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**

ANSA Phase III

Repository Support Software - progress report slides

Andrew Watson

Abstract

The Phase III C4a workpackage deals with representing service type information at run-time. This progress report (intended for the TC) explains how this fits into the current work programme, and briefly describes the current status of the work.

Colour slides for a talk to last 30 minutes.

APM.1411.00.02 **Draft** 27th February 1995

Project Management (confidential to ANSA consortium for 2 years)

Distribution:
Supersedes:
Superseded by:

Repository Support Software - progress report
Task C4a

February ‘95

Nicola Howarth
Mike Beasley
Andrew Watson

The Motivation

- **Basic technology of distributed object-based systems becoming widely-accepted**
  - **Transparent remote invocations upon objects via opaque handles**
  - **CORBA, Microsoft COM, other proprietary designs**
- **Issues in practical, large-scale deployment of distributed objects now becoming more pressing**
  - **Currently-deployed technology oriented towards single workstations**
  - **Management and federation issues need addressing**
  - **How to discover and correctly use services in cross-enterprise systems**

Why run-time type information?

- **Locating services in large-scale information infrastructures requires meta-data**
  - **Simple name lookup inadequate**
  - **Trading**
  - **Potentially describes many aspects of a service**
  - **Descriptions of service required and that found should match for client to be allowed to acquire service reference**
  - **Allows as much correctness checking as possible as early as possible**
- **One important aspect of a service is operations it supports**
  - **... and their parameter types, and result types**
  - **... and the operations *those* types support**
  - **... and so on**
  - **The computational type**

Why run-time type information? (cont.)

- **Important to ensure that service implements operations required, with parameters required, before attempting to invoke**
  - **This can be done no earlier than bind time**
  - **i.e. the point where client acquires reference to service-provider**
  - **... so type information must be available at run-time**

Type Safety Means ...

never having to say “Sorry (method not understood)”

Current technology

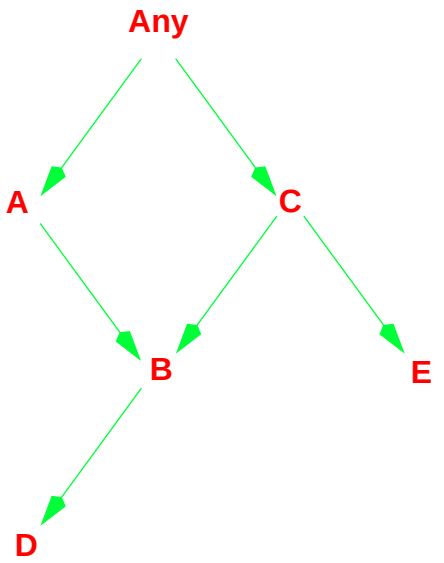
- Now is a good moment to return to prototyping trader technology
  - ODP trader at Committee Draft
  - OMG seem to be about to issue trader RFP
- ANSAware's trader includes a crude type manager
  - Does not store representations of types themselves, only IDL names
  - Relationships between type names asserted by trader administrator
  - Federation of traders relies on names being used consistently everywhere
- CORBA 1.x type (interface) management under-specified
  - Full type information stored at run-time in interface repository
  - ... but only read interface for type information documented
  - Authors expected full type (interface) manager to be specified elsewhere
  - Little support for actually using type information

Current technology (cont.)

- CORBA 2.x will extend 1.x spec in the obvious way
  - ... by adding a write interface
  - Establishes one type-manager-per-ORB assumption
  - Uses global unique IDs to represent equivalence across repositories
  - Also problems with inconsistencies across update
- COM objects (will) do their own type management
  - Types themselves not available at run-time
  - ... identified only by UUIDs
  - Substitutability established only by direct equivalence (of UUIDs)
  - Mechanism for asking object if it supports a particular type (UUID)
  - Does not preclude trading-style mechanisms
  - ... but no provision for them (yet)

Prototyping a better approach

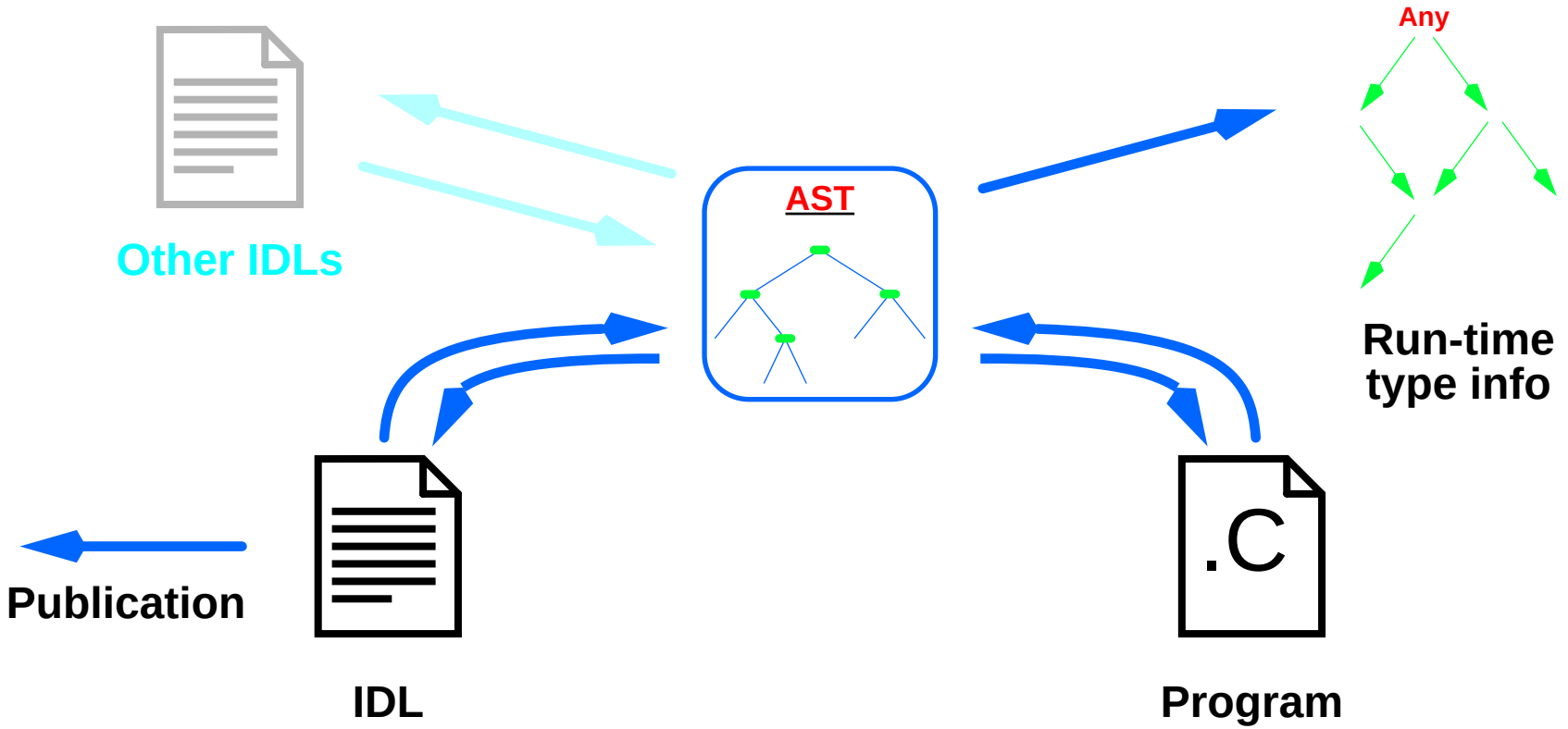
- **Represent abstraction of (anonymous) types at run-time**
  - **Main part of the abstraction is structural substitutability test**
  - **Permits run-time *conformance checks* to determine if provision is compatible with requirement**
- **Type ‘graph’ structure deduced, not asserted**
  - **Permits new types to be inserted anywhere**



Who generates these types?

- **Trading and type checking integrated into the tool set**
  - **Run-time types generated from IDL specifications and/or programs**
  - **Common AST used as intermediate representation**
  - **Inferencer determines client's type requirements from pattern of use**
- **Once the basics are working, move on to type checking factories and collections**
  - **More exploratory**

How ASTs fit in



Current status

- Basic type conformance checker implemented in C++ and thoroughly tested
- Type inference currently being added to Abstract Syntax Tree representation
  - Inference rules documented in APM.1347

The way ahead

- Finish inferencer
- Integration and testing
- Explore type conformance and inference for collections, factories
  - The infamous F-bounded quantification seems the best approach
- Investigate more efficient realisations
  - Initial approach directly implements the programming abstraction
  - Run-time type representations are potentially large, distributed networks of references to services representing types
  - Distributed type conformance checks may be slow
  - Will investigate engineering optimisations like concrete graph representations or naming types in repositories