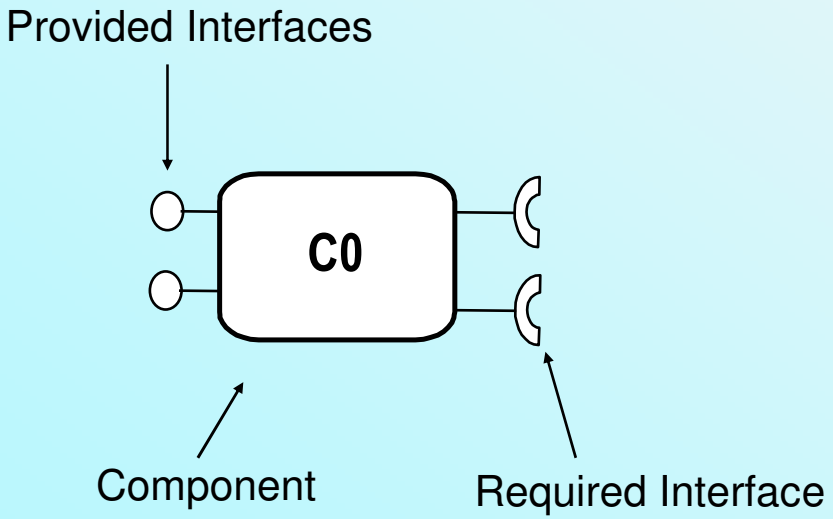
Segmentation of D+C Models



Component Data Model



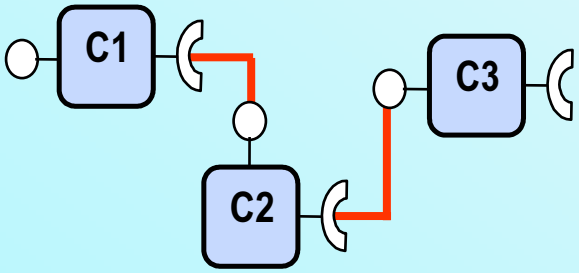
D+C Component Definition



A Component …

… is a replacable unit defined by its ports (required and provided interfaces)

D+C Component Definition (cont'd)

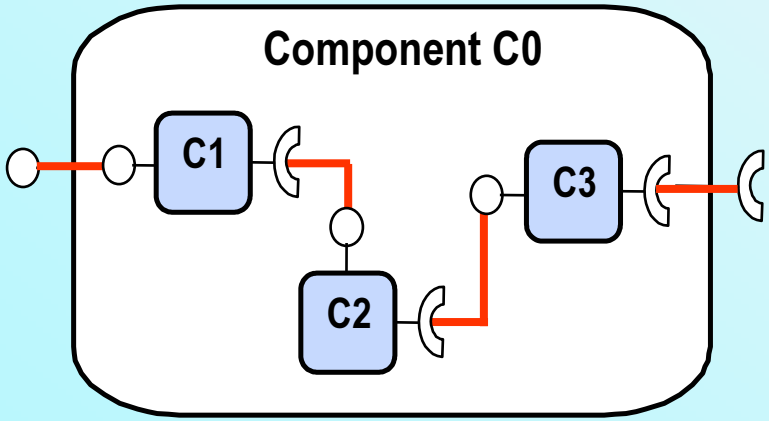


A Component …

… is a replacable unit defined by its ports (required and provided interfaces)

… is connected with other components via its ports

D+C Component Definition (cont'd)



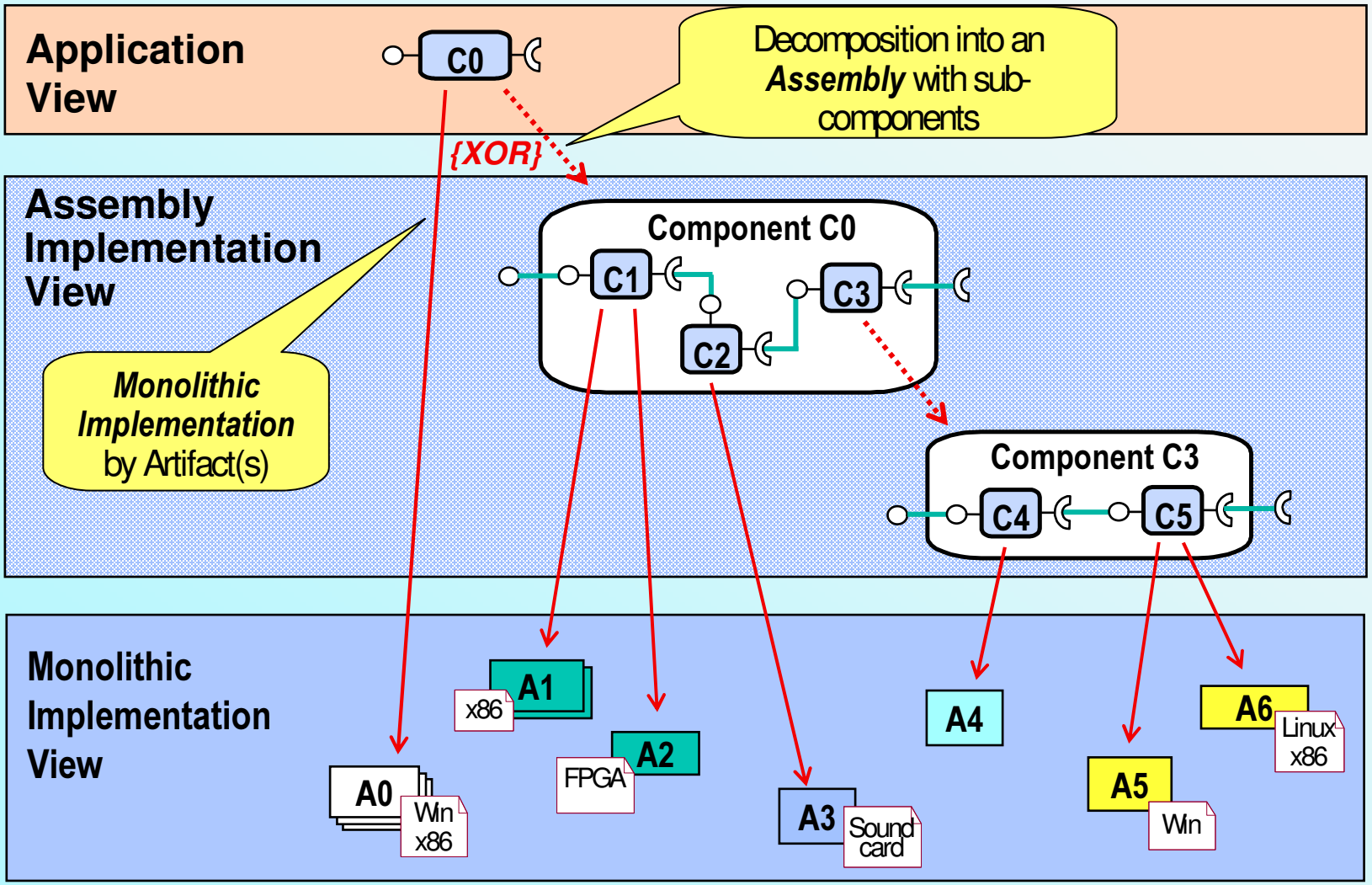
A Component …

… is a replacable unit defined by its ports (required and provided interfaces)

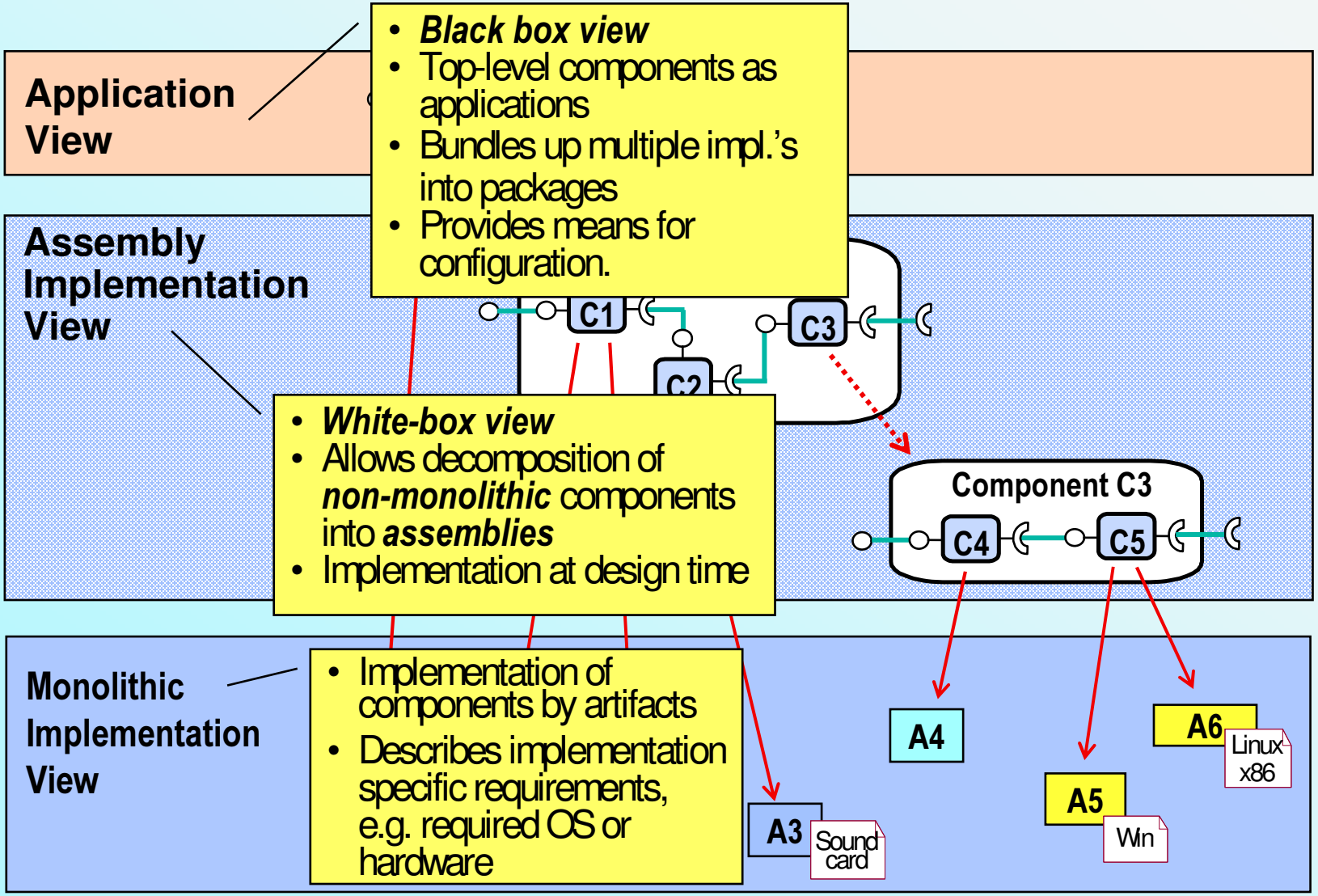
… is connected with other components via ports

… may be hierarchically defined (also called ***Assembly***)

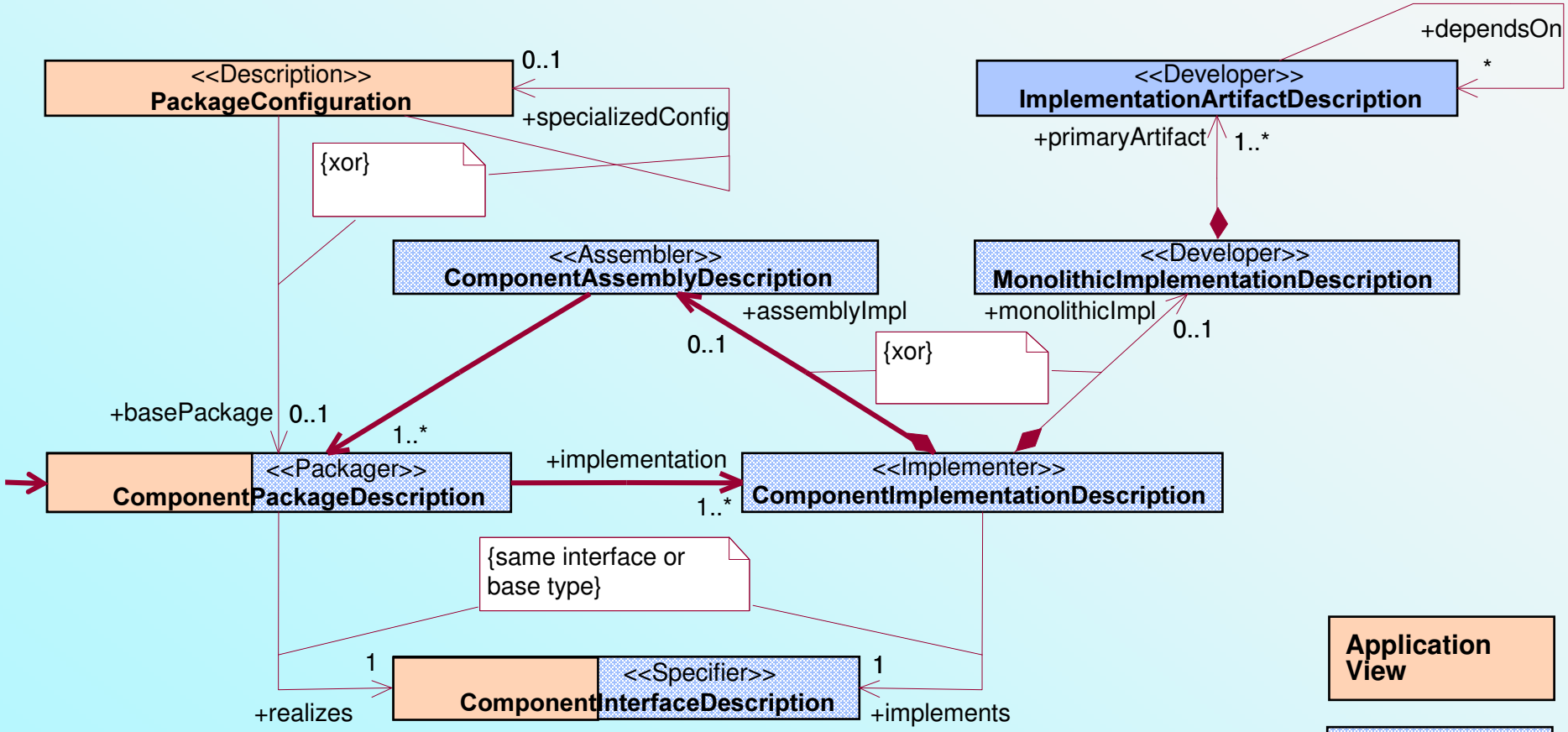
D+C Component Implementation



Different Views on Components

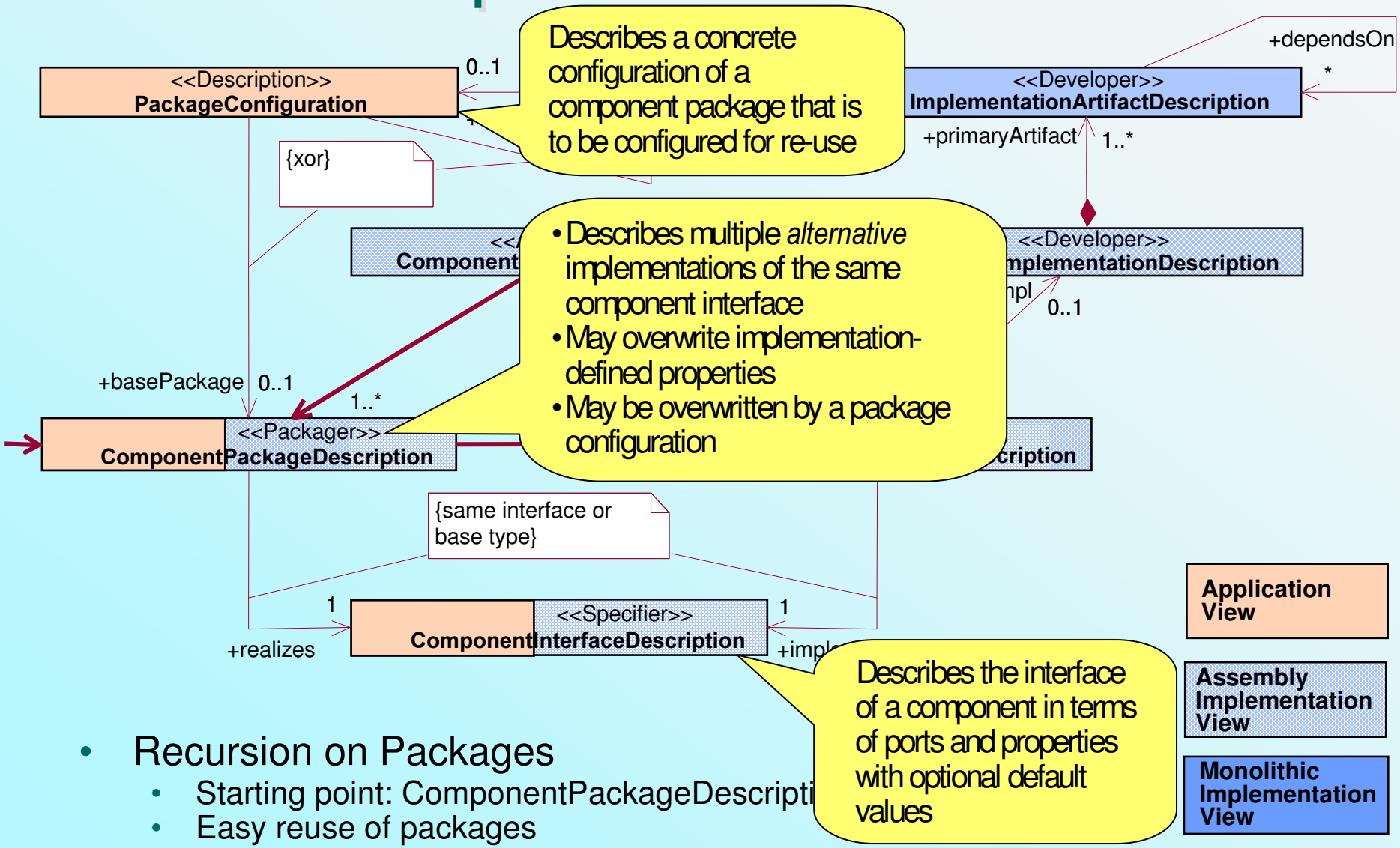


Component Data Model



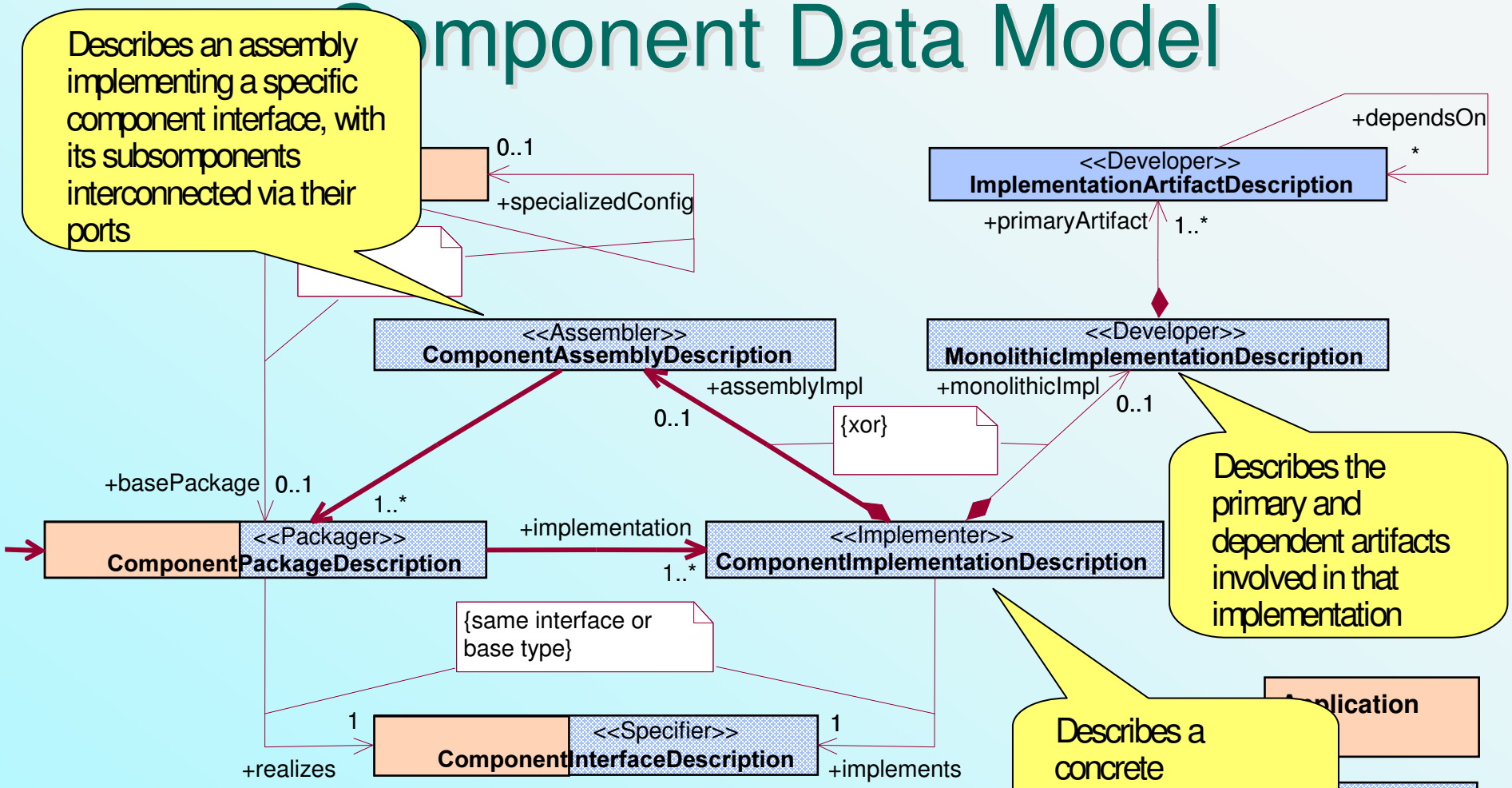
- Recursion on Packages
  - Starting point: ComponentPackageDescription
  - Easy reuse of packages

Component Data Model



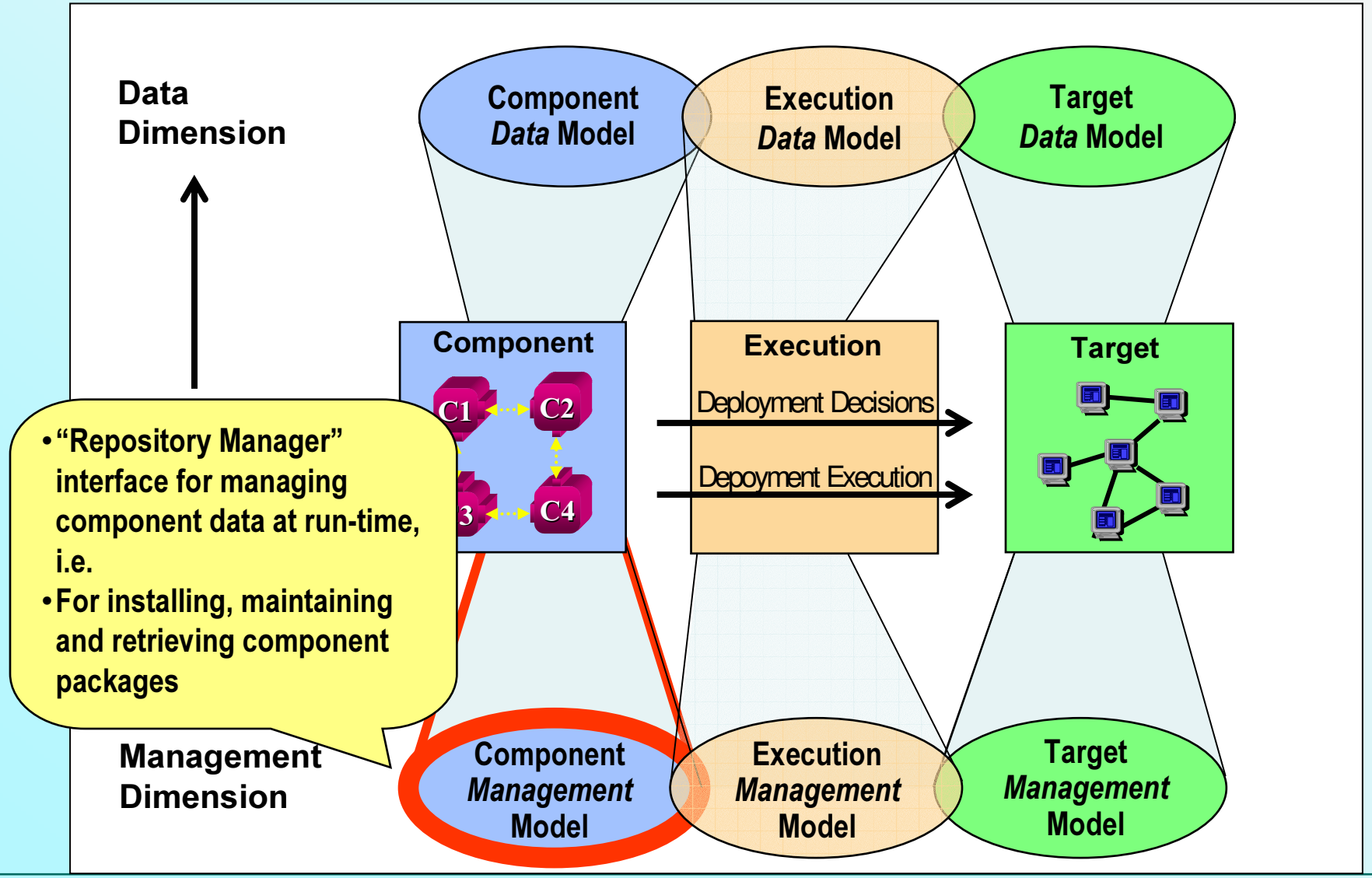
- Recursion on Packages
  - Starting point: ComponentPackageDescripti
  - Easy reuse of packages

Component Data Model



- Recursion on Packages
  - Starting point: ComponentPackageDescription
  - Easy reuse of packages

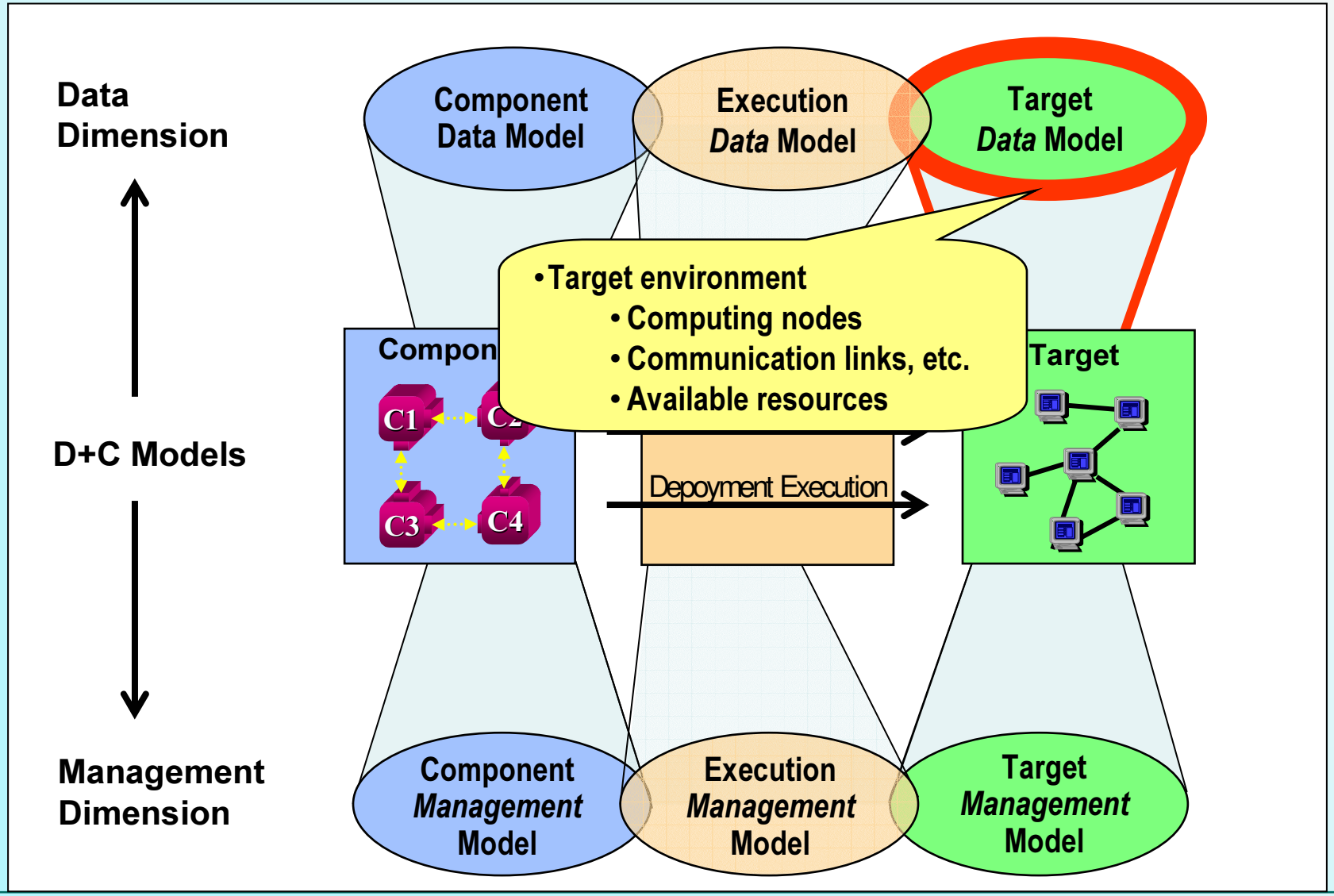
Component Management Model



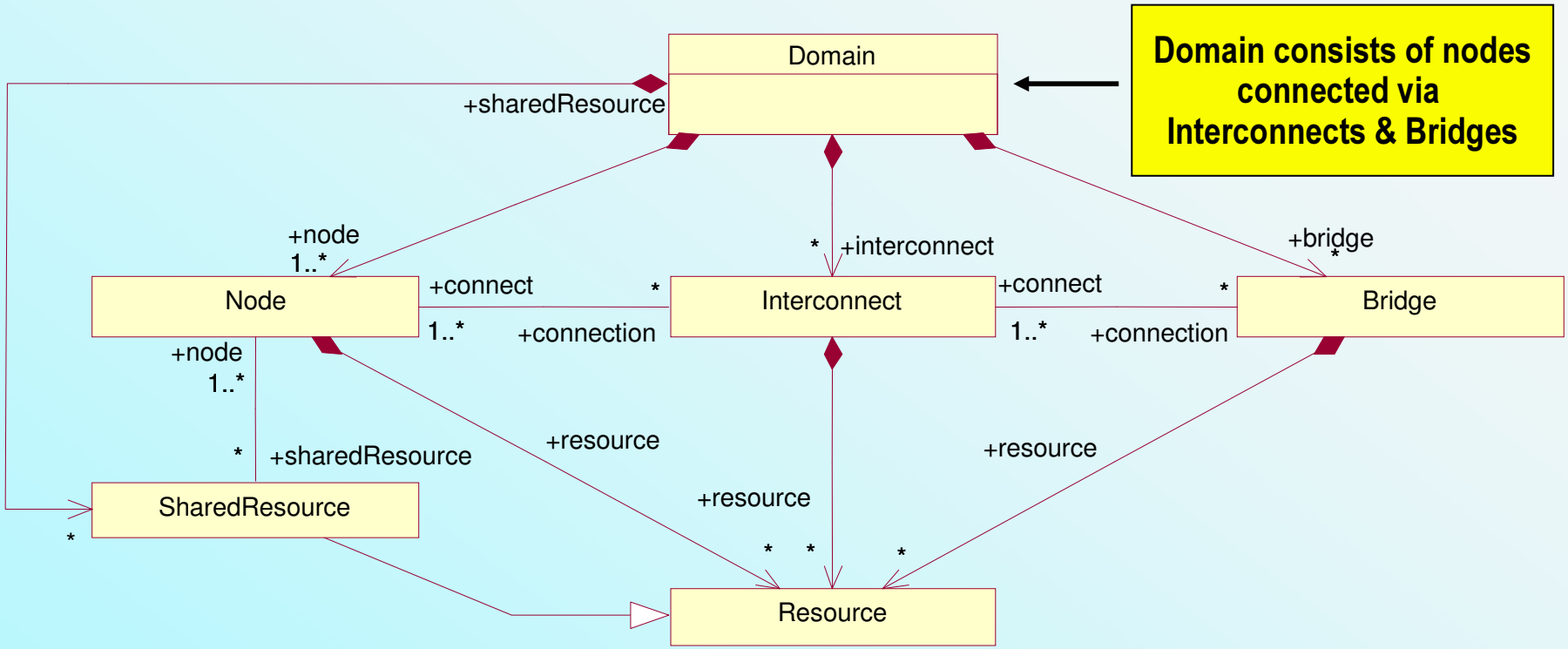
Repository Manager

- Maintains a collection of PackageConfigurations
- Installes, updates, deletes or retrieves PackageConfigurations

Target Data Model

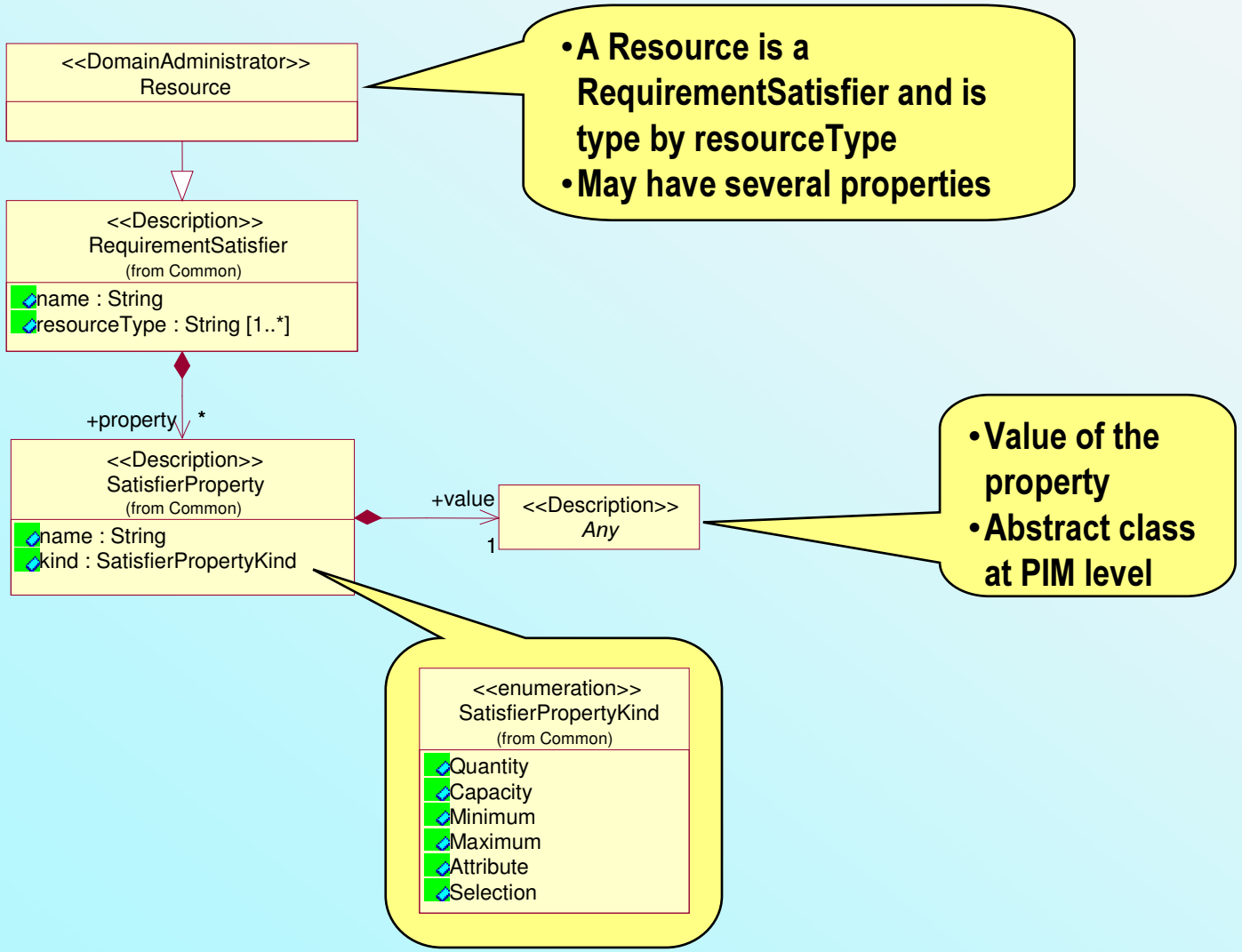


Target Data Model

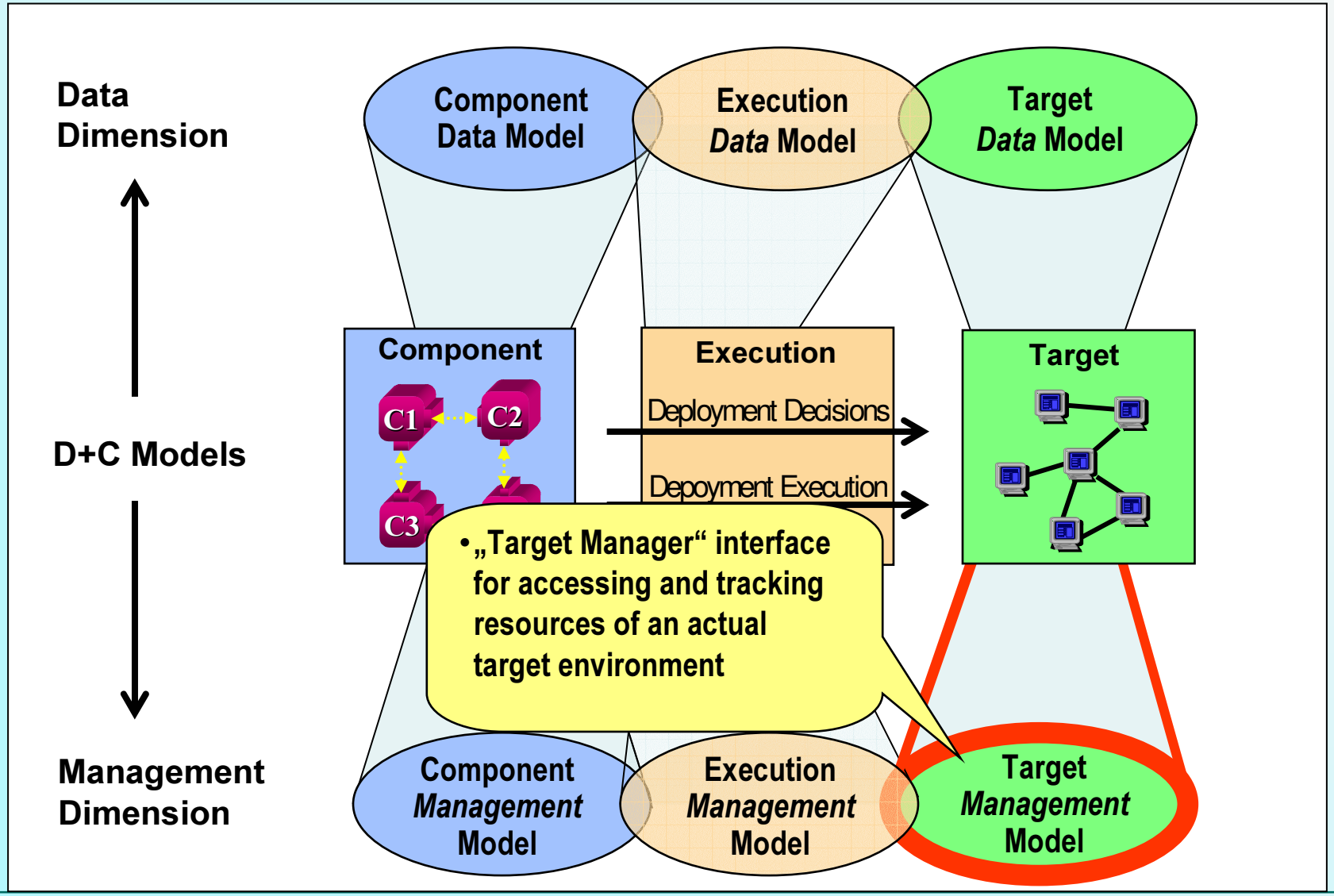


- Nodes are target for execution, Interconnects & Bridges are target for connections
- Node, Interconnect and Bridge have Resources
  - Nodes: e.g. processors, hardware devices, memory, operating system
  - Interconnect: bandwidth, protocol
- Resources may be shared among Nodes

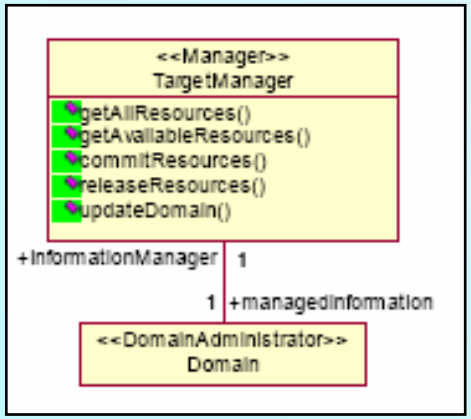
Resources are Typed



Target Management Model

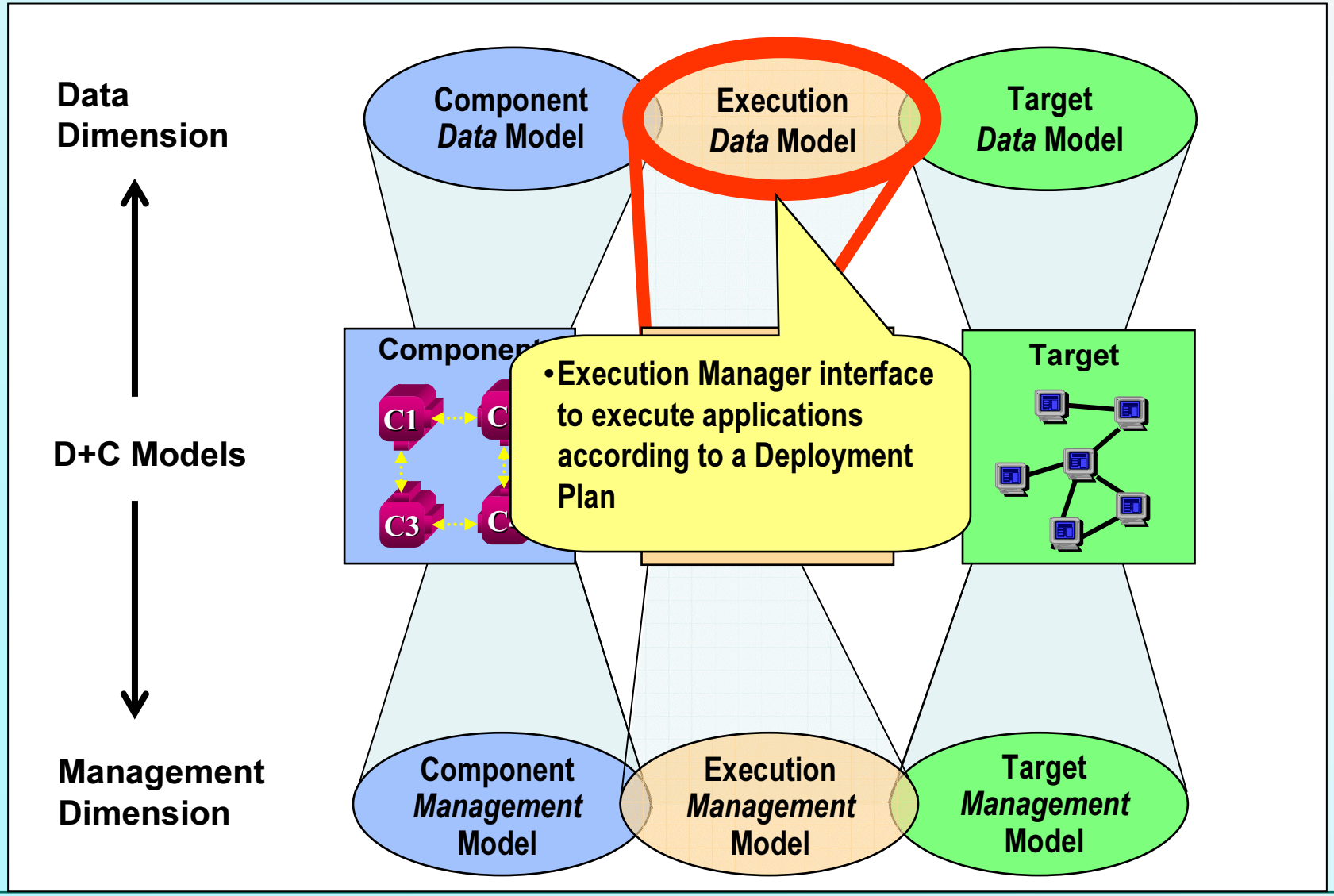


Target Management Model



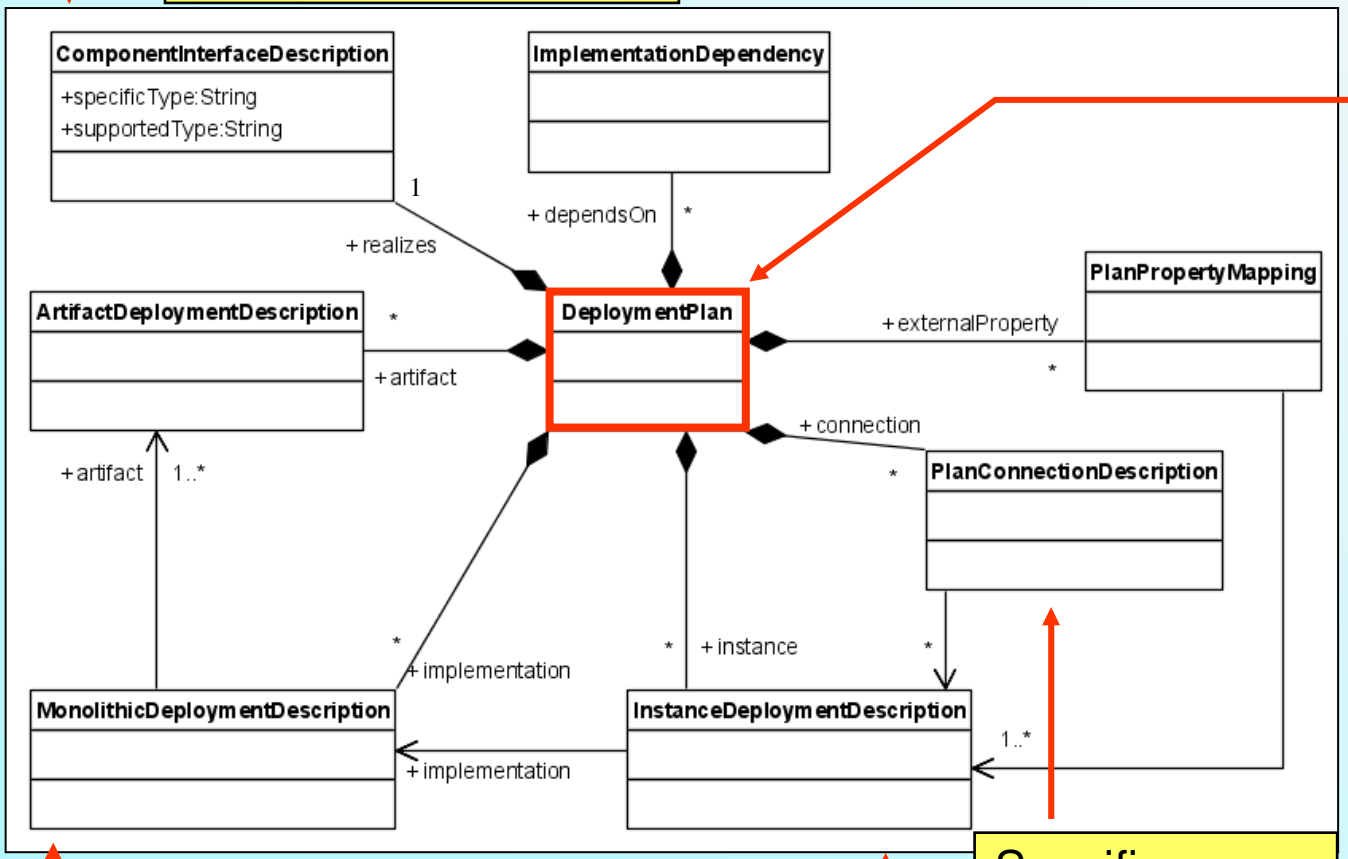
- Provides information about the domain needed for deployment planning
  - getAllResources
  - getAvailableResources
- Manages all available resource in a domain centrally
  - commitResources
  - releaseResources
- Updates domain information

Execution Data Model



Deployment Plan

Component definition



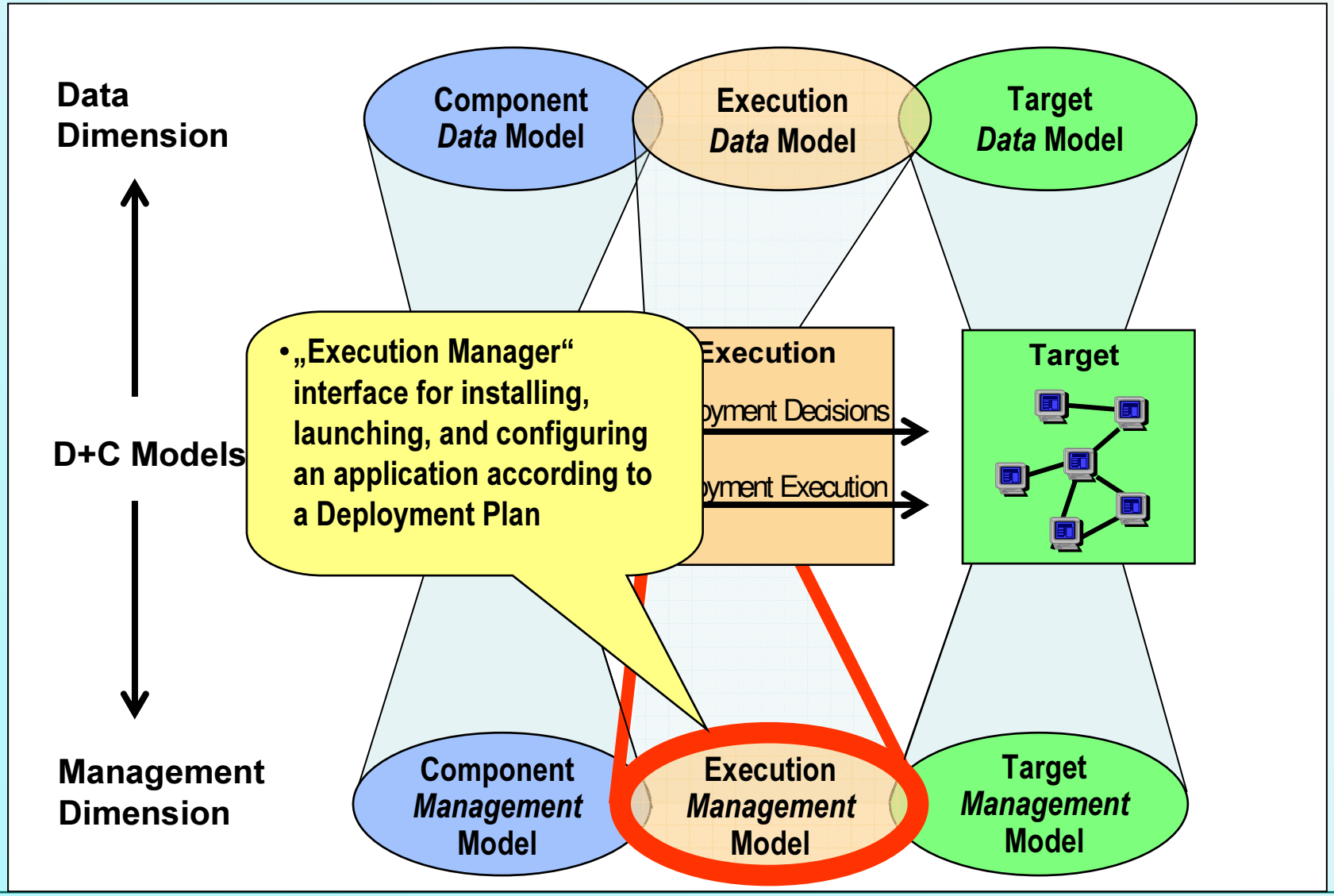
Specifies deployment of a particular component impl.

Specifies interconnected instances

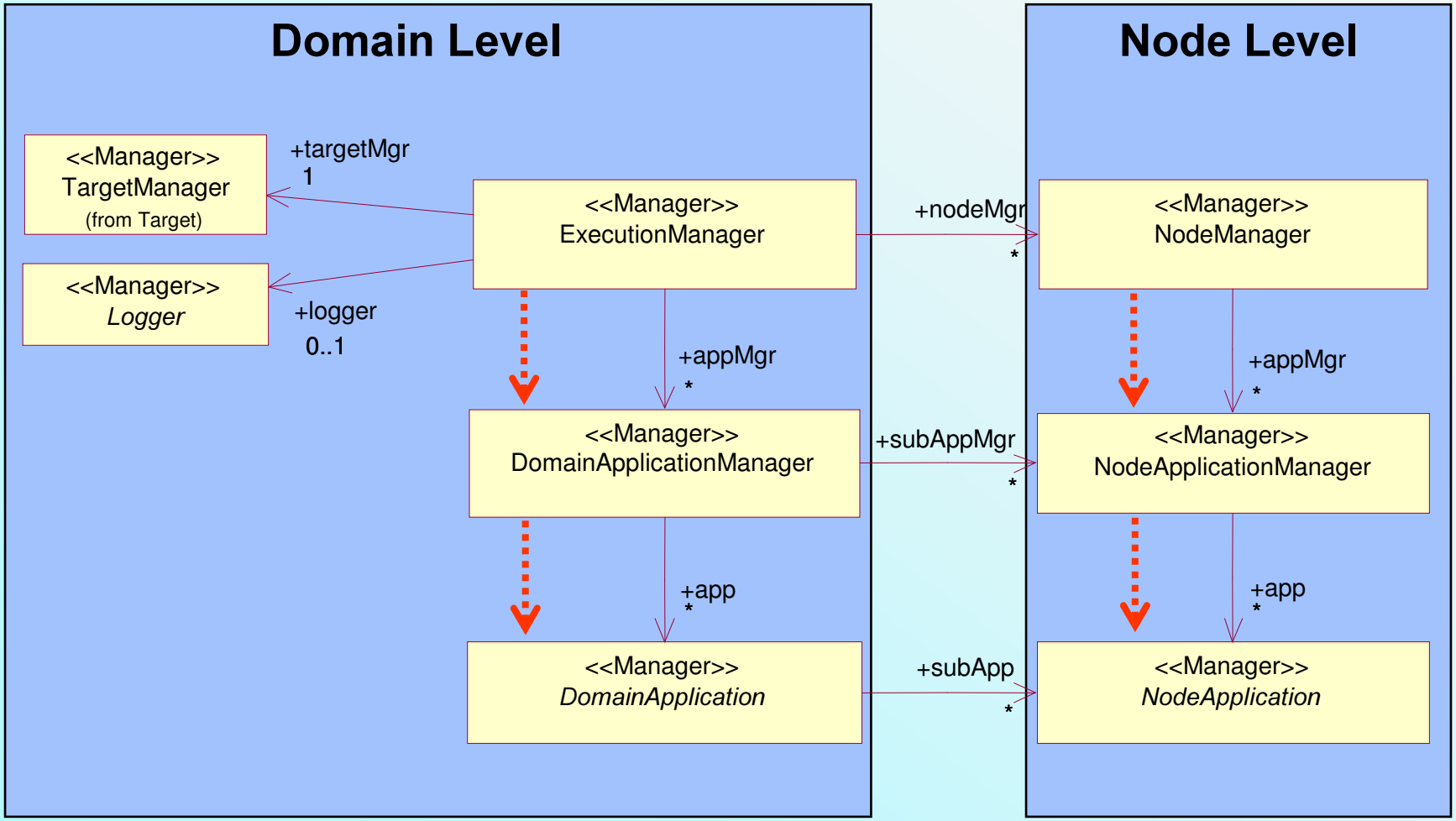
DeploymentPlan:

- Refers to the application's descriptions and the actual target environment
- Maps component implementations to nodes and connections between component instances to bridges and interconnects
- Records matching properties against resources

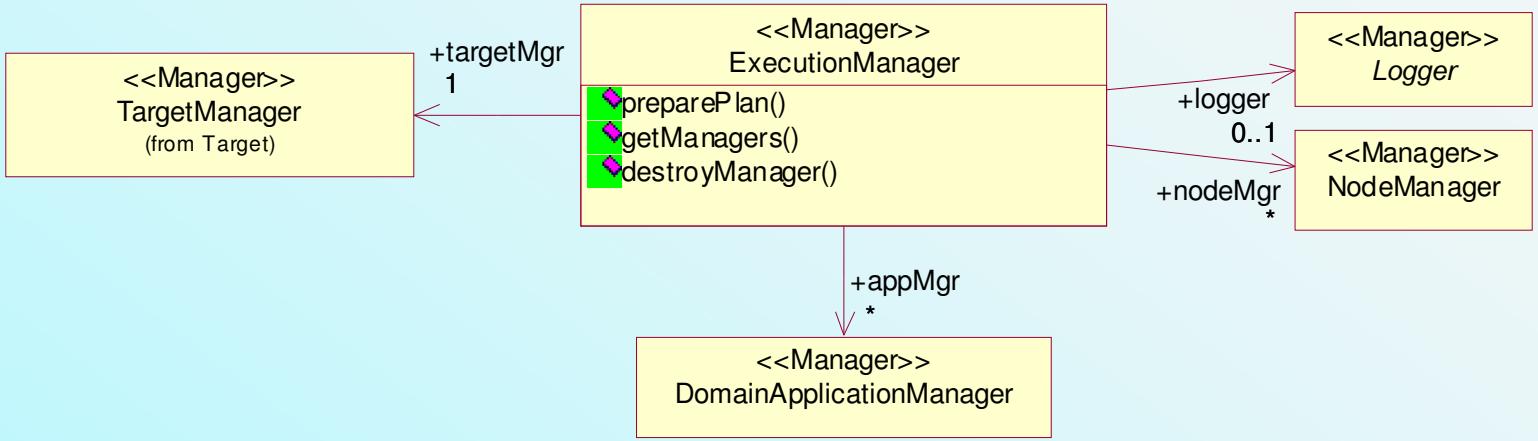
Execution Management Model



Execution Mgmt. Model Overview

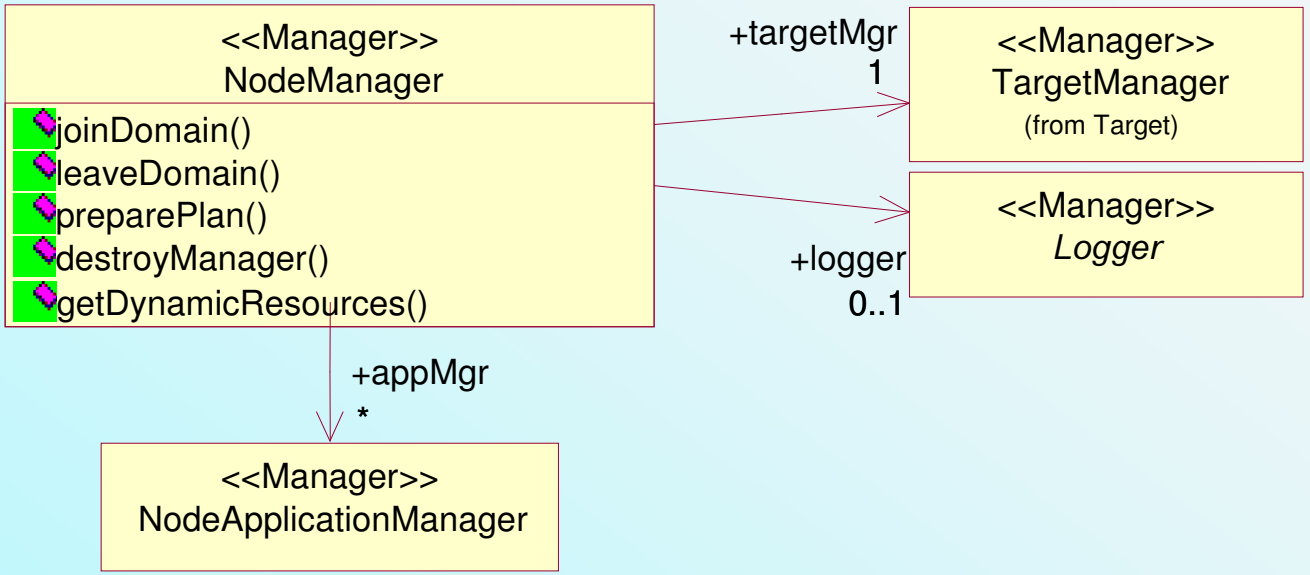


Execution Manager



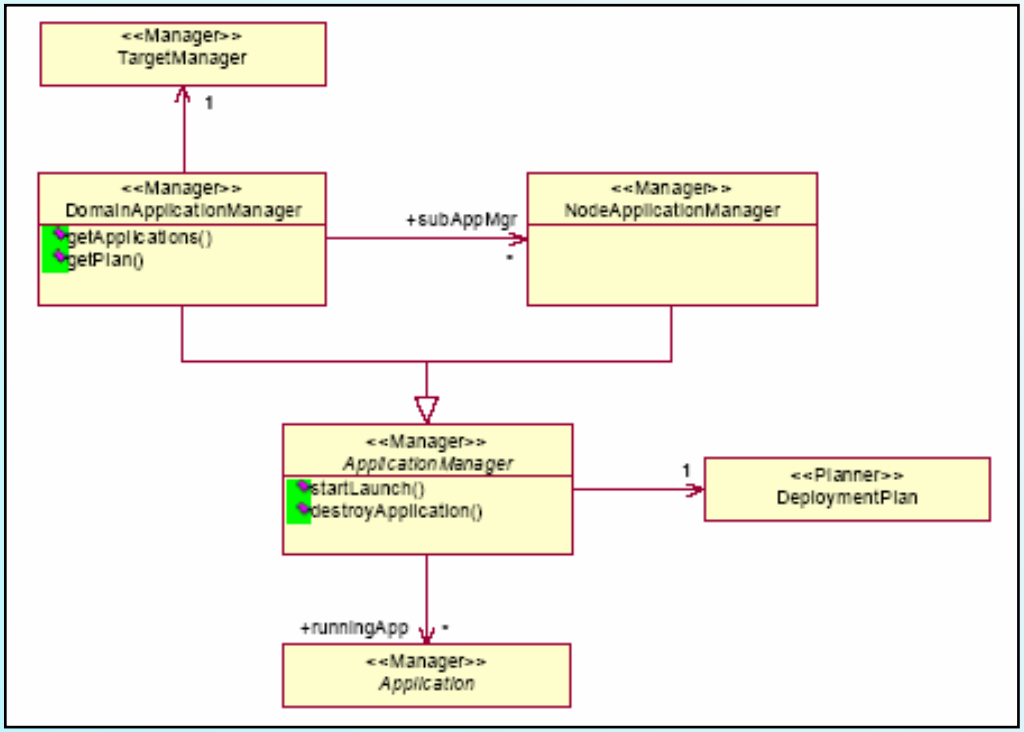
- **DeploymentPlan** contains all deployment decisions
  - What artifact is to be installed on which node
  - How to bootstrap the application
- **ExecutionManager** takes DeploymentPlan and returns a DomainApplicationManager
  - Different approaches possible (early or late XML processing and code upload)
- **DomainApplicationManager** is a generic factory for launching application instances (using NodeApplicationManager)
- Resource management using the **TargetManager**

Node Manager



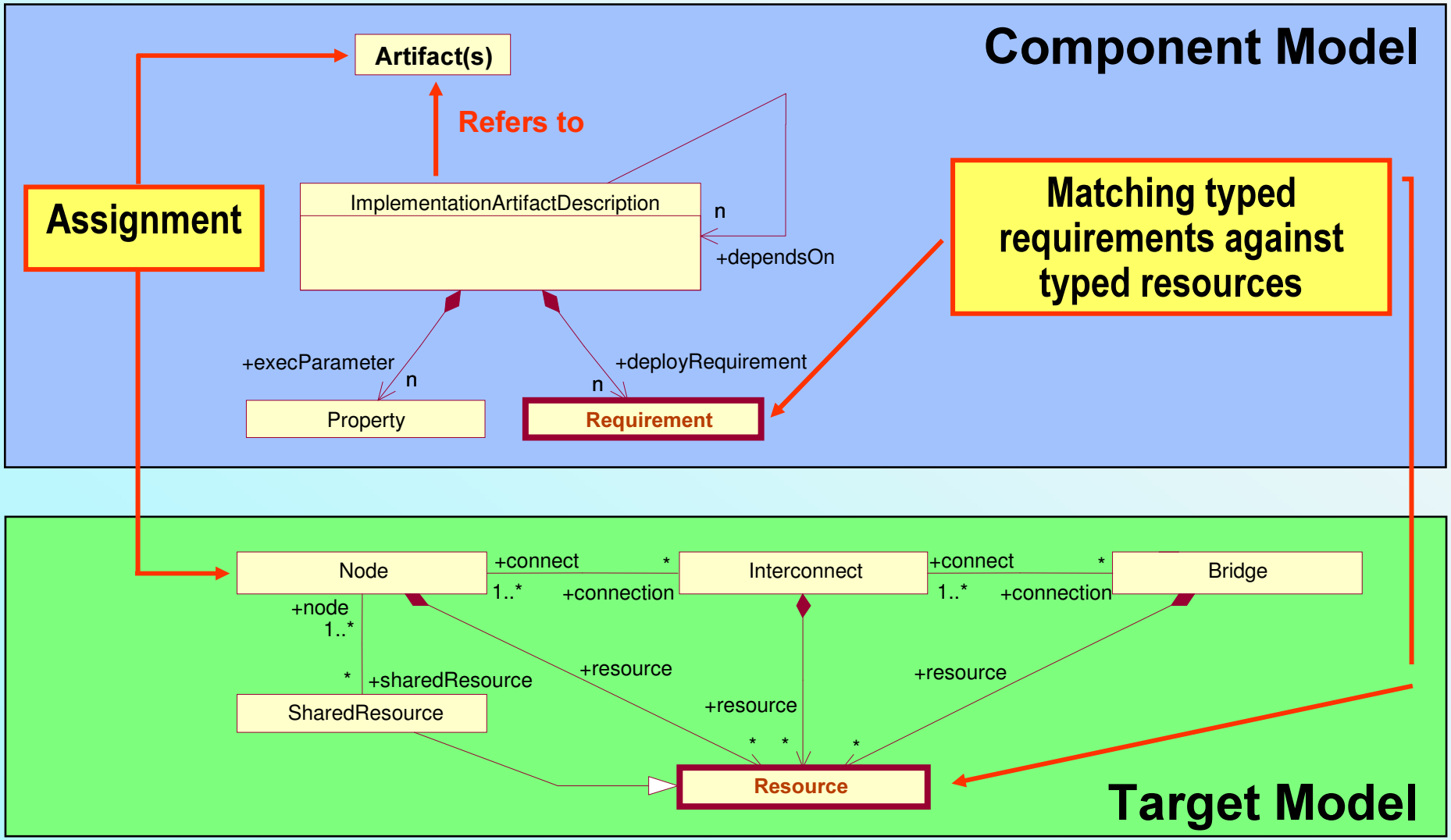
- Registration with ExecutionManager
- Local mirror of ExecutionManager for one node
- Creates NodeApplicationManager
  - NodeApplicationManager manages a partial application limited to its node
- NodeManager important for
  - Heterogeneous multi-vendor assemblies
  - Heterogeneous execution environments integrating special nodes (with special capabilities or operating systems)

Application Managers

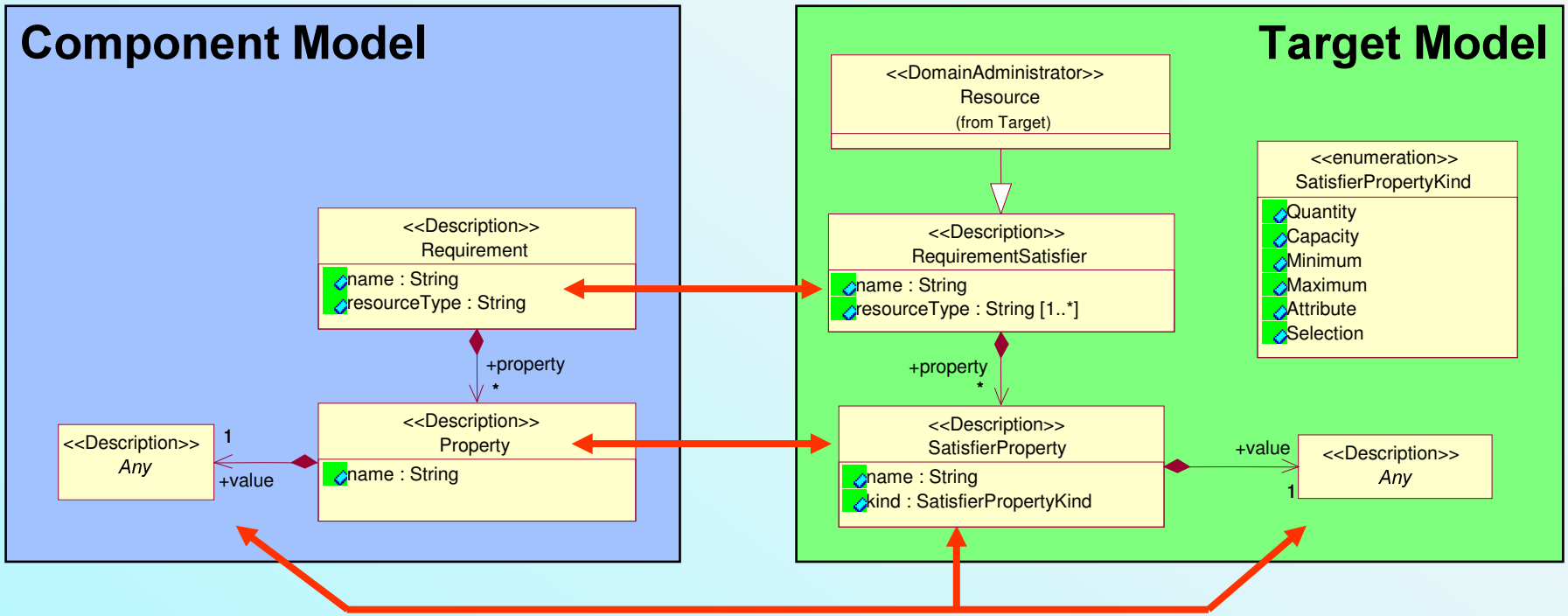


- *ApplicationManager* launches and terminates an application according to its DeploymentPlan
- *DomainApplicationManager* handles “global” application
- *NodeApplicationManager* handles partial application locally limited to a single node

Deployment Matching Mechanism



Deployment Matching Mechanism (cont'd)



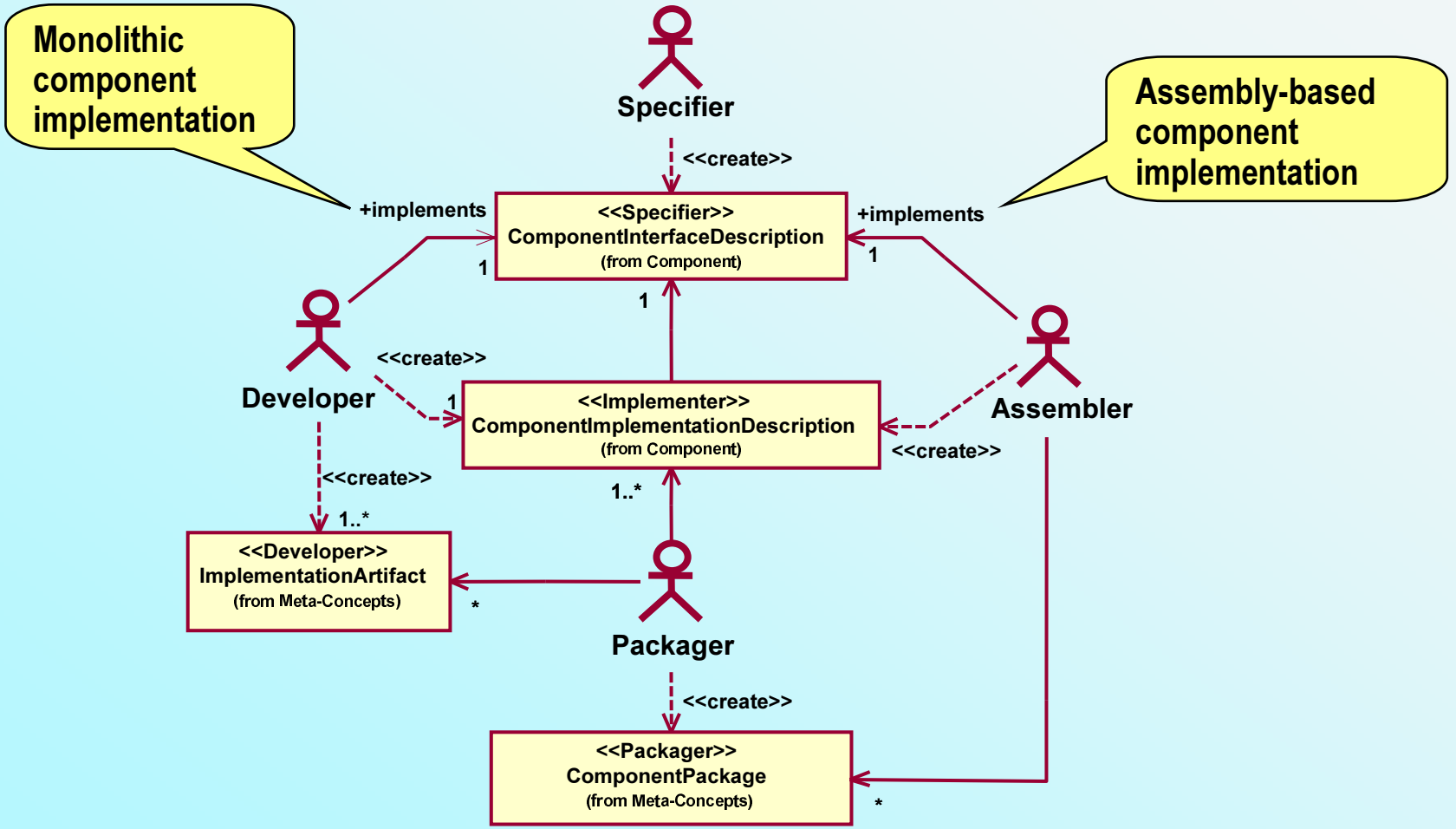
- Generic grammar for defining resources and requirements
- Well-defined, generic matching algorithm

Compliance Points

- **Compliance Points for PIM:**
  - **Goal: to enable different vendor implementations**
  - RepositoryManager
  - TargetManager (offline or online planning at runtime)
  - NodeManager (part of deployment system related to node OS, ORB etc.)
  - ExecutionManager (core of deployment run-time system)
- **Compliance Points for PSMs:**
  - PSM for CCM defines all the above compl. points
  - Other PSMs may define their own compliance points

Table of Contents

Development Actors



Deployment Actors

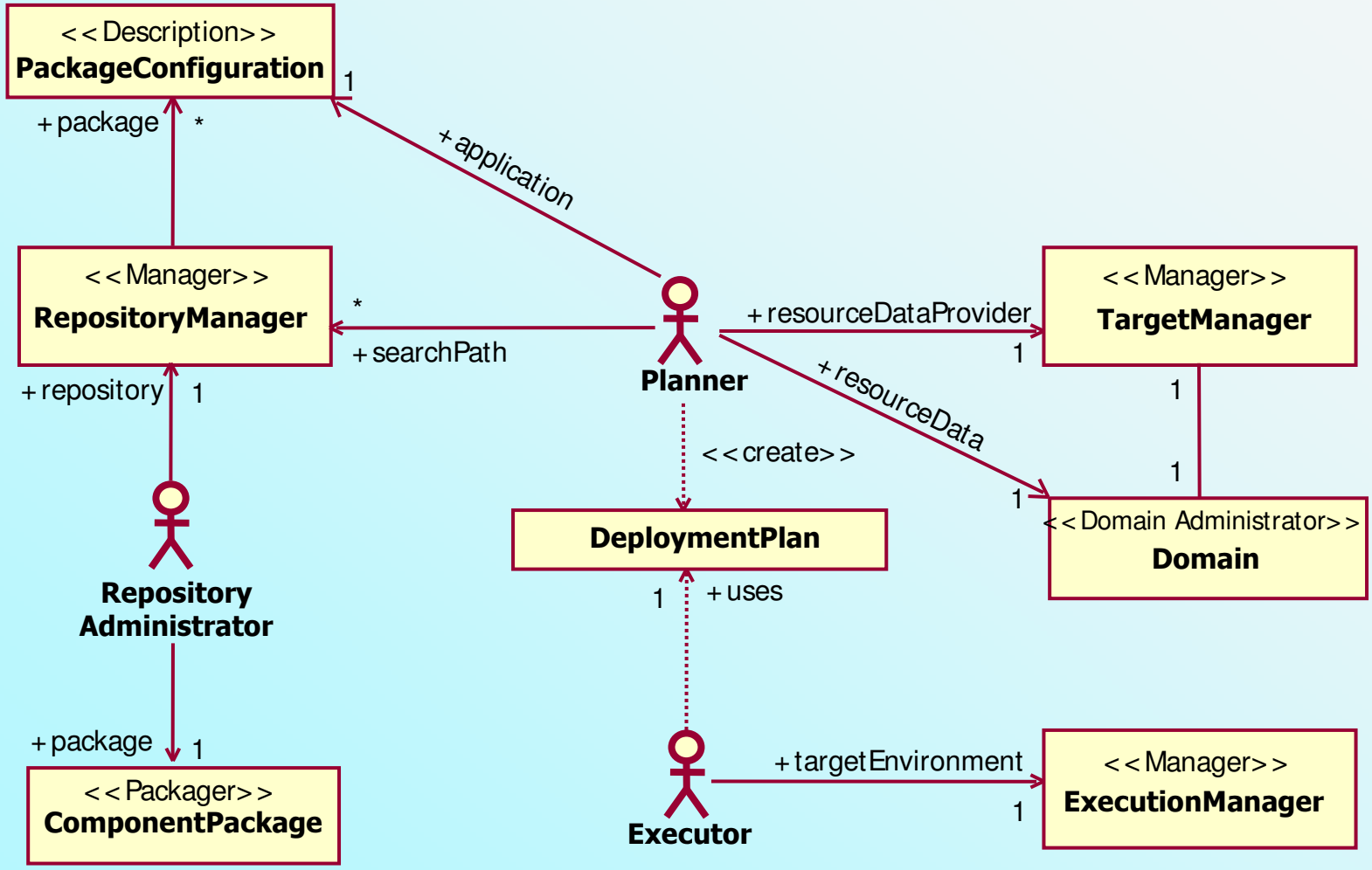
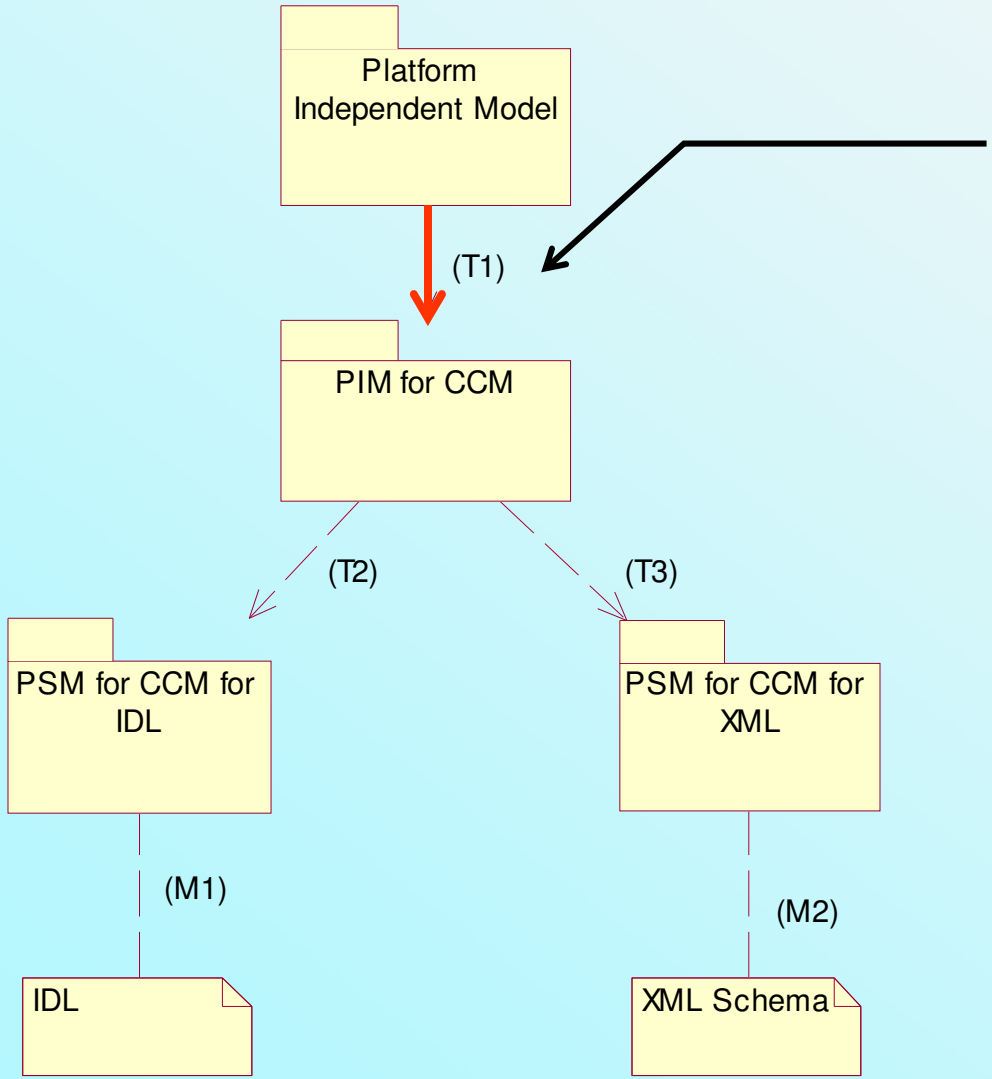


Table of Contents

PSM(s) for CCM

- **Usage of D+C data models for:**
  1. persistent storage and distribution of information
     => mapping to XML schema
  2. Representing data at run-time
     =>mapping management classes to IDL
- **Definition of two PSMs for CCM and the transformation from the PIM to these PSMs**
  - One PSM for XML schema generation
    (for generation of XML files out of data models)
  - One PSM for IDL generation
    (for generation of XML files out of data models)
- **Definition of proper rules for automatic XML schema and IDL generation**

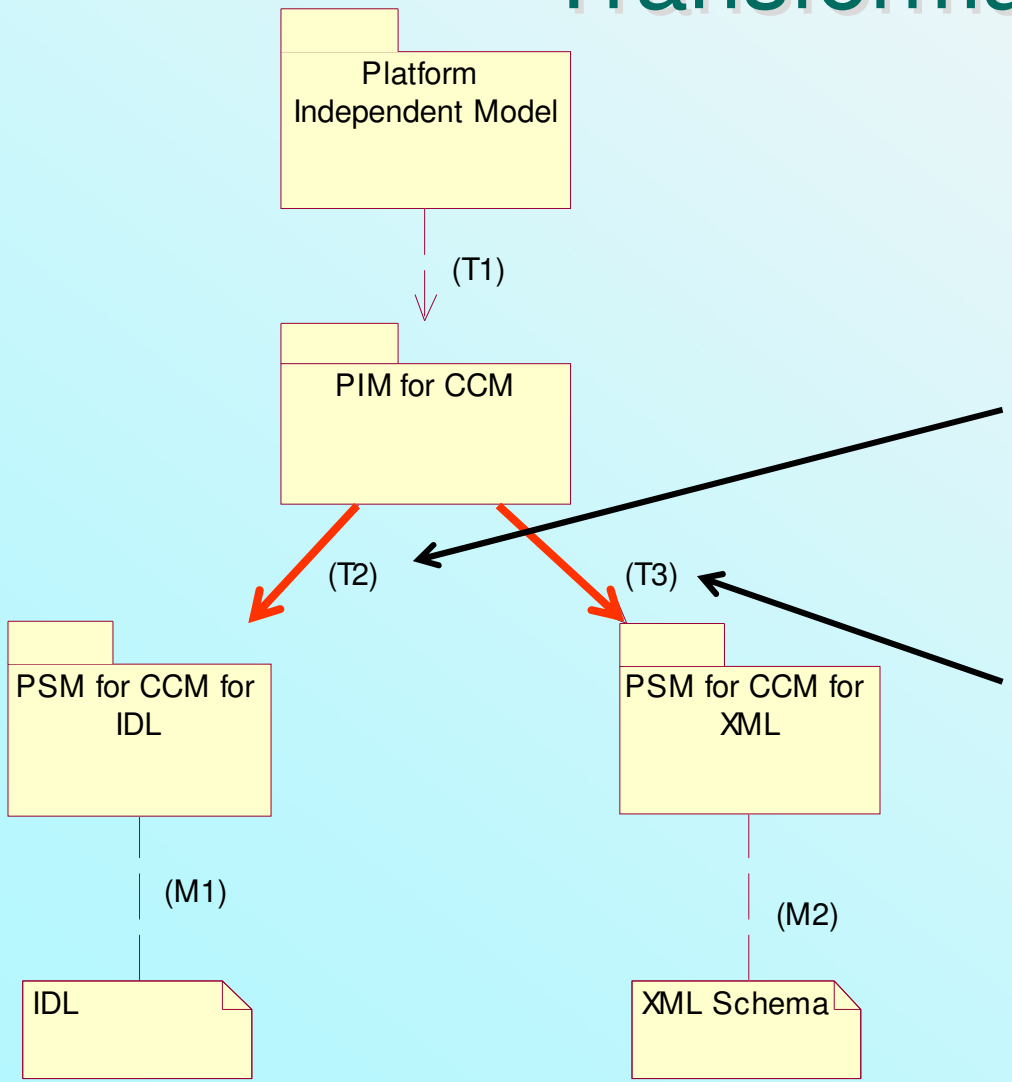
PIM to PIM for CCM Transformation



Transformation (T1):
PIM to PIM for CCM

- **Concretization of abstract meta-concepts**
- **Alignment of other classes with CCM**
- **Changes to attributes, associations and semantics of some classes**

PIM to PSM for CCM for IDL/XML Transformation



Transformation (T2):
PIM for CCM to
PSM for CCM for IDL

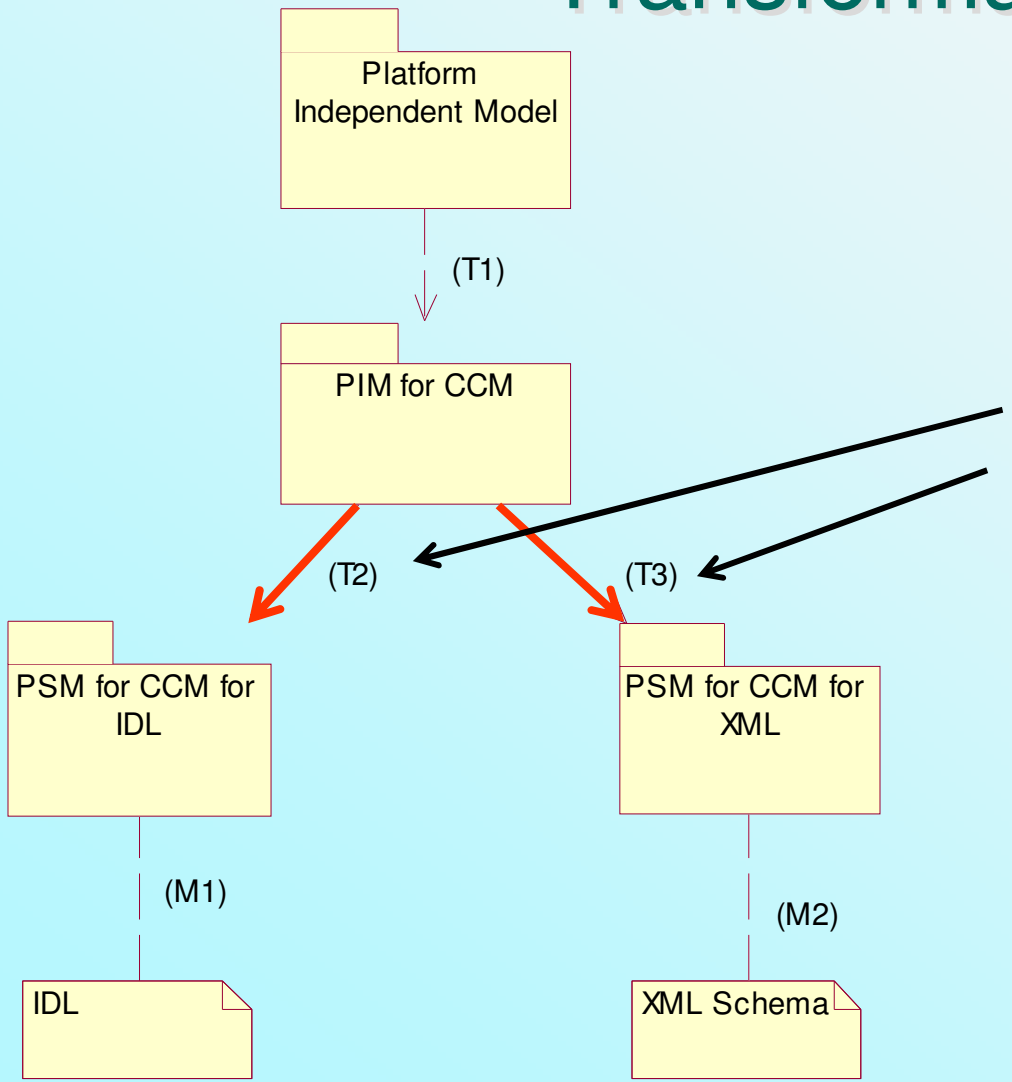
- **Transformation into a model enabling IDL generation**

<<Exception>> => <<CORBAException>>
<<Manager>> => <<CORBAInterface>>

Transformation (T3):
PIM for CCM to
PSM for CCM for XML

- **Transformation into a model enabling XML generation**

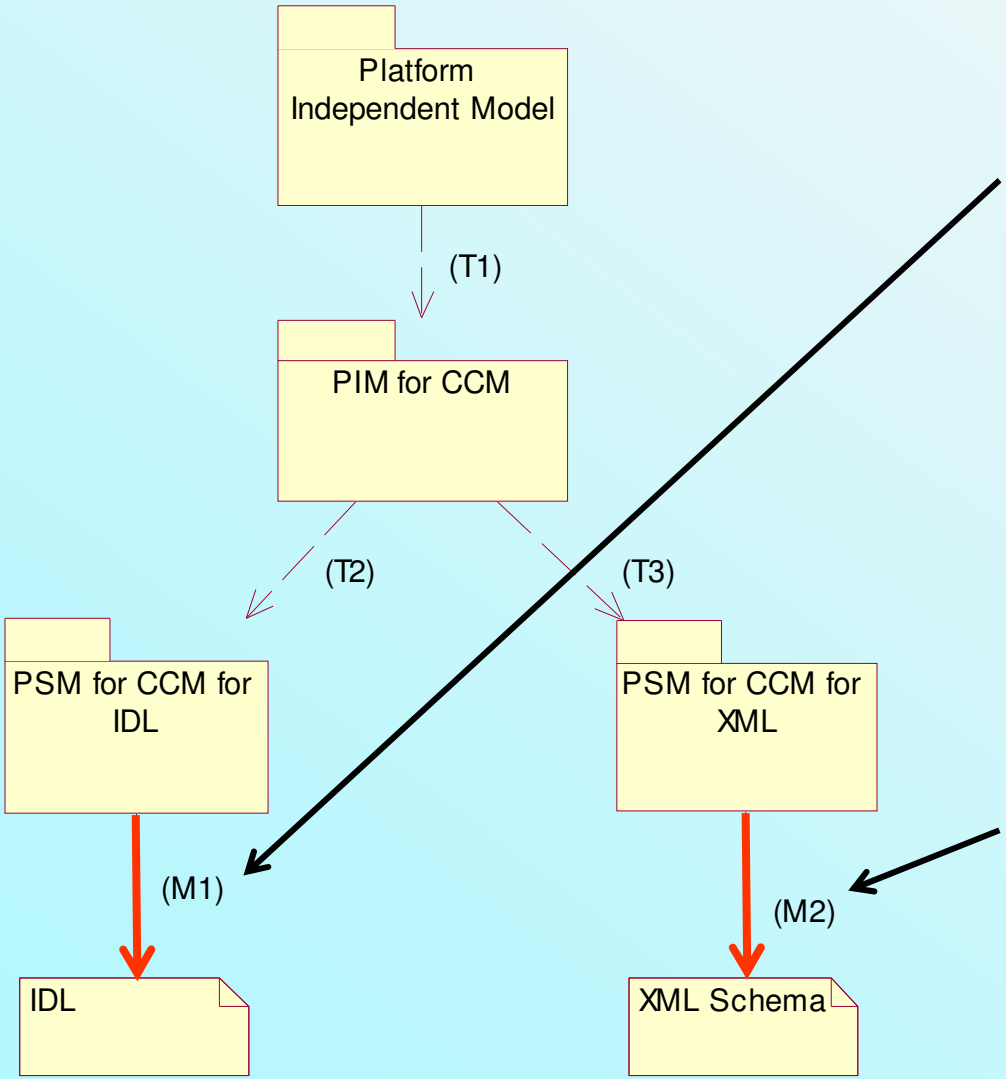
PIM to PSM for CCM for IDL/XML Transformation



Rational for (T2) and (T3)

- **Transformation into PSMs so that generic rule-based mappings (M1) and (M2) can be applied**
- **Reason: Some classes have different representations in IDL and XML (e.g. Any)**
- **Thus IDL and XML schema cannot be generated from same model**

PSM for CCM Mappings



Mapping (M1):
PSM for CCM for IDL

- **Transformation into a model enabling IDL generation**
- **Use of UML Profile for CORBA**

Mapping (M2):
PIM for CCM to PSM for CCM for XML

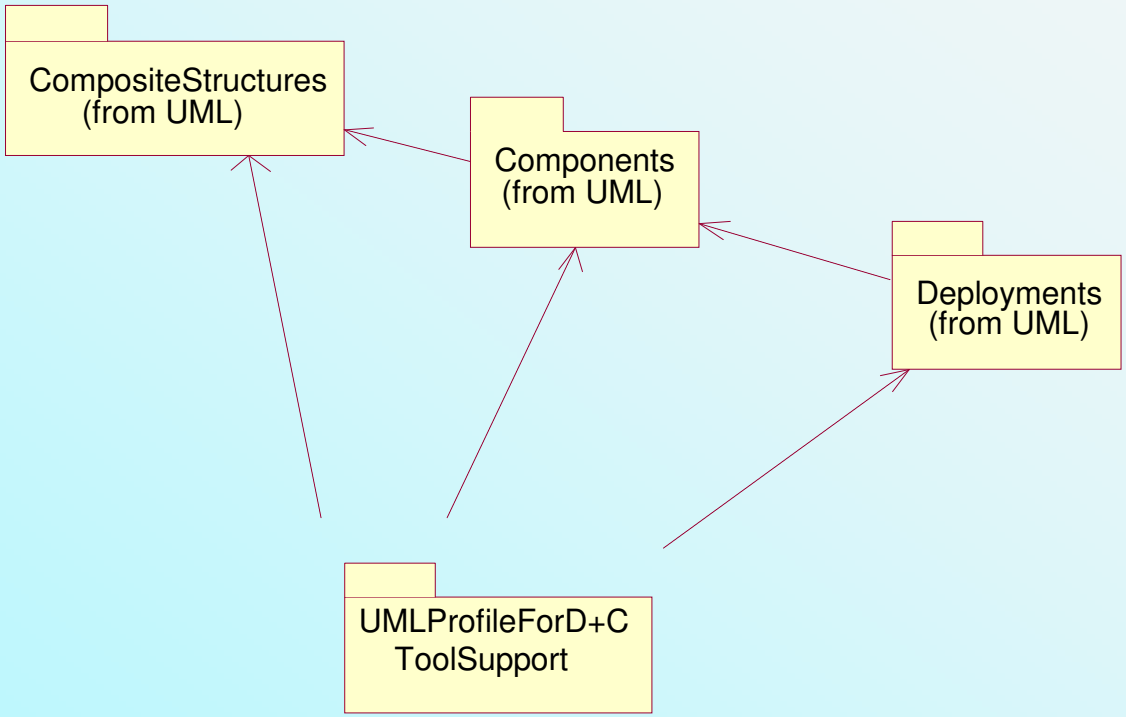
- **Transformation into a model enabling XML schema generation**

Table of Contents

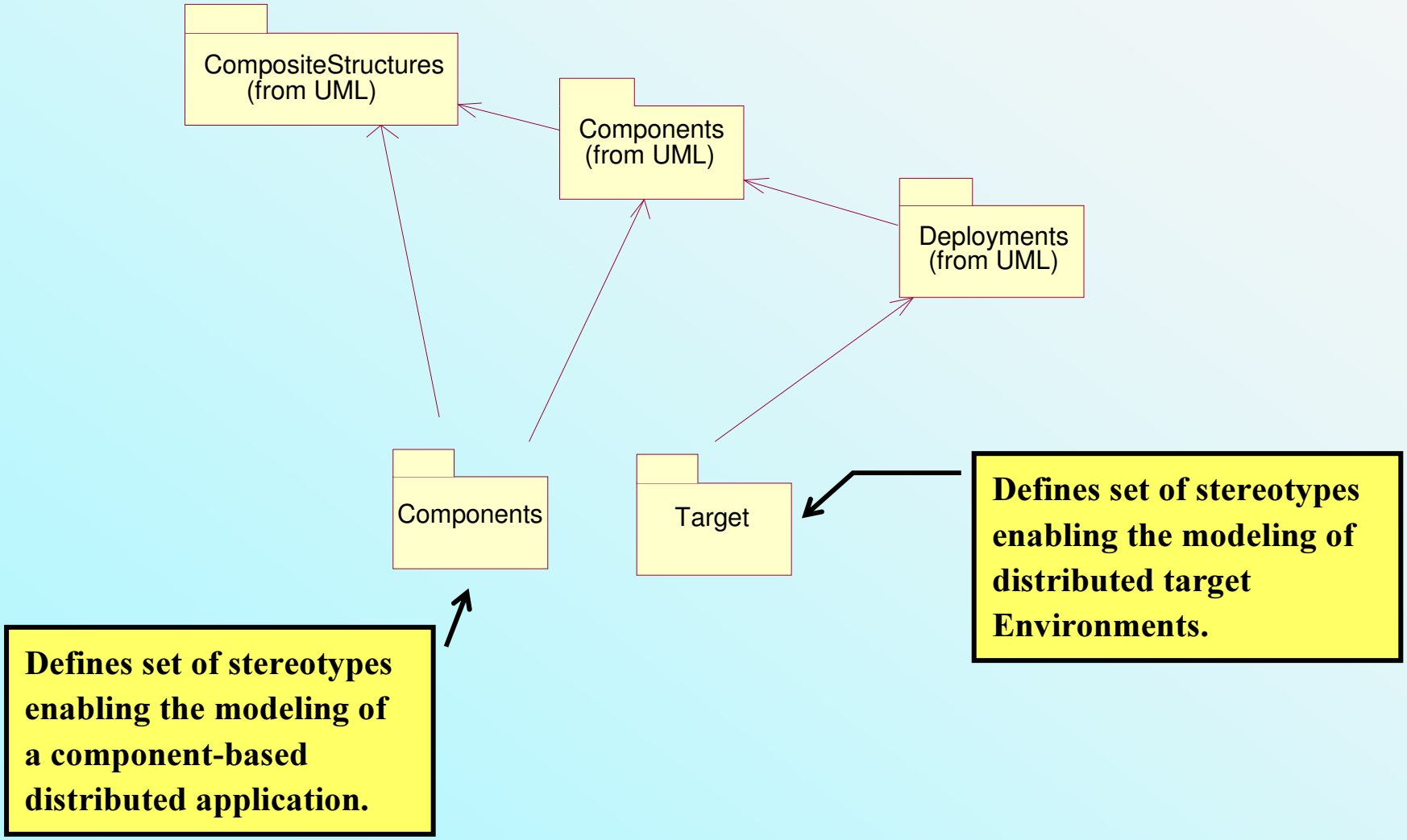
UML Profile for D+C Tool Support

- **Scope of this profile:**
  - components and target env. description (from PIM)
  - does not address deployment infrastructure (execution model as defined by PIM)
- **Major objectives of this UML Profile are:**
  - to define the ***notation*** necessary to support the component-based application development process and target environment description
  - to enable the ***automatic generation of D+C descriptors*** from both component assembly and target models.

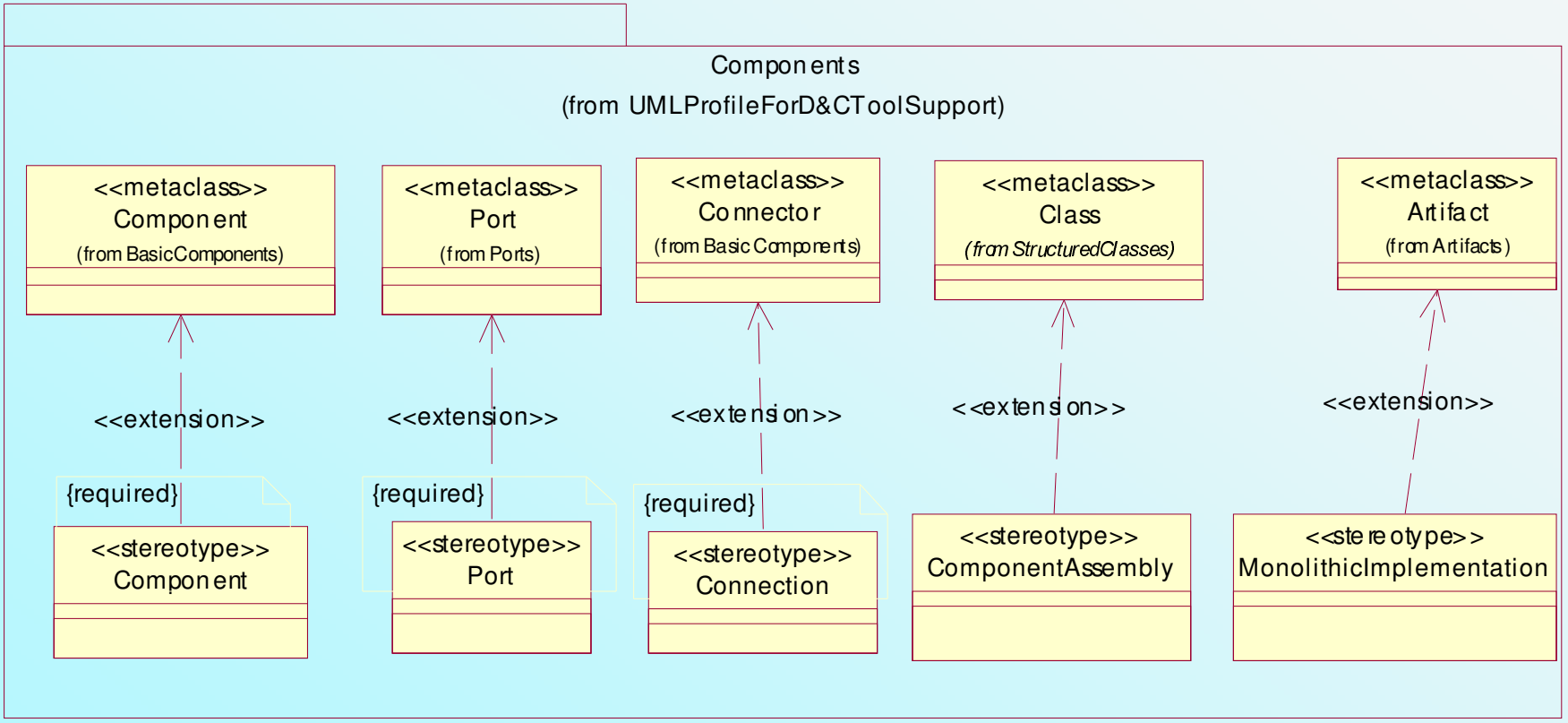
UML Profile Structure



UML Profile Structure



Components Package Overview



Target Package Overview

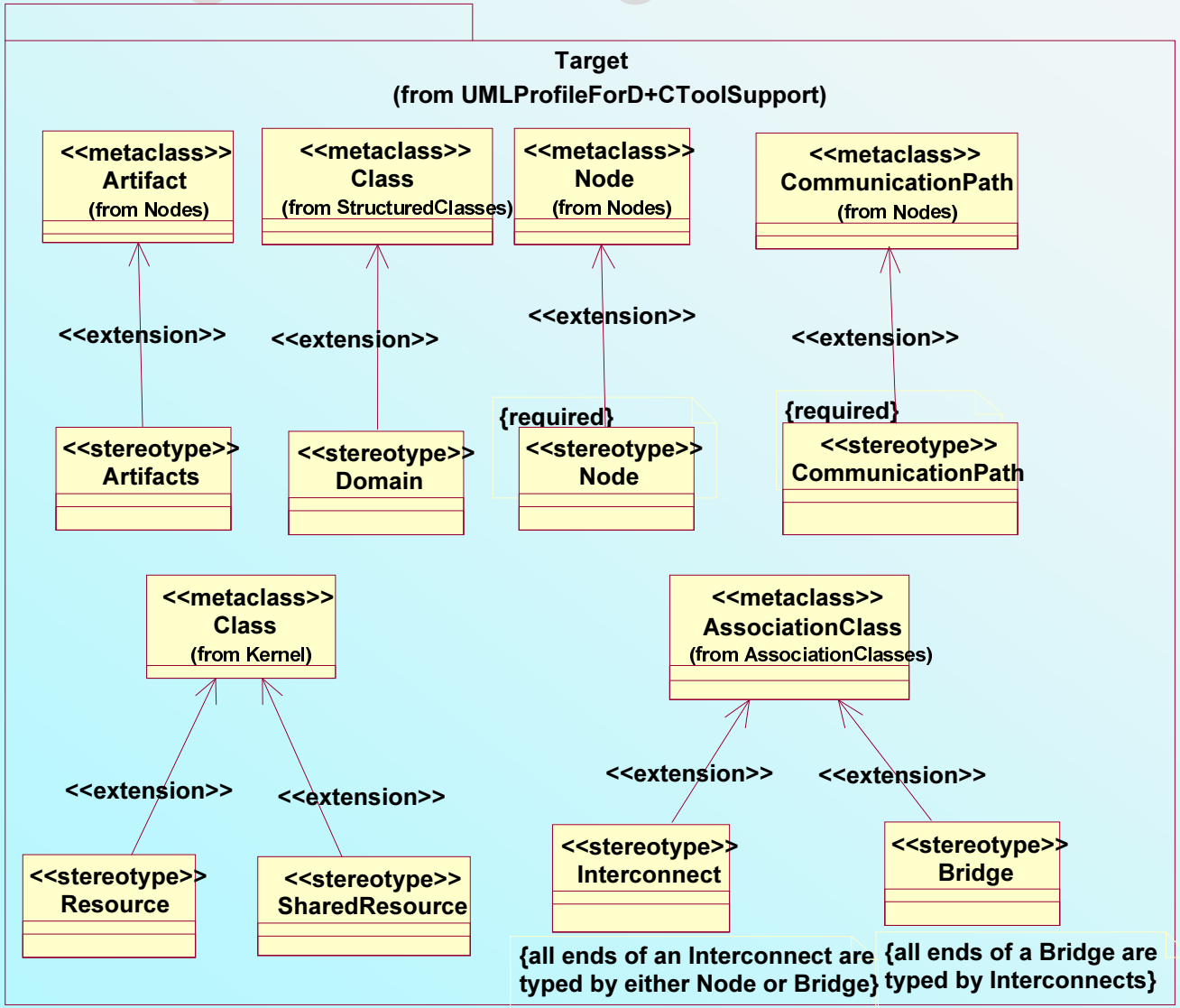


Table of Contents

Comparison of D+C concepts with UML2

- ***Meanwhile UML2 was adopted:***
  - What is the difference between UML2 and D+C?
  - Comparison of D+C's deployment concepts with UML2's ones
- ***Both***
  - are suitable for modeling distributed systems at *platform-independent* (PIM) level
  - according to OMG's Model Driven Architecture (MDA) approach, and
  - have a MOF-based Metamodel.

Overview on UML 2.0

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

Overview on Compared Concepts

	UML 2.0	D+C
Component Modeling	Component	ComponentInterfaceDescription
	Port	ComponentPortDescription
	(nested) Component	ComponentImplementationDescription, ComponentAssemblyDescription
	Artifact (as component manifestation)	ComponentImplementationDescription, MonolithicImplementationDescription, ImplementationArtifactDescription
	(name-value annotation)	typed Requirement
Target Environment Modeling	-	Domain
	Node	Node
	CommunicationPath	Interconnect, Bridge
	(name-value annotation)	typed Resource and SharedResource
Distribution Modeling	([static] deployment diagram)	DeploymentPlan
	Deployment	ArtifactDeploymentDescription
	InstanceSpecification	InstanceDeploymentDescription
	-	<typed Resources meet typed Requirements>
Run-time Management	-	ApplicationManager, Application, Target-,Node-,ExecutionManager

Table of Contents

Improvements to current CCM

- **Common MDA/MOF/XMI conform model for meta-data generates both IDL and XML schema**
- **New meta-data models for**
  - target environment (including topology and resources)
  - implementation capabilities and requirements
  - deployment decisions
- **Nesting of assemblies (using their external interfaces)**
  - allows reuse of assemblies/packages without changes
- **Automated distr. deployment (interoperable between vendors)**
- **Well-defined deployment planning phases (on-line/off-line)**
- **Generic support for QoS features**
- **Support for heterogeneous multi-vendor target environments and assemblies**

Summary

- Follows MDA:
  - Definition of PIM for Deployment and Configuration for distributed component-based applications
  - PSM(s) for CCM and appropriate model transformations
  - Rules for automatic generation of IDL and XML schemata from same model
- Compatible with XMI 2.0, MOF 1.4 and aligned with UML 2.0 concepts
- Several improvements to current CCM
- Support for heterogeneous multi-vendor target environments and assemblies