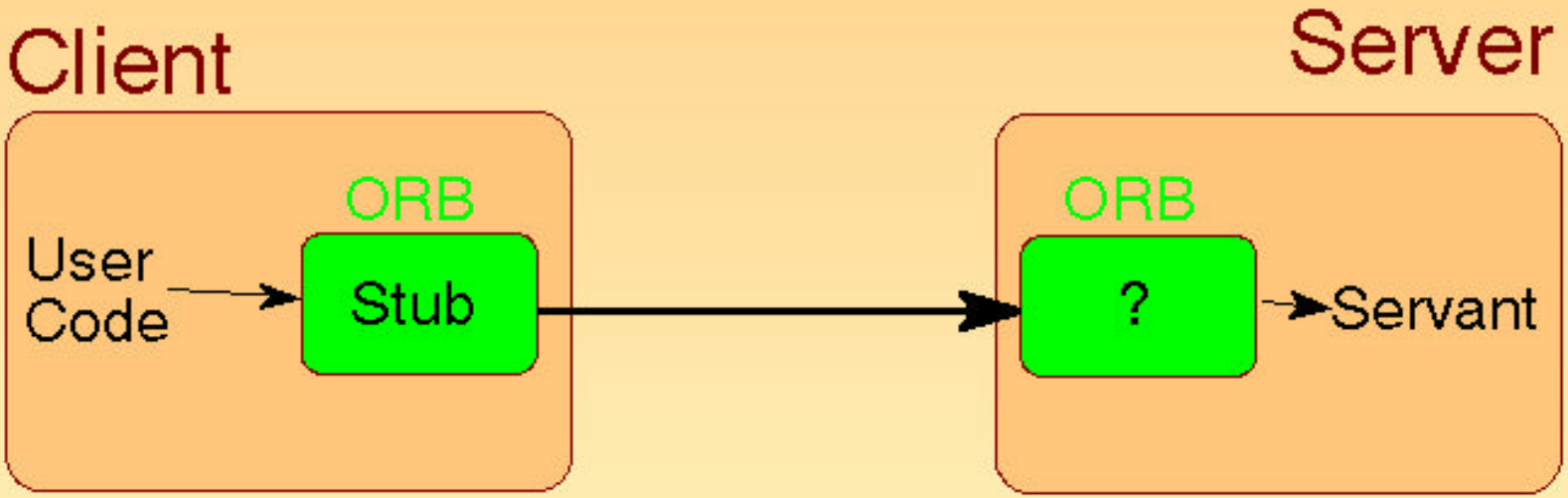
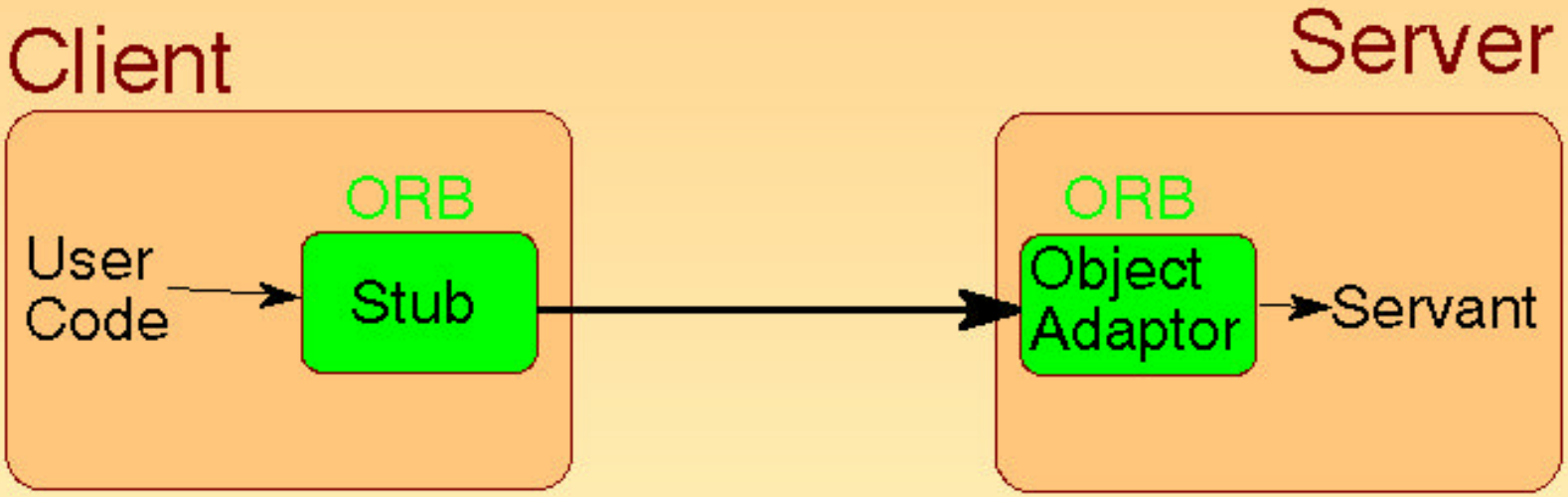
CORBA

Der Portable Object Adapter (POA)

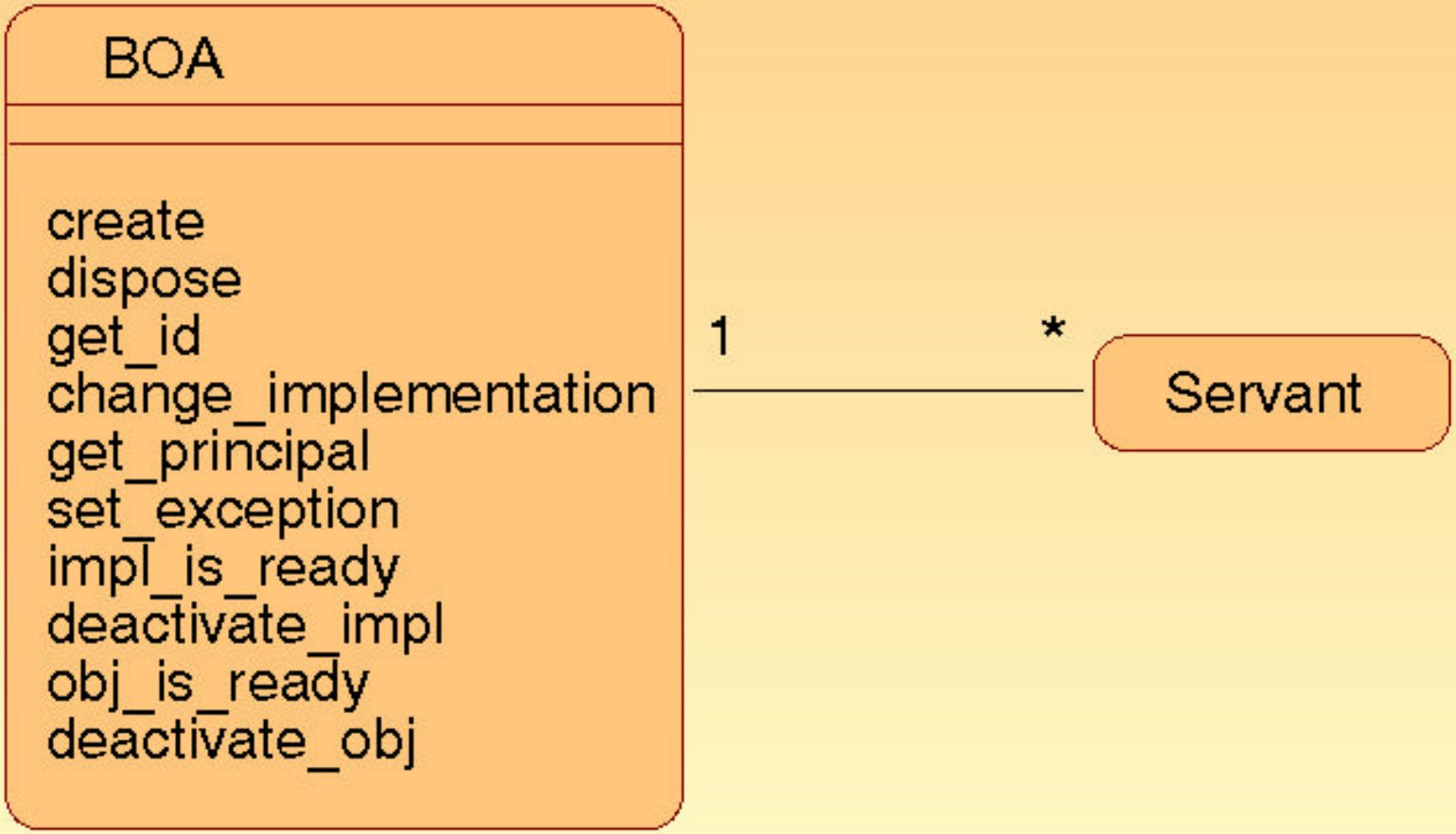
ORB - Architektur



ORB - Architektur



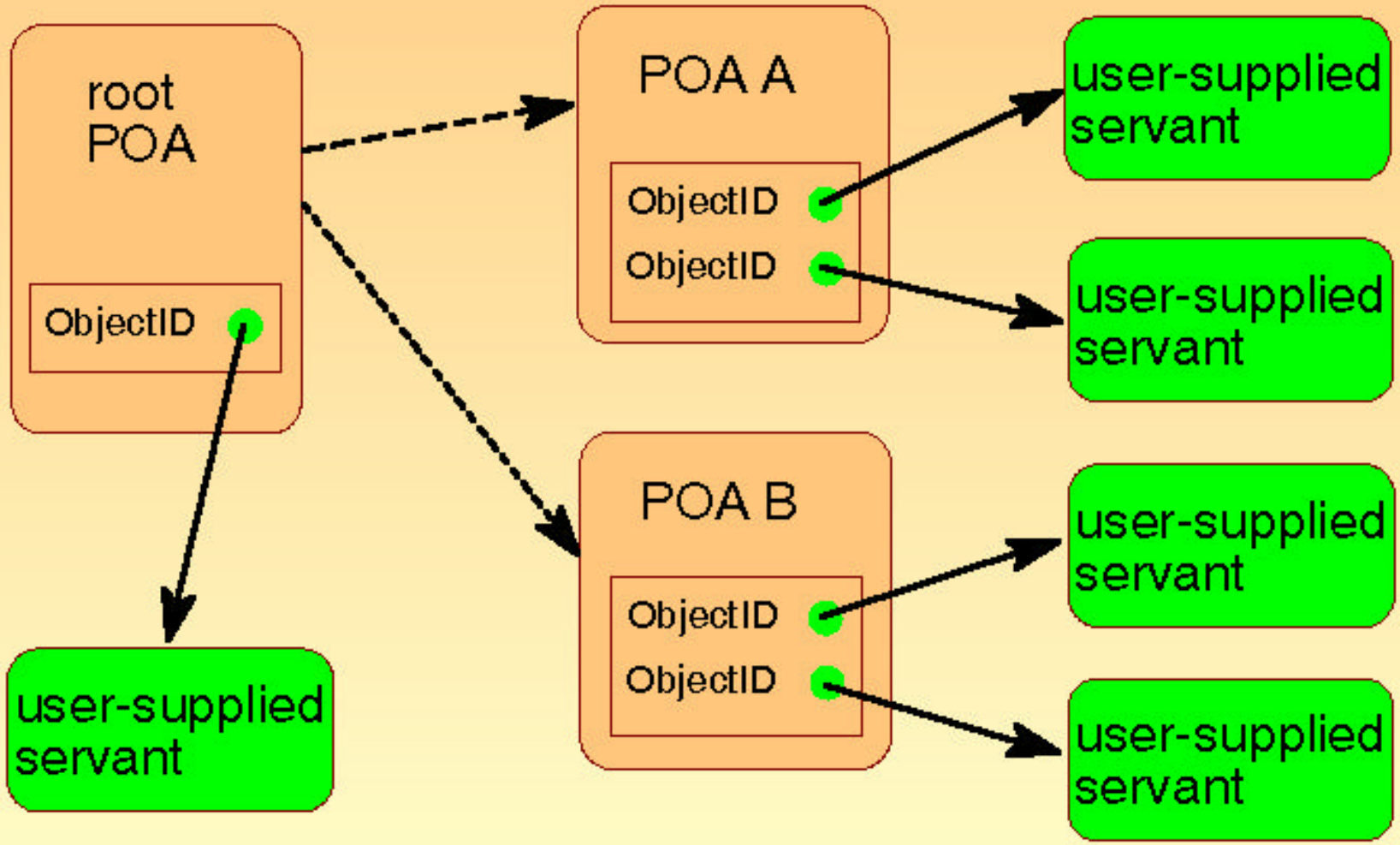
BOA - Basic Object Adaptor



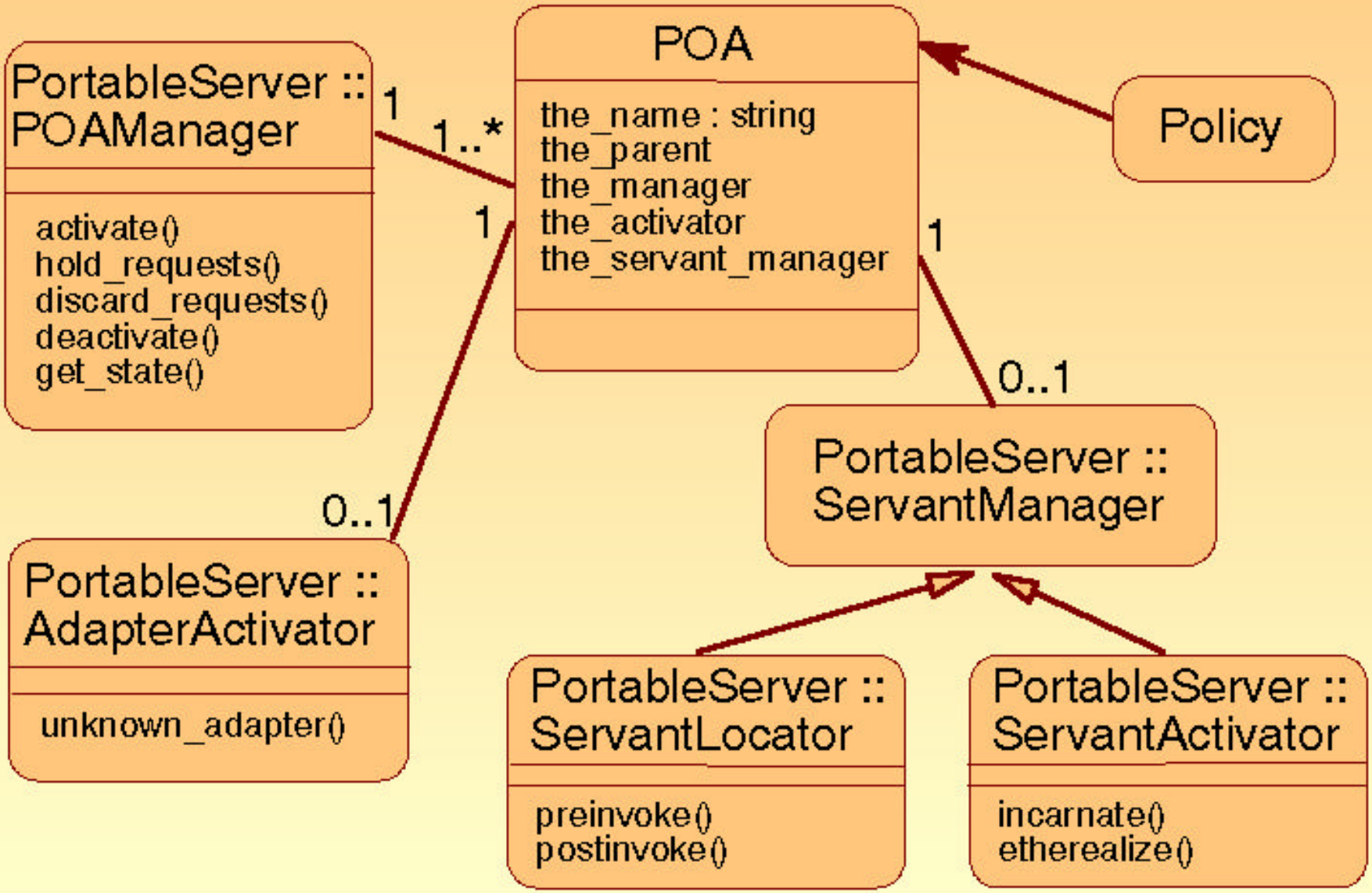
Unzulänglichkeiten des BOA

- keine Unterstützung für persistente Objekte
- keine Unterstützung für implizite Servant-Aktivierung
- keine Kontrolle über die Objekt-Identität (Object-ID)
- kaum Konfigurationsmöglichkeiten des BOAs
- keine Möglichkeit, die Objektimplementation zur Laufzeit auszutauschen

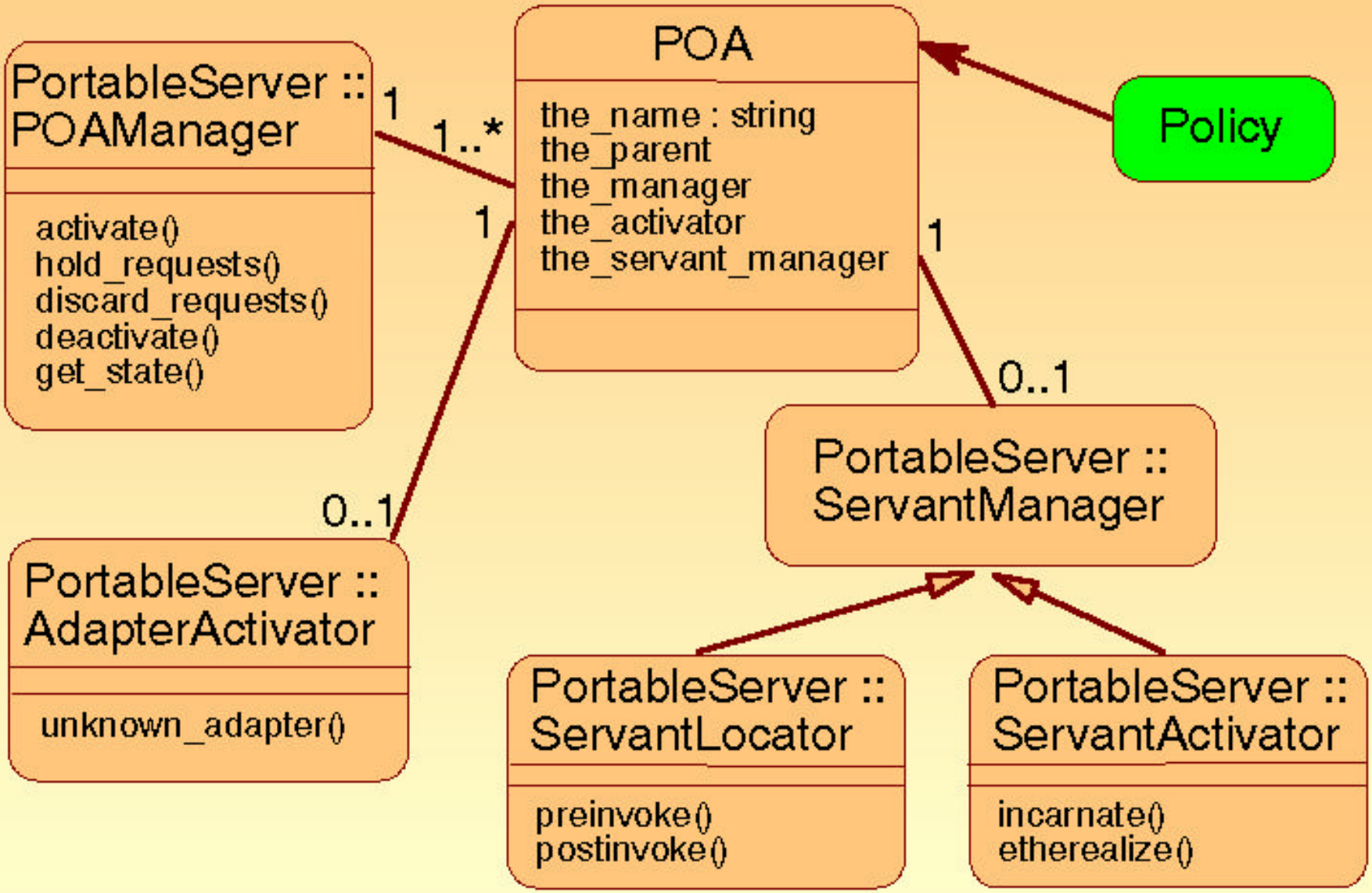
root POA und Sub-POAs



POA-Architektur



POA-Architektur



Policies

ServantRetentionPolicyValue = {RETAIN, NON_RETAIN}

idAssignmentPolicyValue = {USER_ID, SYSTEM_ID}

idUniquenessPolicyValue = {UNIQUE_ID, MULTIPLE_ID}

ImplicitActivationPolicyValue =
{IMPLICIT_ACTIVATION, NO_IMPLICIT_ACTIVATION}

LifespanPolicyValue = {TRANSIENT, PERSISTENT}

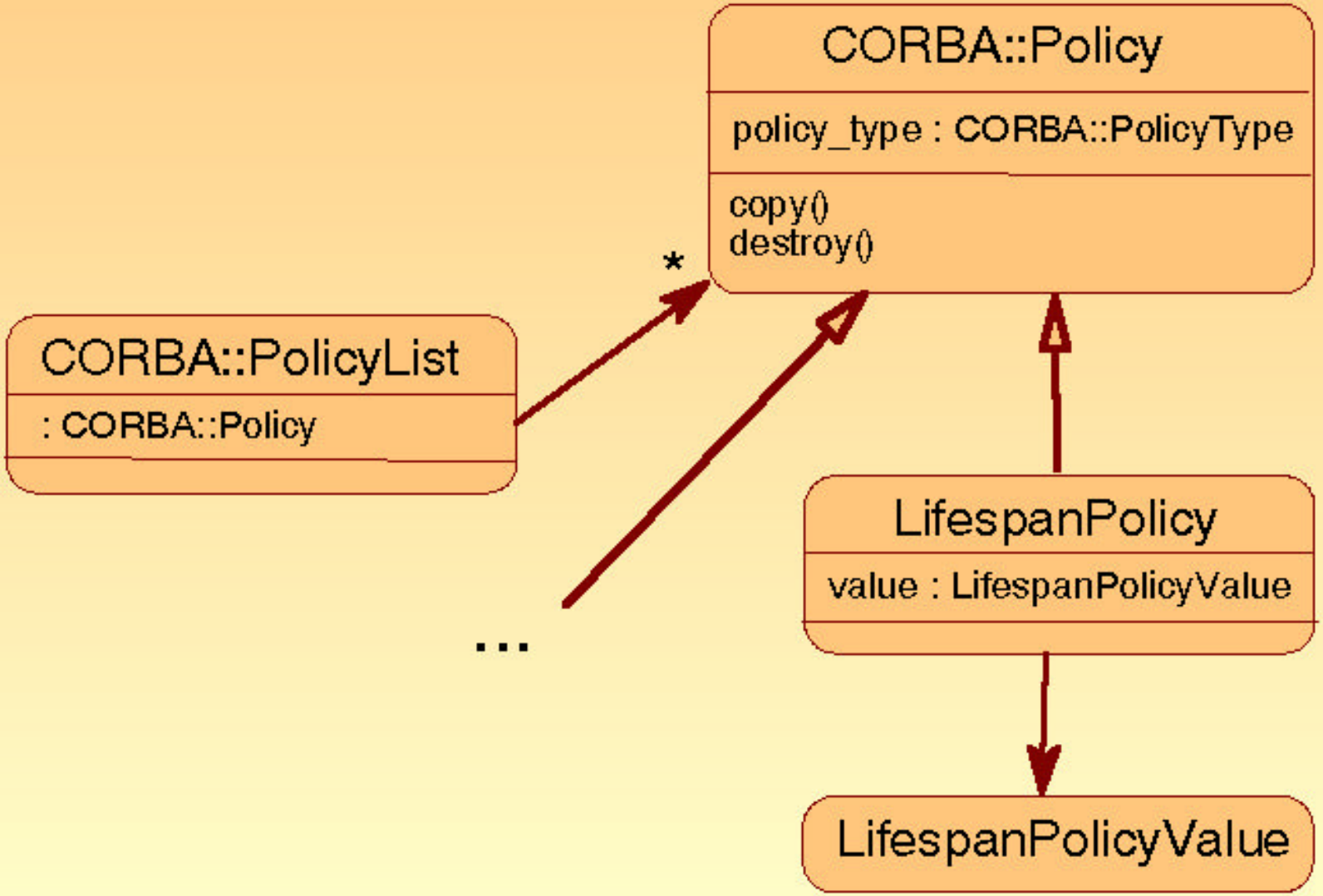
RequestProcessingPolicyValue =
{USE_ACTIVE_OBJECT_MAP_ONLY,
USE_DEFAULT_SERVANT,
USE_SERVANT_MANAGER}

ThreadPolicyValue =
{ORB_CTRL_MODEL, SINGLE_THREAD_MODEL}

Standard - Policies

Thread Policy:
ORB_CTRL_MODEL
Lifespan Policy:
TRANSIENT
Object ID Uniqueness Policy:
UNIQUE_ID
ID Assignment Policy:
SYSTEM_ID
Servant Retention Policy:
RETAIN
Request Processing Policy:
USE_ACTIVE_OBJECT_MAP_ONLY
Implicit Activation Policy:
IMPLICIT_ACTIVATION

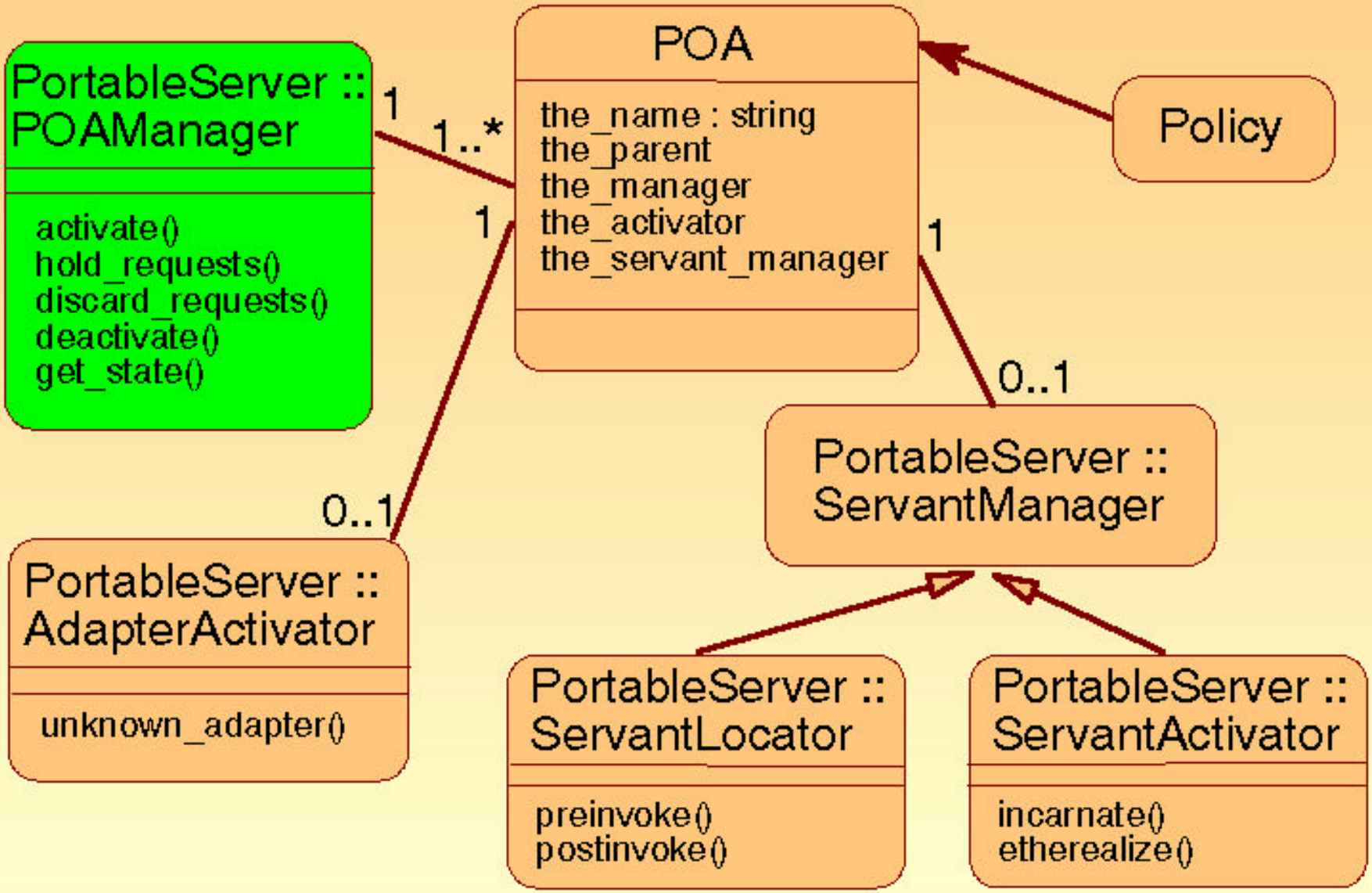
Policies


CORBA::Policy
policy_type : CORBA::PolicyType
copy()
destroy()
*
CORBA::PolicyList
: CORBA::Policy
...
LifespanPolicy
value : LifespanPolicyValue
LifespanPolicyValue

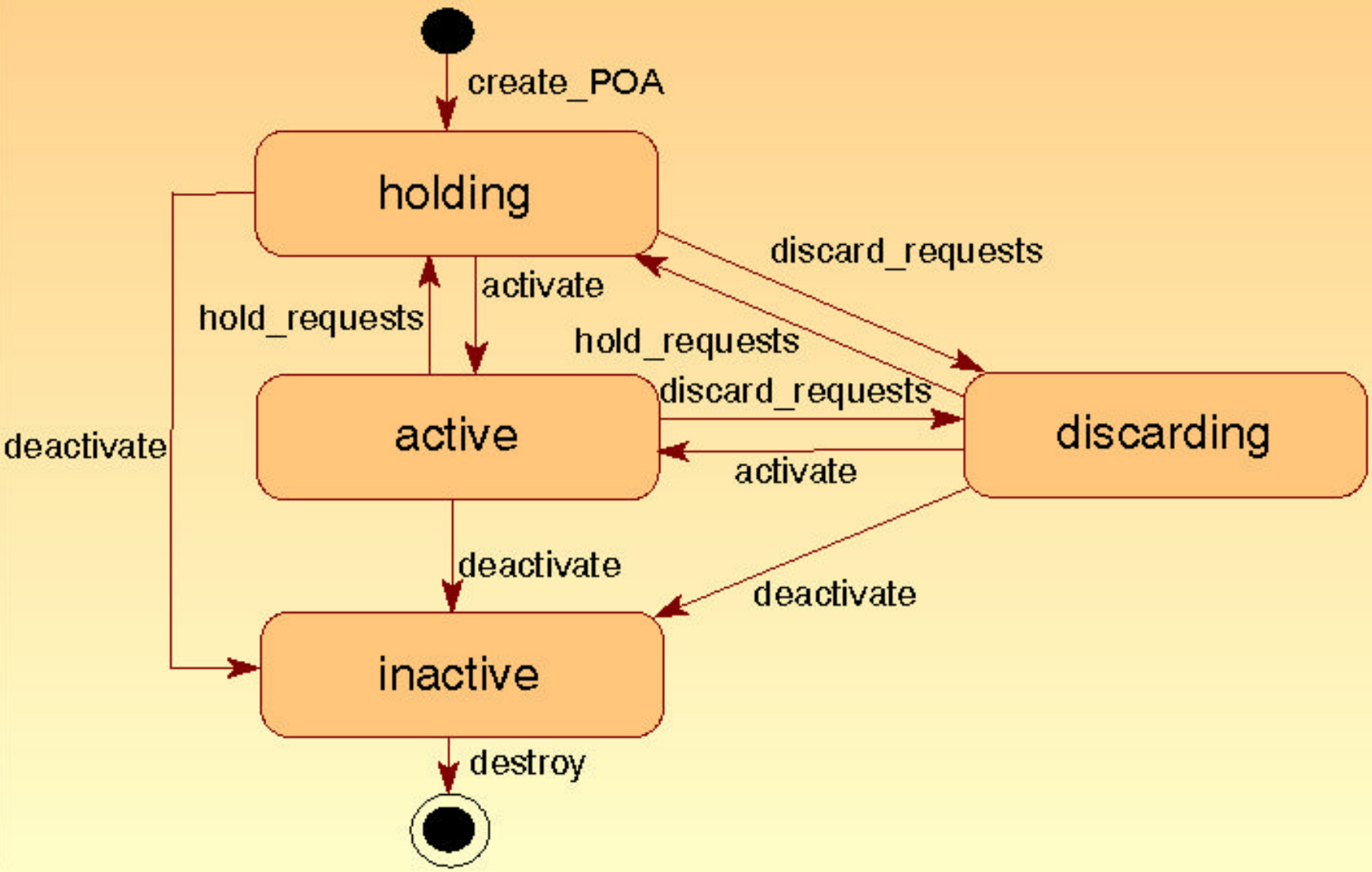
Policies

```
// C++
// rootPOA holen
CORBA::ORB_ptr orb = CORBA::ORB_init(argc, argv);
CORBA::Object_ptr pfobj = orb -> resolve_initial_references("RootPOA");

using namespace PortableServer;

POA_ptr rootPOA;
rootPOA = POA::narrow(pfobj);

// neuen POA erzeugen
::CORBA::PolicyList policies(2);

policies.length(2);

policies[0] = rootPOA -> create_thread_policy(
                ThreadPolicy::ORB_CTRL_MODEL);

policies[1] = rootPOA -> create_lifespan_policy(
                LifespanPolicy::TRANSIENT);

POA_ptr poa = rootPOA -> create_POA(
                "my_little_poa", POAManager::_nil(), policies);
```

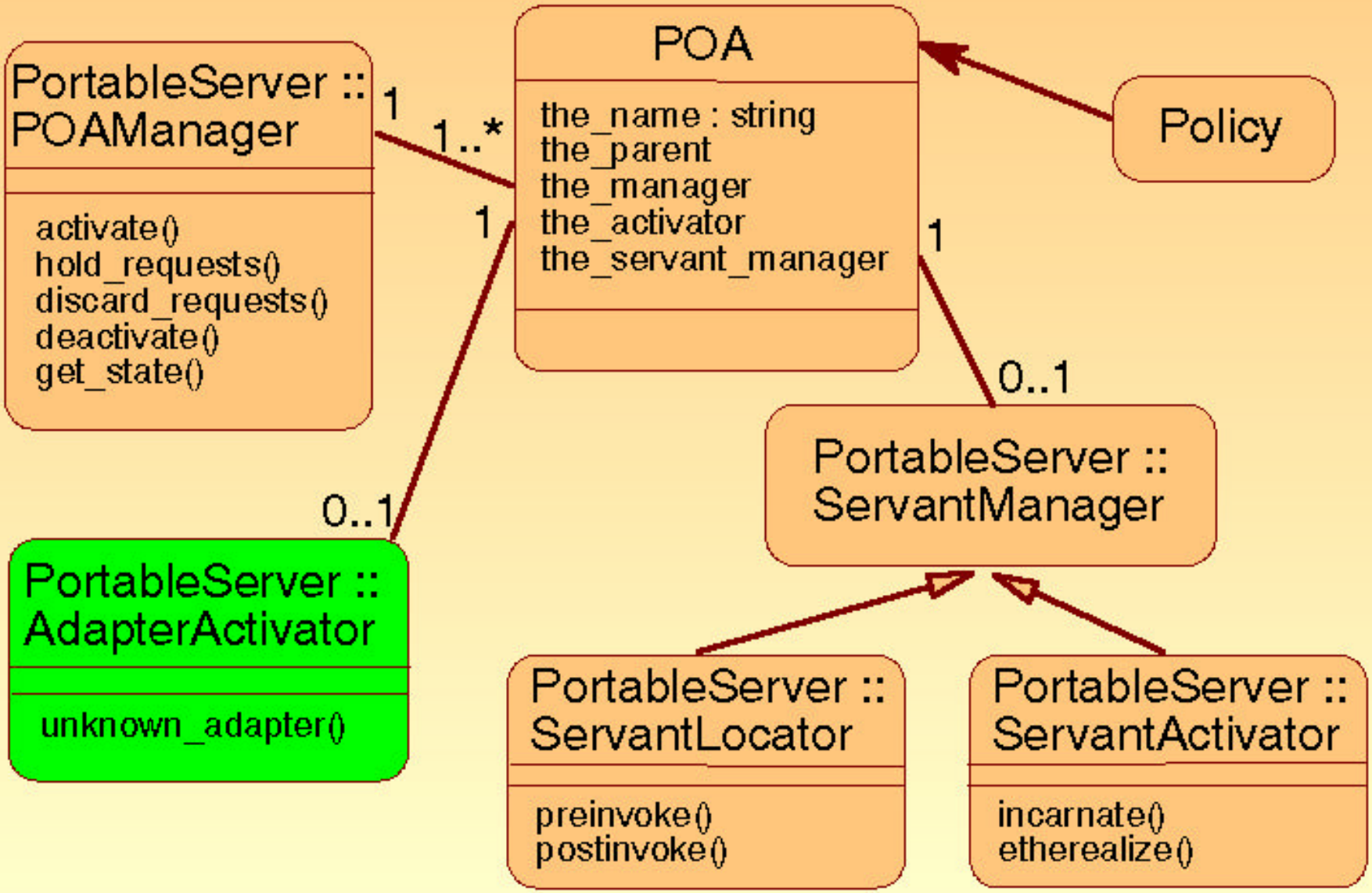
POA-Architektur



POAManager


create_POA
holding
activate
hold_requests
discard_requests
hold_requests
discard_requests
active
discarding
activate
deactivate
deactivate
deactivate
inactive
destroy

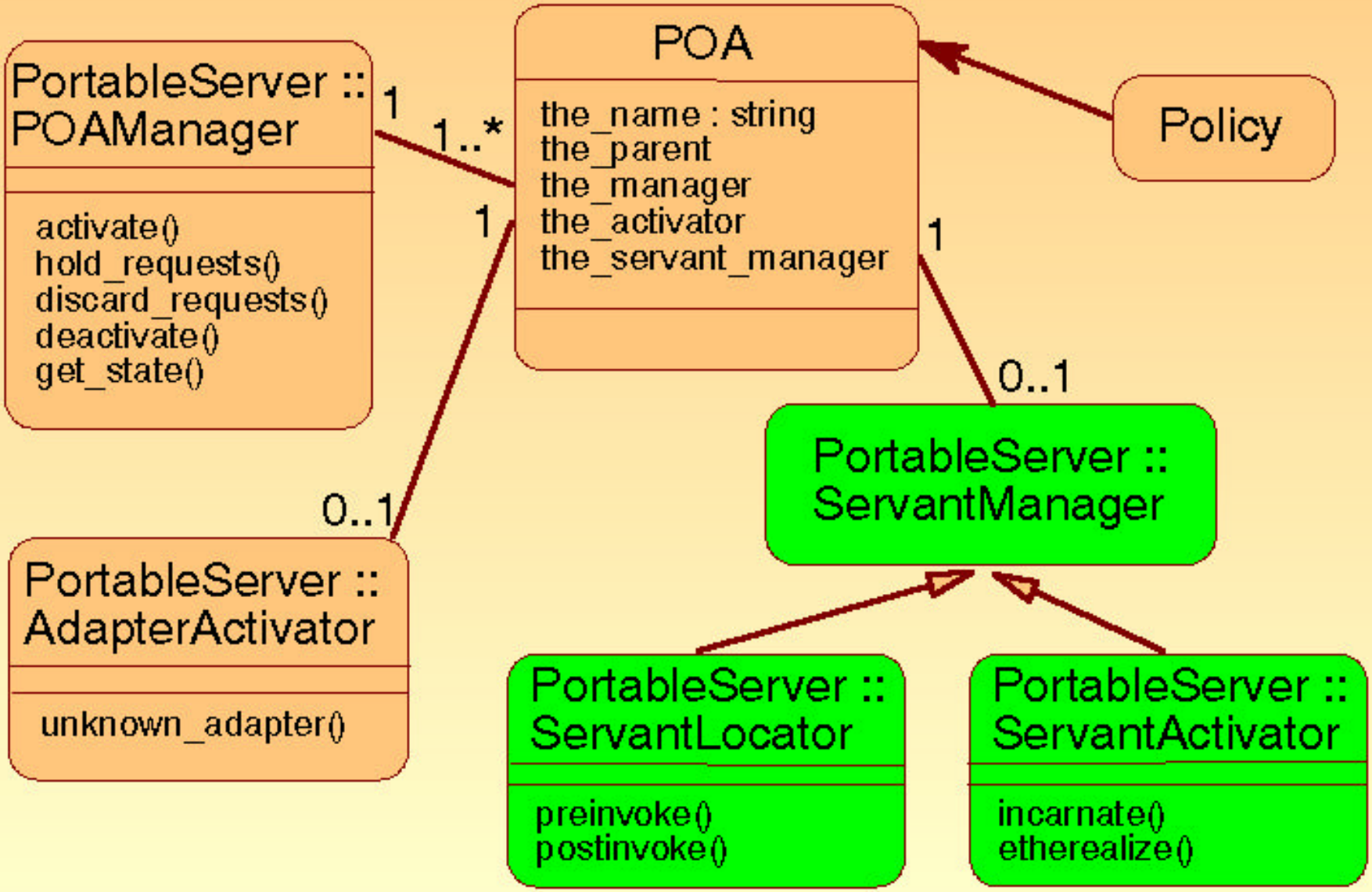
POA-Architektur



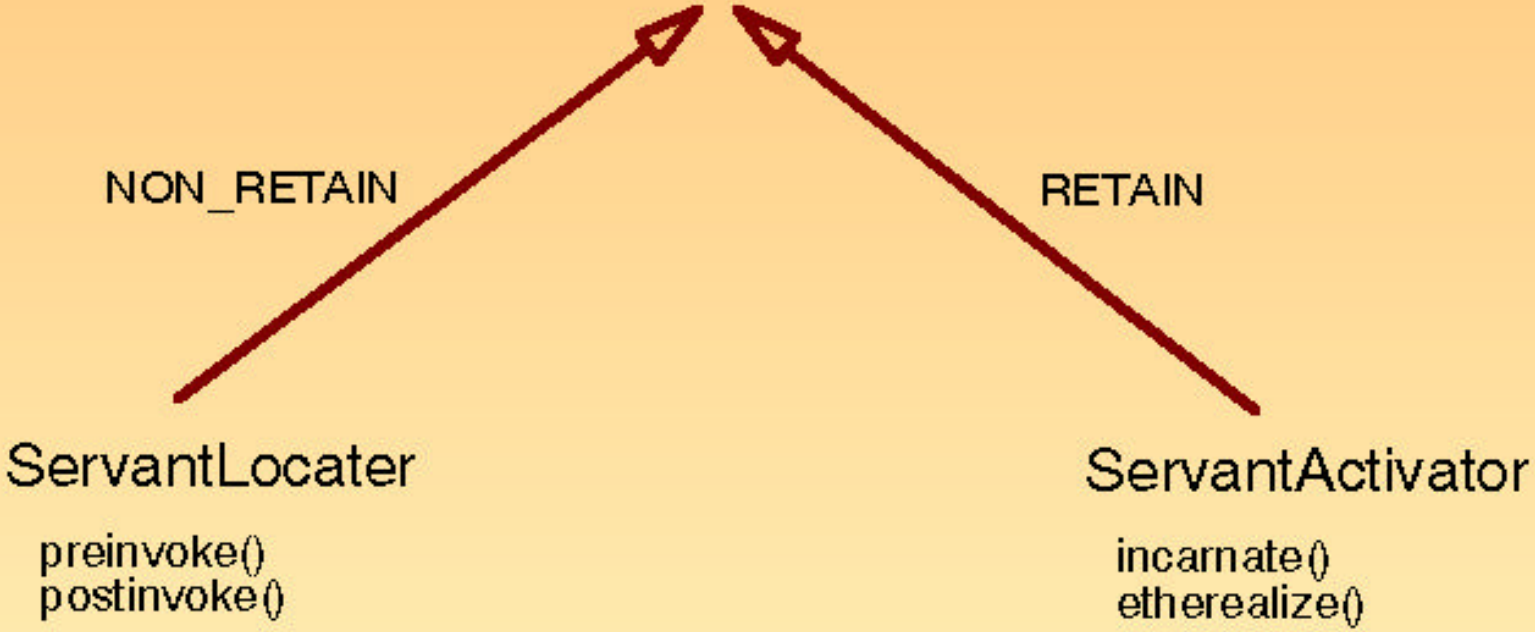
AdaptorActivator

- optional
- wird benutzt, wenn POA::findPOA mit einem unbekannten POA-Namen aufgerufen wird, um einen neuen (Sub-) POA zu erzeugen

POA-Architektur

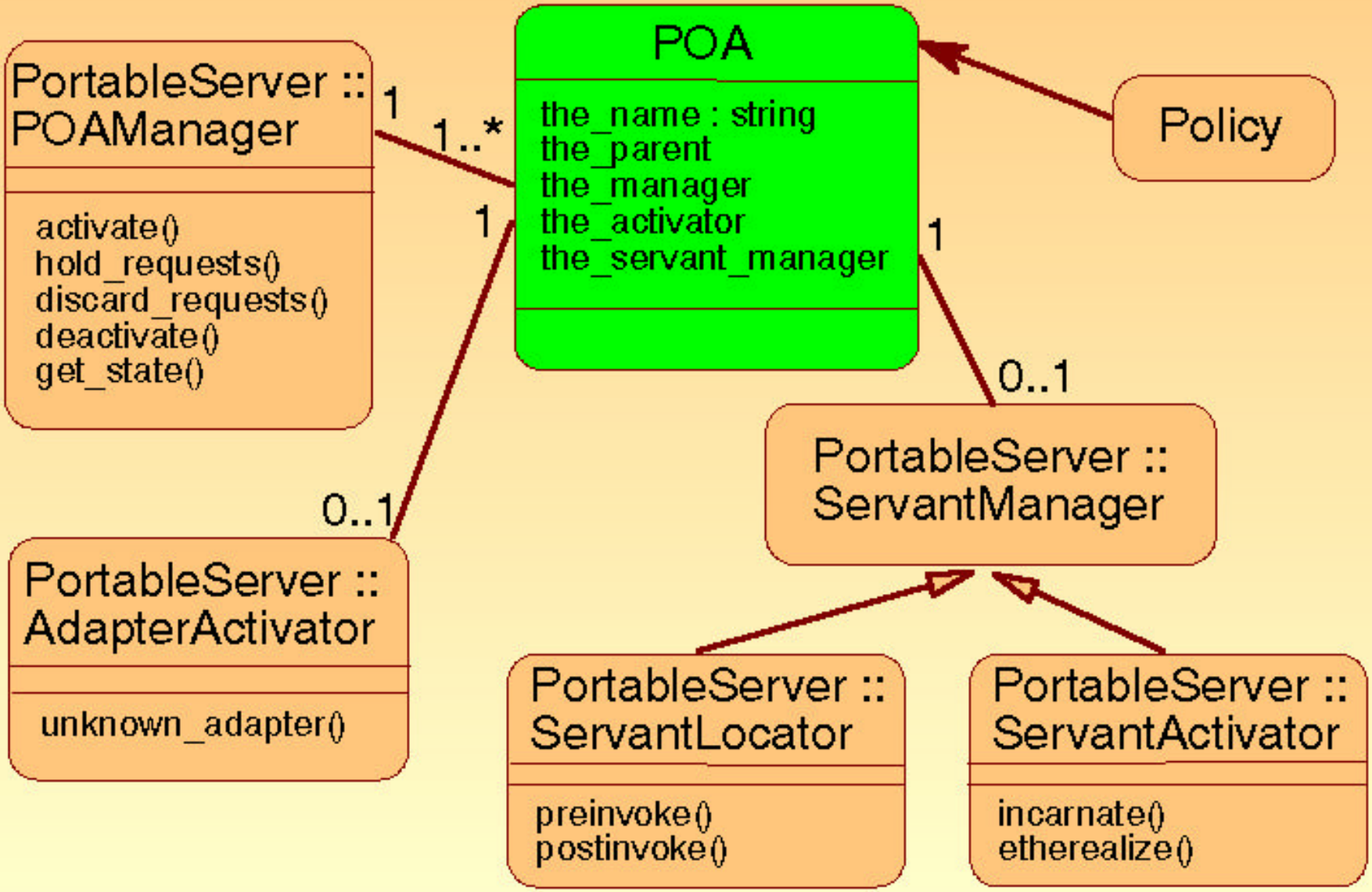


ServantManager



USE_SERVANT_MANGER muß gesetzt sein

POA-Architektur



POA

(Sub-) POA erzeugen:

```
POA create_POA(in string adapter_name,
               in POAManager a_POAManager,
               in CORBA::PolicyList policies);

POA find_POA(in string adapter_name,
             in boolean activate_it);
```

(Sub-) POA zerstören:

```
void destroy(in boolean etherealize_objects,
             in boolean wait_for_completion);
```

POA

Policy Creation

```
ThreadPolicy create_thread_policy
                    (in ThreadPolicyValue value);
...
```

Attribute (public):

```
readonly attribute string the_name;

readonly attribute POA the_parent;

readonly attribute POAManager the_POAManager;

attribute AdapterActivator the_activator;
```

POA

Attribute (protected):

ServantManager:

ServantManager get_servant_manager();

void set_servant_manager(in ServantManager imgr);

(erfordert die USE_SERVANT_MANAGER - Policy)

DefaultServant:

Servant get_servant();

void set_servant(in Servant p_servant);

(erfordert die USE_DEFAULT_SERVANT - Policy)

POA

Aktivierung von Objekten:

ObjectId activate_object(in Servant p_servant);
(erfordert SYSTEM_ID - und RETAIN - Policy)

void activate_object_with_id
(in ObjectId oid, in Servant p_servant);
(erfordert RETAIN - Policy)

Deaktivierung von Objekten:

void deactivate_object(in ObjectId oid);
(erfordert RETAIN - Policy)

POA

```
Object create_reference(in CORBA::RepositoryId intf);

Object create_reference_with_id(
        in ObjectId oid, in CORBA::RepositoryId intf);

ObjectId servant_to_id(in Servant p_servant);

Object servant_to_reference(in Servant p_servant);

Servant reference_to_servant(in Object reference);

ObjectId reference_to_id(in Object reference);

Servant id_to_servant(in ObjectId oid);

Object id_to_reference(in ObjectId oid);
```

PortableServer::Current : CORBA::Current

Eine Instanz von Current wird mittels

CORBA::ORB::resolve_initial_references("POACurrent");

erlangt.

Innerhalb eines Servants kann dann während einer POA-dispatchten Methode dieses Interface genutzt werden:

POA get_POA();

ObjectId get_object_id();