Poseidon House
Castle Park
Cambridge CB3 0RD
United Kingdom

TELEPHONE: **Cambridge (0223) 323010**
INTERNATIONAL: **+44 223 323010**
FAX: **+44 223 359779**
E-MAIL: **apm@ansa.co.uk**

ANSA Phase III

TINA DPE collaboration - progress report

Andrew Watson

Abstract

These colour slides constitute a work-in-progress report to the ANSA Technical Committee on the collaborative work between ANSA and TINAC on the design and construction of a toolset for augmenting standard programming languages (such as C++) with constructs for building distributed applications that conform to the ODP object model. The tool set is intended to reduce the cost of writing and re-writing distributed programs by providing a programmer-friendly interface to the distribution infrastructure, and hiding platform-specific implementation details.

ANSA's part in this work is concerned with the intermediate Abstract Syntax Tree representation used within the toolset, and with type checking.

The talk is aimed to last around 30 minutes.

APM.1251.00.01 **Draft** 25 April 1995

Request for Comments (confidential to ANSA consortium for 2 years)

Distribution:
Supersedes:
Superseded by:

TINAC DPE collaboration - progress report

June ‘94

Andrew Watson
Nicola Howarth

Activity summary

- **Coding of the following TPP components underway:**

  Abstract Syntax Tree (AST)
  Pretty printer
  Conformance checker
  Type inferencer
  Type-safe run-time binding mechanism

- **Low level of collaboration with TINAC to date**
  - **Due to their recent reorganisation and redirection**
- **Liaison visit to TINAC planned for later in June**

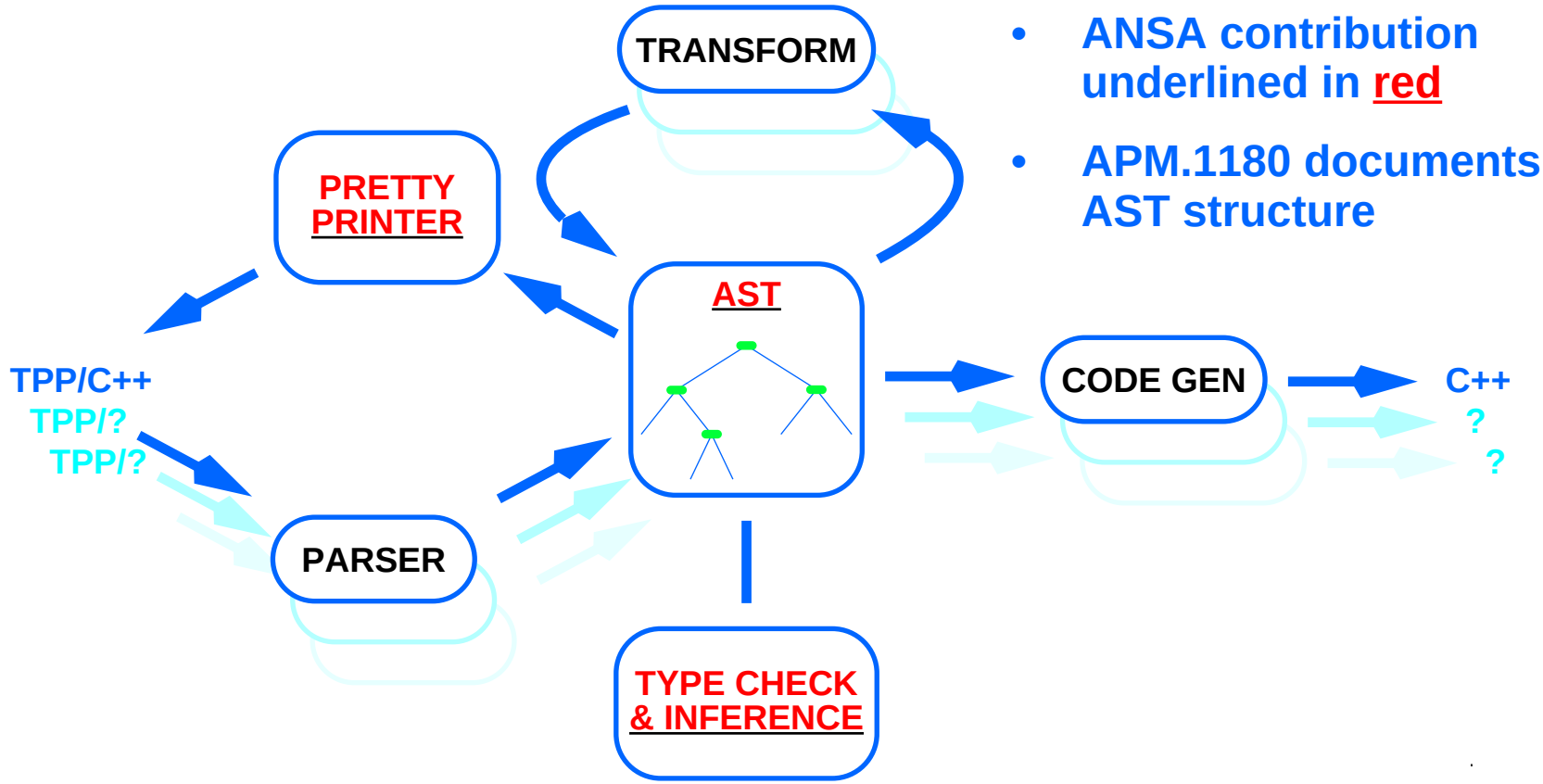
Talk objective

- **Brief TC members on work in progress in this area**
- **Provide background on AST and type checking technology**

The motivation

- **TINAC plan to use a “tools-based” approach to writing cross-platform distributed programs**
- **The alternative (driving infrastructure via procedure calls or class libraries) has shortcomings:**
  - **Incorrect use (wrong order of calls etc) can only be found at run time**
  - **Difficult to apply program transformations to achieve other kinds of transparency**
  - **More difficult to achieve platform-independence**
- **Pooling our manpower helps us both**
- **ANSA has expertise in distribution, type checking & ASTs to share**
- **ANSA gets the benefit of TINAC’s telecoms perspective**

Tool structure



- ANSA contribution underlined in red
- APM.1180 documents AST structure

Where IDLs fit in

- **Interface definitions often separated from code that animates them**
  - **Examples: ANSAware PREPC + IDL, CORBA IDL**
  - **Emphasises encapsulation, separation of interface and implementation**
- **However, TPP specifies interfaces and implementations in a single file**
  - **Distributed Modula 3, Emerald, Commandos do likewise**
- **ANSAware experience: keeping separate interface and implementation files in step is a chore**
- **Strength of a separate IDL is in documentation, communicating between coworkers, inter-language working**
  - **Tools will produce and use TINAC IDL definitions from TPP for these purposes**
- **Conformance-based type checking will ensure interfaces provided and required are compatible**

Type checking - what's planned

- **Rewrite existing first-order conformance checker in C++**
  - Already prototyped in Lisp and DPL
  - Very well understood
  - Main issue is learning C++
- **Adapt experimental type inferencer to the DPE AST**
  - Already prototyped, but AST-specific
- **Add support for type-safe trading via "type wrapper" mechanism**
  - Not previously implemented within ANSA
  - ... but design is independently verified by external papers, so risk is low
- **If time allows, experiment with F-bounded quantification**
  - Provides support for higher-order types
  - A current research problem

Conformance checking

- **Without separate IDL files, ensuring interface provided by server matches that required by client entails a mechanical type check**
  - **Provided interface structure must be substitutable for that required**
- **Several possible substitutability criteria**
  - **Structural equality**
  - **Extension**
  - **Conformance**
- **Conformance provides maximum reconfiguration flexibility**
  - **Server may provide more operations, results may be subtypes, arguments supertypes**
  - **No need to specify type hierarchy, forseeing types of all future versions of a service**
  - **Minimises need to recompile clients to use new service versions**

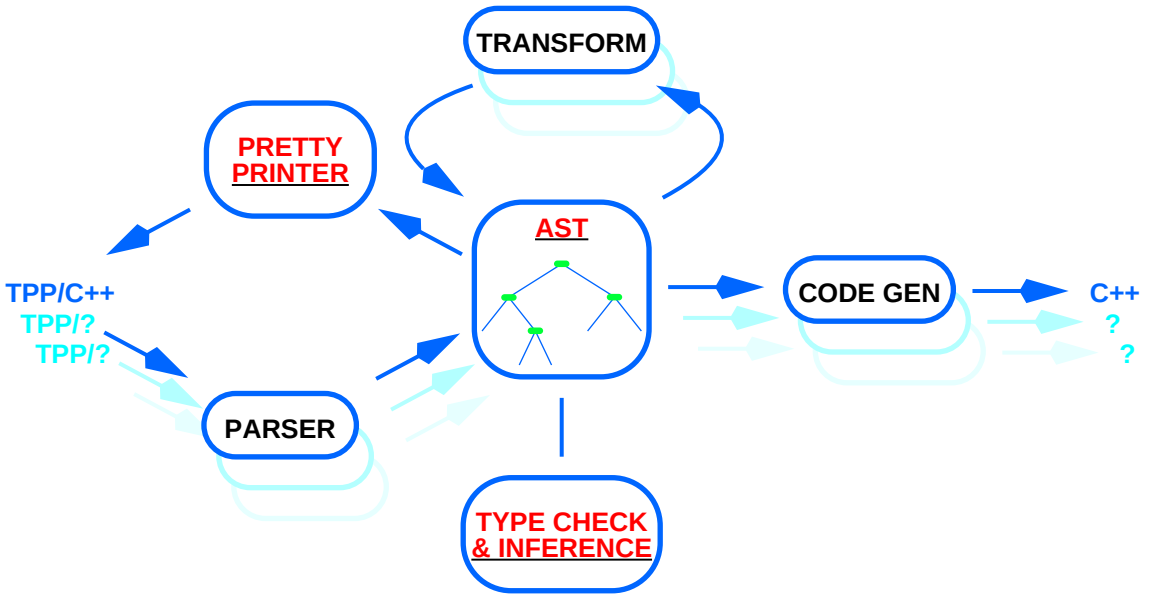
Type inference

- **At least some types are inferred in most programming languages**
  - **Type of anonymous value a+b inferred from types of a and b**
- **Making thorough use of type inference eases the programming burden**
  - **Minimises number pf type (interface) specicifications in the code**
  - **Brings some of the freedom of dynamically-typed languages (e.g. Smalltalk)**
  - **... but without losing safety of compile-time type-checking**
- **Ease-of-use issue (not specifically related to distribution)**

Type-safe trading

- **Traders are heterogeneous collections**
  - Every item may possibly have a different type
- **Programmer writing `import` statement knows type of the instance he wants**
  - But how to be sure his program gets it at run-time?
  - ... and how to type-check the trader itself?
- **Obvious solution: attach run-time type information to all service instances**
  - ... whether needed or not
- **Better: only create run-time type information for traded services**
  - Reduces size of type repository (fewer types needed at run-time)
  - Reduce size of untraded service references
  - May be integrated into TPP mechanism that forces type checking of imports

Summary



- **In all areas except higher-order types our experience permits rapid progress**
  - **However we haven't yet seen the final language design**
  - **TPP designers running down many of the same dead-ends that we did**
  - **AJW to visit TINAC in June to foster links**