CORBA Component Model Architectural Overview

David S. Frankel

Chief Scientist

Genesis Development Corporation

OMG Architecture Board Member

OMG Business Object Initiative WG Co-Chair

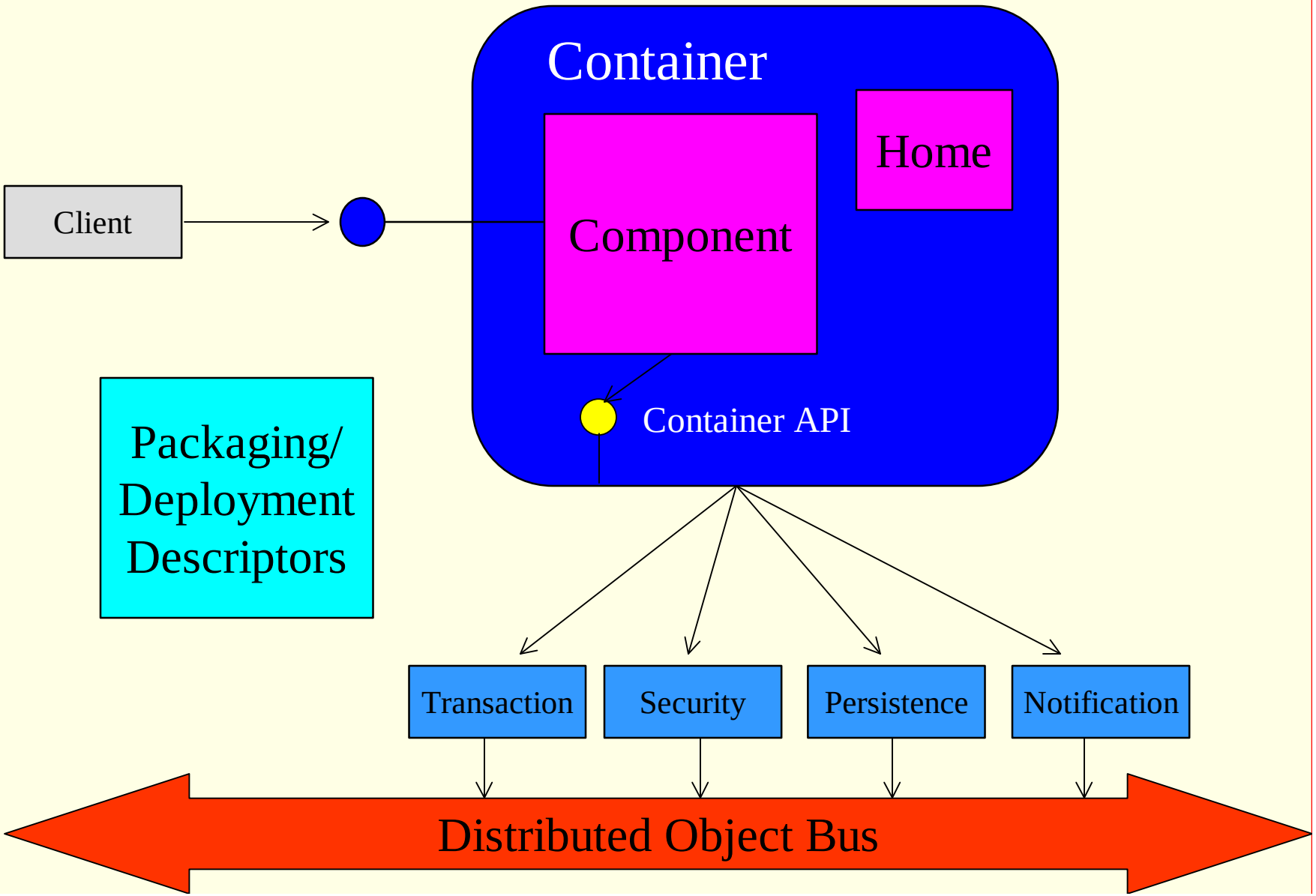
dfrankel@gendev.com

Presentation Contents

- Container/Home Architecture
- CCM and EJB
- Extended CCM Components
- Mapping Component IDL Declarations to Legacy IDL
- Packaging and Deployment Descriptors
- CCM Metamodels
- Market Prospects

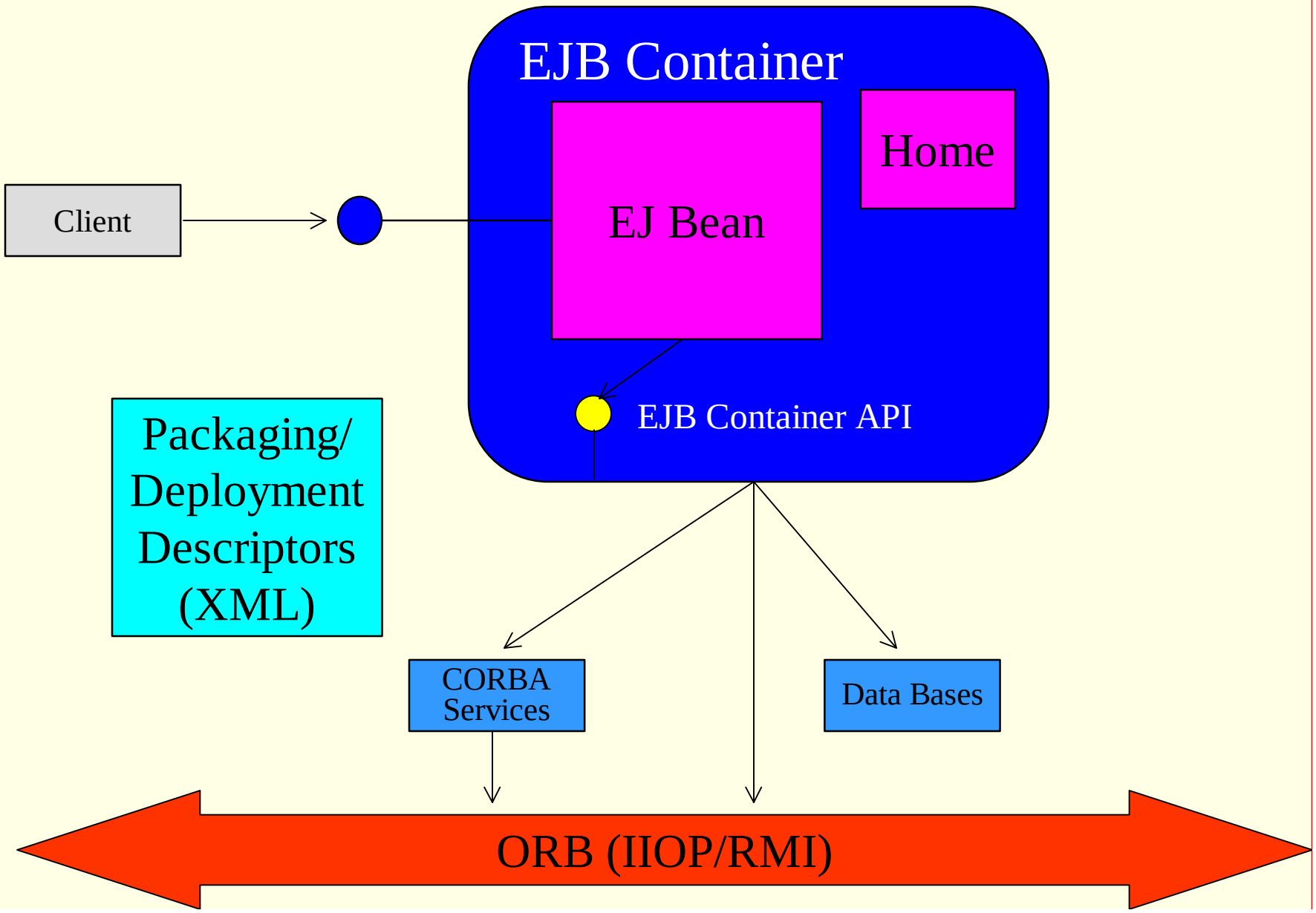
GENESIS Development Corporation

A Component in a Container
Generic Industry Model



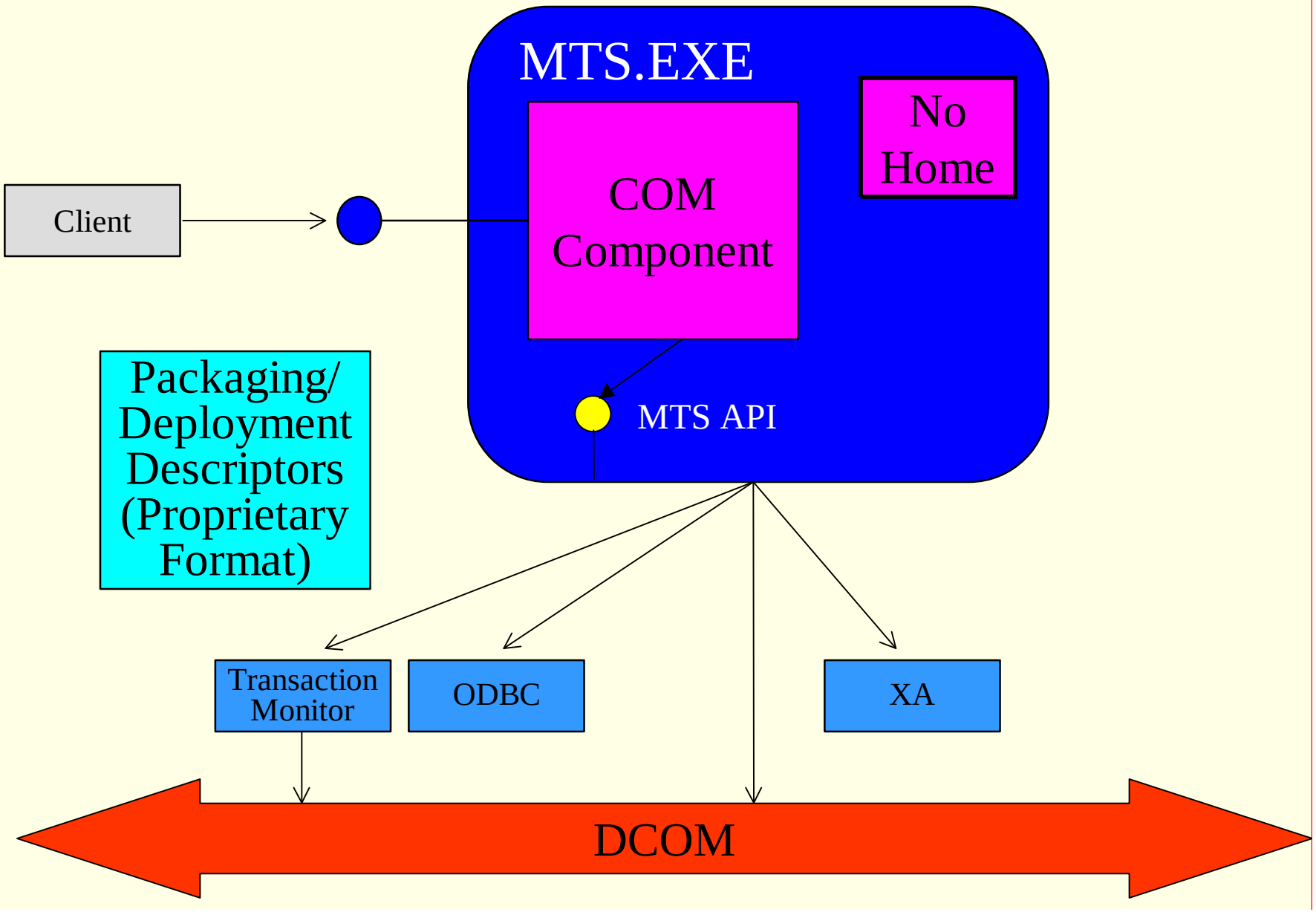
GENESIS Development Corporation

EJB Components and Containers Today



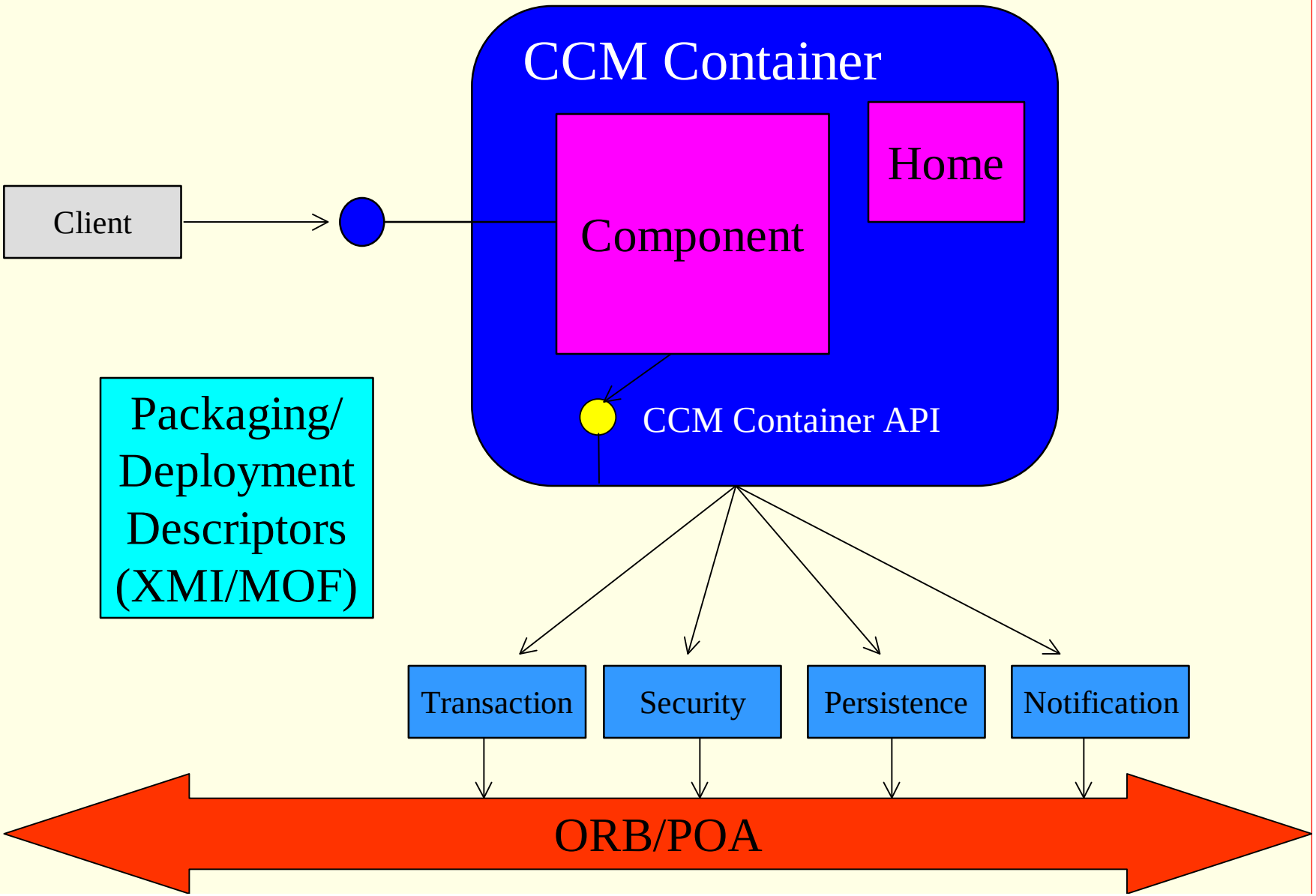
GENESIS Development Corporation

Microsoft Transaction Server



GENESIS Development Corporation

CORBA Components



GENESIS Development Corporation

CCM and EJB

- CCM is a language-neutral superset of the EJB model
  - CCM has additional features
  - CCs that do not use the additional features are *basic* CCs
  - CCs that use the additional features are *extended* CCs
- For basic CCs that are programmed in Java, the EJB 1.1 APIs are mandatory
  - Where this conflicts with the standard IDL-Java mapping, the EJB APIs take precedence over IDL-Java
  - Thus, a basic, Java CC *is* an EJB
- CCM defines a CORBA container for EJBs
  - Standardizes the already-common use of CORBA by EJB containers
- CCM packaging/deployment descriptors more complex than EJB's
  - Programming language must be reflected
    - Can't assume language is Java
  - Operating system must be reflected
    - Can't assume Java VM
- Component Implementation Definition Language (CIDL) is new

GENESIS Development Corporation

An Extended CC Can...

- Expose multiple interfaces
- Declare the interfaces that it calls (*uses*)
- Declare its event emission/publication and consumption
- Have multiple *segments* with different persistence characteristics

GENESIS Development Corporation

Declaring a Component in IDL

```
component RenderableARUpdate
{
   //ARUpdate and Render are interfaces declared elsewhere
   //arUpdate and render are facet identifiers
   provides ARUpdate arUpdate;
   provides Render render;

   //Customer and GeneralLedger are interfaces declared elsewhere
   //customer and gl are receptacle identifiers
   uses Customer customer;
   uses GeneralLedger gl;

   //ARUpdateComplete, CollectionsAlert, and OrderInitiated
   //are valuetypes declared elsewhere
   publishes ARUpdateComplete arUpdateComplete;
   emits CollectionsAlert collectionsAlert;
   consumes OrderInitiated orderInitiated;
};
```

GENESIS Development Corporation

Declaring a Home in IDL

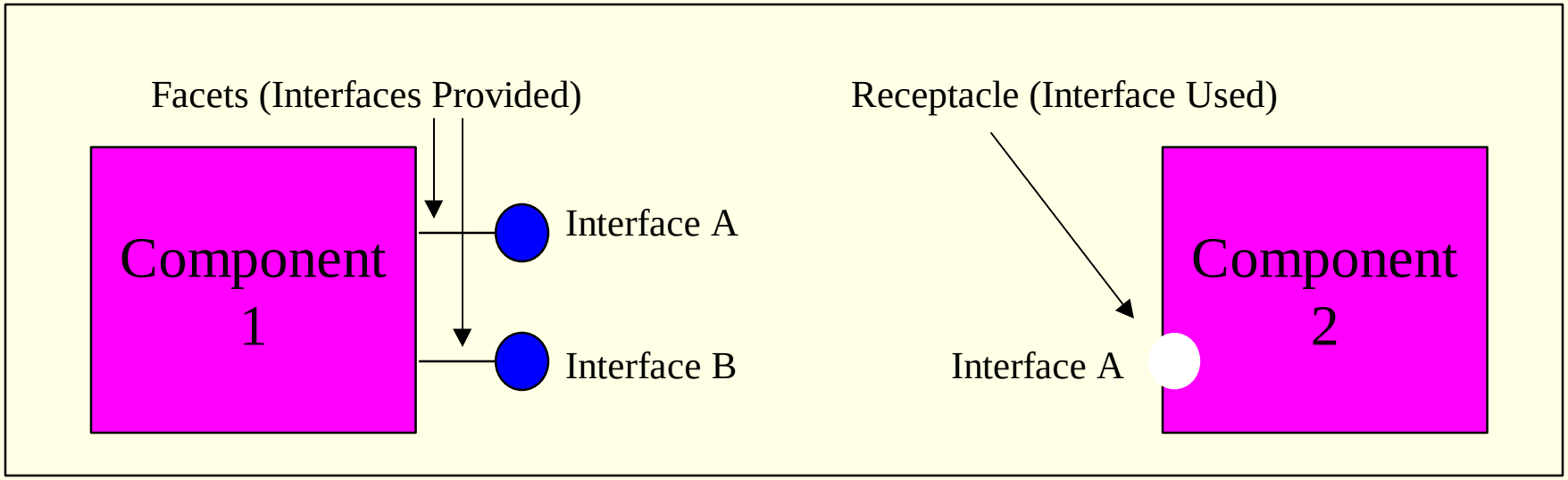
```
home AcmeARUpdateHome manages RenderableARUpdate
{
   //CollectionsPolicy is defined elsewhere
   factory CreateARUpdate (in CollectionsPolicy policy);
   finder FindARUpdate (in CollectionsPolicy policy);
};
```

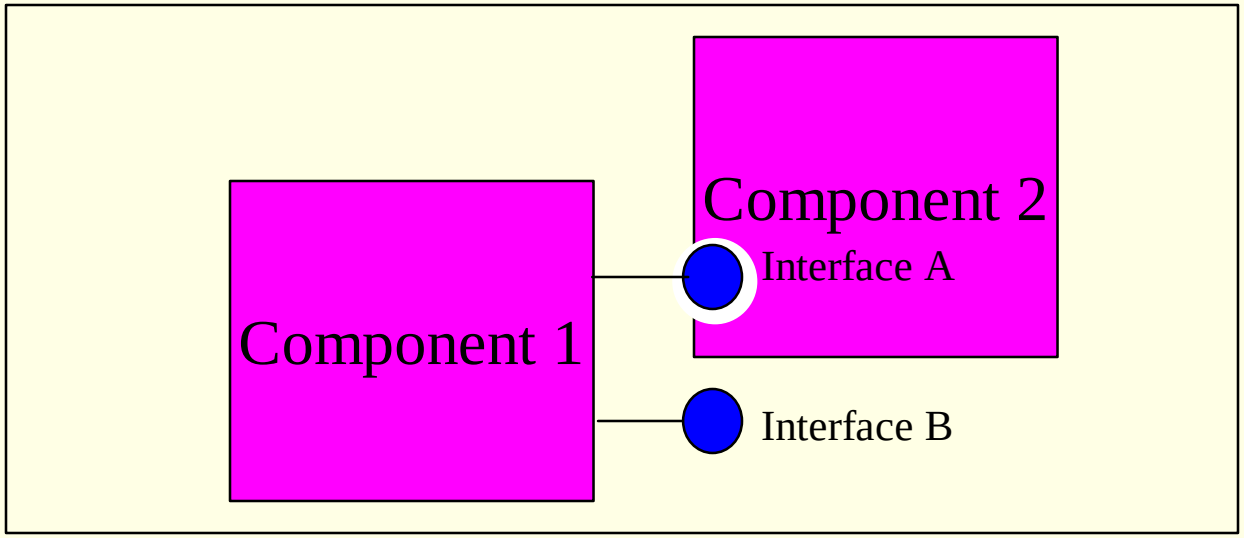
Declaring a Basic Component

```
component ARUpdateComp supports ARUpdate { };
home AcmeARUpdateHome manages ARUpdateComp
{
   //CollectionsPolicy is defined elsewhere
   factory CreateARUpdate (in CollectionsPolicy policy);
   finder FindARUpdate (in CollectionsPolicy policy);
};
```

GENESIS Development Corporation

Facets and Receptacles





Events

- Simple event model
- Events are valuetypes derived from Components::EventBase (empty, abstract)
- Push model
- Container mediates access to CosNotification channels

GENESIS Development Corporation

Event Sources

- Two types: *publisher* and *emitter*
- *Publisher*
  - Client subscribes to event source by directly calling the component
  - Component mediates subscription to event channel
  - Component is only source of events for channel (dedicated channel per component)
- *Emitter* is a simple push consumer
  - Intended for connection to an arbitrary event channel during configuration
  - Client subscription is not by call to component
    - Ordinary means for subscribing to an event channel are used

Declaring a Home With a Primary Key

```
component Customer { };

//SocSec# is a valuetype defined elsewhere
home CustomerHome manages Customer primaryKey SocSec#
{
};
```

GENESIS Development Corporation

The Philosophy of Homes
A Concession to Reality

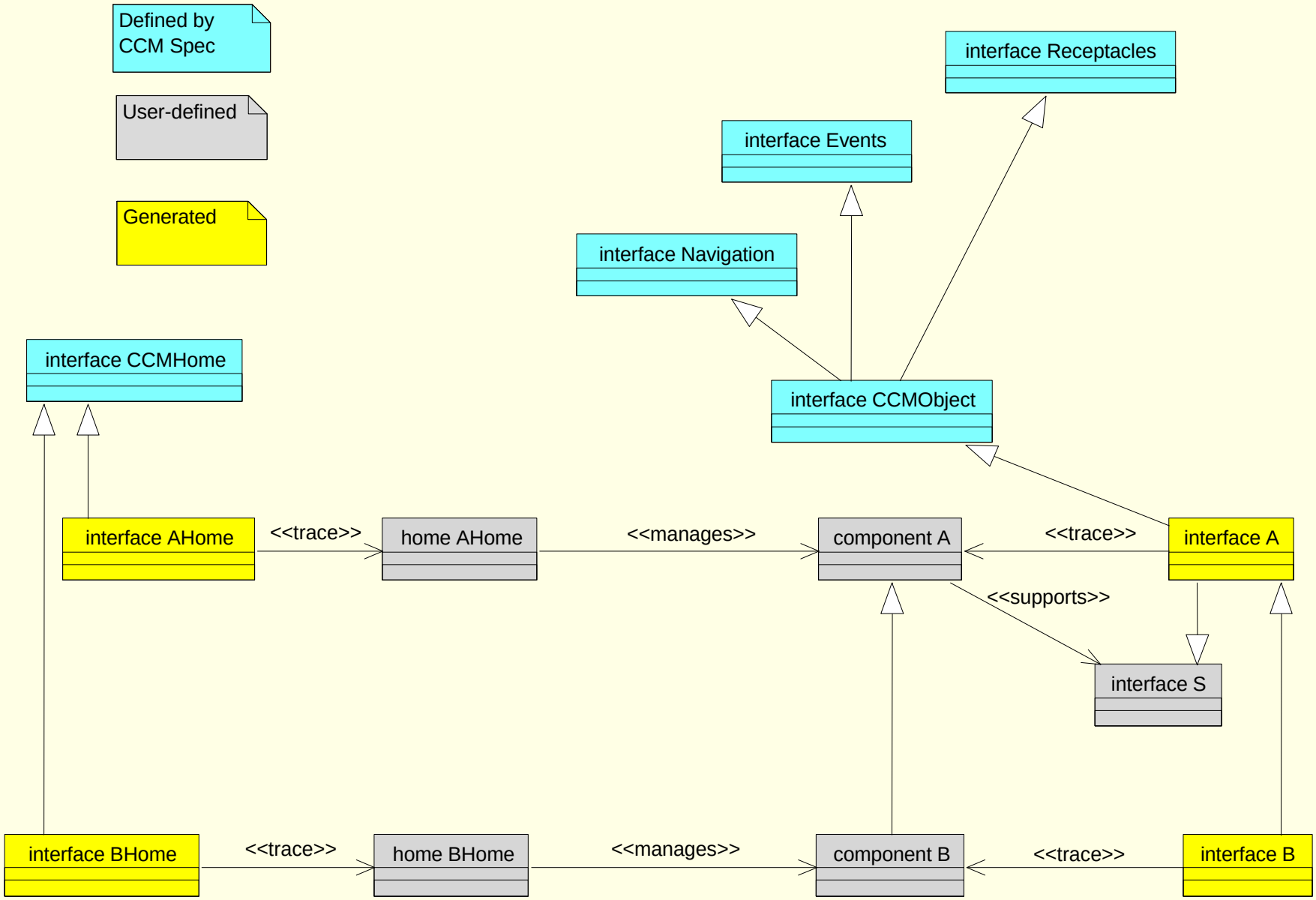
- Predicated on the realization that one grand instance manager for a type is impossible to achieve in practice
  - Very difficult to achieve in one large, hermetically-sealed enterprise
  - Enterprises are increasingly *not* sealed
    - Mergers and acquisitions
    - Business to business electronic commerce coming
- Allows there to be multiple managers of a large type extent
  - One Customer home might use social security#
  - Another Customer home might use an internal customer# as the primary key

Home Finders

```
module Components {

  interface HomeFinder {

    CCMHome find_home_by_component_type(in CORBA::RepositoryId comp_repid);

    CCMHome find_home_by_home_type (in CORBA::RepositoryId home_repid);

    CCMHome find_home_by_name (in string home_name);
  };
};
```

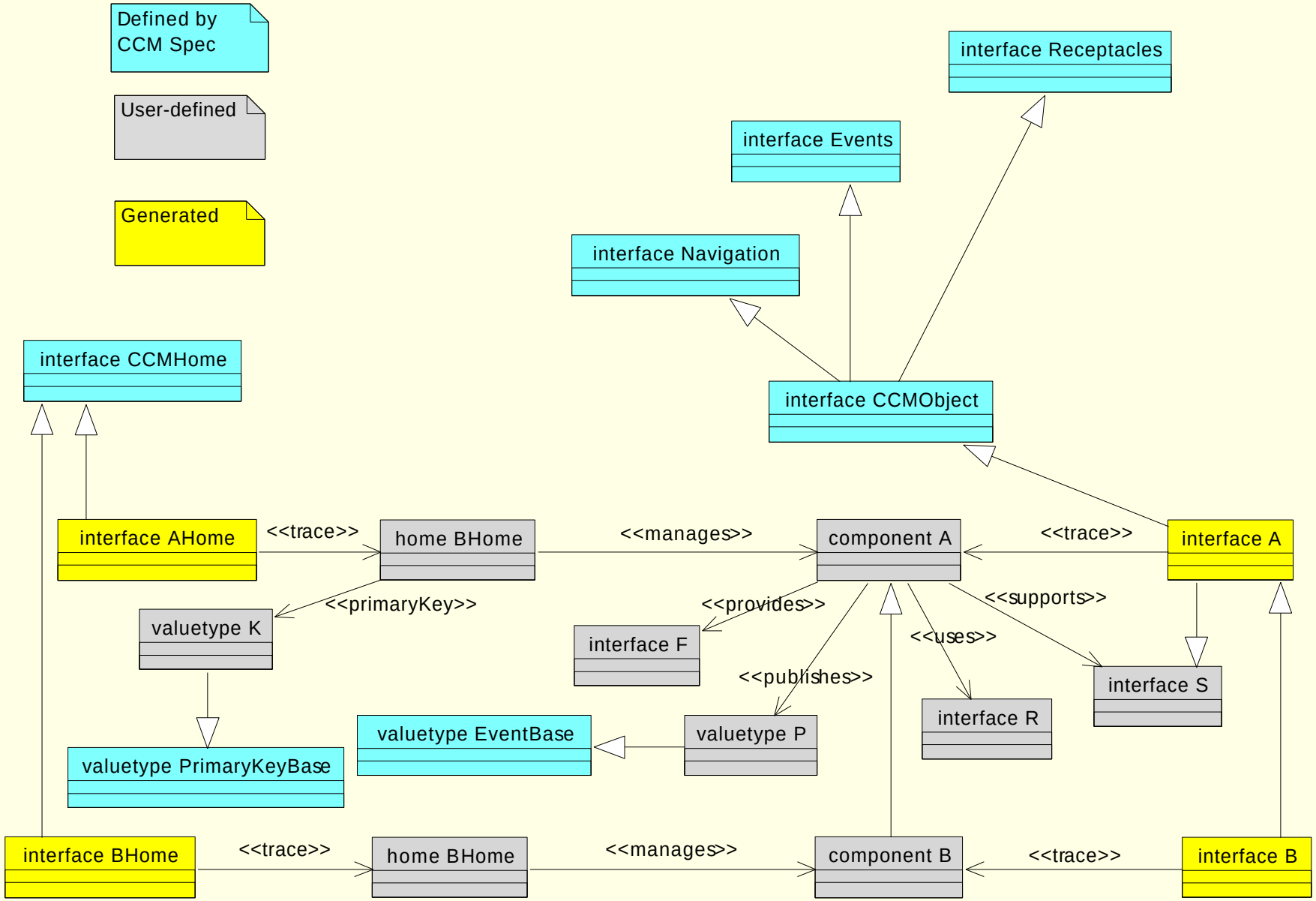
GENESIS Development Corporation

Mappings to Legacy IDL



GENESIS Development Corporation

Mappings to Legacy IDL



GENESIS Development Corporation

Home Operations Generated from Primary Key Declaration

<*component_type*> create (in <*key_type*> key);

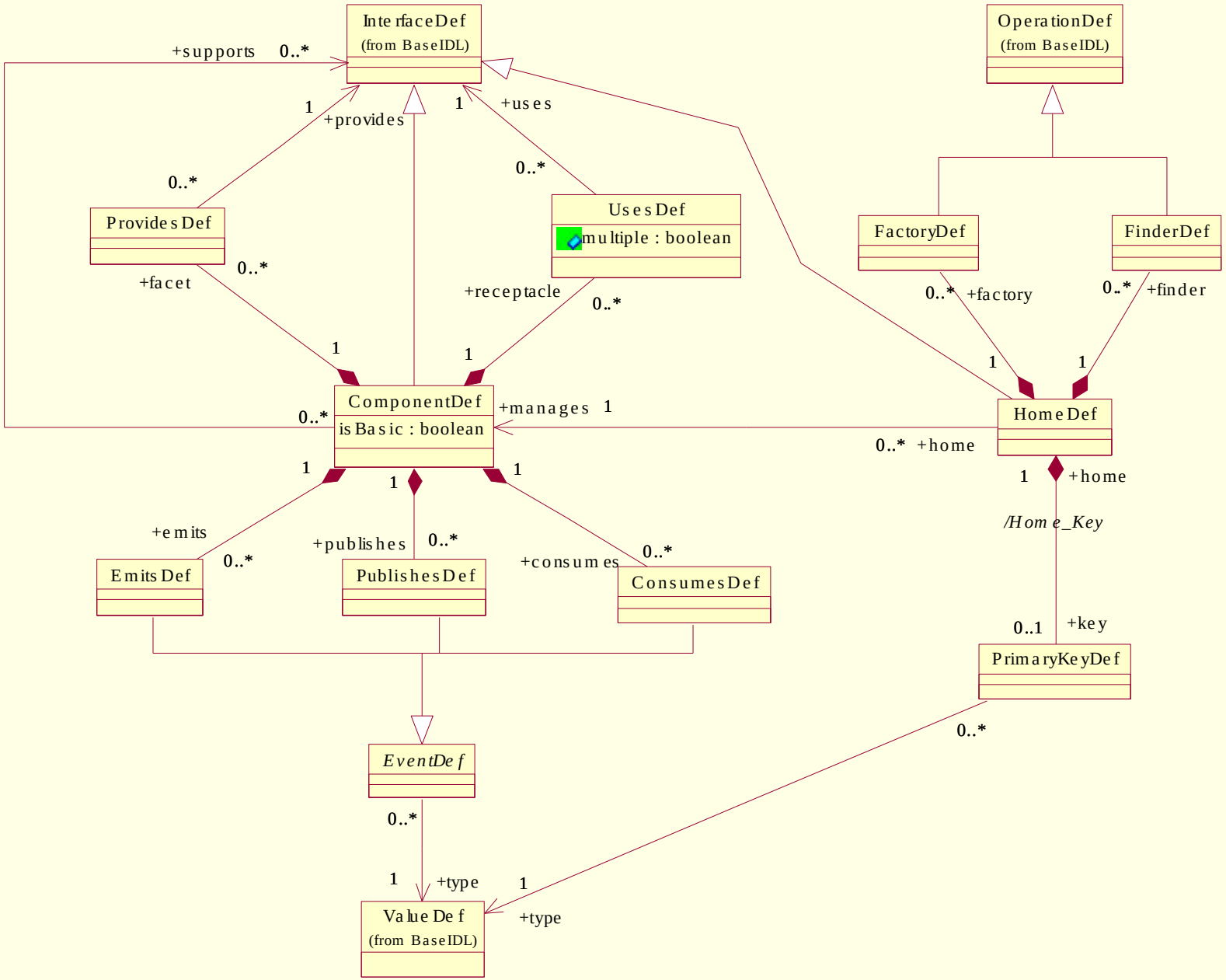
<*component_type*> find_by_primary_key (in <*key_type*> key);

void remove (in <*key_type*> key);

<*key_type*> get_primary_key (in <*component_type*> comp);

GENESIS Development Corporation

Metamodel of IDL Extensions



GENESIS Development Corporation

Component Implementation Definition Language (CIDL)

```
import RenderableARUpdate;
import AcmeARUpdateHome;
composition service AcmeARUpdateComposition
{
   home executor AcmeARUpdateHomeImpl
   {
      implements AcmeARUpdateHome;
      manages AcmeARUpdateImpl;
   };
};
```

GENESIS Development Corporation

Component Categories
Pre-Defined Lifecycle Characteristics

- **Service**
  - Behavior, no state beyond lifetime of transaction, no identity
  - Models single independent execution of an "operation"
  - Examples: CICS tx, "command object," MTS "stateless" component
- **Session**
  - Behavior, transient state, non-persistent identity
  - Models transient state for lifetime of client
  - Examples: iterator, MTS "stateful" component
- **Process**
  - Behavior, persistent state (not visible to client), persistent identity (visible to client only through user-defined operations)
  - Able to roll back its state changes
  - Models a long-lived business process (workflow)
  - Example: loan application
- **Entity**
  - "Uninteresting" behavior (attribute accessors and mutators), persistent state, identity via primary key
  - Models business entities in the real world with non-transient existence
  - Examples: customers, accounts
  - Able to roll back its state changes

Container API Categories

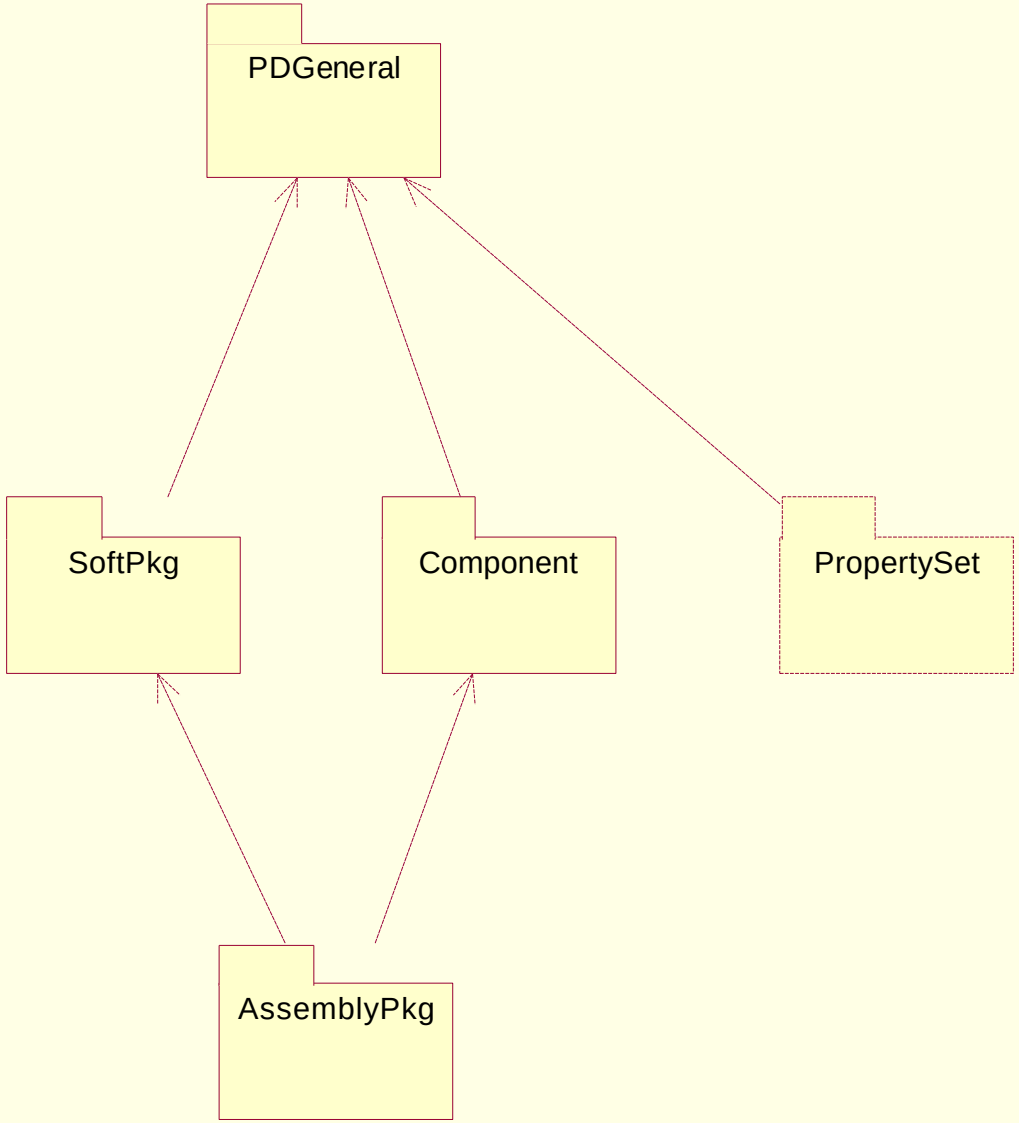
- Service
- Session
- Process
- Entity
- EJBSession
- EJBEntity

Basic vs. Extended Containers

- **Basic**
  - **Corresponds to EBJ 1.1 container functionality**
  - **Simple persistence (single segment), container manages objref creation**
- **Extended**
  - **Adds advanced persistence (multiple segments), event model, component involved in objref creation**

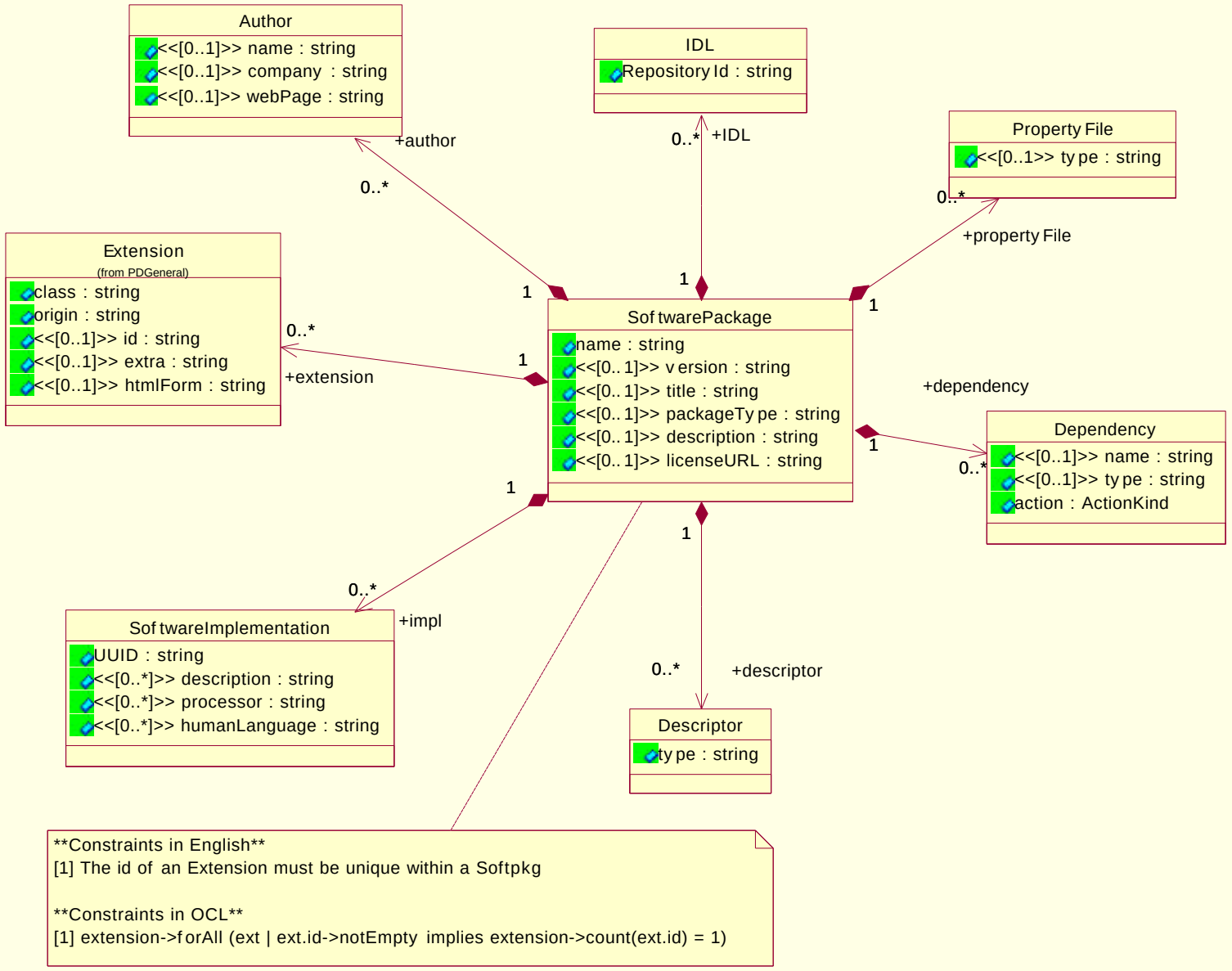
GENESIS Development Corporation

Packaging and Deployment Descriptors



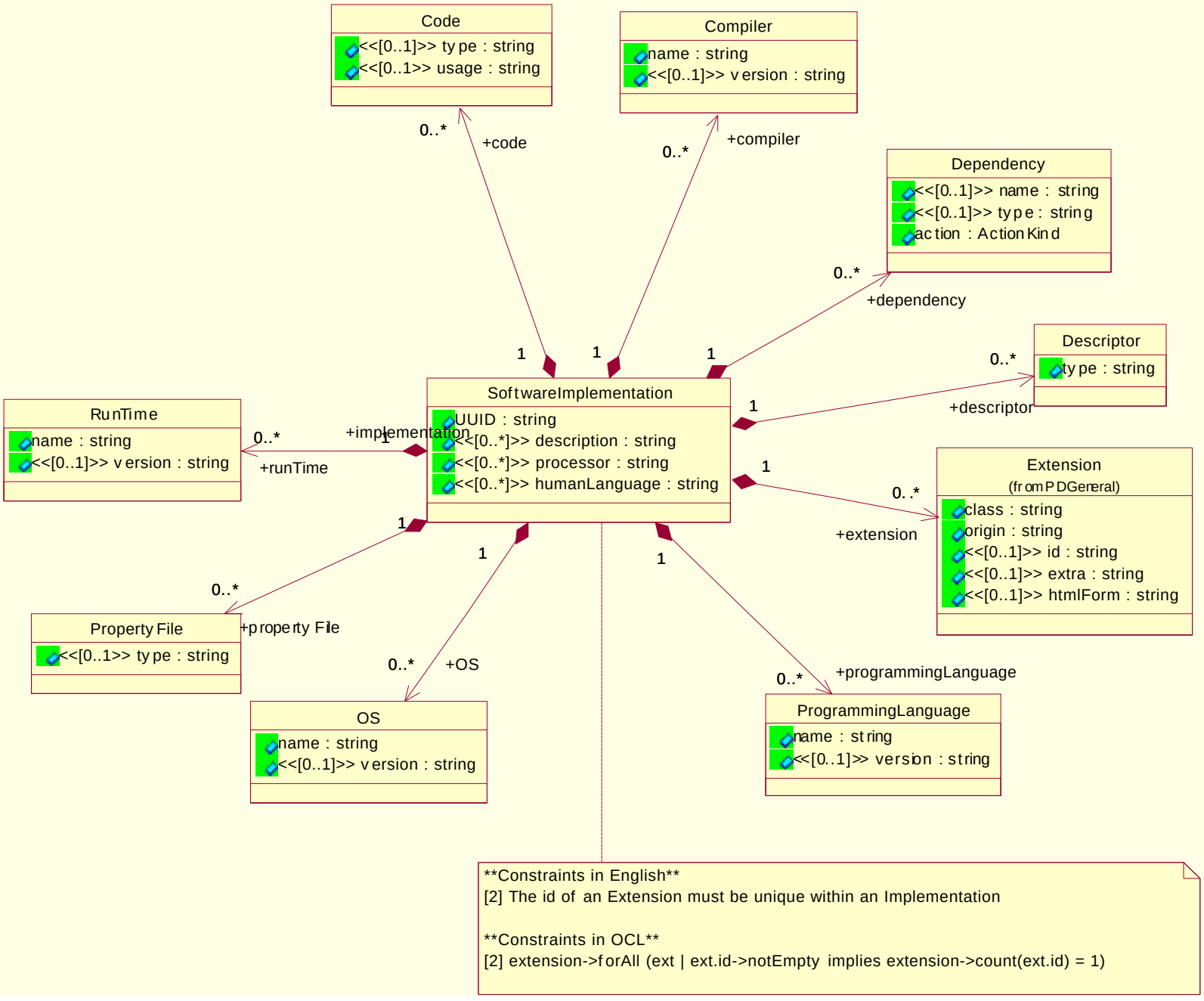
GENESIS Development Corporation

The Softpkg Descriptor



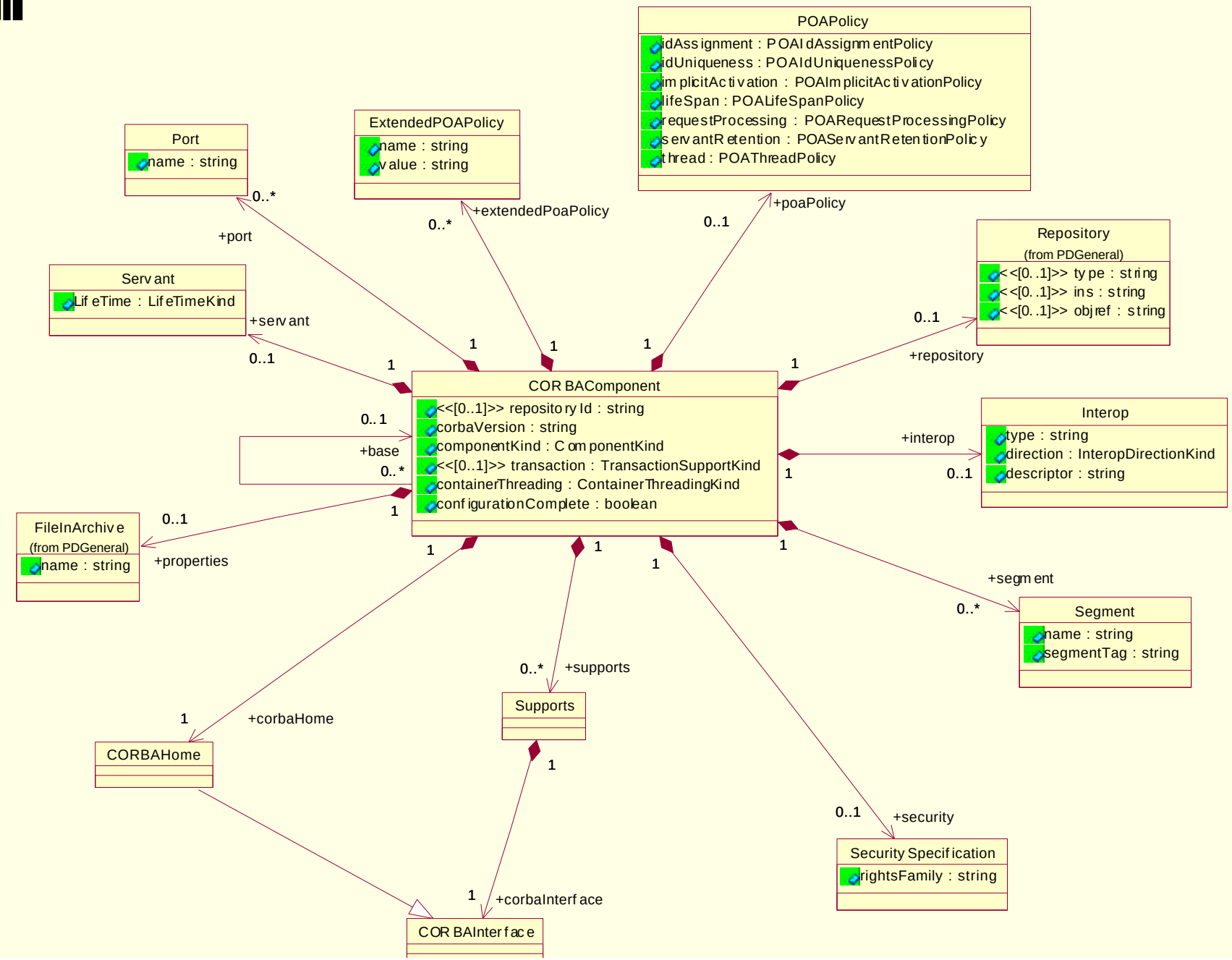
GENESIS Development Corporation

The Implementation Descriptor



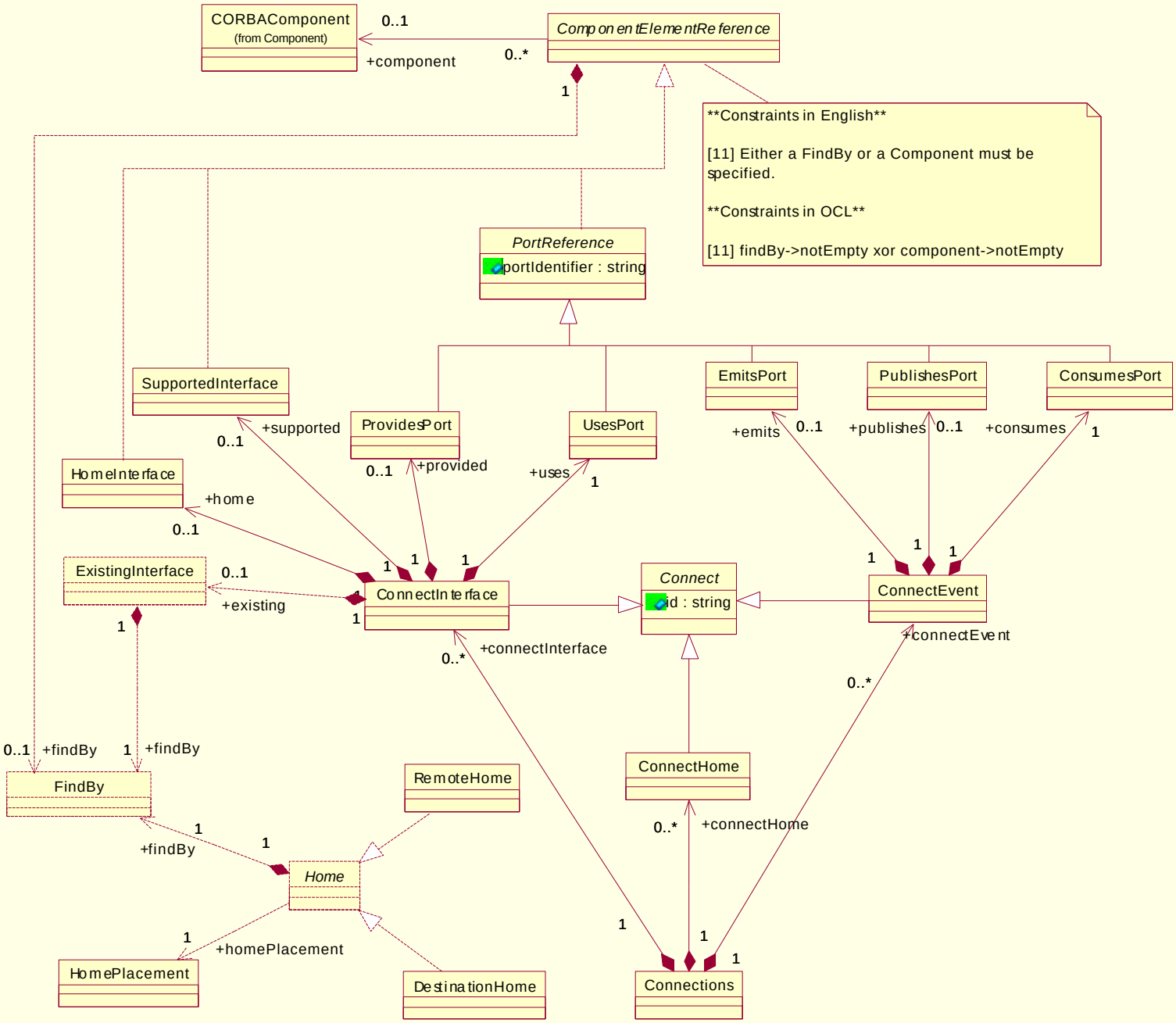
GENESIS Development Corporation

The Component Descriptor



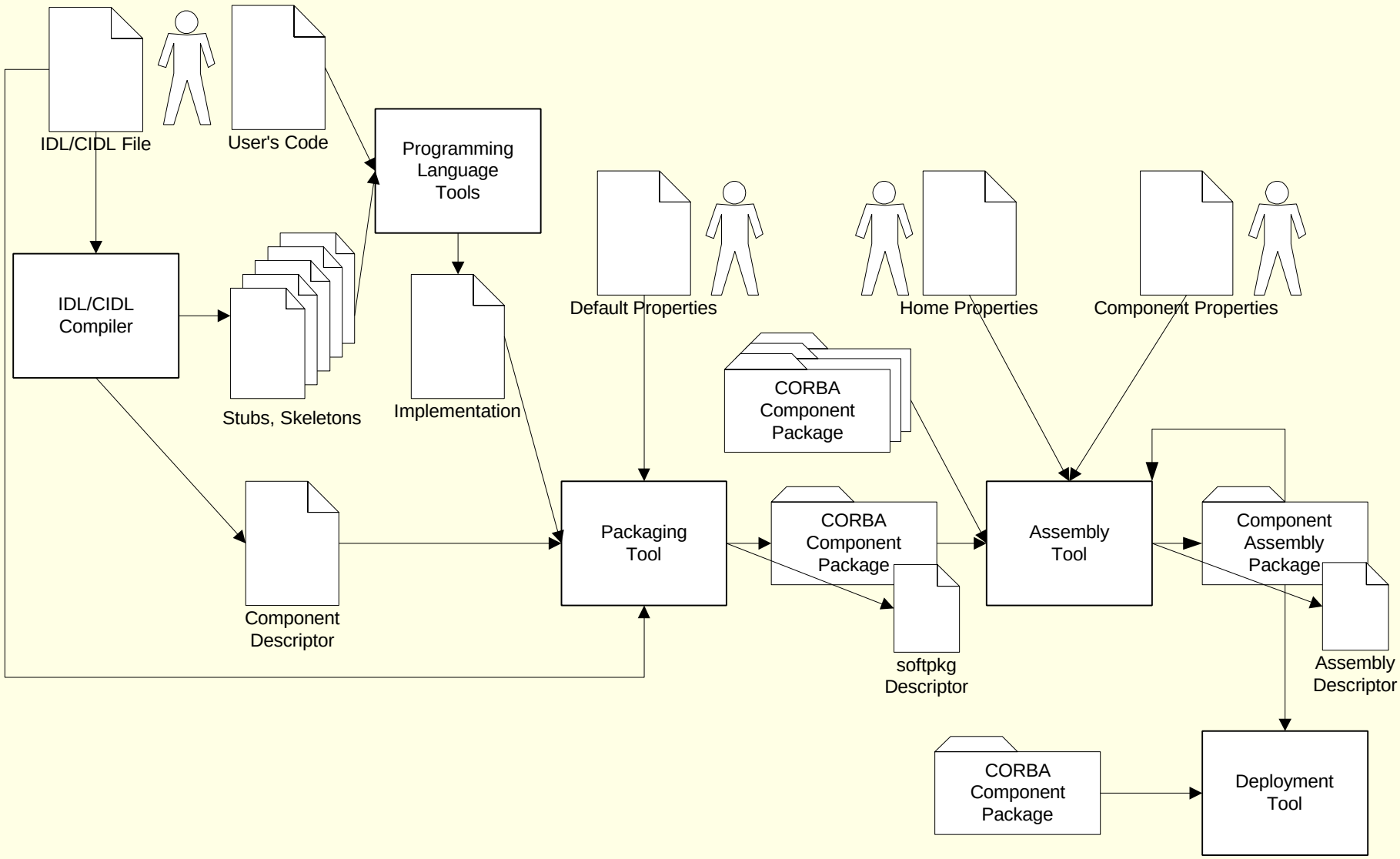
GENESIS Development Corporation

The Assembly Descriptor



GENESIS Development Corporation

Typical Construction/Deployment Scenario



GENESIS Development Corporation

Configuration APIs
Implemented by Installation/Assembly Tools

Points to location of component package

```
interface ComponentInstallation {
   boolean install(in string implUUID, in string component_loc) raises InvalidLocation;
   boolean replace(in string implUUID, in string component_loc) raises InvalidLocation;
   boolean remove(in string implUUID) raises UnknownImplId;
};
```

Points to location of assembly package

```
interface AssemblyFactory {
   Cookie create(in string assembly_loc) raises InvalidLocation;
   Assembly lookup(in Cookie c) raises InvalidAssembly;
   boolean destroy(in Cookie c) raises InvalidAssembly;
};

enum AssemblyState {INACTIVE, INSERVICE};

interface Assembly {
   boolean build();
   boolean tear_down();
   AssemblyState get_state();
};
```

GENESIS Development Corporation

CCM Metamodels
MOF Compliance

- The OMG Metaobject Facility (MOF)
  - A concession to reality
  - Predicated on the realization that one grand metamodel is unrealistic
  - Instead, provides a modest set of common constructs for defining metamodels
  - The common constructs allow common tools to manage the storage, retrieval, and manipulation of models in metadata repositories

GENESIS Development Corporation

Changes to ORB Core

- Directly related to components
  - The string, "ComponentHomeFinder" is added to the list of valid ObjectID values input to ORB::resolve_initial_references
  - New operation on CORBA::object
    - get_component
- General cleanup
  - Attributes can raise exceptions
  - Optional keyword "local" as modifier of interface
    - Local interfaces are CORBA object interfaces, but the resulting type cannot be marshaled or remotely invoked.
    - Intended to obviate the need for PIDL, and to formalize the current practice of defining special "locality-constrained" cases of CORBA interfaces
  - "import" <imported_scope> ;
    - Mechanism for importing external name scopes
    - Obviates need for doing so in terms of file scopes
    - <imported_scope> may be fully qualified scoped IDL name or a repository id
  - Repository id declaration
    - Portable, standard mechanism, obviates #pragmas
    - AB insisting on using repository ids for versioning

GENESIS Development Corporation

CCM's Market Prospects

- Negatives
  - App server vendors already using EJB programming model
  - ORB vendor commitment to extended components is questionable
- Positives
  - The distributed component industry is still in a very early stage
    - Iona, Inprise, BEA, Component Broker comprise only a "first wave" of ORBs
  - There are dozens of in-house corporate ORBs
  - Need for interoperable, portable components with multiple interfaces and declarable events will increase over time
- User interests
  - The CCM standard is controlled by an open process
  - EJB controlled by one company
    - JSR process not nearly as open as OMG
  - Users should emphasize to Sun that EJB 2.0 should align with CCM
    - Just as CCM aligned with Java 1.1