


HUMAN-COMPUTER INTERACTION THIRD EDITION DIX FINLAY ABOARD BEALE

chapter 21

hypertext, multimedia and the world-wide web

extract for MSc/MRes AISD


dynamic web content



ALAN DIX, JANEY FINLAY, GREGORY D. ABOARD, ROBERT W. BEALE
HUMAN-COMPUTER INTERACTION
Third Edition

dynamic web content


what happens where
technology and security
local interaction, search
remote & batch generation
dynamic content



ALAN DIX, JANEY FINLAY, GREGORY D. ABOARD, ROBERT W. BEALE
HUMAN-COMPUTER INTERACTION
Third Edition

the active web


- early days of the web
 - static pages ... mostly text
 - some gateways (ftp, gopher)
 - usability ... easy - one simple model (except frames break the model!)
- dynamic content
 - what is the model/metaphor ???
 - passive pages or active interface
 - each leads to different user understanding
 - no easy answers!



ALAN DIX, JANEY FINLAY, GREGORY D. ABOARD, ROBERT W. BEALE
HUMAN-COMPUTER INTERACTION
Third Edition

what happens where?


- architectural design is about what happens where
- this affects:
 - feedback
 - seeing results of one's own actions
 - feedthrough
 - seeing effects of other people's actions
 - also affects complexity of implementation and hence maintenance



ALAN DIX, JANEY FINLAY, GREGORY D. ABOARD, ROBERT W. BEALE
HUMAN-COMPUTER INTERACTION
Third Edition

user view

- what changes?
 - media stream, presentation, content
- by whom?
 - automatic, site author, user
 - other users - feedthrough
- how often?
 - pace of change: days, months, seconds



ALAN DIX, JANEY FINLAY, GREGORY D. ABOARD, ROBERT W. BEALE
HUMAN-COMPUTER INTERACTION
Third Edition

technology

where does it happen

client

- applets, Flash, JavaScript & DHTML

server

- CGI scripts, Java servlets, JSP, ASP, PHP, etc,

another machine

- author's machine, database server, proxy

people

- socio-technical solutions

security

- for computation
 - code and data at same place!
- problem
 - data - needs to be secure
 - web-server - least secure machine
 - client machine even worse

... and networks!

local interaction (at client)

```

graph LR
    User((User)) -- "(ii) user interacts locally" --> Client[script / Java running in client]
    Client -- "(i) page loads once" --> Server[web server]
  
```

- fixed content
- use Java applets, Flash, JavaScript+DHTML
- pros: rapid feedback
- cons: only local, no feedthrough
- after interaction ... what does 'back' do ??

examples

two horse races

coin race uses JavaScript

dancing histograms are a Java applet

search

```

graph LR
    User((User)) -- "(i) user fills field in form" --> Client[web page with text field for search words]
    Client -- "(ii) search results returned" --> Server[web server]
    Server -- "CGI script looks up words in index" --> Index[index file pre-computed]
  
```

- create indices off-line
- fast lookup when needed

see <http://www.hcibook.com/e3/search/>

automatic generation

- dilemma;
 - hand crafting ... leads to web stasis!!
 - so need database driven sites
- early days ad hoc, now many tools
- options:
 - client-end applet or Flash access remote DB
 - server-end CGI driven by web forms (limited UI)
- hybrid solutions
 - CGI generated pages can contain JavaScript etc.
 - JavaScript can 'write' web pages on the fly!

Java applet & JDBC

```

graph LR
    User((User)) --> Client[Java applet]
    Client -- "JDBC accesses database" --> Server[web server]
  
```

- pros: interactive DB access
- cons: bandwidth, security

CGI script accesses database

```

graph LR
    User((User's machine)) -- "(i) request to server" --> WebServer[web server]
    subgraph WebServer
        CGI[CGI script]
        DB[(database)]
        CGI -- "(ii) CGI script accesses database using SQLJDBC" --> DB
        DB -- "(iii) server returns generated pages" --> User
    end
  
```

- pros: up-to-date, use existing DB
- cons: not proxy/index friendly

batch generation

- for slow varying data
 - update local database
 - periodically generate pages and upload
- many technologies
 - C, Java, HyperCard, Visual Basic

```

Set db = openDatabase("C:\test.mdb");
sql = "select Name, Address from Personnel;";
Set query = db.OpenRecordset(sql)
Open "out.html" For Output As #1
Print #1, "<h1>Address List</h1>"
query.MoveFirst
While Not query.EOF
Print #1, "<p>" & query("Name") & " " & query("Address")
query.MoveNext
Wend
Close #1
query.Close
  
```

batch generation of web pages

```

graph LR
    User((User's machine)) -- "(iii) server returns generated pages" --> WebServer[web server]
    subgraph WebServer
        GenPages[generated pages]
    end
    subgraph ThirdMachine[third machine]
        DB[(database)]
        GenOff[pages generated off-line from database]
    end
    GenOff -- "(i) pages generated off-line from database" --> DB
    GenOff -- "(ii) pages copied to web server via ftp" --> GenPages
    GenPages -- "(iii) server returns generated pages" --> User
  
```

- pros: indexable, secure
- cons: slower turnaround

dynamic content

- really 'active' web pages ...
 - data updated as well as presented on the web
- presentation
 - any of the previous means: CGI, applet-JDBC
- update
 - web form/interface -> server script -> update db
 - e.g. book theatre seats
- issues
 - authentication and security
 - multiple transactions due to 'back' button
 - right pace/control - do we want human in the loop?

n-tier architecture

```

graph LR
    User((User's machine)) -- HTML --> WebServer[web server]
    subgraph WebServer
        JSP[JSP]
    end
    subgraph EnterpriseServer[enterprise server]
        JEB[JEB]
    end
    subgraph Database
        DB[(database)]
    end
    JSP -- XML --> JEB
    JEB -- JDBC --> DB
  
```

- one or more intermediate layers
- 'business logic' in layers
- web standard components and protocols