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

ANSAwise - Development tools and methods in distributed systems

Rob van der Linden

Abstract

Efficient development of distributed systems requires new tools and changes in the development environment. This module introduces the new challenges and changes.

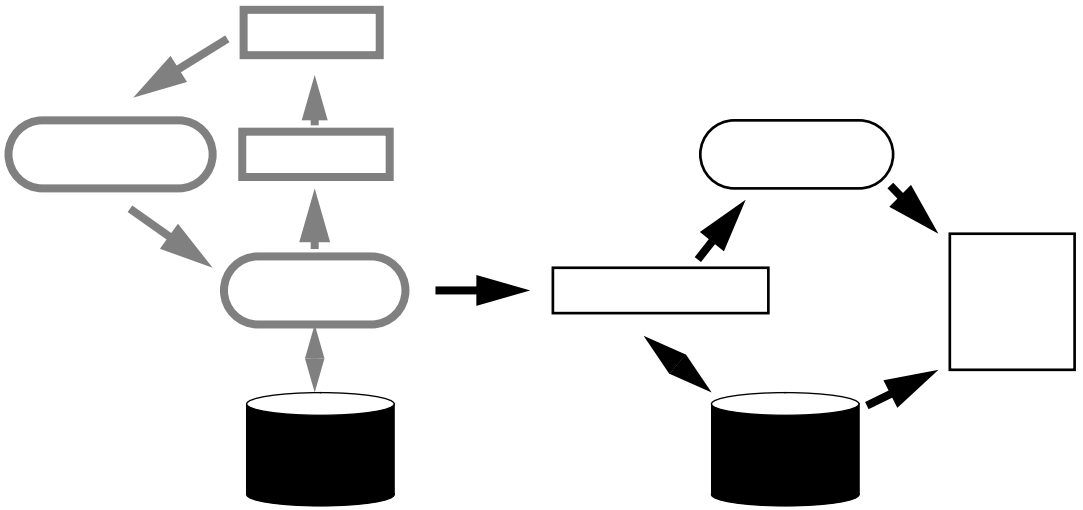
Project management issues are not discussed. It is not the aim of this module to deliver a critique on existing and proposed software development methodologies.

APM.1363.01 **Approved** 25th November 1994

Briefing Note

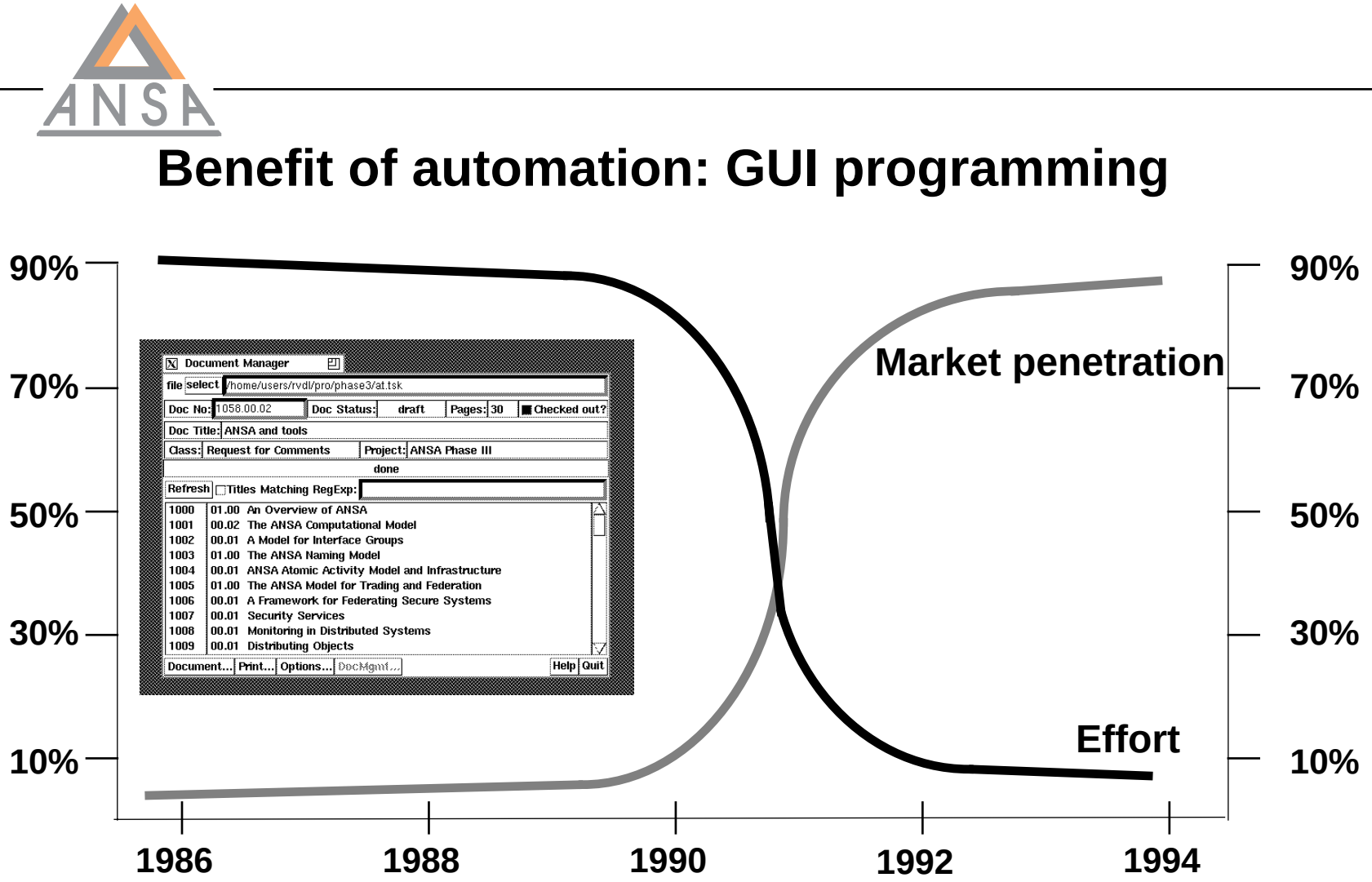
Distribution:
Supersedes:
Superseded by:

Development Tools and Methods in Distributed Systems



In this session

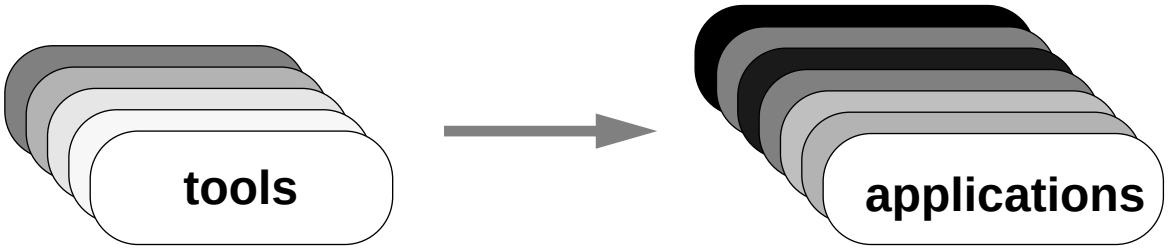
- **Explain why tools are important for distributed systems**
- **Provide help in tool classification**
- **State the characteristics of distributed systems design tools**
- **Assess how existing tools meet the challenge**


ANSA
Benefit of automation: GUI programming
90%
70%
50%
30%
10%
Market penetration
Effort
1986
1988
1990
1992
1994
Document Manager
file select /home/users/rvdl/pro/phase3/at.tsk
Doc No: 1058.00.02
Doc Status: draft
Pages: 30
Checked out?
Doc Title: ANSA and tools
Class: Request for Comments
Project: ANSA Phase III
done
Refresh
Titles Matching RegExp:
1000 01.00 An Overview of ANSA
1001 00.02 The ANSA Computational Model
1002 00.01 A Model for Interface Groups
1003 01.00 The ANSA Naming Model
1004 00.01 ANSA Atomic Activity Model and Infrastructure
1005 01.00 The ANSA Model for Trading and Federation
1006 00.01 A Framework for Federating Secure Systems
1007 00.01 Security Services
1008 00.01 Monitoring in Distributed Systems
1009 00.01 Distributing Objects
Document...
Print...
Options...
DocMgmt...
Help
Quit

Classification

- functionality
- intelligence
- implementation
- methodology

- vertical market
- environment
- object orientation



Tool Functionality

- Cater for part of a system
  - e.g. GUI builders
  - ...database application builders (SQL based tools)
  - ...database internal design (System Architect)
  - ...transaction design (DEC/Forte, ONTOS)
  - ...configuration management (DEC/Forte, SNAP)
  - ...and others...
- Cater for part of the design process
  - programming support (SNiFF+, Visual Basic/C etc.)
  - ...most information modelling tools have code generators

Intelligence

- **Expert system based tools**
  - **e.g. Art*Enterprise, MagicCAP™, Ptech, SNAP**
- **Used to state and interpret business rules and constraints**
  - **enterprise and information modelling support**
- **Used to support the design process itself**
  - **e.g. analysis and design methodologies, expressed as set of rules**
  - **...design can be checked for completeness against these rules**

Implementation

- Design and/or code may be kept in a set of files...
  - e.g. SNAP, SNiFF+
- ...or in a central database
  - e.g. Ptech, Magic CAP™
- Database can be a closed system
  - e.g. internal and conceptual schemata in Tenet are fixed
- The “internal model” is often the heart of a tool set
  - and therefore commercially protected, preventing openness

Methodology

- Some tools enforce the use of a particular methodology
  - e.g. OOA/RD for Shlaer/Mellor
- Some impose methodological constraints...
  - e.g. PTech, ObjecTeam
- ...or very few, if any, constraints
  - e.g. SNAP
- There are well over 20 object oriented or object based design methodologies
  - e.g. see OMG survey of Object Analysis and Design
  - ...many are supported by tools

Vertical Market

- Tools designed specifically for certain types of applications:
  - Statemate for hardware designs
  - ONTOS for training marketing and communications applications
  - OpenBase (Prism) for the petroleum industry (POSC)

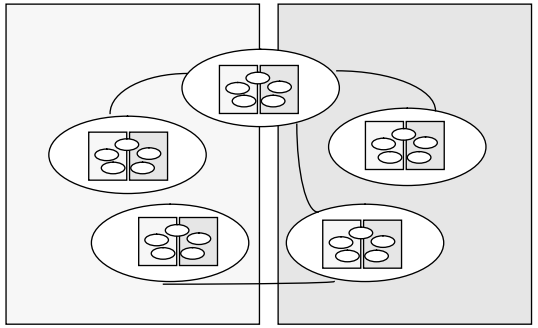
Environment

- Tools specifically for developing
  - UNIX applications
  - PC Windows or X-Windows applications
    (GUI development tools are becoming generic with specific back-ends)

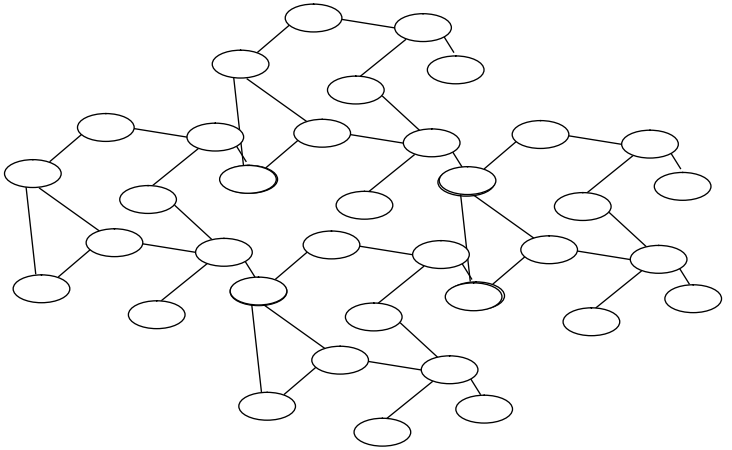
Object Orientation

- Every tool supplier now claims object support
  - sometimes a simple reworking of structured design
  - sometimes language support (e.g. C++, ObjectiveC, SmallTalk)
- Libraries are needed
  - Smalltalk did not get off the ground until a decent library was supplied
- Extensions to relational databases make these object-oriented
- Inheritance may cause problems with distribution

Systems development



Development process



System

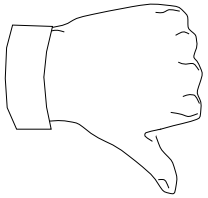
- distributed, large design teams
- hide complexity
- exchange of designs
- exchange of completed components

- scales
- easy to configure
- easy to change & maintain

Desirable system development characteristics

- **Tools**
  - **are distributed**
  - **allow prescriptive / declarative programming styles to support transparencies**
  - **support early type checking for confidence**
- **Systems**
  - **scale to very large sizes**
  - **allow late binding for flexibility**

State of the Art

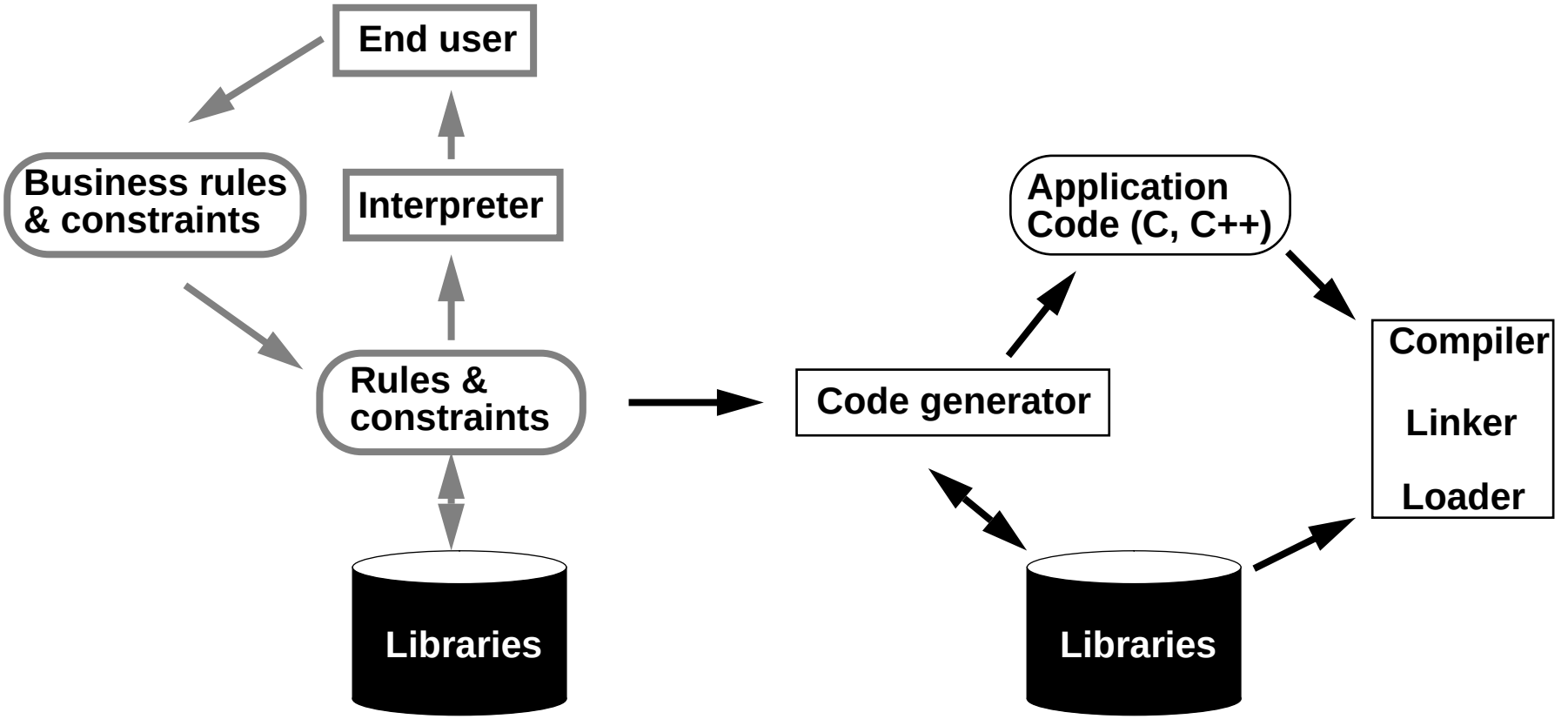


- some complexities transparent
- late binding and early type checking

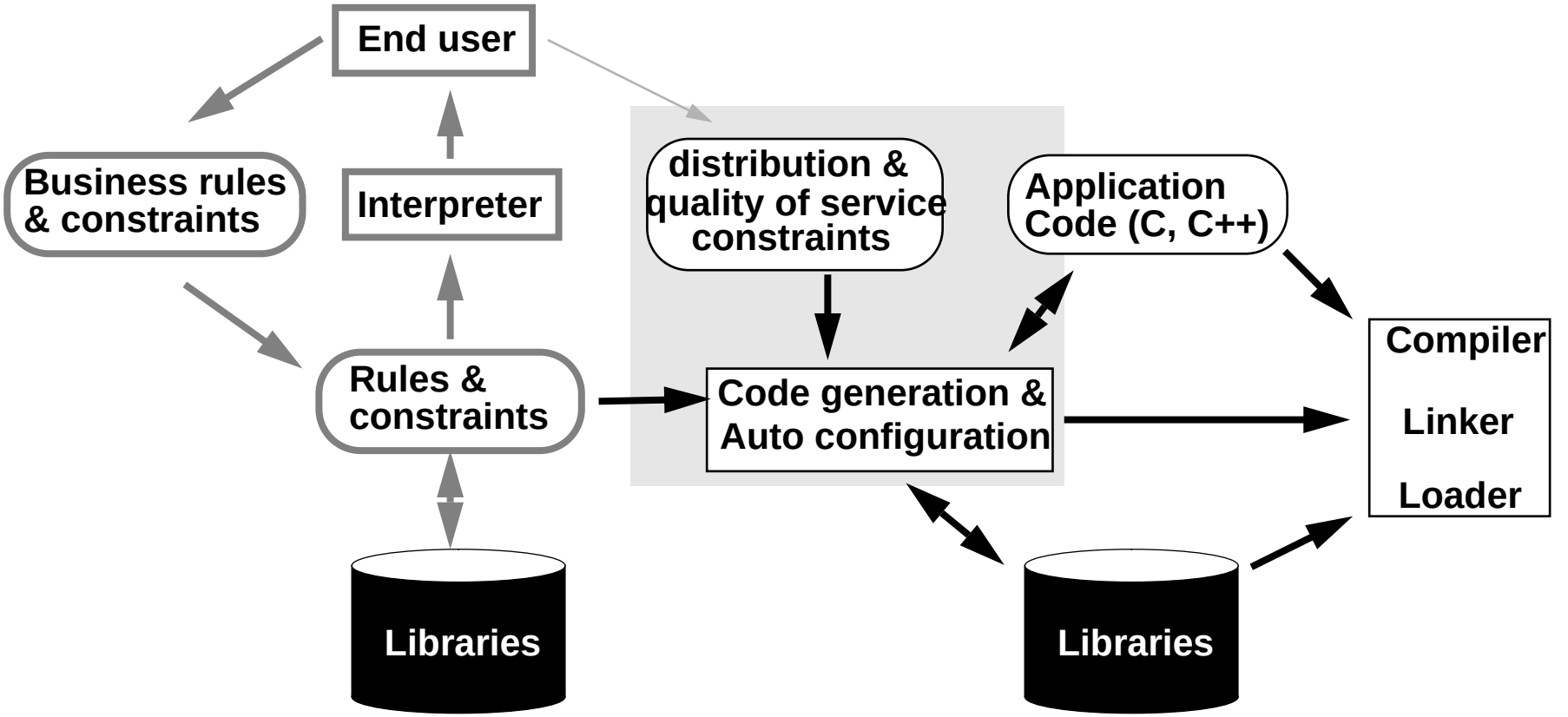
- tools limit size of application
- communications libraries out-dated (DCE support emerging)
- no agreed way to describe interfaces
- no support for application properties
- browsing limited
- tools are (still) closed applications

Challenge

Common tool environment



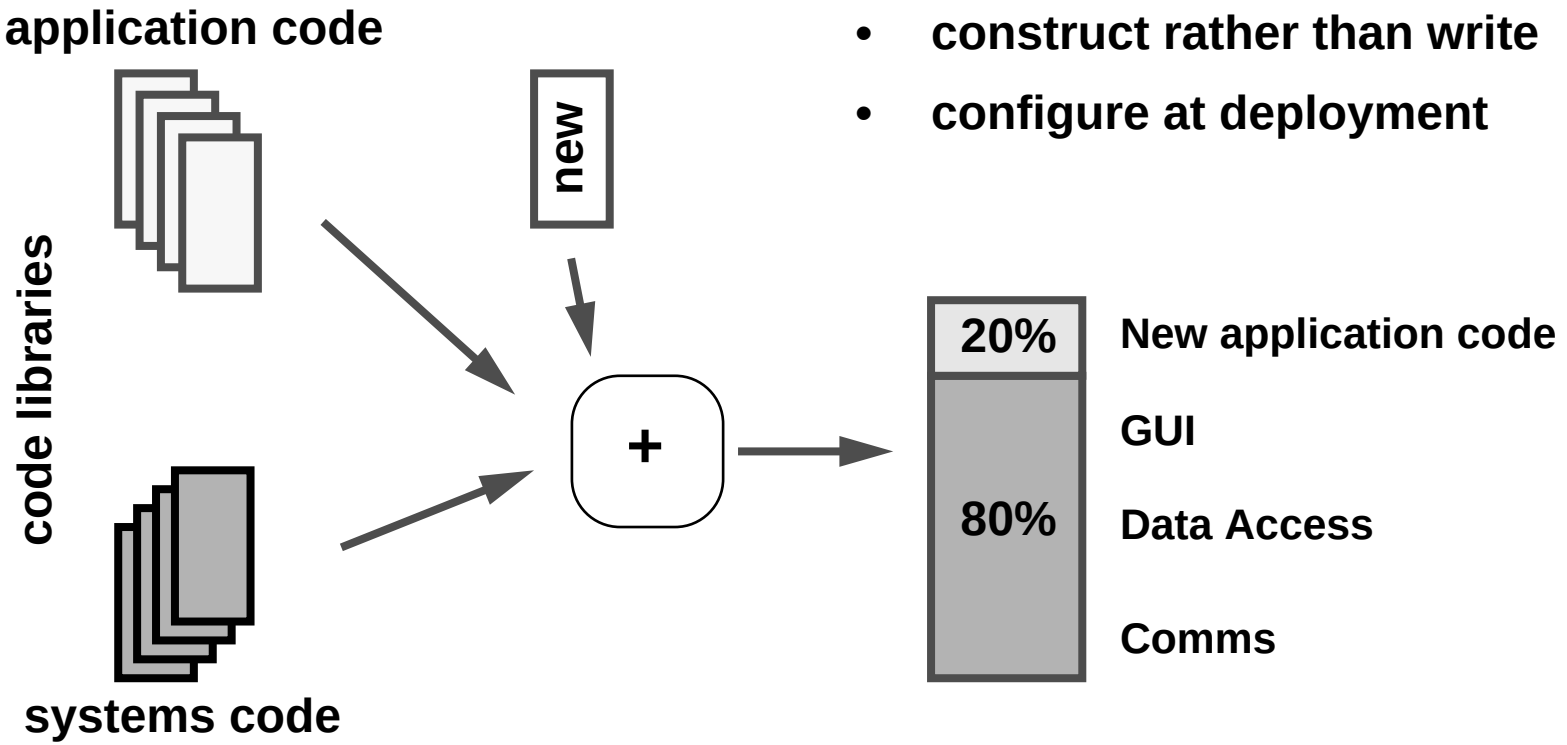
Advanced tool environment



Multiple Library problem

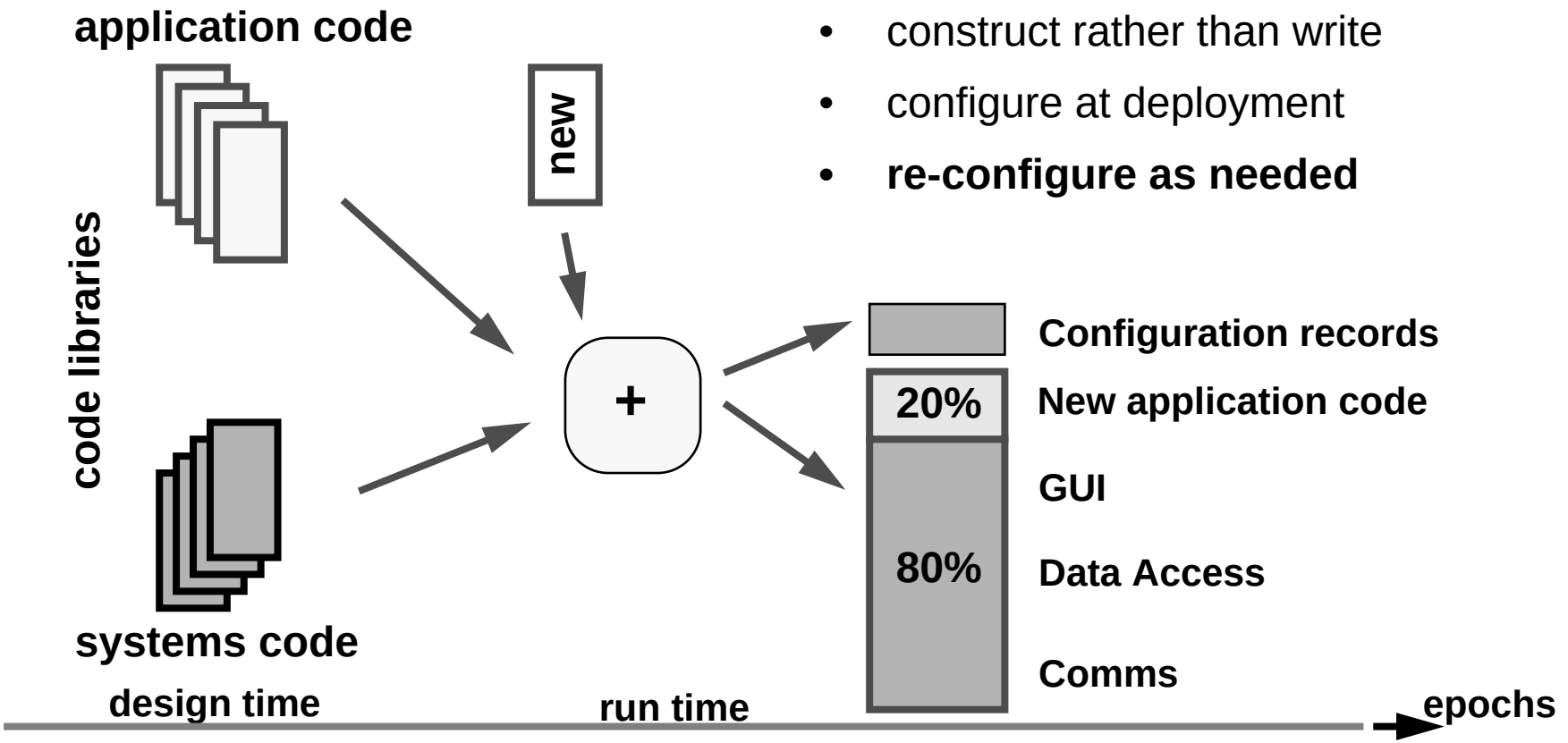
- Tool modularity allows us to buy different libraries and plug them in
  - there is little change control
- Differences in libraries cause portability problems
- The PC world is especially problematic
  - no general standards for add-ins...
  - ... spreadsheets, graphics package filters, custom controls...

Tool structure trend



- construct rather than write
- configure at deployment

Advanced tool structures



- construct rather than write
- configure at deployment
- **re-configure as needed**

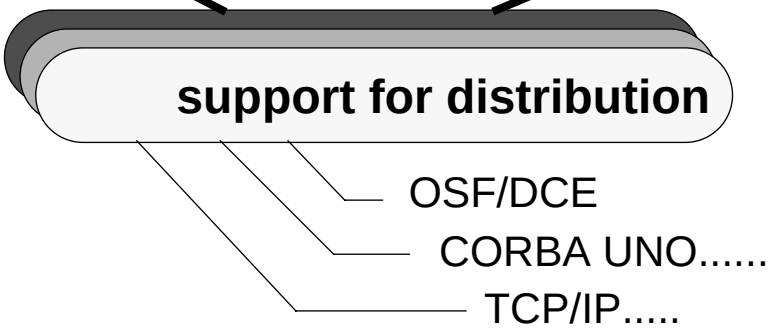
Interoperability issues

distributing the design
& development process

extending the design process to all epochs

tool interworking
- specification exchange
- common libraries
- common class hierarchies

application interworking



Tool interoperability - Control integration

- One tool registers its interest in certain events and gets notified
  - e.g. HP Broadcast Message Server (BMS) and Sun ToolTalk
- CASE Communique (set up by HP in 1991) helped bring over 50 tools together under BMS
- CASE Interoperability Alliance (set up by SUN) centred round ToolTalk and did the same
- ANSI X3H6 now subsumes CASE Communique and CASE Interoperability Alliance

Tool interoperability - Data integration

- Tools agree a format for the storage or exchange of design details
- PCTE: Portable Common Tool Environment
  - grouped round a repository which stores development data only
  - e.g. Emeraude (based on the NIST/ECMA Model for Frameworks of Engineering Environments)
- IRDS: Information Resource Dictionary System
  - centred on a repository which stores development and application data
- EIA/CDIF: CASE Data Interchange Format
  - enables data (e.g. dataflow diagrams) to be exchanged between tools

Distributing the development process - some common problems

- You cannot find out what exists (was built by others)
  - except in the context of a small project or by reading about it
- You cannot find out what will be there tomorrow
  - what are others doing?
- You do not know just what a component does, how to access its functionality, what side effects it can create
  - it is easier to write a new component from scratch
- You feel de-skilled if all you do is bolt other people's stuff together
- Information flows in distributed projects are hard to manage

Support for the distributed development process - the need

- Open distributed repositories must be provided
  - containing complete descriptions of software components (objects)
- There will not be one integrated repository
  - federation is the answer
- There is a need for common access from a variety of existing tools
- The repositories cannot be integrated: federation is the answer

Distributed systems supported by distributed repositories

- **Distributed repositories and traders**
  - allow visibility of specifications of components
  - allow use of software written by others (reuse)
  - need to store complete specifications, not just IDL (as in OMG interface repository)
- **Distributed systems**
  - containing their own descriptions
  - ... to aid maintenance and (re)configuration

No such systems exist today

- **...so put up with the tools provided with your distributed systems platform**

Summary

- **Tools for building distributed systems are still primitive**
  - DCE and CORBA implementations require substantial C/C++ programming effort
  - several vendors are venturing DCE support (e.g. SNAP)
  - tool suppliers still see this as a high risk area
- **Distributed database development tools are available**
  - but tied to a particular database vendor - not open
- **Tools do not support large distributed design enterprises**
  - this is likely to remain that way for some time
- **Tool integration and interoperability is being developed quite separately from open distributed systems technologies**

For more information

- **For a survey of object analysis and design...**
- **...produced by the Object Analysis and Design Special Interest Group of OMG**
  - **see *Object Analysis and Design: Description of Methods* ed. Andrew Hutt (OMG/Wiley)**
  - **see *Object Analysis and Design: Comparison of Methods* ed. Andrew Hutt (OMG/Wiley)**