Information Systems and Databases

Richard Cooper

Part 1 Information Systems Using Spreadsheets and the Internet

L1 Introduction

- Systems for managing and communicating information as documents

L2-3 Spreadsheets

- Systems for managing and communicating information in documents with simple kinds of calculation
- Spreadsheets for data analysis and accounts
- Numerical errors in computation

L4-10 Hypermedia Systems

- L4 Multimedia
- L5 The Internet and the World Wide Web
- L6 Introduction to XML
- L7-8 XHTML
- L9 Style and Design on the Web
- L10 Advanced Web Topics

Part 2 Database Systems – details in Lecture 11

Course Information

Lectures

- Tuesdays and Thursdays, weeks 2-13, Maths 326
- Wednesdays, weeks 2-6, Maths 325

Tutorials

- Thursdays 11-12 , weeks 2-13, Boyd Orr C

Labs

- Tuesdays 2-4, weeks 2-13

Text Books

- *Database Systems*, Fourth Edition, Connolly and Begg, Addison Wesley, ISBN 0-321-21025
- *Foundations of Database Systems*. Fifth Edition, Elmasri & Navathe, Addison-Wesley, ISBN 0-321-41506-x
- *Web Design*, Chapman & Chapman, Wiley, ISBN 0-470-06089-1

Web Site

- http://www.dcs.gla.ac.uk/~rich/ISD09
- Reachable from Moodle

Coursework

In the first few weeks, there will be some unassessed pieces of work to hand-in:

- A simple Word document describing yourself
- A spreadsheet for calculating your qualification
- A simple web site
- A small database modification in Microsoft Access

Your main piece of work will be to build a database in Oracle, including:

- a conceptual design (ER)
- a set of tables
- a set of data
- a set of queries
- indexes, privileges, etc
- and a simple Java application built on this (a minor edit of a template program)

Lecture Outline

The Development of Computer Applications

- Managing Numbers
  - spreadsheets and databases
- Managing Text
  - word processing, etc.
  - style and formatting
  - integrating documents
- Hypermedia
  - multimedia
  - the web and distributing information
- Distributed Applications
  - distributing processing
  - ubiquitous computing
  - grid computing

ISD and the MScIT Programme

The practice of Computing Science and IT is about developing and using languages which

i. describe the **processes** we use and
ii. describe **information structures**
iii. in ways which a **computer can understand and use**

Programming achieves the first goal

ISD is mostly about the second goal

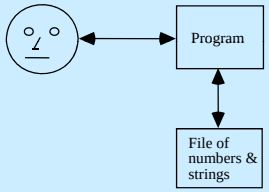
- describing documents representing collections of information in structured ways

Examples

- word processed documents, spreadsheets and databases

Numerical Documents

The initial use of computers was to store and process data which was essentially numerical - perhaps extended with simple textual annotation



Repeated creation of such programs has led to a number of **abstractions** - identifying the essential nature of certain classes of application/document

This in turn has led to new kinds of program which are **generic** - not intended to be used solely for one application

Abstracting Numerical Documents which Require Simple Kinds of Calculation

Very often the numerical program would require the management of a number of values which are inter-connected by some simple formulæ

Accounts are like this for instance

- You enter a number of monetary values and then have them summed in various ways

	A	B	C	D	E
1	**Richard's Account**				
2		**Expenditure**	£1,128.85	**Income =**	£1,381.48
3				**Balance =**	£252.63
4		**Outgoing Summary**		**Incoming Summary**	
5		Food	£468.85	Pay	£1,003.48
6		Rent	£660.00	Expenses	£378.00
7					

Sheet1 Sheet2 Sheet3

This has led to the generation of **spreadsheet programs** - which are "direct manipulation" programs for inputting inter-related formulæ and getting immediate feedback on their values

- The monetary values are stored in cells labelled A1, A2, B1, etc.
- The total is calculated by a function which adds up the values in identified cells
- The cells are displayed graphically in a rectangular grid and are easily edited

Abstracting Numerical Documents which Must Manage Large Amounts of Data

Another kind of application consists of those which rely on access to large amounts of data

This brings up issues of ensuring fast access, providing access to multiple users, reliability, security, etc.

Database Management Systems are designed to tackle these issues and they describe the data in three ways:

- They are stored in files whose structure may be obscure to humans but are fast to access
- They are described at intermediate level in a structure which is the same for all applications – most commonly as tables
- They are described to the human user in ways which make sense in the context of the tasks to be achieved
  - e.g. using a graphical interface

Textual Documents

The next class of application deals with documents which consist wholly of **text**

These bring up issues of **layout** and **formatting** and the need to provide good editing facilities

The management of textual document also requires facilities to impose **structure** on the document

- identifying the meaning of parts of the text – e.g. headings, etc.
- inter-relating parts of the text, creating one document out of parts and so on

Notice the switch between the use of computers as **powerful calculators** to machines which **facilitate communication**

Practical computing is now often called Information and Communication Technology (ICT)

Kinds of Textual Document

At least **five** main kinds of text processing program have appeared:

Text Editors - these concern themselves largely with the content rather than formatting - e.g. *Notepad*, *emacs*

Word Processors - these provide editing and formatting of single linear documents. There are two types that you typically find:

WYSIWYG Word Processors - What You See Is What You Get - as the editing takes place the formatting takes effect on the screen (and formatting is often achieved by graphical means) - *AppleWorks*, *Word*, *WordPerfect*, etc.

Text Formatters - the editing uses a Text Editor and the formatting commands are embedded directly by typing them in and the effect is only seen through a viewer or when the document is printed - *LaTex*, *Troff* and *HTML*

Page Makeup Programs - These allow you to manage a number of documents to create newspaper-like layout - *PageMaker* and *Ready-Set-Go*

Slide Preparation Programs - These provide mechanisms for creating sets of slides – with backgrounds, titles, etc. - *PowerPoint*

Styles: Structure and Formatting

Text programs typically allow you to affect two aspects of the text:

- **structure** – headings, chapters, paragraphs and so on
- **formatting** – emboldening, underlining, margins, etc.

These two easily become confused, e.g. HTML and Microsoft Word Styles.

- both have styles which names which imply structure
  e.g. <h1>, <h2>, ... in HTML and *Heading1*, *Heading2*, ... in Word describe different levels of heading
- XML and XHTML separate these – which is much, much better

A **style** is a collection of formatting choices which can be applied to a paragraph

- e.g. on this slide – *Title*, *Normal*, *IndentHalf*, *IndentHalf* and *Normal* are the first five lines
- you can change which style is associated with a paragraph
- you can alter a style, thus changing the look of every paragraph associated with that style

Styles: Why They are Important

It is almost always best to create a set of styles for a document:

- this can control the indentation, spacing and so on for a number of paragraphs spread through the document to **make your work look consistent and thus professional**
- you can edit the format of **a whole set of paragraphs in one go**
- you can **share the styles** amongst a set of authors
  e.g. Springer-Verlag styles for conference proceedings
- a good Word Processor will also give you a number of **facilities which exploit the existence of the structural styles**
  - e.g. making a table of contents out of the heading styles

Never do alignment through the use of multiple spaces, tabs or blank lines, since this is not easy to maintain -

- note the usability issue of **consistency** and PSD issue of **maintainability** crop up already!

Graphical Documents

The next addition to documents were figures and diagrams

With the advent of better quality screens and printers, it became possible to create programs which allow the user to manage graphical figures

There are two main kinds:

- **Diagrams or Drawings** - these consist of a collection of shapes and text strings which together make up the overall document - the document is remembered as shapes and text;
- **Pictures or Images** - these consist of a description of the pixels which make up the picture - the meaning and editability of the text is largely lost

AppleWorks and Microsoft Office have components for both of these.

Other Media

There is now widespread use of sound or video

- Most computers come with CD and DVD drives and driver programs, e.g.
  - *RealAudio* is a program which retrieves and plays sound documents
  - *RealPlayer* is a program which retrieves and plays video documents

Multimedia is the term used to describe data which consists of numerical, textual, graphical, aural and video data – also 3D for virtual reality. There are two classes of file format:

- ones which model the **structure of the data** (e.g. drawings); and
- ones which merely record the **perceptual information** (e.g. images)

Two problems for multimedia:

- the computer must describe the data with sufficient accuracy that the sense information can be perceived with sufficient clarity
- the data must not swamp disk space and network traffic
  - **compression systems** are created to help with this problem

Local Integration

Maintaining separate numerical, textual and graphical data is useful, but documents produced for communication could usefully have parts which correspond to each of these

Increasingly, therefore, software is available which integrates these:

- The **Clipboard** (introduced on the Macintosh) is a generic facility which allows any kind of data to be extracted from a document, held in a buffer and then inserted into another document - e.g. pictures in text documents
- **Integrated packages**, such as *AppleWorks*, provide separate components for each kind of document - but then allow you to embed parts from one inside another
- *Office* is Microsoft's packaging together of previously separate programs - *Word*, *PowerPoint*, *Excel*, *Access* and so on

OLE and OpenDoc

Office depends on a model for documents which describes how they can be inter-related

Object Linking and Embedding is the name of the model and describes two kinds of links between documents. For instance, if Document A (e.g. a text document) must have Document B (e.g. an image) contained as part of it:

- **Linking** makes Document A refer to Document B in such a way that changes to B are reflected in A
- **Embedding** takes a copy of Document B and puts inside Document B so that, from then on, it is part of A

OpenDoc is Apple/IBM's version of OLE – but is more extensive

Copying External Objects into a Word Document

There are several facilities in Word which seem to do the same thing, but vary in:

i) whether the external object is turned into a Word object (e.g. spreadsheets become tables)

ii) whether the object is linked or embedded

Some examples of taking a spreadsheet and embedding it into a text document:

The *Object* drop-down menu in the *Insert* tab *Create New* creates a blank table which when selected becomes a spreadsheet editable within Word

Create from File in the same drop-down menu creates a table with data from the spreadsheet which also becomes a spreadsheet editable within Word

Edit::Copy in Excel followed by *Edit::Paste* in Word creates a table which is never turned into a spreadsheet

Edit::Copy in Excel followed by *PasteSpecial* in the Home tab also creates a table which can be edited, but this is a link and the editing returns you to Excel and the original is also edited

Plug-ins

Another way of integrating various kinds of function is to allow one program to accommodate others as components

These are called **plug-ins** (or by Microsoft) **add-ins**

- For instance, a word processor might include a drawing component or a spreadsheet component

Plug-ins are accommodated in a particular slot in the application and make the application extensible

Examples:

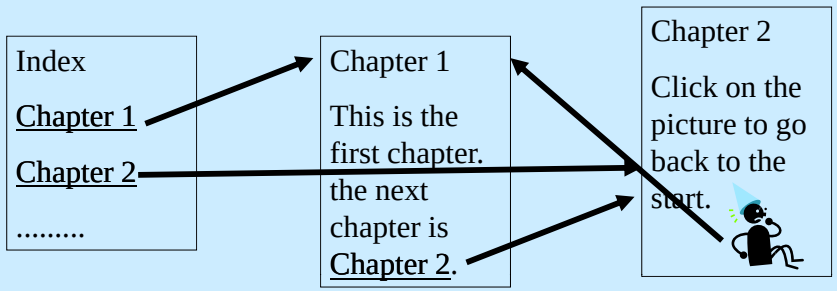
- In Word, you can create an object – i.e. a component of the document which is managed by a distinct component of Word. This may be a plug-in
- In Internet Explorer and other browsers, plug-ins allow different document types to be viewed – e.g. pdf files
- Eclipse is a program specifically designed to accommodate plugins (it has a tiny core) and there are a huge number of them, e.g. the Java editor you use

Hypermedia

Another extension of the basic documents found in Word Processors is the extension from linear forms to non-linear forms

Hypertext is the name for *textual* documents which have links between parts so that they can be read in any order

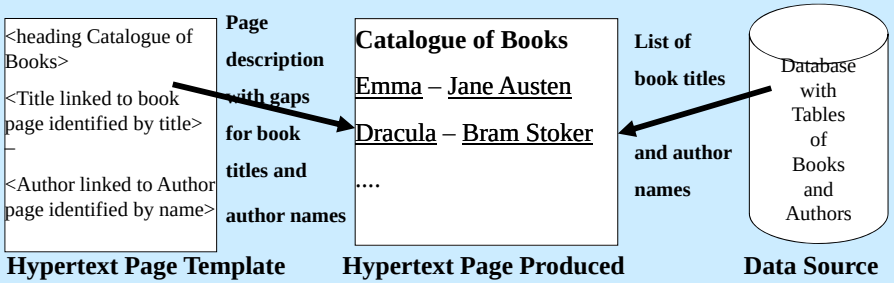
Hypermedia is the name for *multimedia* documents which have links between parts so that they can be read in any order



The World Wide Web

The web is a hypermedia document which is broken up into pages which can be viewed by following links from other pages

But increasingly this involves **dynamic pages** – ones which involve programming to create the page content



The Web use started as a mechanism for information dissemination, became a major means of commerce and is now the basis of a rich variety of social networking

Web 2.0

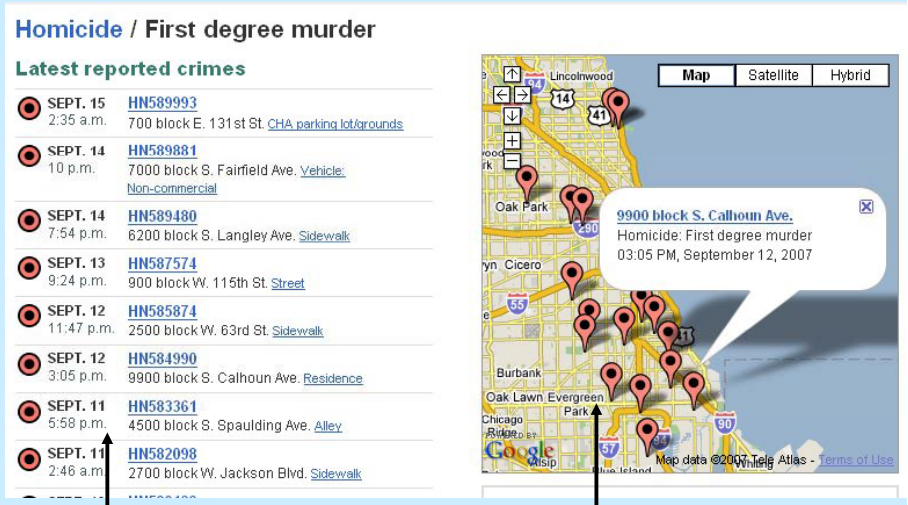
Increasingly web sites resemble desktop applications

- You can manage a set of data
- You have sophisticated interaction facilities

Web 2.0 is the term used to encompass

- Collaborative and social web sites
  - Facebook, Flickr, web logs, etc.
- built using advanced technologies such as
  - Flash for animation
  - Ajax – refreshing part of a page
  - Mashups – putting together several techniques to make one page
- and communicating through simple protocols

Example of a Mashup



Database Retrieval of Data

Google Maps with locations superimposed

Accessibility

There is an increasing need to make applications available to people with different backgrounds and abilities

- e.g. providing screen readers or magnified interfaces
- this is increasingly a legal requirement

A key concept here is that the content to be sent to a user should be well formatted

- which makes it easier to deliver in different ways
- e.g. an image can be described rather than shown if it is put into a structure which also includes a textual description

There are guidelines to make this easier to achieve

Distributing Processing I

So far we have concentrated on data – what about programs?

Initially, a program ran on one computer and was disconnected from any other

- This is called a **standalone application** and is still the way you probably use a word processor or spreadsheet

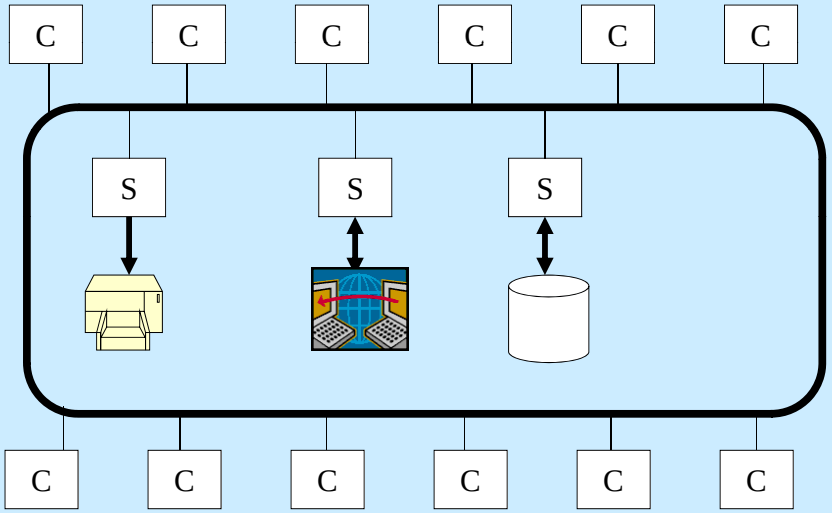
Then it became clear that it was better to connect computers to share their computing power

The first way of doing this was to have one computer (**server**) look after a resource, such a printer, and have the others that people are using (**clients**) connect to it to print

- This is called a **client-server architecture**
- Large database systems run like this
  - e.g. bank databases providing cash points as clients

Peer-to-peer systems blur the distinction between clients and servers

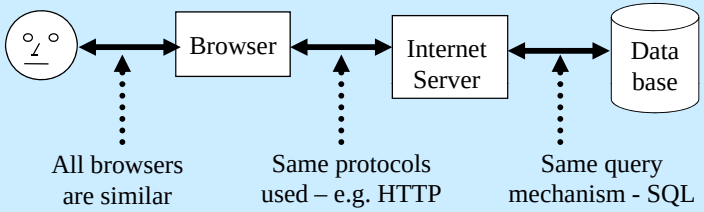
Client Server Architecture



Distributing Processing II

Then further splitting of computation created extra layers

- e.g. a web browser connects to an internet server which may connect to a database server
- these are called 3-tier or **n-tier architectures**



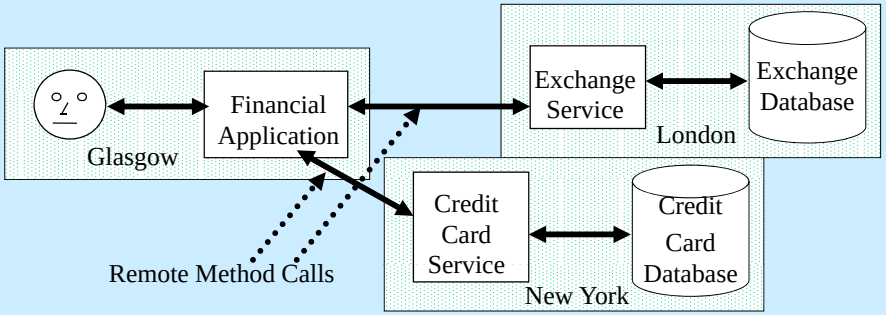
These architectures depend on a fixed configuration of components

- e.g. browsers, internet servers and database servers all look the same, and they connect the same way

Distributed Computing Services

What is now emerging are architectures which support different new components to be written and to be able to work together

- The components are called **services**
- **Web services** allow an application developer to write an application using components which are actually housed on different machines
- e.g. a financial application may use a currency exchange calculator which is run by Reuters in London and a credit card facility run in New York



XML

Extensible Markup Language has become the main mechanism for structuring data to be transmitted between programs

- Example glossary of terms:

```
<?xml version="1.0" ?>
<glossary>
<entry>  <term>XML</term>  <desc>A markup meta-language...</desc>  </entry>
.....  more entries
</glossary>
```

An XML file is verbose and self-describing and so can be transmitted to a program expecting a file of that structure, examples:

- XHTML – the well structured version of HTML that can be used to meet accessibility requirements
- SOAP – the structure for passing parameters between web services

XML files can have their structure described in a Document type Definition or an XML-Schema description

Grid Computing

Grid computing (or the use of a *computational grid*) is applying the resources of many computers in a network to a single problem at the same time

- usually to a scientific or technical problem that requires a great number of computer processing cycles or access to large amounts of data
- e.g. **SETI@Home** in which home computers are joined together to search for "rational" signals from outer space hoping to find extra-terrestrial intelligence

In Britain, the National E-Science Centre is trying to build systems to help scientists exploit computing power in several areas:

- Particle Physics
- Biomedical Applications
- Earth Observations

Note all of these have massive computation needs

Ubiquitous Computing

The other main development is of systems which allow people to use computing technology without having to work in a specific place in which there is a computer tied physically to a network

Ubiquitous (or pervasive) computing depends on two physical components:

- **wireless networks** which allow computing devices to continue to function as they are moved around
- computing devices which are **portable** and not as physically large or heavy as computers traditionally are
  - interactive devices such as laptops, handheld devices, PDAs and mobile phones
  - data capture devices such as bar code readers and sensors

Such a combination of computing and communication equipment promises the ability to move around and access (and manage) information anywhere

- Another term used is **wearable computing** – implying devices that you can wear like glasses

Software Problems for Ubiquitous Computing

Ubiquitous computing adds a new dimension to computer applications – **context**

Ubiquitous application have to respond differently depending on:

- **where they are**
  - should they be silent?
  - should they display a map focussed on where you are
- **which device** is being used
  - portable devices typically have smaller screens, harder to use keyboards and slower network connections
- personal **preferences**
  - for preferred formats
  - for disabilities (legal requirements may play a part here)
  - are you in a bad mood
  - etc.

Summary

Software structures have evolved to match the growth in computing power and network capabilities

Applications have evolved to deal with

- structured calculations
- large amounts of data
- multimedia
- distributed data
- different devices

Software is increasingly linked together

- in Office-style grouped applications
- in distributed systems

The goal is for applications which run anywhere in appropriate ways