XML in the Development of Component Systems

XML and the CORBA Component Model

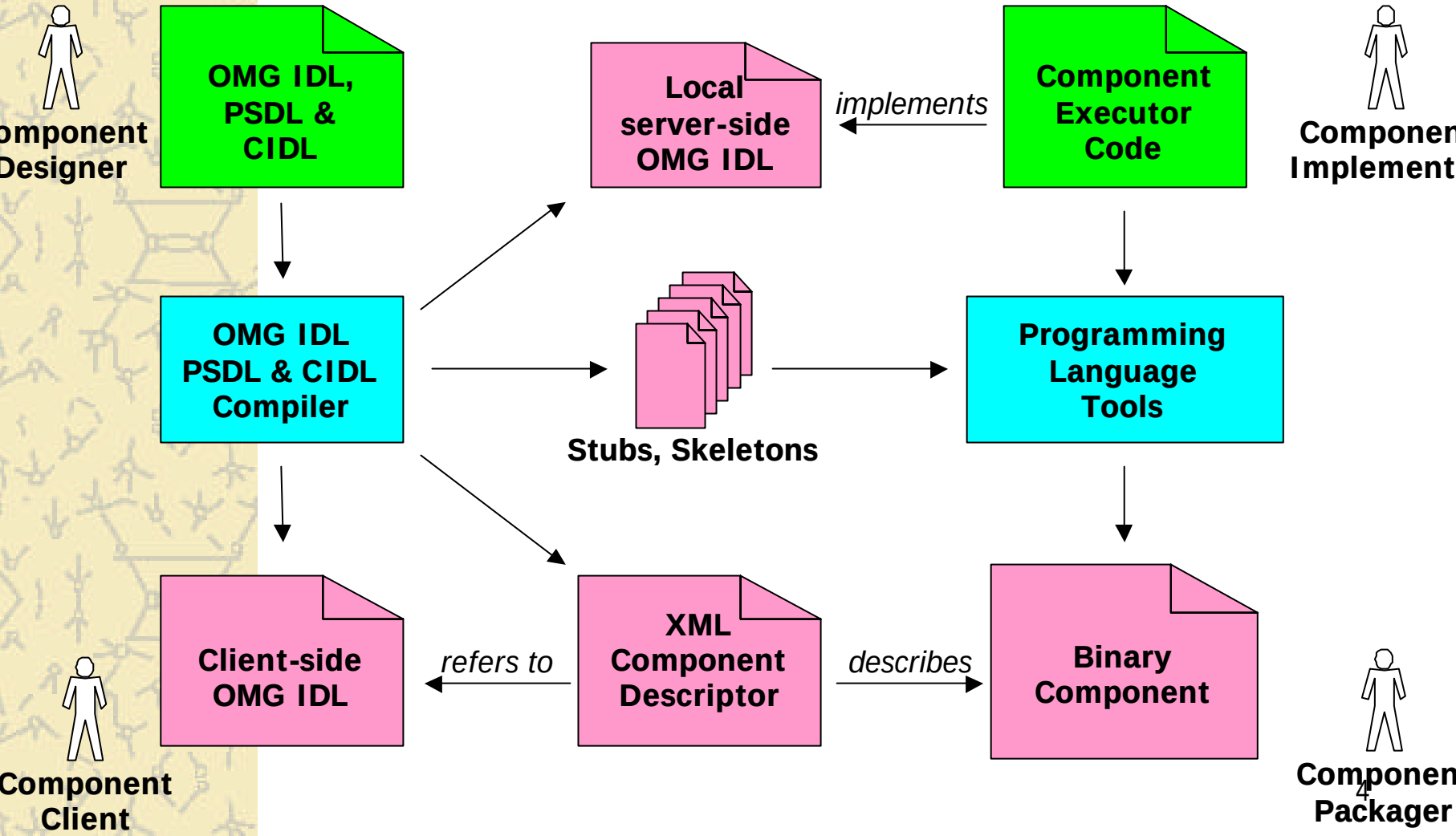
Components

- Szyperski (in „Component Software“):
  - A component is a unit of independent deployment
  - A component is a unit of third-party composition
  - (A component has no persistent state)
- Terminology: Objects, Components, Instances
  - Component Instances?

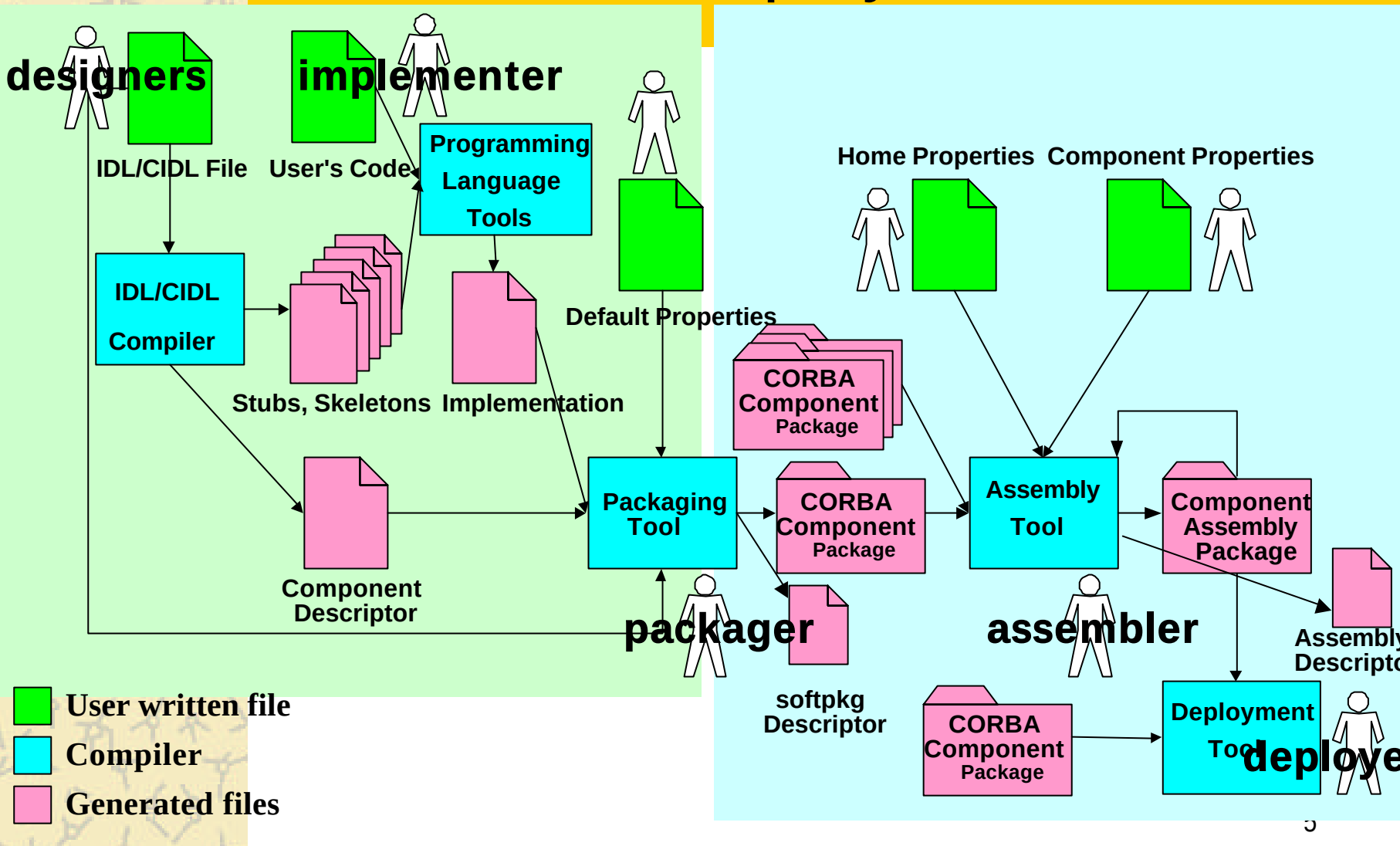
CORBA Component Model

- CCM Architecture:
  - IDLv3: Component Definitions
  - CIDL (Component Implementation Definition Language)
  - Container Architecture
  - (Supporting Services: Security, Transactions)
- Implementation framework and container framework to simplify state management and persistance
- Packaging and Deployment specification
- http://www.omg.org/technology/documents/formal/components.htm

From Design to Packaging ...



... to Deployment



Packaging

- “Classic” CORBA: No standard means of ...
  - Configuration
  - Distribution
  - Deployment
- Packaging and Deployment of Components
  - Components are packaged into a self-descriptive package
  - Packages can be assembled
  - Assemblies can be deployed
- Helped by XML descriptors

Component Package

- Archive (ZIP file) containing
  - one component, consisting of
    - one or more implementations (e.g. for different OSs)
    - IDL file
    - CORBA Component Descriptor (.ccd)
  - Property File Descriptor (.cpf) defining default attribute values
  - Software Package Descriptor (.csd) describing package contents
- Self-contained and self-descriptive, reusable unit
- Usually done by the component implementor

Component Assembly

- Archive (ZIP file) containing
  - one or more component packages, either
    - including a package's contents
    - including the original package
    - referencing the package by URL
  - Property File Descriptors defining initial attribute values
  - Component Assembly Descriptor (.cad)
    - defines home instances to be created
    - defines component instances to be created
    - defines connections between ports to be made
- Self-contained and self-descriptive unit
- For automatic and easy "one step" deployment
- No programming language experience necessary

XML Descriptors Overview

- Software Package Descriptor (.csd)
  - Describes contents of a component software package
  - Lists one or more implementation(s)
- CORBA Component Descriptor (.ccd)
  - Technical information mainly generated from CIDL
  - Some policy values editable by user
- Component Assembly Descriptor (.cad)
  - Describes initial virtual configuration
    - homes, component instances, and connections
- Component Property File Descriptor (.cpf)
  - name/value pairs to configure attributes

Software Package Descriptor (.csd)

- Descriptive elements
  - title, author, company, webpage, license
- Link to IDL file
- Link to property file
- Implementation(s)
  - information about Implementation
    - Operating System, processor, language, compiler, ORB
    - dependencies on other libraries
  - link to implementation file
    - shared library, Java class, executable
  - entry point

Software Package Descriptor Example

```
<?xml version='1.0'?>
<!DOCTYPE softpkg>
<softpkg name="PhilosopherHome">
  <idl id="IDL:DiningPhilosophers/PhilosopherHome:1.0">
    <fileinarchive name="philo.idl"/>
  </idl>
  <implementation id="*">
    <code type="DLL">
      <fileinarchive name="philo.dll"/>
      <entrypoint>create_DiningPhilosophers_PhilosopherHome</entrypoint>
    </code>
  </implementation>
</softpkg>
```

CORBA Component Descriptor (.ccd)

- Structural information generated by CIDL
  - component / home types and features
  - ports and supported interfaces
  - component category and segments
- Container policies filled by the packager
  - threading
  - servant lifetime
  - transactions
  - security
  - events
  - persistence
  - link to property files

CORBA Component Descriptor Example

```
<corbacomponent>
  <corbaversion>3.0</corbaversion>
  <componentrepid>IDL:DiningPhilosophers/Philosopher:1.0</componentrepid>
  <homerepid>IDL:DiningPhilosophers/PhilosopherHome:1.0</homerepid>
  <componentkind><session><servant lifetime="component"/></session></componentkind>
  <threading policy="multithread"/>
  <configurationcomplete set="true"/>
  <homefeatures name="PhilosopherHome" repid="IDL:...PhilosopherHome:1.0"/>
  <componentfeatures name="Philosopher" repid="IDL:...Philosopher:1.0">
    <ports>
      <publishes publishesname="info" eventtype="IDL:DiningPhilosophers/StatusInfo:1.0">
        <eventpolicy/>
      </publishes>
      <uses usesname="left" repid="IDL:DiningPhilosophers/Fork:1.0"/>
      <uses usesname="right" repid="IDL:DiningPhilosophers/Fork:1.0"/>
    </ports>
  </componentfeatures>
</corbacomponent>
```

Property File Descriptor (.cpf)

- Contains zero or more name/value pairs to configure attributes
- Referenced by...
  - Software Package Descriptors to define default values for component attributes
  - CORBA Component Descriptors to define default values for component or home attributes
  - Assembly Descriptors to configure initial values for home or component instances

Property File Descriptor Example

```
<?xml version='1.0'?>
<!DOCTYPE properties>
<properties>
  <simple name="name" type="string">
    <value>Socrates</value>
  </simple>
</properties>
```

Component Assembly Descriptor (.cad)

- References one or more Component Software Descriptors
- Defines home instances and their collocation contraints
- Defines components to be instantiated
- Defines that homes, components or ports are to be registered in the Naming or Trading Service
- Defines connections to be made between component ports

Component Assembly Descriptor Example

```
<?xml version='1.0'?>
<!DOCTYPE componentassembly>
<componentassembly id="demophilo">
  <componentfiles>
    <componentfile id="ObserverHome">
      <fileinarchive name="ObserverHome.csd"/>
    </componentfile>
    <componentfile id="PhilosopherHome">
      <fileinarchive name="PhilosopherHome.csd"/>
    </componentfile>
    <componentfile id="ForkHome">
      <fileinarchive name="ForkHome.csd"/>
    </componentfile>
  </componentfiles>
```

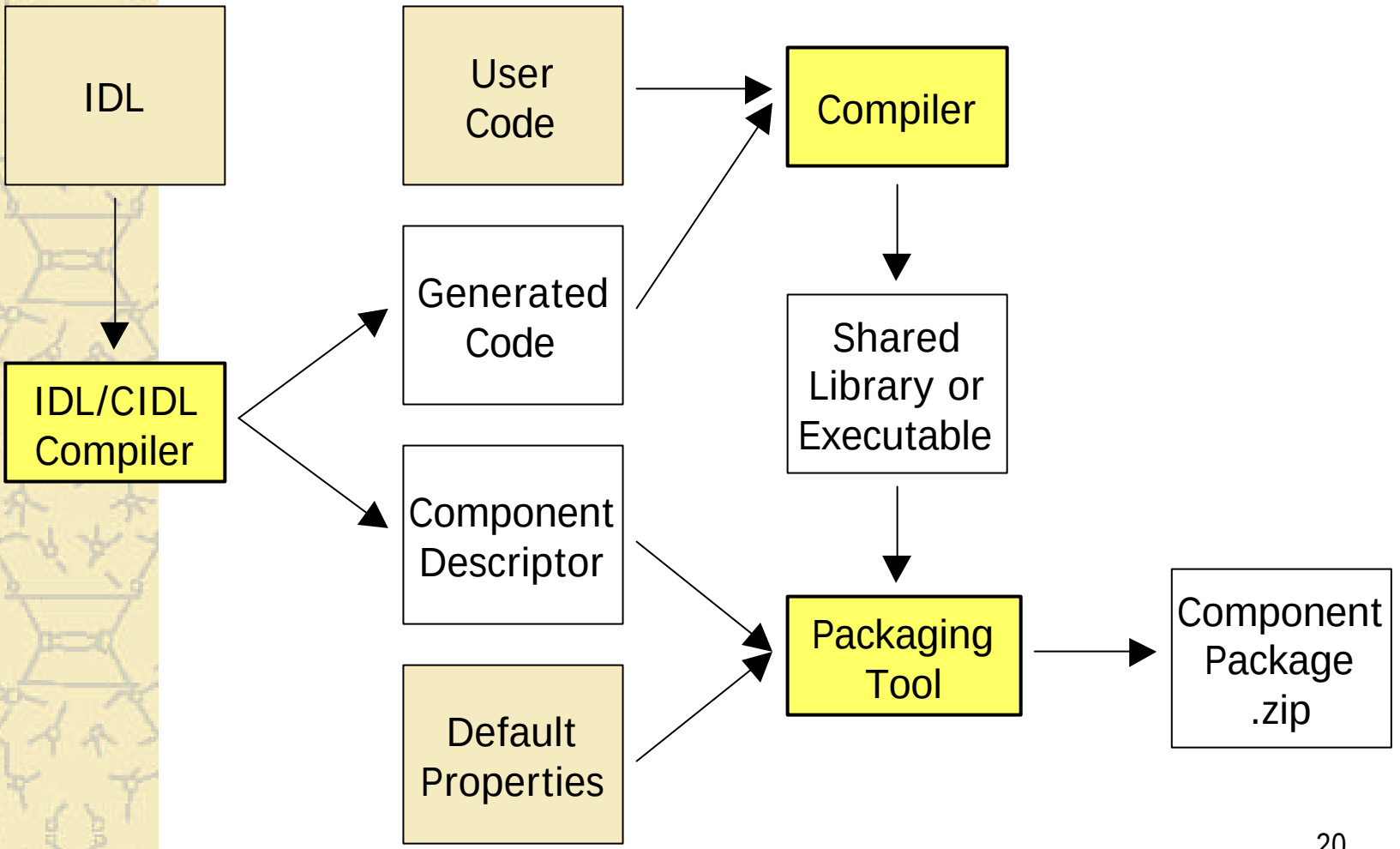
Assembly Descriptor Example (2)

```
<partitioning>
  <homeplacement id="ObserverHome">
    <componentfileref idref="ObserverHome"/>
    <registerwithnaming name="ObserverHome"/>
  </homeplacement>
  <homeplacement id="PhilosopherHome">
    <componentfileref idref="PhilosopherHome"/>
    <registerwithnaming name="PhilosopherHome"/>
  </homeplacement>
  <homeplacement id="ForkHome">
    <componentfileref idref="ForkHome"/>
    <registerwithnaming name="ForkHome"/>
  </homeplacement>
</partitioning><connections/></componentassembly>
```

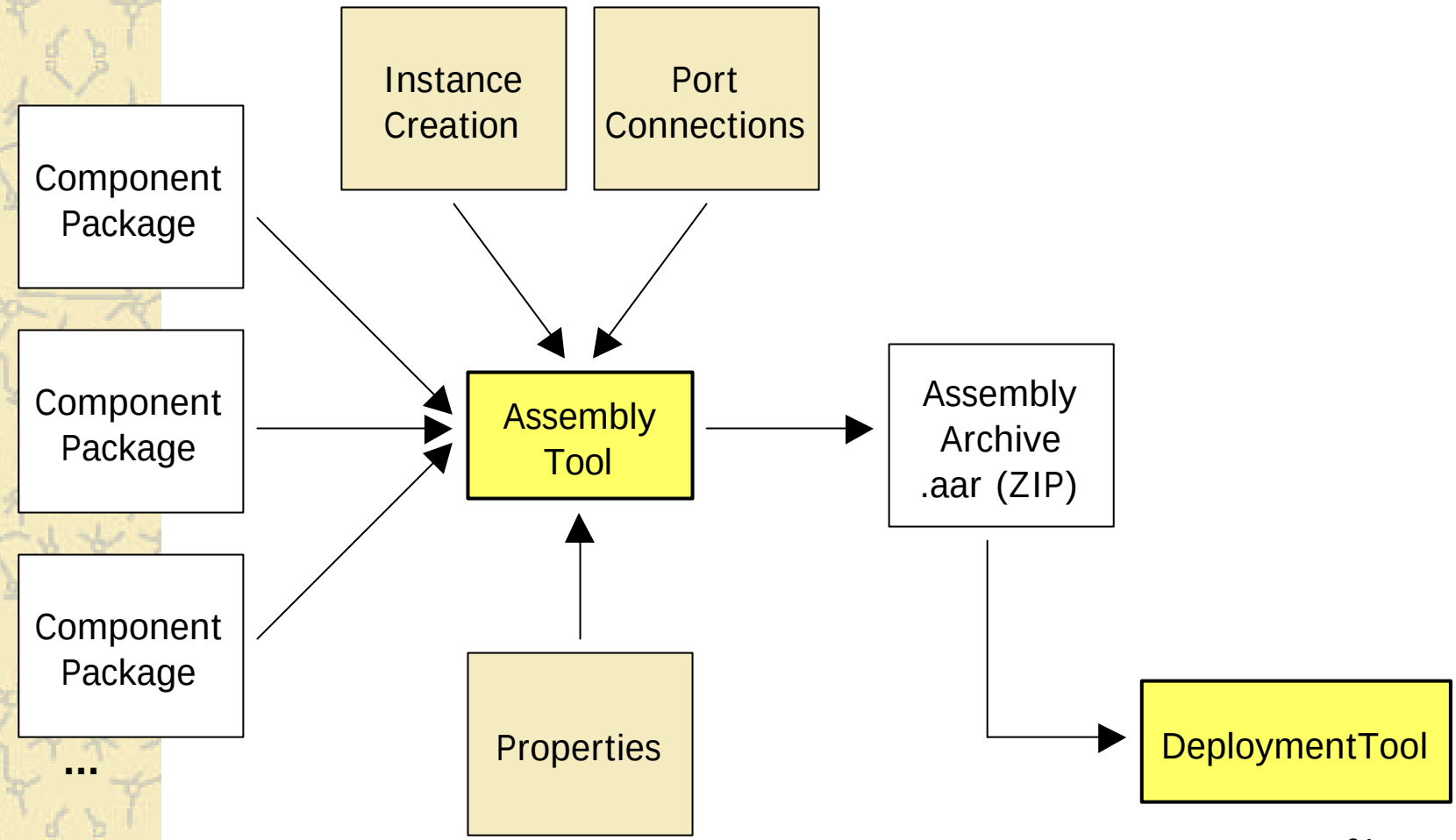
Assembly Descriptor Connection Example

```
<connectinterface>
  <usesport>
    <usesidentifier>left</usesidentifier>
    <componentinstantiationref idref="Philosopher (1)"/>
  </usesport>
  <providesport>
    <providesidentifier>the_fork</providesidentifier>
    <componentinstantiationref idref="ForkManager (1)"/>
  </providesport>
</connectinterface>
```

Component Packaging



Component Assembly



CCM XML Specification

- OMG document formal/02-06-65
- Descriptors specification in chapter 6 (Packaging and Deployment)
- DTDs in chapter 7 (XML DTDs)

Deployment and Configuration RFP

- Draft specification ptc/03-07-02
  - "Deployment and Configuration of Component-based Distributed Applications Specification"
  - Generalizes deployment and configuration beyond CCM
- Model-Driven Architecture (MDA)
  - Platform-Independent Model (PIM)
    - Specified as a UML model
  - Platform-Specific Model (PSM) for CCM
    - Specified as model extension + XML Schema

Deployment Model

- Components: compiled code (monolithic) or assemblies
- Deployment activities:
  1. Precondition: Getting hold of packages
  2. Installation: Bringing packages into a repository
  3. Configuration: Determining functional parameters
  4. Planning: Determining how to use distributed resources
  5. Preparation: Making software ready to use (traditional installation)
  6. Launch: Bringing the application into the running state

D&C Notations

- D&C PIM, specified in UML
  - UML profile for denoting the PIM
- D&C Tool Support UML profile, to denote specific models
- D&C PSMs, e.g. for CCM
  - PIM for CCM
    - mapped to PSM for CCM for IDL (online model exchange)
    - mapped to PSM for CCM for XML (offline model exchange)
      - XML Schema