The Landscape of Enterprise Applications
- a personal view

Geoff Sharman

© Copyright geoffrey.sharman@bcs.org.uk
www.dcs.bbk.ac.uk/~geoff/
Typical Web Application?

- Static pages served from web server/content management system
- Dynamic pages assembled by applications on application servers
Did you do any of these today?

- Buy something in a supermarket?
- Buy a ticket for travel or entertainment?
- Make a telephone call (mobile or fixed)?
- Use a cash machine or debit card?
- Pay for something with a credit card?
- Use electricity, gas or water?

The chances are you used a traditional online transaction system (running on a mainframe?)
That's Cloud Power?
What's an Enterprise, or an Enterprise System?
**What is a System?**

**Outputs**
- information
- energy
- material

**Inputs**
- information
- energy
- material

- control function

Internal organisation, possibly including sub-systems

Physical, chemical, biological, social systems - real-time dynamic behaviour
An Enterprise is a System for delivering economic & social outputs

Related pairs of inputs & outputs are often referred to as transactions.
Interesting

What's an Enterprise?

- Large numbers of customers
- Large numbers of transactions
- Large financial throughputs
- Complex behaviour/operations
- Sustainable operation

→ High scale for an extended period
An Enterprise System is the automated part of an enterprise = a real-time model of the enterprise
Where would you find an ES?

- **Probably not here:** (primary industry, 5% of economy)
  - Agriculture, fisheries, forestry, water extraction
  - Mining, oil & gas extraction

- **Possibly here:** (secondary industry, 40% of economy)
  - Construction, utilities
  - Transport, distribution, communications
  - Manufacturing (discrete & continuous)

- **Probably here:** (tertiary industry, 55% of the economy)
  - Financial & business services, media, retail
  - Education, healthcare, tourism, entertainment
  - Public admin
Brief History of Enterprise Systems
Pioneers of Enterprise Systems

• 1952 – Joe Lyons & Co. Leo system
  - batch accounting & payroll operations

• 1965 – American Airlines Sabre system
  - online flight reservations & check in

• 1993 – Amazon.com
  - direct customer service, just-in-time delivery

• 2001 – Google.com
  - customised search using large amounts of data

• 2008 – Apple iPhone
  - mobile applications
Getting Closer to the Customer

System Type
1=Batch, 2=Online, 3=Network, 4=Web, 5=Social/mobile
What were the Key Innovations?

- 1950s – main storage (delay lines, ferrite cores), secondary storage (magnetic tape), batch job scheduler
- 1960s – operating system, direct access storage, database management, time sharing terminals
- 1970s – TP monitor, relational database
- 1980s – personal computer, networking
- 1990s – World Wide Web
- 2000s – search engine, social networking, mobile
Programming Paradigms
Time Sharing/Conversational

- At logon time, operating system allocates:
  - Memory address space for application
  - Operating system process
  - Files, communications channels, etc.
  - These remain dedicated to user until logoff

- Paradigm is widely used, but:
  - No sharing of resources
  - Not scalable beyond few hundred users
TP Monitor PseudoConversations

- TP Monitor acquires & retains shared resources
  - Applications, memory, processes, threads, files, databases, communication channels, etc.
  - On receipt of user transaction request, provides *concurrent* access to resources for application
  - Frees resources as soon as output message sent

- Highly scalable to 10s of thousands of users
  - Requires stateless application programming
  - Conversation state held in “scratchpad” files
Representational State Transfer

• Underlying paradigm for Web hypertext transfers
  • Commonly abbreviated as REST
  • Web servers manage network & provide **concurrent** access
  • Defines stateless clients for rendering data
  • Highly scalable to 10s of thousands of users

• Does not define how to build update applications on the Web
  • Disallows “cookies” - no scratchpad
  • Does not define server application model
Google Applications

• Underlying paradigm for Google search and other applications
  • Uses GAE (Google App engine)
  • Requires stateless clients
  • Concurrent access to “scratchpad” storage via GFS/BigTable
  • Highly scalable to 10s of thousands of users

• Especially suitable for applications using read-only data, e.g. search data, maps, etc.
Why do these Paradigms Work?

• All these paradigms embody the many-to-one relationship between customers and the enterprise

• The TP, Restful, & Google paradigms provide scalable concurrency & enable the enterprise to exploit economies of scale

• None of them is a complete description of what modern enterprise systems need
What Paradigm is Needed?

• Stateless applications provide the highest scalability and work well for read-only requests

• But commercial applications, e.g. web shopping, need conversation state & concurrent update

  • Use HTTP because it supports any-client-to-any server, unlike object-based protocols
  • Hold state on client or replicated server file system
  • Collect updates that form part of a transaction
  • Permanently save data at end of conversation
Current Enterprise System Challenges
Enterprise Business Challenges

Enterprise business people care about two primary objectives:

• Reducing costs:
  • automating/eliminating internal processes
  • reducing operating costs for enterprise systems
  • reducing ownership costs for enterprise systems

• Increasing revenue:
  • Winning new customers
  • Retaining existing customers
  • Getting more business from existing customers
Enterprise System Challenges

1) **Multi-channel applications**
   - acting consistently to the customer

2) **Multi-business service**
   - providing multiple offers consistently

3) **Effective customer knowledge**
   - acting more intelligently to the customer

4) **Effective market knowledge**
   - foreseeing what customers will want next
Multi-Channel Applications

- Many enterprise systems are designed to support particular sales channels, eg.:
  - Store checkout systems
  - Kiosk/ticketing systems
  - Call centre systems
  - Web based systems
  - Mobile systems

- Business offer may depend on channel, but

- Applications should *treat the customer consistently*, whichever channel he/she uses
Typical $M^2$ Architecture

- **Presentation / Channel server(s)**
  - Static/ Dynamic web pages
  - Portals
  - Channel specific

- **Integration server**
  - Tight coupling
  - Loose coupling
  - Stand-in processing
  - Flow ctrl/compensation

- **Line of business Application and Data server(s)**

$M^2 = $ Multi-Channel, Multi-Business
Customer Knowledge

• Many web systems allow customer to explore options before & after a transaction:
  • high “browse to buy” ratio in web shopping
  • evaluations of product, service, etc.

• If we identify the customer, we can study:
  • search patterns
  • history of actual transactions
  • customer likes & dislikes

• May enable better offers to the customer
  • need more data & may require real time parallel computation
Market Knowledge

• Many enterprise systems collect data about a mass of customer transactions:
  • Collected/refined in data warehouse
  • Linked with tools for analytics / Bus Intelligence
  • Used to produce periodic reports & analyses

• This process may be ineffective:
  • Too slow/costly for business needs
  • Only structured data – much unstructured data
  • New methods use very large data sets

• Best practice uses *highly parallel* processing
“Highly Parallel” Processing

• Google is the best known exponent
  • Many processes crawling the Web in parallel
  • Combine results using MapReduce technique
  • Store results in Google File System
  • Effectively substitutes concurrency for parallelism

• Also widely used in scientific applications:
  • e.g. SETI @Home used subscriber PCs

• IBM “Blue Gene” protein modelling project
  • 4K processors generated 10 μsec simulation
  • Uses hardware cluster plus GPRS
Summary

• When building an Enterprise System, we are building a model of (part of) the enterprise:
  • Model must be real time and scalable
  • Customer can use anywhere, anytime, any device
  • Access any business offering consistently
  • Know and respond *intelligently* to each customer

• Meta-Enterprise System should analyse system & aggregate behaviour of customers
  • Detect trends and respond to them