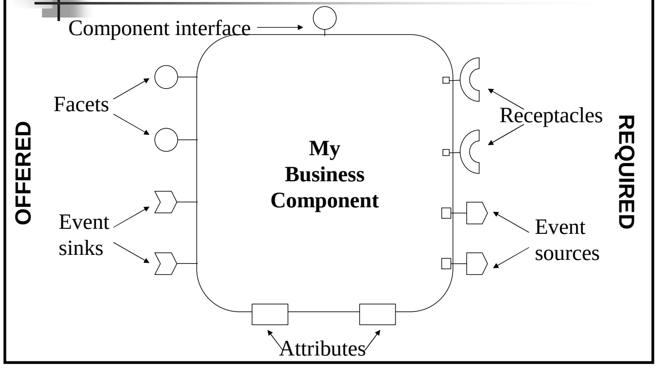
Komponenttiväliohjelmistot

CORBA Component Model (CCM) jatkoa... korjatulla esitysjärjestyksellä

CORBA-komponenttimalli CCM

- abstrakti komponenttimalli
- komponenttien suoritusaikainen ympäristö
  - Component container programming model
- komponenttien toteutus
  - Component Implementation Framework CIF + Component implementation language CIDL
- Komponenttien pakkaaminen ja käyttöönsaatto

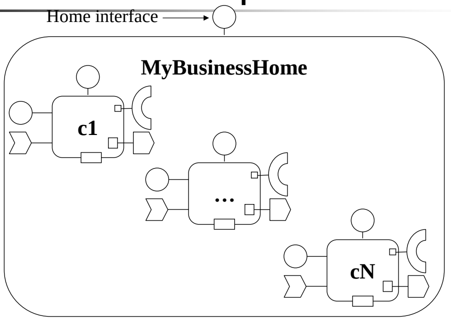
A CORBA Component



Component runtime support

- Component home
  - Controls a set of components using the component equivalent interface
  - Provides component factory facilities
    - Specializations
- Container
  - Defines the environment for supporting dynamic collections of components
  - The core of the CCM is the container: a running piece of code that you buy from a vendor and install on your server machine. The container includes an ORB with a POA. Corba components are server-side objects; your system administrator installs components into the container, which takes charge of them when they run.

A CORBA Component Home



Component Home

- Is instantiated at deployment time
- Manages a unique component type
  - More than one home type can manage the same component type
  - But a component instance is managed by a single home instance
- Allows life cycle characteristics or key type to vary/evolve without changing component definition
- Optional use of `primarykey` for business component identity and persistency primary key
- Standard `factory` and `finder` business logic operations
- Extensible with arbitrary user-defined business logic operations

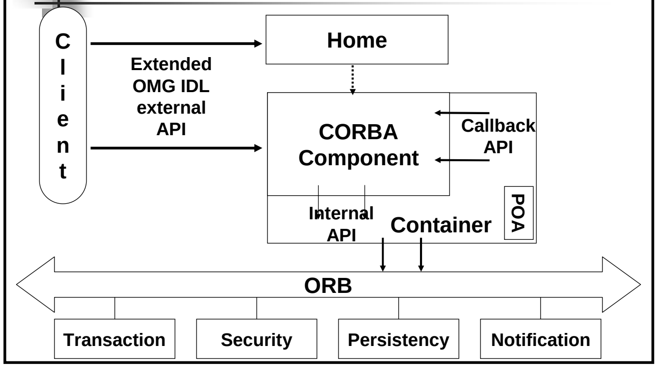
Primary Keys

- Values exposed to clients to create, find, and destroy component instances
  - Uniquely identifies a component instance within a home
  - Assigned at creation time, or in pre-existing database
  - Must be a value type derived from Components::PrimaryKeyBase (empty, abstract)
- Association between a primary key and a component is defined and maintained by its home
  - Different home types may define different key types (or no key) for the same component type
  - Primary key is not necessarily a part of the component's state

Komponenttien suoritusaikainen ympstö

- component container model
  - Framework for component application servers
  - Mostly built on the POAs
    - Automatic activation/deactivation
    - Resource usage optimization
    - Provides simplified interfaces for corba services
    - Uses callbacks for instance management
  - Container encapsulates 1-N POAs
  - References exported thourhg component home finder, naming or trading

The Container Architecture



CCM Container model

- CORBA Usage Model
  - Describes the interaction between the container, the POA and the CORBA services, ie reference persistence and servant to ObjectID mapping
  - Types: stateless, conversational, durable
- Component categories
  - Combination of internal and external APIs

usage model	container API	comp.category	object ref	servant/OID
stateless	session	service	transient	1:N
conversational	session	session	transient	1:1
durable	entity	process	persistent	1:1
durable	entity	entity	persistent	1:1

CCM Container model

- Components define their runtime requirements through (in the deployment descriptor)
  - Usage model
  - Component category
  - Activation and servant lifetime management
  - Transaction policies
  - Security policies
  - Events
  - Persistence
  - Compoent level (basic-ejb-compat, extended)

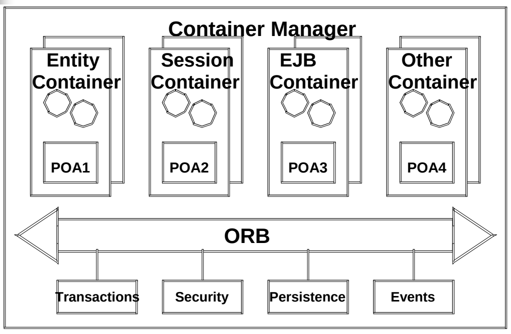
CCM container model

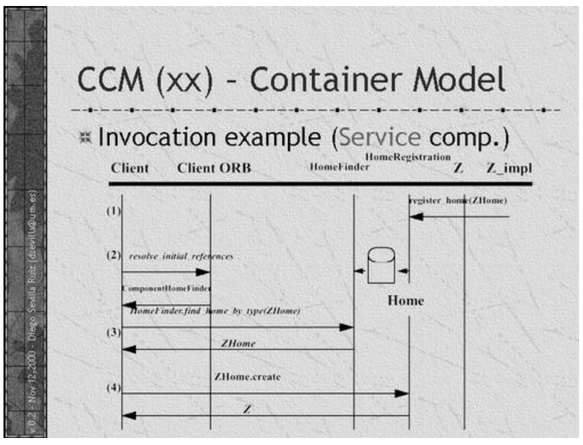
- component activation and servant lifetime management
  - Each container has a POA and servantLocator
  - Policies
    - Method: activate/passivate by method-basis
    - Transaction: lifetime tied to a transaction
    - Component: component decides itsef on deactivation
    - Container: lifetime tied to container's lifetime
  - Depends also on component category

characteristics	property (service)
internal interface	session context (basic) or Session2Context (extended)
callback interface	session component
usage model	stateless
extenral API type	keyless
client design pattern	factory
servat lifetime mgmt	method (only)

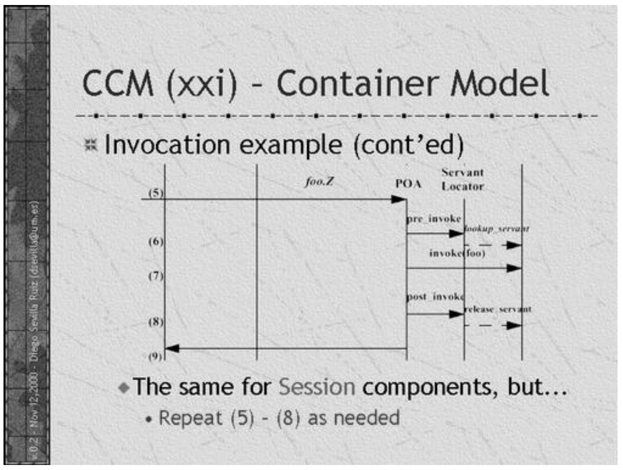
characteristics	property (entity)
internal interface	entitycontext(basic) or entity2context(extended)
callback interface	EntityComponent
usage model	durable
extenral API type	keyfull
client design pattern	factory or finder
servat lifetime mgmt	any

The Container Server Architecture




CCM (xx) - Container Model

- Invocation example (Service comp.)


CCM (xxi) - Container Model

- Invocation example (cont'ed)

- The same for Session components, but...
  - Repeat (5) - (8) as needed

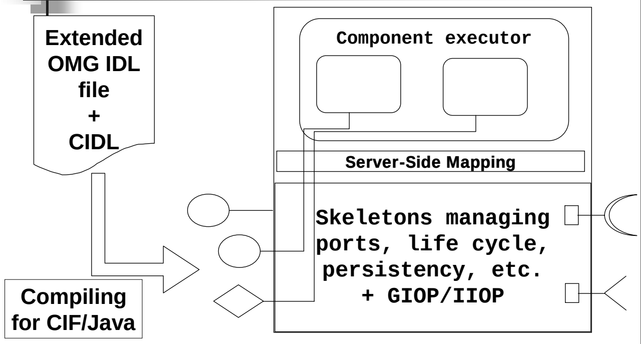
Cmponent Implementation Framework

- CIF defines a programming model for constructing component implementations
  - How components should be implemented
  - Generates executors: implementation of behavioural elements (homes, containers, ...)
- Facilitates component implementation
  - “only” business logic should be implemented
    - Not activation, identify, port management and introspection

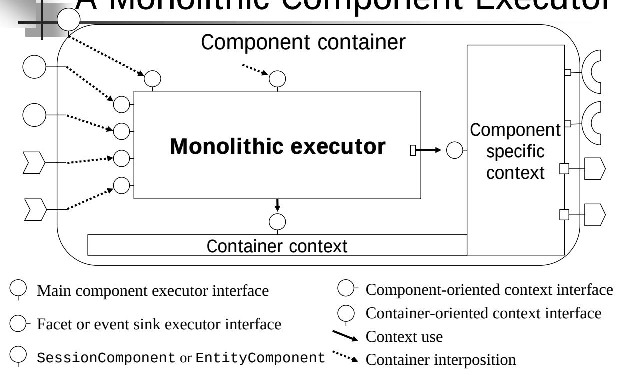
  => Local server-side OMG IDL mapping
    - Interactions between implementations and containers
- Manages segmentation and persistency

  => Component Implementation Definition Language

Component Implementation Framework to Component Skeleton Generation


Extended OMG IDL file + CIDL
Compiling for CIF/Java
Component executor
Server-Side Mapping
Skeletons managing ports, life cycle, persistency, etc. + GIOP/IIOP

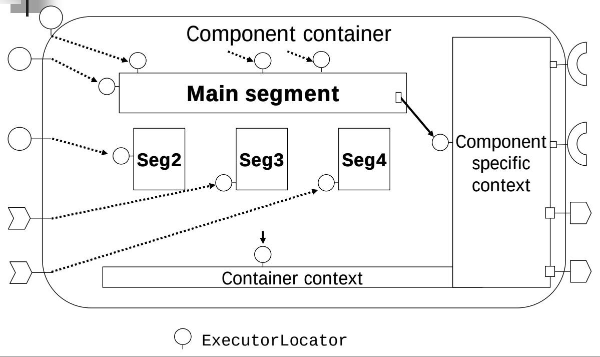
A Monolithic Component Executor


Component container
Monolithic executor
Component specific context
Container context
Main component executor interface
Facet or event sink executor interface
SessionComponent or EntityComponent
Component-oriented context interface
Container-oriented context interface
Context use
Container interposition

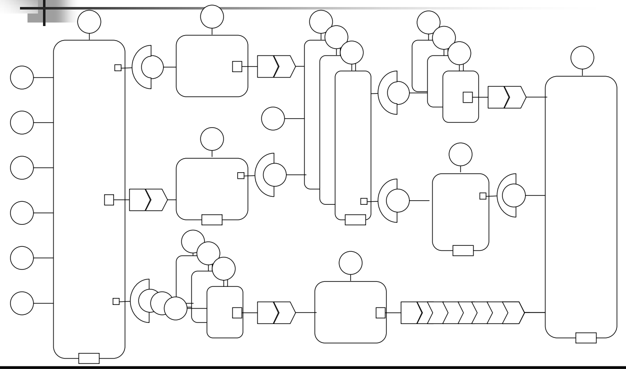
CIF

- implementations can be segmented
  - Segments are physical partitions of implementations
  - Can define independent state and can be independently activated
  - Help in managing, partitioning and sharing a high number of facet implementations
  - Only in process and entity categories

A Segmented Component Executor



Building CCM Applications = Assembling CORBA Component Instances



CCM development project stages

1. Analysis/design
   - UML + business object profiles
2. Component declaration
   - Define component’s methods and home with extended IDL and compile to produce
     - for clients: Operations, navigation operations, stubs
     - for servers: skeletons, IR entries, some code, packaging and deployment descriptors in XML

Esimerkki

```
component ShoppingCart {
   provides ShoppingCartIntf Cart1;
   uses CheckoutIntf CheckOut1;
}
home ShoppingCartHome manages
  ShoppingCart{};
```

CCM development project stages

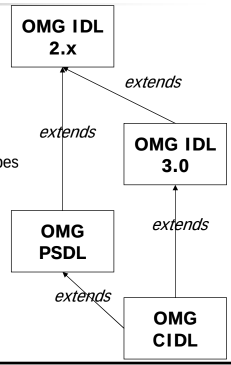
3. component implementation
   - declare component’s persisent state in PSDL and some behaviour aspects in CIDL
   - compile to get skeletons; fill in with business logic; compile to joint result to get compiled libraries

Esimerkki

```
abstract storagetype CustomerState{
   state long AcctNum;
   state ...
}
storagetype PortableCustomerState implements CustomerState{};
typedef sequence <Customer> CustomerList;
abstract storagehome CustomerStorageHome of Customer {
      primary key AcctNum(AcctNum);
      factory create(AcctNum, Name, ...);
}
storagehome PortableCustomerStorageHome implements
    CustomerStateHome{};
```

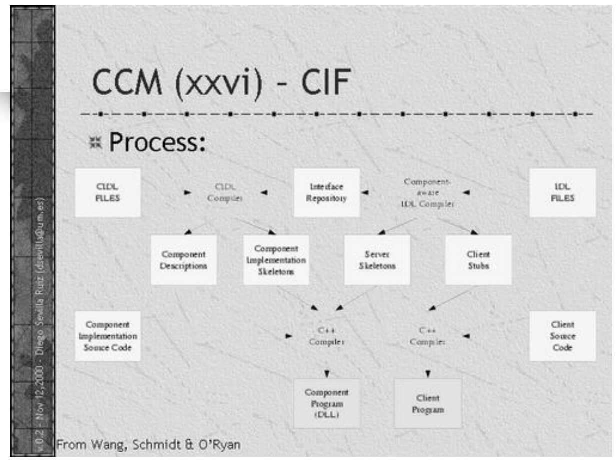
Relations between OMG Definition Languages

- OMG IDL 2.x
  - Object-oriented collaboration
  - i.e. data types, interfaces, and value types
- OMG IDL 3.0
  - Component-oriented collaboration
  - i.e. component types, homes, and event types
- OMG PSDL
  - Persistent state definition
  - i.e. [abstract] storage types and homes
- OMG CIDL
  - Component implementation description
  - i.e. compositions and segments



Component Implementation Definition Language CIDL

- describes component composition
  - Aggregate artifacts required to implement a component and its home
- Manages component persistent state
  - Persistent state definition language PSDL
  - Links storage types to segmented executors
- Generates executor skeletons providing
  - Segmentation of component executors
  - Default implementations of callback operations
  - Component’s state persistency



CCM (xxvi) - CIF

Process:

From Wang, Schmidt & O’Ryan

Composition

Defined in CIDL
- Category
- Home executor
- Executor
- Bindings
- State management (storage)

- 7 component categories:

4 CCM, 2 EJB, 1 clustomizable
- servant lifetime policies:

method, transaction, component, container

category

usage	container model	primary API type	key
service	stateless	session	no
session	conversational	session	no
process	durable	entity	no
entity	durable	entity	yes

Esimerkki

```
composition entity CustomerImpl {
    implements Customer;
    home executor CustomerHomeImpl
      delegatesTo abstractstoragehome
      CustomerStateStorageHome;
}
```

CCM development project stages

4. component packaging
   - (use interactive tools to produce)/write component descriptor (in XML) to tell CCM runtime how to connect up and manage the implementation
   - package up implementation and component descriptor into a component archive file CAR

5. component assembly
   - An application or part composed of some components with predefined interaction pathways in an assembly archive file AAR
   - Customizes configuration, connections, partitioning to different computers

CCM development project stages

6. component deployment and installation
   - Prior to this: system admin has installed and configured runtime environement
     - Installer program for CARs
   - Deploy component factories and managers
7. Runtime: component instance activation
   - Part of the application logic
   - Components are activated by the container POA

Esimerkki

```
<componentassembly id=”374...2304”>
<description>Assembly descr for example
</description>
<componentfiles>
<componentfile id=“ShoppingChartFIle”>
<fileinarchve name=“shoppingchar.car”>
</componentfiles>
<partitioning>
<homeplacement id =“ShoppingCartHome”>
<componentfileref idref=“ShoppingCartFIle”>
</homeplacement>
...
</partitioning>
<connections>
<connectinterface>
<usesport>
<usesidentifier>Check1</usesidentifier>
<homeplacementref idref=“ShoppingChartHome”>
...
```

The Client Programming Model

- Component-aware and -unaware clients
- Clients see two design patterns
  - Factory – Client finds a home and uses it to create a new component instance
  - Finder - Client searches an existing component instance through Name Service, Trader Service, or home finder operations
- Optionally demarcation of transactions
- Could establish initial security credentials
- Invokes operations on component instances
  - Those defined by the client-side mapping

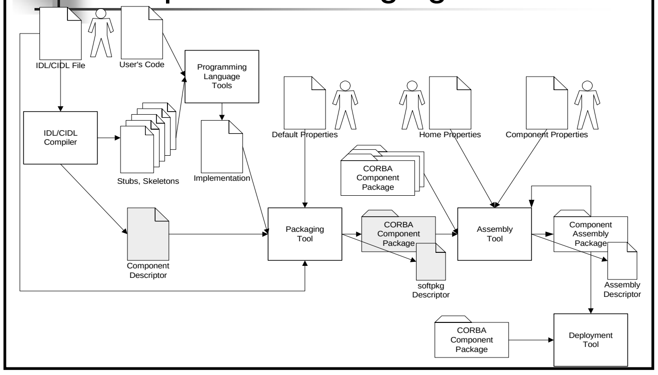
Packaging and Deployment

- 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
- Packaging and Deployment model Allows interoperability between deployment tools and containers

CCM Applications Deployment

- It is necessary for an application to
  - List component instances
  - Define logical location and partitioning
  - Specify connections between components
- It is necessary for a component to
  - Specify its elements
    - interfaces, implementations
  - Describe system requirements
    - OS, ORB, JVM, library releases, …
  - Specify its initial configuration
- It is necessary for a connection to
  - Associate related component ports

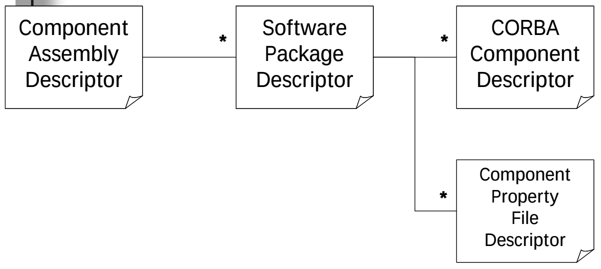
Component Packaging Artifacts



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 container managed policies filled 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

Relationship Between CCM XML Descriptors


Component Assembly Descriptor
*
Software Package Descriptor
*
CORBA Component Descriptor
*
Component Property File Descriptor

Component Assembly Package

- 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

Component Assembly Descriptor (.cad)

- References one or more Component Software Descriptors
- Defines home instances and their collocation and cardinality constraints
- Defines components to be instantiated
- Defines that homes, components or ports are to be registered in the ComponentHomeFinder, Naming or Trading Service
- Defines connections to be made between component ports, e.g. receptacles to facets and event sinks to event sources

Software Package Descriptor (.csd)

- Descriptive general elements
  - title, description, author, company, webpage, license
- Link to OMG IDL file
- Link to default property file
- Implementation(s)
  - Information about Implementation
    - Operating System, processor, language, compiler, ORB
    - Dependencies on other libraries and deployment requirements
    - Customized property and CORBA component descriptor
  - Link to implementation file
    - Shared library, Java class, executable
  - Entry point

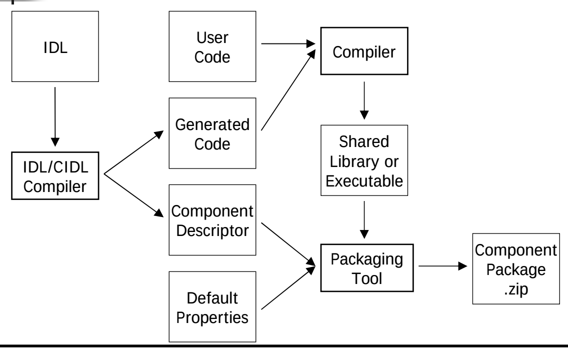
CORBA Component Descriptor (.ccd)

- 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
  - Extended POA policies
- Link to component and home property files

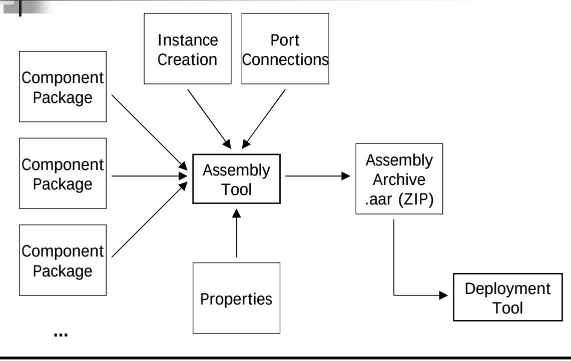
Property File Descriptor (.cpf)

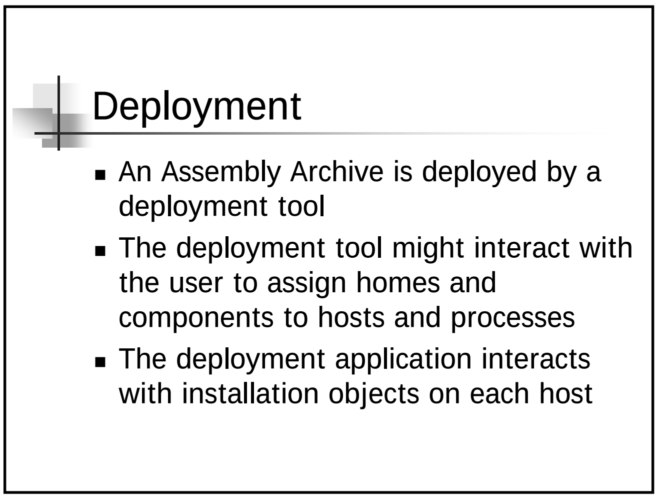
- Used to set home and component properties
  - However, it could be used for anything
- Contains zero or more name/value pairs to configure attributes
- Referenced by...
  - Software Package Descriptors to define default values for component attributes
  - CORBA Component Descriptors to define default values for component or home attributes
  - Assembly Descriptors to configure initial values for home or component instances

Component Packaging



Component Assembly





Deployment

- An Assembly Archive is deployed by a deployment tool
- The deployment tool might interact with the user to assign homes and components to hosts and processes
- The deployment application interacts with installation objects on each host