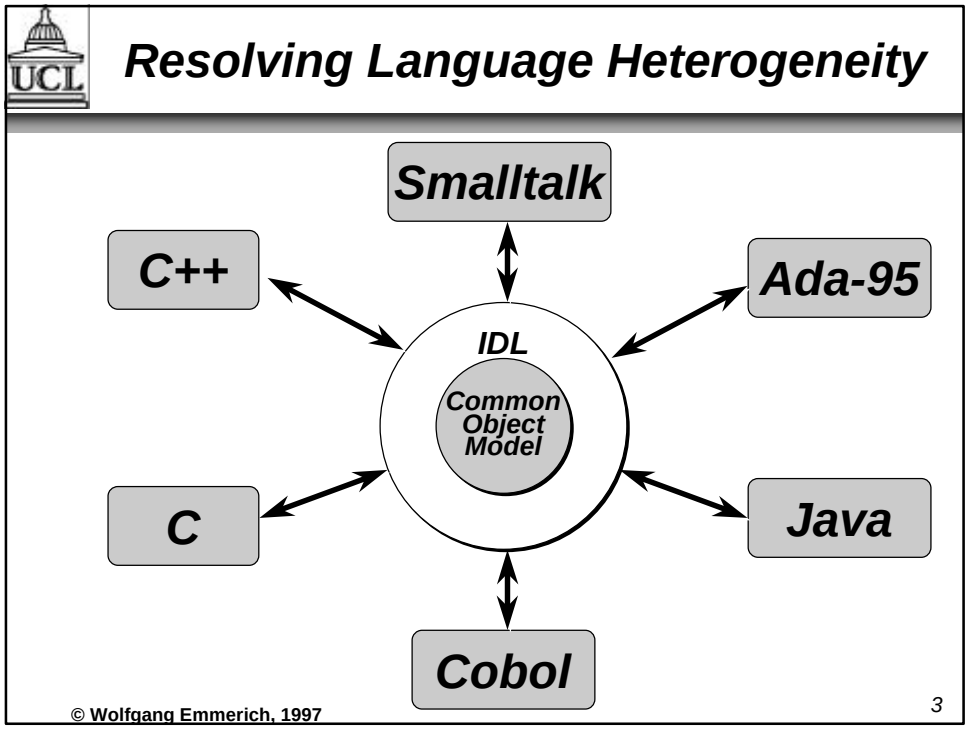
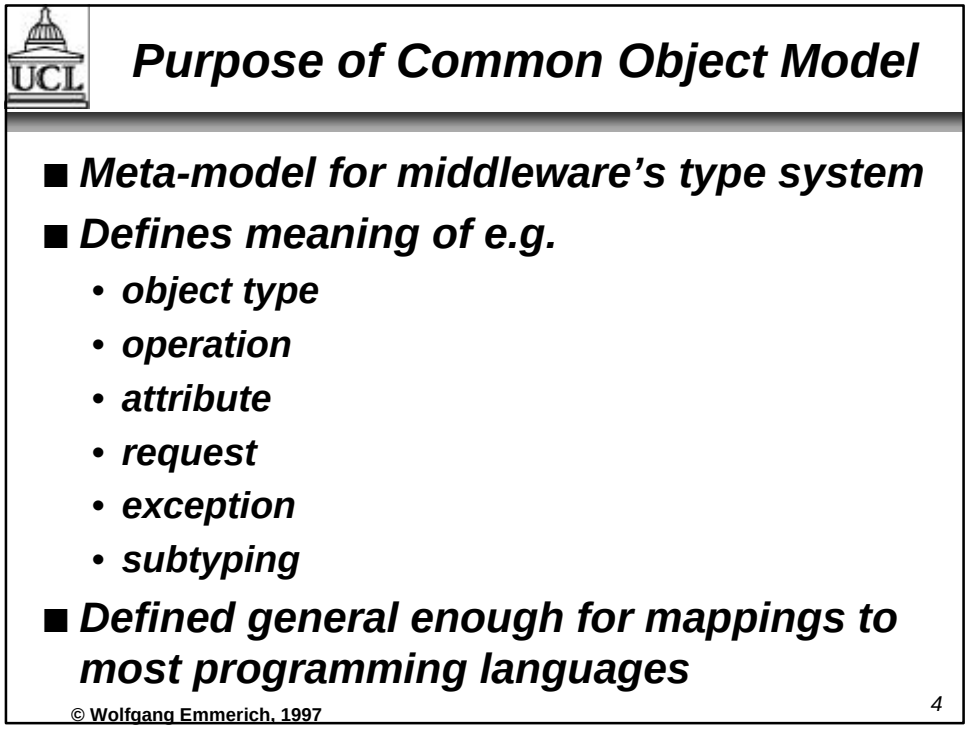
Resolving Language Heterogeneity

Motivation

- *Components of distributed systems are written in different programming languages*
- *Programming languages may or may not have their own object model*
- *Object models largely vary*
- *Differences need to be overcome in order to facilitate integration*


Resolving Language Heterogeneity
Smalltalk
C++
Ada-95
IDL
Common Object Model
C
Java
Cobol
© Wolfgang Emmerich, 1997
3


Purpose of Common Object Model
■ Meta-model for middleware's type system
■ Defines meaning of e.g.
• object type
• operation
• attribute
• request
• exception
• subtyping
■ Defined general enough for mappings to most programming languages
© Wolfgang Emmerich, 1997
4

Interface Definition Language

- *Language for expressing all concepts of the middleware's object model*
- *Should be*
  - *programming-language independent*
  - *not computationally complete*
- *Bindings to different programming languages are needed*
- *As an example: OMG object model and OMG/IDL*

Attributes of Objects

- *Attributes have a name and a type*
- *Type can be object or non-object type*
- *Attributes are readable by other components*
- *Attributes may or may not be modifiable by other components*
- *Attributes correspond to one or two operations (set/get)*

Example IDL Attributes

```
readonly attribute ATMList ATMs;
readonly attribute BankList banks;
```


Exceptions

- Service requests in a distributed system may fail
- Exceptions explain reason of failure to service requester
- Operation execution failures may be
  - generic
  - specific
- Specific failures are explained in type specific exceptions

Example IDL Exceptions

```
exception InvalidPIN;
exception InvalidATM;
exception NotEnoughMoney {
          short available;
};
```


Operations

- Operations have a signature
- Signature consists of
  - name
  - list of in, out, or inout parameters
  - return value type
  - list of exceptions operation may raise

IDL Operation Examples

```
void accept_request(in Requester req,
                    in short amount)
             raises(InvalidPIN,
                    NotEnoughMoney);
short money_in_ATM (in ATM dispenser)
             raises(InvalidATM);
```


Object Types

- Objects export attributes, operations and exceptions
- Multiple objects may export the same properties
- Only define the properties once
- Attributes, operations and exceptions are defined in object types

Example IDL Interfaces

```
interface ATM;
interface TellerCtrl {
 typedef sequence<ATM> ATMList;
 exception InvalidATM;
 exception NotEnoughMoney{...};
 readonly attribute ATMList ATMs;
 readonly attribute BankList banks;
 void accept_request(in Requester req,
                     in short amount)
    raises(InvalidPIN,NotEnoughMoney);
};
```


Subtyping

- Properties shared by several types should be defined only once
- Types organised in type hierarchy
- Subtypes inherit attributes, exceptions and operations from super-types
- Subtypes can add properties
- Subtypes may redefine inherited properties

Example of Inheritance in IDL

```
interface Controllee;
interface Ctrl {
 typedef sequence<Controllee> CtrleeList
 readonly attribute CtrleeList controls;
 void add(in Controllee new_controllee);
 void discard(in Controllee old_ctrlee);
};
interface ATM : Controllee {...};
interface TellerCtrl : Ctrl {...};
```


Programming Language Bindings

- Atomic data types / type constructors
- Constants
- Interfaces and multiple inheritance
- Object references
- Attribute accesses
- Operation execution requests
- Exception declaration / handling
- Modules
- Middleware interface invocations

Standardisation of Bindings

- *Facilitate portability with respect to:*
  - *Object requests*
  - *Object implementations*
  - *ORB interface invocations*
- *Decrease learning curve of developers*