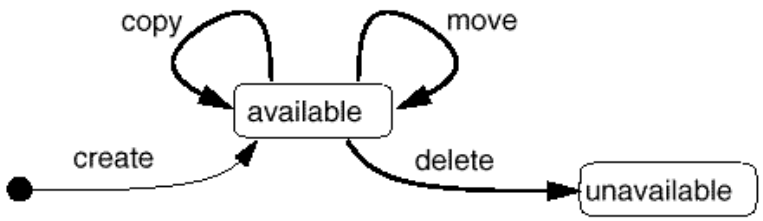
Distributed Object Lifecycle

The Lifecycle



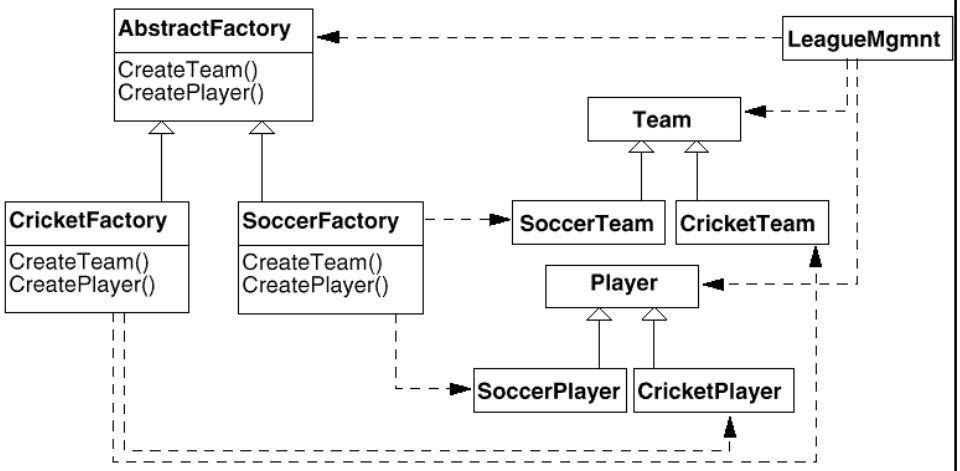
Motivation

- *Distributed object life cycle different from local object life cycle*
- *Creation:*
  - *Where to create an object*
- *Migration:*
  - *Where to copy/move an object to*
  - *How to resolve heterogeneity in data and object code representation*
- *Deletion:*
  - *Garbage collection does not work*

Object Creation

- *Clients might wish to create objects on remote machines*
- *Achieved by factories.*
- *Remote machines have to be identified in a location transparent way*
- *Achieved by factory finders.*

Factories

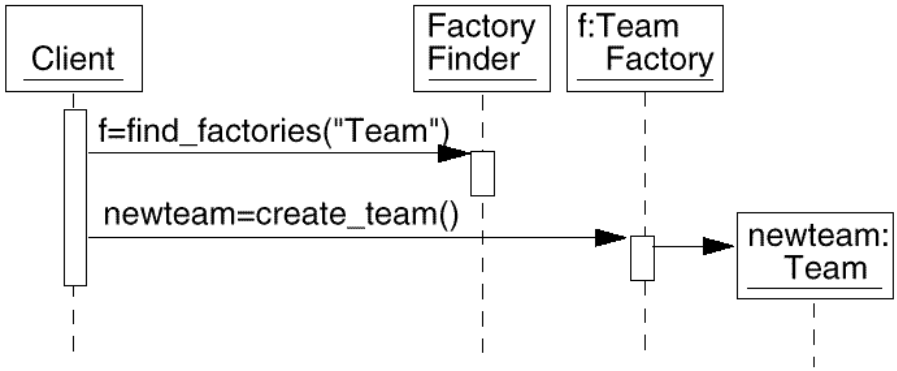


- *In distrib. systems: factories hide location*

Factory Finders

- *Location should not be transparent to everybody*
- *Administrators should be able to decide where to create new objects*
- *Policies are implemented using FactoryFinder objects.*
- *FactoryFinders export an operation find_factories that returns a suitable factory and thus implements the location policy.*

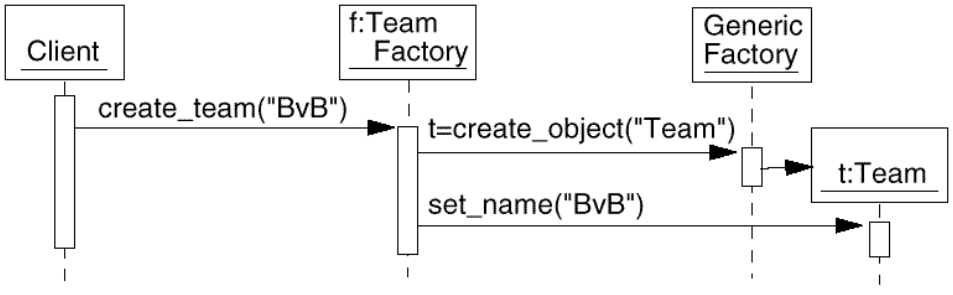
Creating Distributed Objects

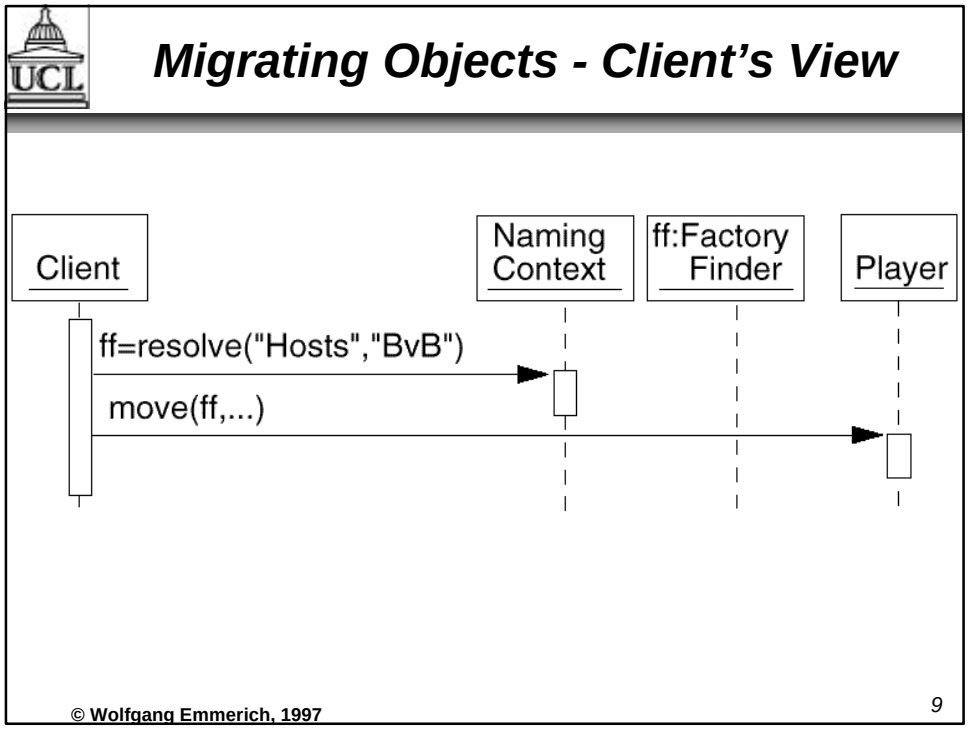


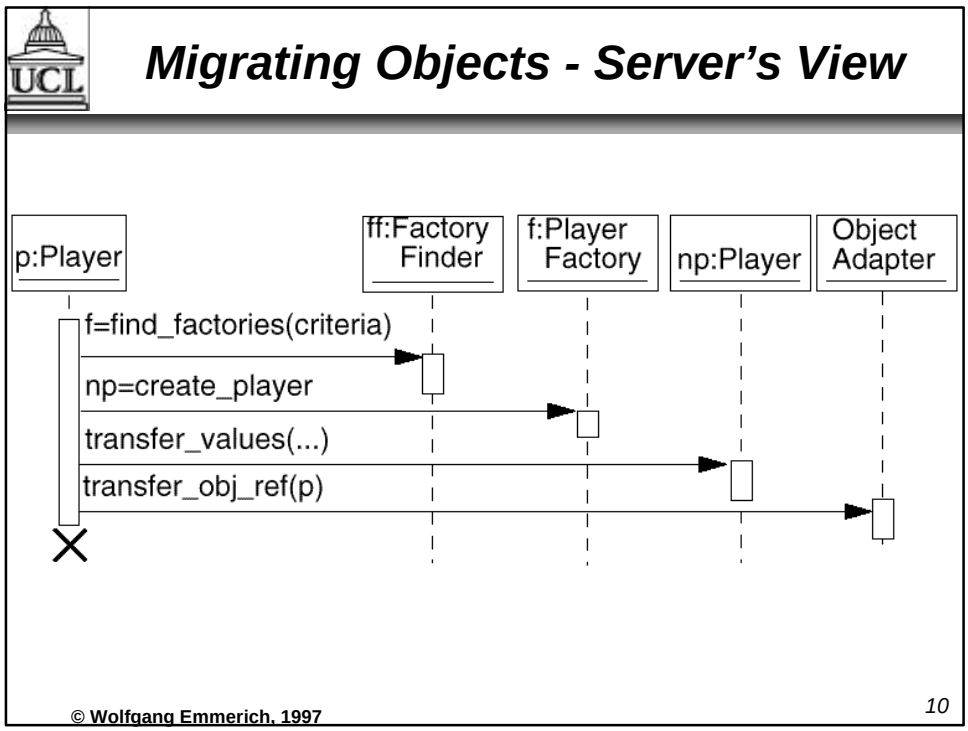
- ***New team object will be created on machine that hosts TeamFactory f.***

Implementing Factories

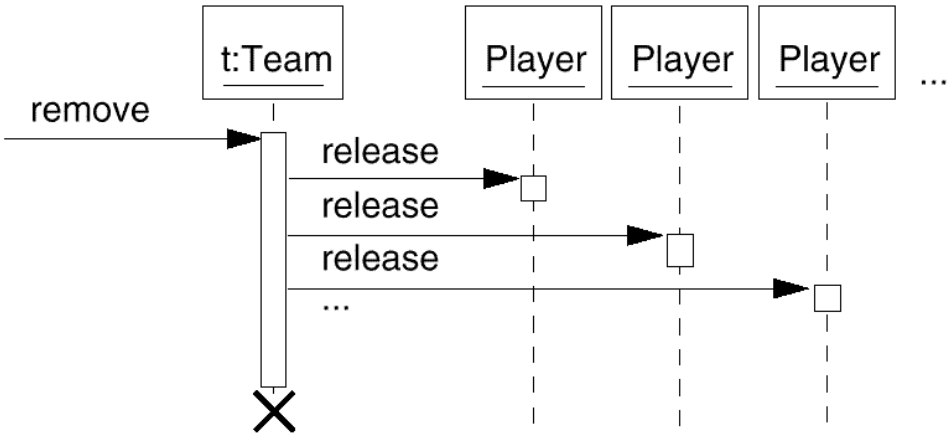
- ***Encapsulation of generic factories in type specific factories.***




Migrating Objects - Client's View
Client
Naming Context
ff:Factory Finder
Player
ff=resolve("Hosts","BvB")
move(ff,...)
© Wolfgang Emmerich, 1997
9


Migrating Objects - Server's View
p:Player
ff:Factory Finder
f:Player Factory
np:Player
Object Adapter
f=find_factories(criteria)
np=create_player
transfer_values(...)
transfer_obj_ref(p)
© Wolfgang Emmerich, 1997
10

Deleting Objects

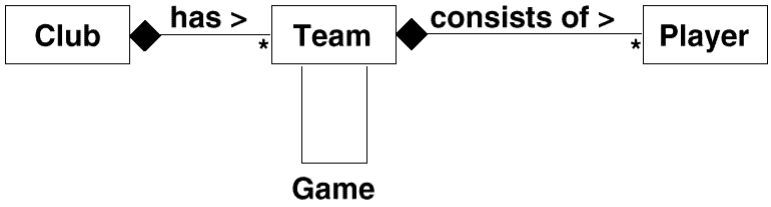


What´s Missing: Replication

- *Copies made by life cycle service are separate and do not evolve together.*
- *Life cycle service cannot be used for replication.*
- *Replicated objects reflect each other´s state changes and hence evolve together.*
- *Replication used for*
  - *load balancing*
  - *fault-tolerance.*

Composite Objects

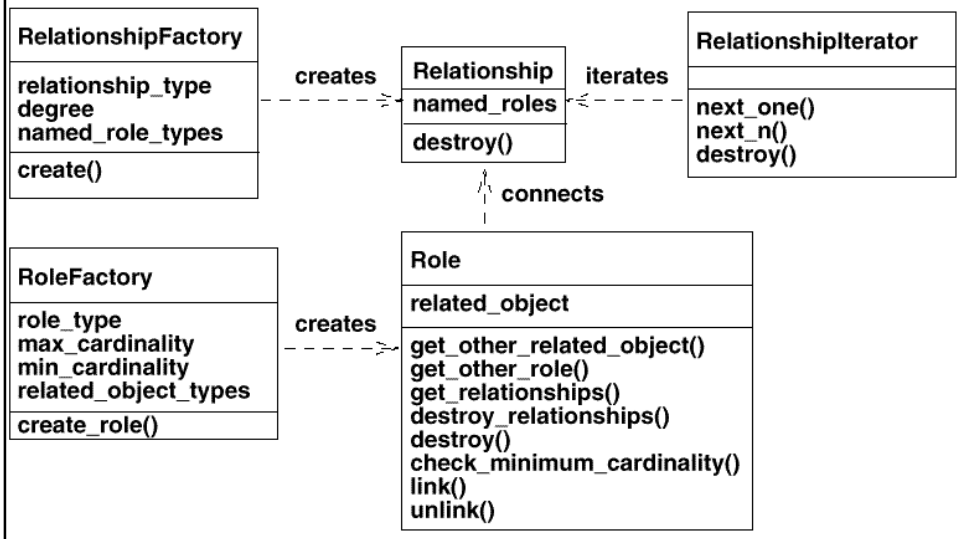
- *Consist of atomic objects*
- *Control life cycle of atomic objects*
- *Modelling of composite objects:*



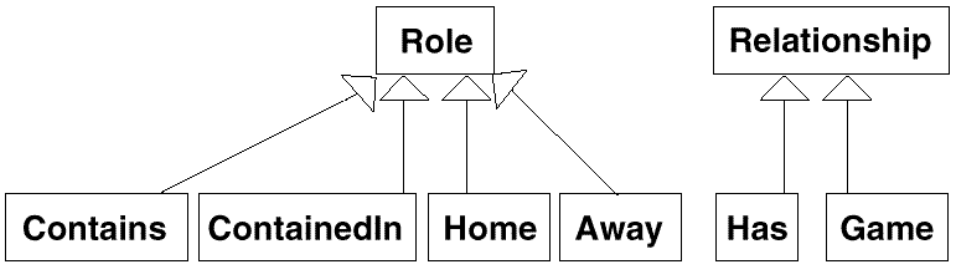
Base Relationships

- *Defined by a set of roles two or more objects play (ownership, containment, reference, employment, ...).*
- *Objects can play different roles.*
- *Cardinality defines maximum number of relationships in which a role is involved.*
- *Degree defines number of roles of a relationship (e.g. binary or ternary).*
- *Relationship may have attributes.*
- *Related objects form a graph of nodes (objects) and edges (relationships).*

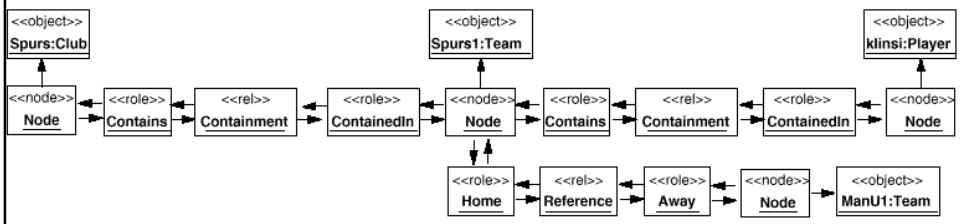
CORBA Relationship Service



Relationship and Role Examples

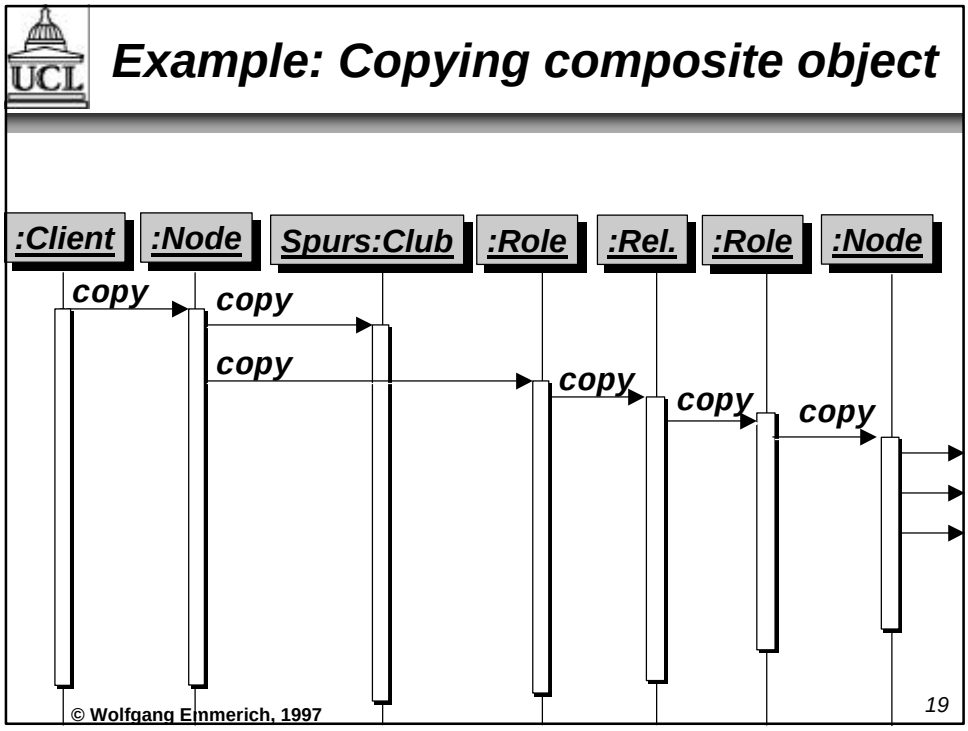


Graph of Related Objects

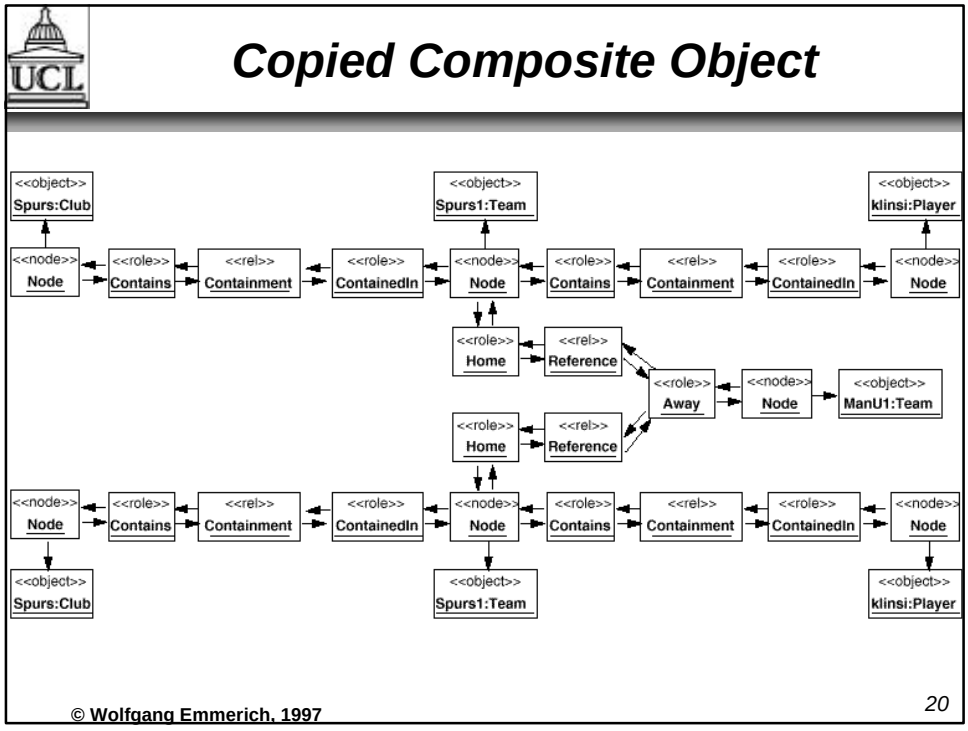


Composite Object Lifecycle

- *Apply lifecycle operations to root nodes*
- *All nodes that are in transitive closure of containment relationship will be copied/moved/deleted.*
- *All relationships internal to that closure will be copied/moved/deleted.*
- *All objects that are connected to these nodes will be copied/moved/deleted.*



Example: Copying composite object



Copied Composite Object