Advanced Internet Programming

Introduction
Topics

• Enterprise systems
  - History
  - Requirements

• Software development methodologies
Review of prior knowledge

• Remember, this is an “Advanced” subject

• Most importantly, you are expected to be competent at Java programming

• Additionally, you should be able to create web pages using HTML, CSS, JavaScript

• Finally, you should have some understanding of CGI and how it works
References for prior knowledge

• Books:
  – a range of other books on HTML, CSS, JavaScript

• Web (tutorials, reference, examples):
  – www.w3schools.com
  – www.webreference.com
  – www.webdeveloper.com
  – www.hotscripts.com
  – etc.
Topics

• Enterprise systems
  - History
  - Requirements

• Software development methodologies
“Enterprise systems encompass those distributed, scalable, multiuser and business-critical systems that are intimately related to enhancing the productivity of a corporate or organizational enterprise via information technology”

[Perrone & Chaganti, p. 2]
"Enterprise applications can be those that cater to end-users via the Internet, partners via the Internet or private networks, various business units within the enterprise via various kinds of user interface. In essence, enterprise applications are those that let an enterprise manage its business activities”

[Subrahmanym et al., p. 11]
Enterprise systems ...

• Most introductory Java courses focus on client-side programming
  - standalone Java applications
  - Java applets embedded in web pages

• Enterprise systems focus on the Java features for developing server-side (enterprise) applications:
  - Enterprise JavaBeans, Servlets, JSP
  - Database connectivity, Transactions, Security, etc.

• Some of this is learning more about Java, while some is about understanding the architecture of enterprise systems
Enterprise systems ...

• In essence, enterprise systems focus on the parts of an application that run “behind-the-scenes” within an organisation, rather than what happens on the client’s desktop, e.g.

  - B2C e-commerce - selling to consumers
  - B2B e-commerce - communication with business partners
  - Web (Internet/Intranet) enabling of legacy applications
  - Enterprise Application Integration (EAI)

• Current trend is **WEB ENABLEMENT**
History of web applications

- The web began as an information system
  - Although there was provision for running CGI programs, these were originally aimed at creating search/indexing facilities, and simple feedback forms

- The web has evolved towards service delivery
  - Online shopping, banking, bill payment, etc

- But the web infrastructure has not fundamentally changed to support service delivery
  - Stateless interaction, no support for transactions
  - Relatively primitive user interfaces (HTML/CSS/JavaScript)
  - Security has improved though
Enterprise system requirements

- Web-enabled enterprise applications must be:
  - robust / reliable / highly available
  - scalable
  - secure
  - transactional
  - distributed
  - etc.

And …
Enterprise system requirements ...

• Applications must integrate with existing (web and non-web) business systems
  - Databases
  - Enterprise Resource Planning (ERP) systems
  - Customer Relationship Management (CRM) systems
  - Proprietary applications

• Flexibility needed to cope with diversity of existing enterprise systems to integrate with!

And …
Enterprise system requirements ...

- Applications must offer a variety of possible user interfaces:
  - Plain Web (HTML/CSS/JavaScript)
  - Web with embedded application (e.g. Java applets)
  - Standalone client programs (e.g. Java applications)
  - Mobile phones via WAP (Wireless Application Protocol)

- Flexibility needed to cope with future user interface requirements!
Satisfying these requirements

• Clearly, developing applications to meet such demanding requirements is not easy

• To develop these applications, you need:
  – an understanding of architectures for enterprise systems
  – an understanding of the technologies available for building enterprise applications
  – a software environment that supports the development and deployment of such applications
  – knowledge of relevant class libraries / API calls

• This subject offers an opportunity to acquire some of the basic skills to develop enterprise systems
Requirements in detail

Technical

- Functionality
- Performance
- Scalability ‡
- Flexibility
- Security †‡
- Reliability †‡
- Availability †‡
- Maintainability †
- Safety †

Other

- Usability
- Site/application design
- Information architecture

† Perrone et al, Ch 24 ‡[Subrahmanym et al, ch 1]
Functionality

- Functional requirements are the “obvious” ones – what tasks the application is expected to perform

- Usually defined in a functional specification document
Performance

• Performance indicators (aka metrics), e.g.:
  – interactive response time
  – transactions per second (throughput)
  – processing time
  – network delay (latency)

• Often graphed against number of concurrent users

• Need:
  – workload model
  – measurement/monitoring strategies
Scalability

- Ability to cope with increasing demands
  - typically more users

- Can’t always just get faster hardware

- Requirement is usually from “management”
  - unlike performance where requirement comes from users of the system
  - but … poor scalability leads to poor perceived performance
Flexibility

• Ease with which an application can be adapted to changing business environment
  – changed business requirements
  – new business requirements

• Reuse is one key to flexibility
  – motivation for component-based apps
Security

• Security is a large topic in itself
• Traditional security aspects (ISO 7498-2):
  – Identification & Authentication
  – Access control and Permissions
  – Secure communication

• Typically an application has requirements from all of these aspects

[ISO/IEC 7498-2:1989 Information processing systems; Open Systems Interconnection; basic reference model; Part 2: Security architecture]
Reliability

- Measure of application correctness
- Unreliable behaviour may be caused by:
  - programmer bugs
  - support environment (app server/OS) bugs
  - unexpected external conditions (still a bug?)

- Different manifestations, e.g.:
  - system crash
  - subtle errors in data
Availability

• Measure of application “uptime”

• Lack of availability usually caused by failure to satisfy some other requirement, e.g.
  - unreliable application crashes system → downtime
  - insecure application hacked and crashed → downtime
  - non-scalable application overloaded → denial of service

• Despite this, availability is often a separate criteria
  - e.g. 99.97% availability (2.6 hours per year downtime)
  - e.g. 99.997% availability (15 mins per year downtime)
Maintainability

- Ease of maintenance of application

- Results in longer term problems
  - difficult to fix bugs (reliability)
  - difficult to adapt to changing business needs (flexibility)
  - expensive to maintain
Safety

• Safety-critical applications typically require additional design/development
  – “safety-critical” implies that a human life depends upon correctness and availability

• Not common to in the e-business domain
  – more related to command and control apps
Usability

• How easy the application is for humans to use

• Often a critical success factor
  – especially if your application is deployed on the Internet

• Internet users typically:
  – don’t read detailed instructions on web pages
  – give up if they can’t find what they need
  – give up if a task seems too complicated
  – etc.
Site/ application design

• One part of usability

• Can include:
  – brand design
  – menu design
  – navigation design

• Hopefully by a web designer, not a programmer!
  – need to interact, particularly w.r.t. navigation
  – design should influence programming, never the other way around
    • Users more tolerant of a usable but less functional application than one which functions perfectly but is unusable!
Information architecture

• Requirements on what data needs to be stored

• Driven by business needs, present and future
  – business needs come from user demand, service differentiation, legal requirements, etc.

• This may impose requirements on apps
  – collection and storage of data
  – business logic using that data
  – usage/updating of legacy data
Requirements summary

- Requirements analysis is not unique to web apps
  - however web apps introduce new/different requirements

- Specification of a system typically needs to address most/all of the requirements described

- An important part of developing any system
Topics

• Enterprise systems
  - History
  - Requirements

• Software development methodologies
Software Development

• “Traditional” software engineering models are not suited to web development
  - Waterfall
  - Spiral

• Enterprise applications must typically be:
  - quick to market
  - quick to respond to change
  - component-based and distributed
SDLC - Waterfall

Requirements Analysis → Design → Build → Test → Deploy & Maintain
SDLC - Iterative

Determine objectives, alternatives, constraints

Evaluate alternatives
identify & resolve risks

Create and evaluate prototype

Review & further Commitment

Requirements plan
Life-cycle plan

Development Plan

Integration & test plan

Requirements

Design

Implement

Test

Develop and Verify

Deploy

Plan next phase

(Source: Boehm, B. 1988)
In recent years, there have been trends to more appropriate methodologies for developing web-based apps:

- Rapid Application Development (RAD)
- Relationship Management Methodology (RMM)
- Hypertext Design Model (HDM) & OO-HDM
- Rational Unified Process (RUP)
- Extreme Programming (XP)
- Aspect Oriented Programming (AOP)
SDLC - Rational Unified Process

• Methodology to:
  - enhance productivity of developers
  - use software engineering best practices

• Guidance on:
  - web architectures
  - business modelling
  - web testing
SDLC - RUP

SDLC - RUP & UML

• RUP assumes UML as a modelling tool
  – Unified Modelling Language

• UML is a set of languages for modelling different parts of a system
  – Use cases (user requirements)
  – Class diagrams (data modelling)
  – Activity diagrams
  – Interaction diagrams
  – State Transition diagrams
  – Package diagrams
SDLC - UML Use Cases

• Used to describe how a user interacts with the system

• Informal language – understandable by non-technical domain experts
SDLC - UML Class diagrams

• For modelling object-oriented systems
• Describes each class in the system, including:
  – name
  – attributes & methods
  – relationships to other classes (inheritance, aggregation)
  – constraints

• Can be used with some tools to auto-generate code skeletons
SDLC - Extreme Programming

• Less structured, or lightweight methodology
• Emphasises customer satisfaction, not rigidity
• Follows the way many programmers actually work when left to their own devices

• See:
  – The Rules and Practices of Extreme Programming
    http://www.extremeprogramming.org/rules.html
SDLC - summary

- Newer methodologies more relevant to web applications

- For development in this subject, you may choose whichever methodology you prefer
  - but you should be conscious that you are choosing

- Can mix and match
  - Just because you choose an XP approach doesn’t mean you can’t write UML class diagrams!
Rapid Development

• One technique is to use Model Driven Architecture (MDA) and Aspect Oriented Programming (AOP).
• Others include Rule Driven engines and frameworks such as Spring MVC
• You can use automation tools such as Ant & Xdoclet & other build tools to reduce manual processing
• You also should use Integrated Development Environment (IDE) to help you build your application
IDE

• Advantage: ease of development by having single interface to many tools
• Many tools to choose from, but in J2EE world, a few stand out

• **ECLIPSE**: - open source, java based, very popular. Many 3rd party tool vendors, including IBM (Rational Application Developer), JBoss, Oracle (J Developer), BEA (Workshop for Weblogic) etc
• **NetBeans**: - from Sun
• **J Builder**: - from Borland/Codegear
• **J Developer**: from Oracle
Eclipse
Coming up next ...

- Next session focuses on architectural issues
  - a sound knowledge of enterprise system architectures is vital to assimilating all the different topics in this subject

- After that, the coding begins ...

- And at the end we revisit application architecture/design
Summary

• Prior knowledge is essential, especially Java
  – no chance to “catch up as you go”

• Enterprise systems are here now!
  – Complex applications with many requirements
  – Requirements on the application itself
  – Requirements for integration with existing systems
  – Requirements for flexible presentation to users

• Software development
  – new methodologies for new kinds of applications
This week’s lab

• Go into the Subject Documents -> Architecture folder.

• Do the install weblogic lab
  – (this installs weblogic)

• (OPTIONAL): Do the familiarisation1 lab
  – (this installs workshop for weblogic, do not install if you have your own Eclipse setup)

• Do the familiarisation2 lab
  – This let’s you play with the webLogic admin console
Questions?