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

Seminar: Distributed Systems - A Busted Flush?

Andrew Herbert

Abstract

Distributed systems technology has been researched since the mid seventies and only a small frationhas found its way into products. This begs the question how usable the tehnology is, and whether further research in the field is worthwhile.

Distributed Computing is akin to “The Force” in the film “Star Wars”: it has a light side - the ability to build open systems; a dark side - increased complecity for programmers and system managers; it hold the universe together by enabling interoperability between existing applications.

This talk reviews the use of distributed computing in industry and outlines priorities for ongoing research.

APM.1200.01 **Approved** 25th October 1994

External Paper

Distribution:
Supersedes:
Superseded by:

Distributed Systems:

A Busted Flush?

Andrew Herbert

ANSA Chief Architect

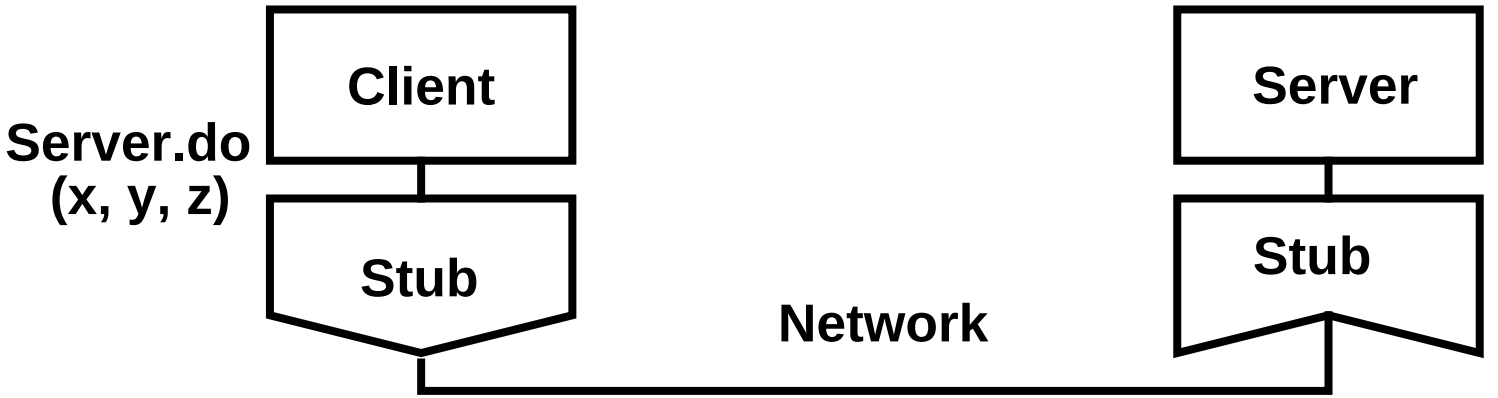
Technical Director
Architecture Projects Management Limited

Some dates

- 1973 CAP Project Minutes - “Beware the Ring”!
- 1978 SIGOPS Conference - LAN-based distributed systems
  - file servers, remote procedure call, transactions
- 1982 Project Universe, Grapevine - WAN-based distributed systems
- 1984 Distributed Objects, interactive multi-media
  - Mayflower, Eden (objects); ISLAND Etherphone (interactive multi-media)
- 1986 Dependability
  - Arjuna, Argus, ISIS, PSync
- 1992 OSF DCE 1.0
- 1993 OMG CORBA implementations

Remote Procedure Call (RPC)

- build protocols out of request-reply interactions
- give programmer a local procedure call interface to the remote end
- hide all the buffering, connection establishment and message passing
- use a stub generator to build an optimized infrastructure



Current RPC Systems

- **Research RPC systems**
  - type safe
  - support for threads and exceptions
  - uncluttered programming interface (by careful choice of language)
  - protocol embedded inside operating system for performance
- **Main products**
  - SUN Open Network Computing RPC
  - OSF Distributed Computing Environment RPC

RPC Problems

- Try to pretend distribution is invisible - optimize for the sequential non-failing case
- Procedure calls in most languages are asymmetric - getting results back is hard
- Exception schemes are clumsy
- Thread management is clumsy
- Binding is clumsy
- Users complain about complexity
  - unless they are used to something worse.....
- Layered above the operating system gives performance problems

Transactions

- **Give computations the ACID property - abstractions to simplify concurrency and failure handling**
  - **atomicity - all or nothing**
  - **consistency - state of data touched by threads within a transaction remains correct with respect to some invariant**
  - **isolation - partial effects of a transaction not visible outside the transaction**
  - **durability - the effects of transactions are not undone by system failures**
- **Research topics**
  - **nested transactions for modularity**
  - **object level granularity for modularity**
  - **semantics-driven locking and recovery for performance (via reduced blocking)**
  - **optimized, non-blocking distributed transaction commit**
  - **transactional programming languages to hide protocols, locking and state versioning**

Current Transaction Systems

- **Proprietary protocols**
  - most flat single level transactions, some nested
  - highly optimized: either for database record access or robust message passing between remote processes (but not both)
  - low level (assembly code) interfaces
  - vendor specific
- **Tuxedo**
  - flat single level transactions
  - programmer responsible for recovery and locking (resource managers)
- **Encina**
  - does flat and nested transactions
  - layered over DCE RPC
  - extended stub generator and generic components (e.g. logs, data volumes) to hide some of the book-keeping

Fault Tolerance

- **Exploit redundancy for faster access and fault tolerance**
- **Research into “broadcast process groups”**
  - **multi-cast communication**
  - **lightweight synchronization (all members see a consistent view, eventually)**
  - **population control (“view management”), including failure detection**
- **Products**
  - **(transparent) hardware fault tolerance for single node (e.g. Stratus)**
  - **embedded software (e.g. System X) - entagled up in the application**
  - **ISIS**

Transaction and Replication Problems

- **overheads too high**
  - **layering above the OS doesn't help, especially for fine-grained objects**
  - **transactional objects are too big for database style, too small for process style**
- **lack of integration with legacy TP**
- **level of skill required to use tool kits is a major problem**
  - **the academic models are too exotic**
  - **users want packaged solutions to standard problems**
- **MANETHO algorithm by Zwanenpoel et al and Workflow / rule based systems show a way ahead**
  - **don't have to be consistent until you expose state to the outside environment**
  - **optimistic activity coordination view c.f. pessimistic object ACID view**

Security

- **Research**
  - cryptography
  - logics for reasoning about security
  - authentication and access control systems
- **Kerberos-based authentication and acess control in OSF DCE**
  - emphasis on confidentiality and access control, whereas commercial need is integrity
  - complex - just watch the mail on the OSF security SIG
  - complex systems don’t get used (properly)

Distributed Objects

- **Concepts**
  - an object pointer can be sent across the network, so objects an be shared and abstractions preserved
  - subtyping (if done properly) allows separate client and server evolution
- **Research (DEC SRC Network Objects, COMANDOS)**
  - single language
  - single developer
  - fixed transparency
  - closed world
- **OMG Distributed Objects Vision**
  - multi-language support
  - repositories for re-use between developers
  - object services

Awkward Questions

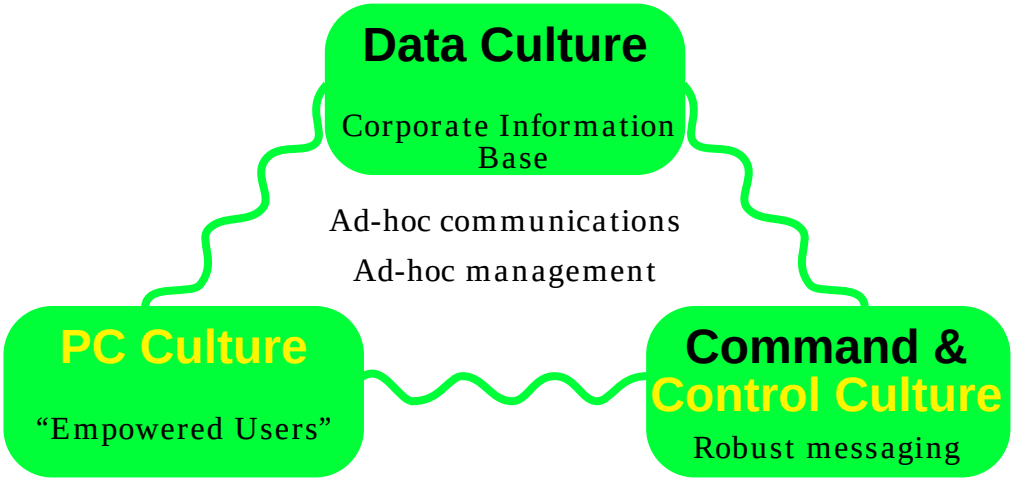
- Do distributed systems research results get out into the field?
- Do those that make it deliver real benefit?
- Why does it take so long?
- Is industry getting ahead of distributed systems research (at least in its goals)?
- Lets have a look at where the technology can be applied.....

The End User's Integration Wish List

- **Integrate products from many vendors - exploit innovation, price/ performance trade-offs**
- **Span application domains - enable information to flow between departments easily**
- **Hide system boundaries - size should only affect performance, not functionality**
- **Protect enterprise boundaries - preserve autonomy and enable flexibility**
- **Enable inter-organisation computing - controlled federation - doing business doesn't require a merger**
- **Preserve existing investments - enable evolutionary change**
- **Match IT style to Enterprise requirements - make the business drive the system, not the other way round**
- **Allow rapid, low-risk adoption of new technology - no-one wants to be first, no-one can risk being last....**

Distributed Systems Today

- Three important and quite separate distributed computing cultures exist

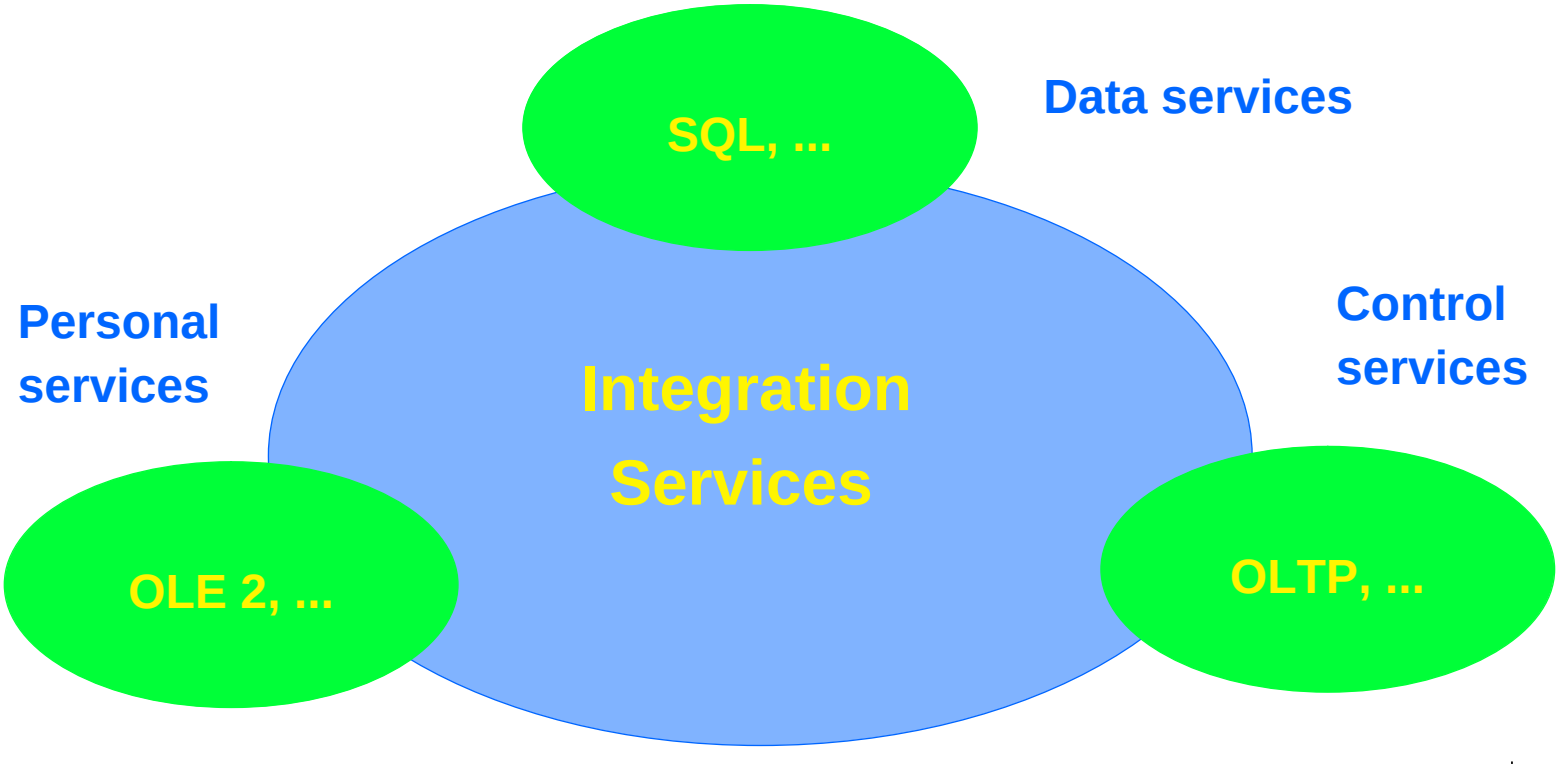


- Each has its own means of distribution and application integration
- They meet different needs - no single one will absorb the others
- Legacy of existing technology choices and applications will persist

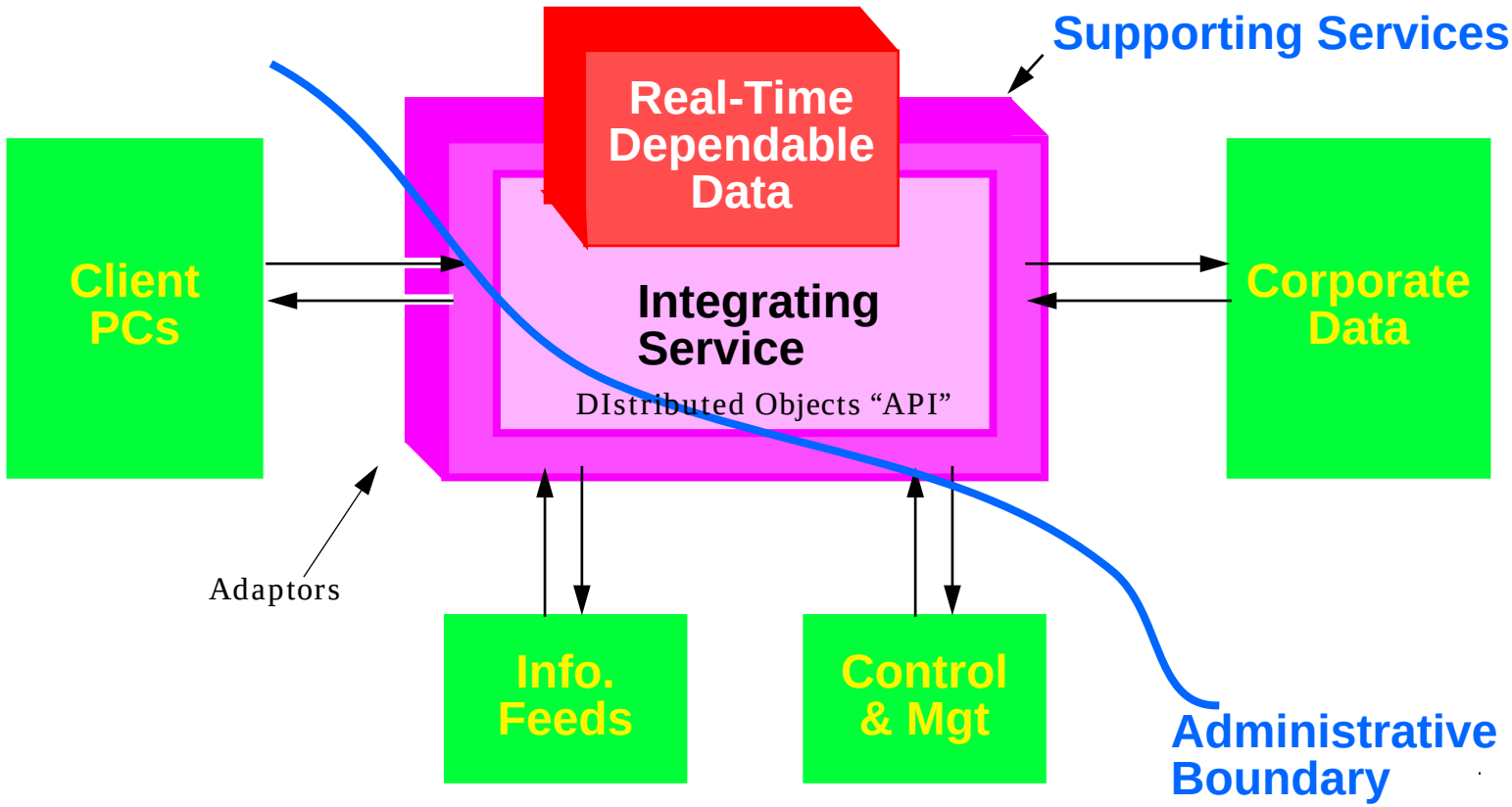
Strengths, Weaknesses

- **Data Culture**
  - Remote data access - mostly proprietary remote login SQL access
  - Federated databases - typically read-only access especially if heterogeneous
  - Stored procedures - for high throughput critical business transactions
  - Object repositories - for more flexible structuring, fine grained locking
- **PC Culture**
  - Single user productivity services (based on OLE 2, CAIRO etc.) - scaling issues
  - File and printer sharing - scaling, security and management issues
  - Group productivity services - very proprietary, not very robust, document based
  - Mobile computers, universal personal digital communication
- **Control Culture**
  - On-line transaction processing - reliable message routing and server activation
  - Strong on throughput, failover and security
  - Systems programmer oriented API
  - Workflow - high level description of transactions and data types - preconceived business models

Distributed Computing = Integration Services = Interoperability and Management



Support for Integration Services - Management Engines



Management Engines

- Management engines provide high level supervisory control
- A Management engine is built using distributed object computing
- A Management engine brings structural integrity to the systems it links
- The ‘standards’ in the other cultures give a narrow interface for integration with distributed object abstractions to hide the complexity

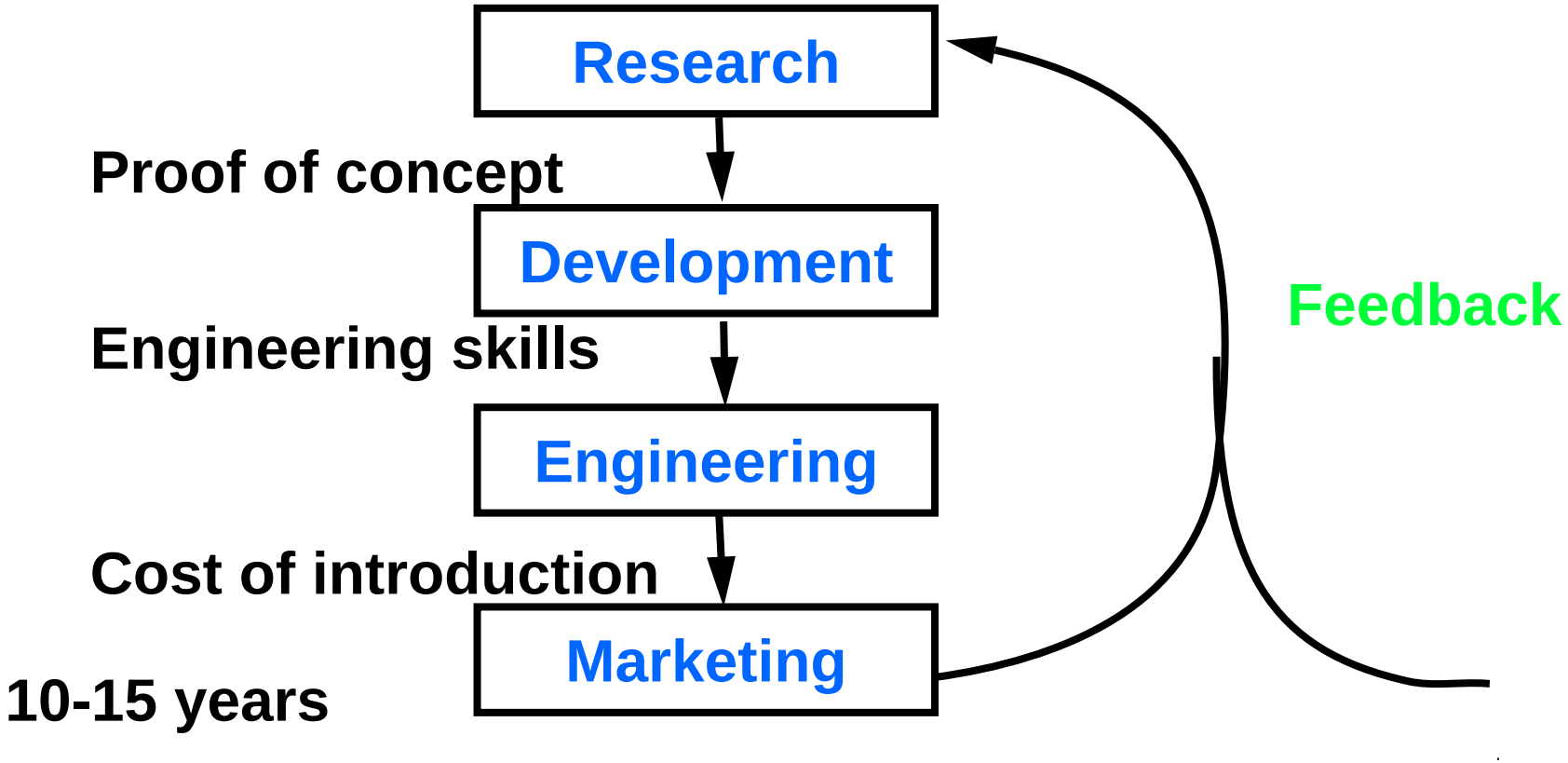
Management Engine Principles

- **Program systems using application concepts**
- **Define a robust high level model for distributed programming**
  - **use objects for encapsulation / separation**
  - **include time and failure as first class citizens**
  - **minimize programmer book-keeping**
- **Use tools to automatically generate the engineering detail**
  - **be as declarative as possible**
  - **generalize the idea of an RPC stub generator**
- **Define structures for integration (including non-native objects)**
  - **cradle to grave object management and monitoring**
  - **as much checking as possible, as early as possible**
  - **decentralized management - objects manage and secure themselves**
  - **selective transparency of engineering detail**

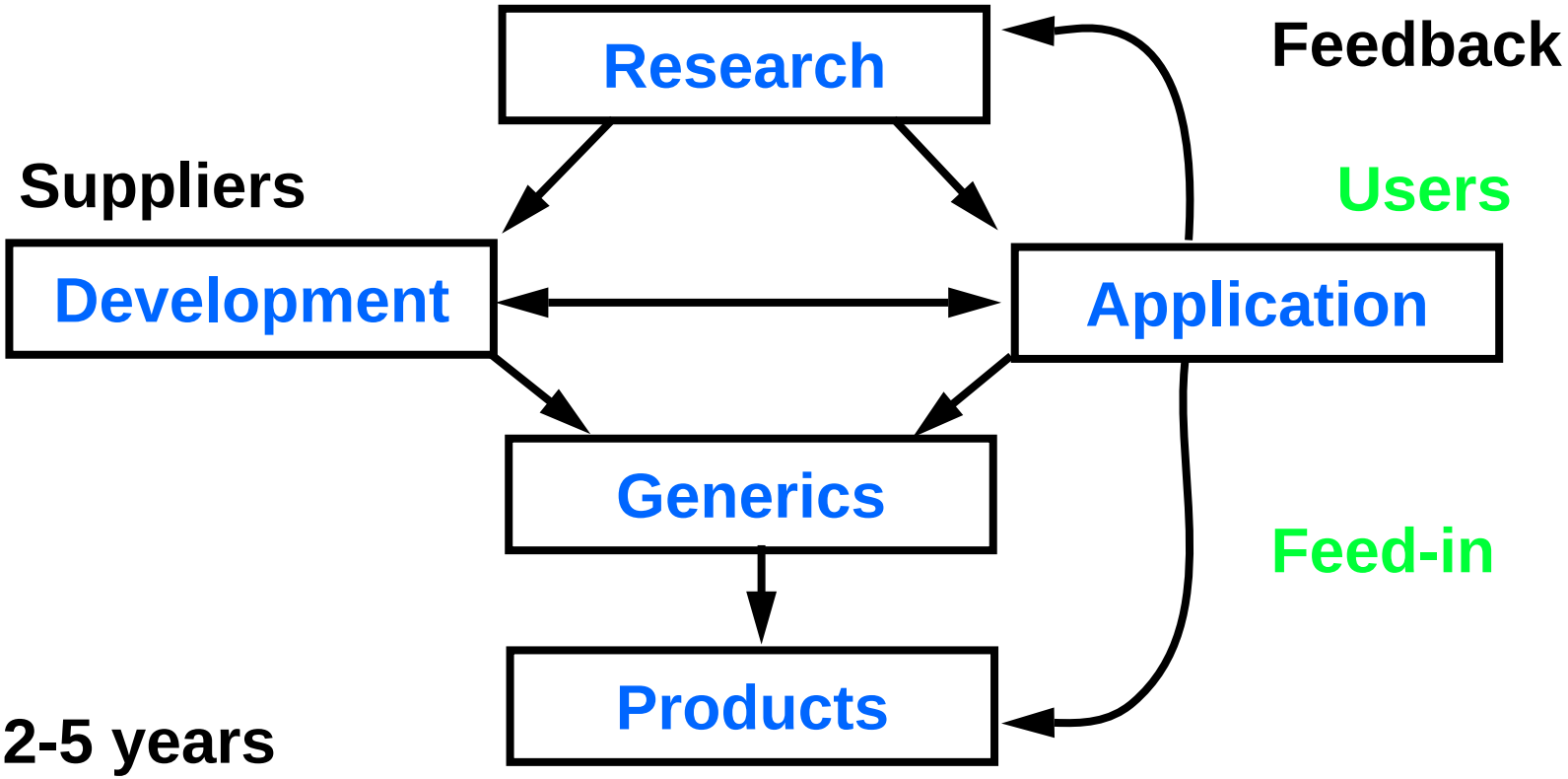
Research: Invention or Innovation?

- **INVENTION is doing something NEW**
- **INNOVATION is doing something NEW and making a SUCCESS out of it**
- ***Technology without application is a waste of effort***

Traditional Pathway For Hard Technology



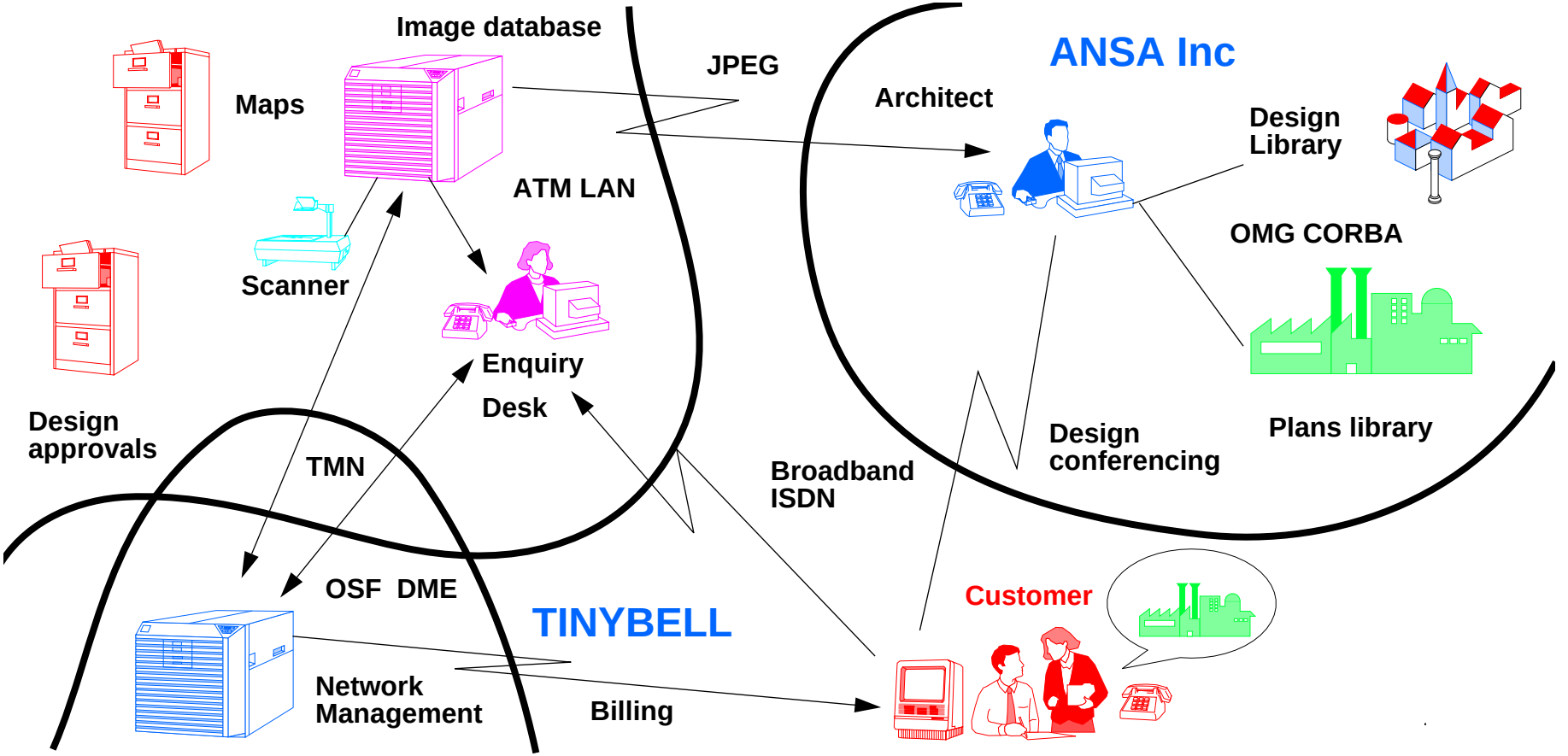
A Better Model for Software Technology



Do We Know How to Use the Technology We've Got?

- Consider the challenge facing the Telecommunications Information Networking Architecture Consortium
- (TINAC = all the phone operators and vendors you can think of)
- (TINA = distributed computing meets the phone system)

TINAVILLE - An Example Scenario



TINA Service Management - The Technical Questions

- How can we analyse business requirements to identify potential services?
- How can we ensure service engineering covers all options and meets all constraints
- How can services be developed rapidly to meet market windows?
- How can existing services interwork with new services?
- How can deployment and management of services be automated?
  - to reduce people costs
  - for faster, accurate response
- How can services be evolved to accomodate new requirements and adapt to new technology?

TINA Service Management - The Business Questions

- How will cooperative services between organizations be set up?
- What are the obligations, liabilities and sanctions (sensitivity, charging, regulation)?
- How will service contracts be negotiated?
- How will feature interactions (policy conflicts) be identified?
- How will the worth of information and information services be assessed (copyright, licensing)?
- How much of this will be supported by service engineering tools?
  - service validation
- How much automation will be available?
  - service generation
- Be manageable - you need to know what's out there and who's using it,....

Some “Facts”

- largest software industry sector is embedded systems (consumer electronics, automation, etc, etc)
- more programs written by users than professional programmers, mostly using packages (spreadsheets, etc etc)
- “applications builder” and “visual programming” tools can automatically generate 80% of the final code
- the Internet World Wide Web is growing by 2000% p.a. - and it is all networking rather than distributed computing