Poseidon House
Castle Park
Cambridge CB3 0RD
United Kingdom

TELEPHONE: **Cambridge (01223) 515010**
INTERNATIONAL: **+44 1223 515010**
FAX: **+44 1223 359779**
E-MAIL: **apm@ansa.co.uk**

Training

OO Language (Intro to ANSA)

Yigal Hoffner

Abstract

The business problem addressed is...

The technical problem created by that business problem is ...

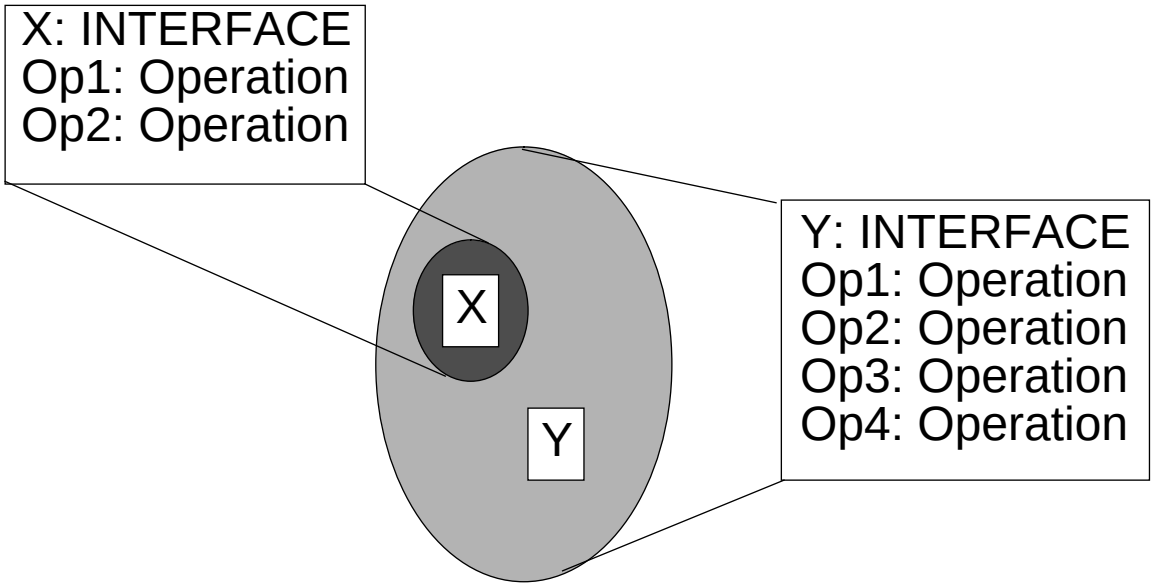
The solution being offered is....

APM.1648.00.01 **Draft** 2nd November 1995

Briefing Note

Distribution:
Supersedes:
Superseded by:

Languages and implementation techniques



In this session

- Discuss OO programming languages
- Introduces OO implementation concepts
- Consider their applicability to distributed systems

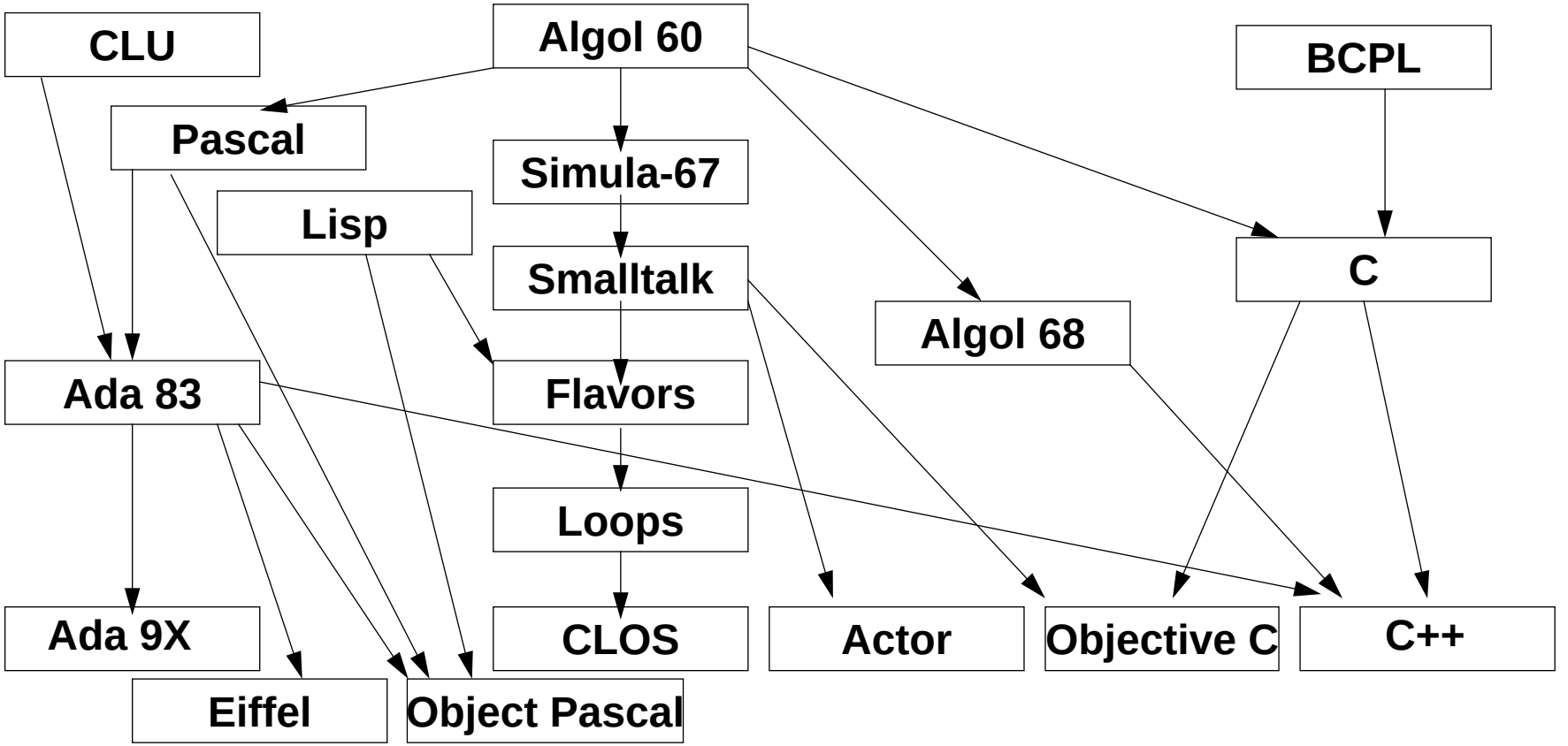
Objects - beyond modular programming

- ***Object programming is 'modular programming' that scales***
  - **to very large and complex systems**
- ***Object implementation techniques include***
  - **encapsulation**
  - **polymorphism**
  - **inheritance**
- ***In distributed systems these concepts need to be examined again***

Encapsulation in programming languages

- ***Access only via explicit interfaces***
  - C++ class
  - Smalltalk class
  - Ada package
- ***Hide the implementation***
  - the data structures
  - the algorithms
  - the state
- ***Keep the data with the code***

Evolution of OO programming languages

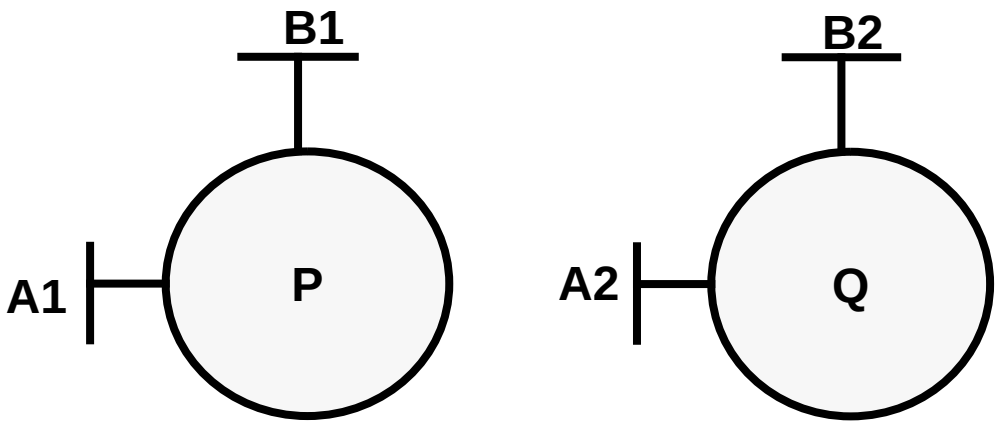


Polymorphism

- ***The same operation can apply to different types of object***
  - **you can print a text file, a structured document, and a bitmap**
- ***Each object can be expected to "know how to print itself"***
  - **it has a standard interface for printing**
  - **the object determines *how* to "print itself"**

Inheritance for objects

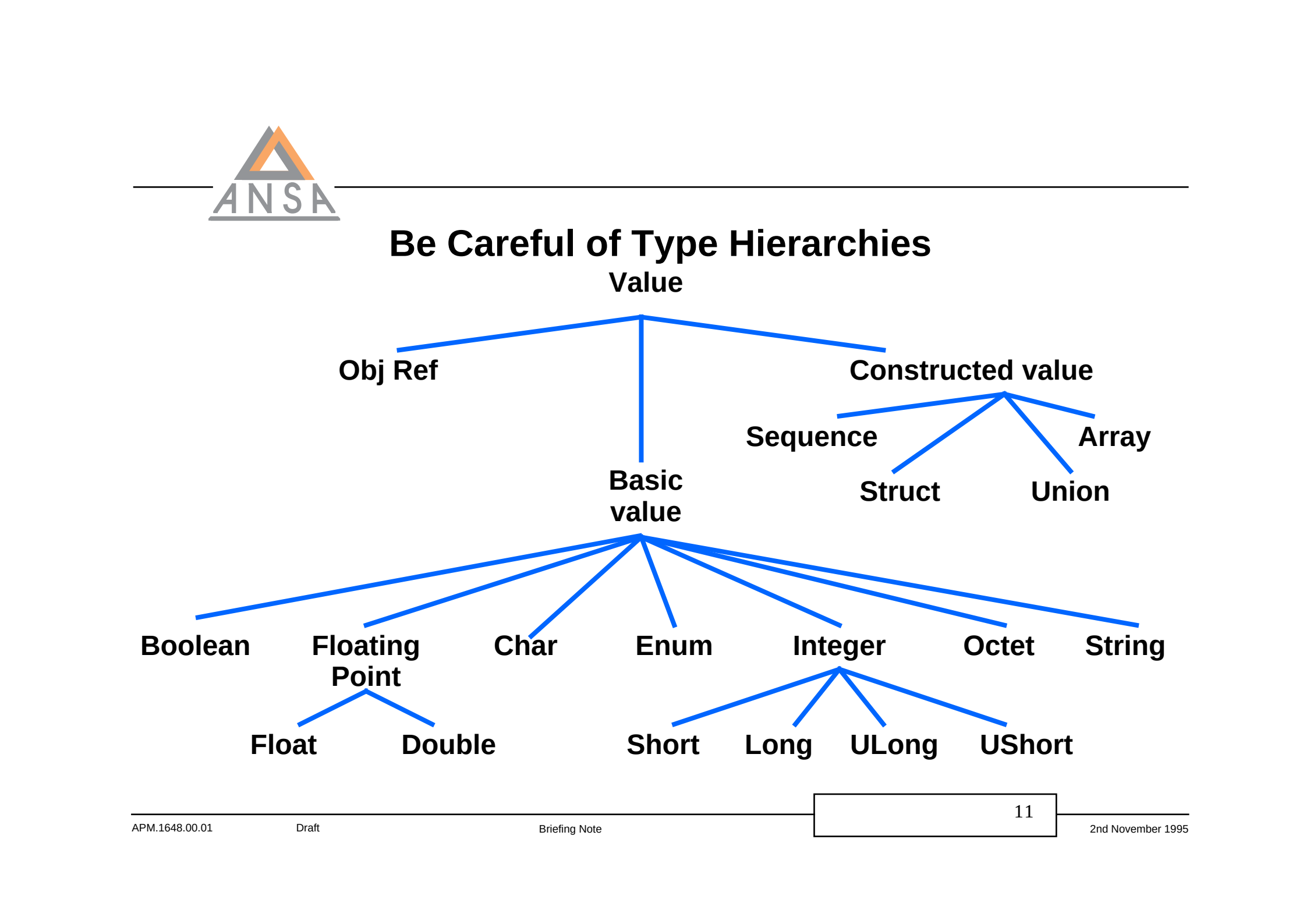
- *Consider two similar objects*



- *They can have similar interfaces A and B*
- *They can have similar implementations P and Q*
- *...these are quite separate properties*

Views of inheritance

- ***In non-distributed object systems we may be interested in similar implementations***
  - **sharing or reusing some of the code...**
  - **... for example the code for the A interface**
- ***In a distributed system this is not relevant***
  - **sharing any code between distributed objects is impossible**
- ***The important thing is compatibility of interfaces***
  - **type conformance (*substitutability)***
- ***Some programming languages try to use ‘class’ for both ideas***

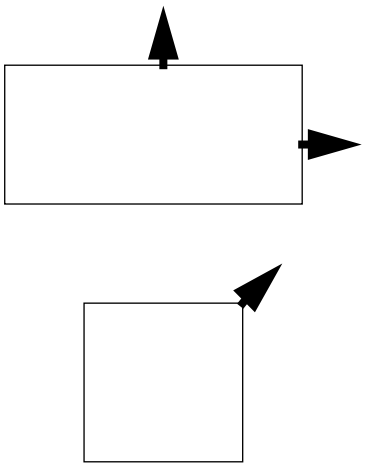

ANSA
Be Careful of Type Hierarchies
Value
Obj Ref
Basic value
Constructed value
Sequence
Array
Struct
Union
Boolean
Floating Point
Char
Enum
Integer
Octet
String
Float
Double
Short
Long
ULong
UShort

What type hierarchies can mean

- ***When someone shows you a picture like that, it could have several meanings***
- ***Does it show***
  - **type conformance relationships?**
  - **implementation relationships?**
  - **something else?**

‘Is-a’ relationships

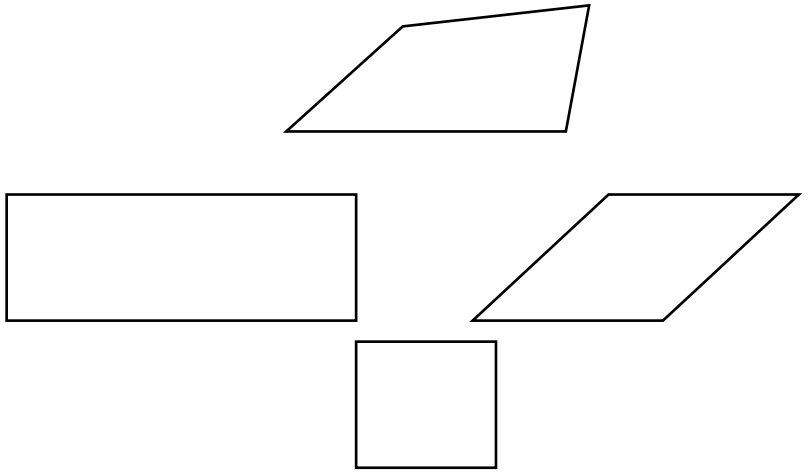
- ***A square ‘is a’ rectangle...***



- ***But their ‘resize’ operations do not conform***
  - **rectangles have separate operations for resizing horizontally and vertically**
  - **squares have only one resize operation**

Is-a relationships and type conformance

- *Type conformance does not match is-a relationships*



- *There is no 'natural' type hierarchy*
  - **all that matters is conformance**

Object lifetime

- ***Objects outlive the processes that create them***
  - **conventional pointers are inadequate...**
  - **... a more general kind of ‘handle’ is needed**
- ***In a distributed system, it must be possible to pass these ‘handles’ around between machines***
  - **their representations must be mutually recognized**
- ***Some systems have a global ‘unique id’ (UUID) for each object***
  - **this is not actually necessary**

Summary

- *The key concepts are encapsulation and type conformance*
- *You can use object systems without an object programming language*
- *Distributed systems allow you choose freely the front-end and tools technology*
  - Programming language
  - User interface/toolkit
  - Database
- *Distributed systems development is possible without objects*
  - but is harder work
- *For more on this topic*
  - see *Distributing Objects* (TR.018)