# Programming with XML

Richard Connor

Distinguished Professor of
Computer Science,
University of Strathclyde

# With help from…

- Initial ideas
  - Ron Morrison, Stan Zdonik
- The team (and ex-team)
  - Colin English, David Lievens, Paolo Manghi, Steve Neely, George Russell, Keith Sibson, Fabio Simeoni, Anna Stavrianou, Scott Wilson
- With support from
  - EPSRC, BBSRC, European Union, University of Strathclyde, Reuters plc
- Despite antagonism from
  - UK database community!

# Outline of day

- Talk 1: Background to XML
  - Why it's important
- Talk 2: How to program over it
  - World as α–test platform
- Talk 3: How to really program over it
  - Our own research agenda


PRESENTS
G HIS
ESENTA
ERE TO
ORMANCES
1 AM
NOON
PM
PM
PM

# When is a document not a document?

- When it's XML
- *Despite its relatively humble origins as a document standard, XML encaptures a data modelling framework that fits with the computational world of the 21st Century. Once it has received the same research and financial investment that the relational framework has enjoyed, it will far outstrip it in utility.*

# This talk:

- 1. Introduction to XML
  - Semi-structured data on the web
- 2. Typeful programming by convention
  - Not by restrictive practice

# HTML → XML

- An evolution
- HTML contains much valuable information
  - And details of how to present it
- Idea:
  - Separate data and presentation
    - Clever!

# Presentation and Content

- As the HTML web grew…
  - people started to program over it
- Problem: mixed presentation and content
- Solution (Amazon – early days….)
  - Employ a lot of programmers
  - Pay them “on-call” fees
  - Sound physical alarm when page layout changes

# Reinvention: data on the web

- Small standard data language (XML)
- Data models by convention
  - DTD, XMLSchema, xmlns
- Presentation separated
  - XSL, XSLT
- Anyone can join
  - But who can be trusted?

# What is XML?

- A large, poorly “designed” standard
  - Upwards compatible
    - Superset language of HTML4.01 (almost…who cares?!)
- A large number of associated standards and technologies:
  - DTD, XMLSchema, XSLT, XQUERY
    database front-ends, content tools, …
- Contains the semi-structured data model
  - This last bit is good!

# Semistructured data

- Implicitly structured
  - self-describing
- Multiply invented, close to canonical data model
- Possible semantic interpretation: set of facts
- Let's invent it…

# Definition of ss-data

- data ::= scalar | factSet
- factSet ::= { $fact^{+}$ }
- fact ::= attributeName = data ;

# Semantics

- An edge-labelled graph
  - Not tree…
- Nodes are objects
- Edges are attributes
- Labels are attribute names

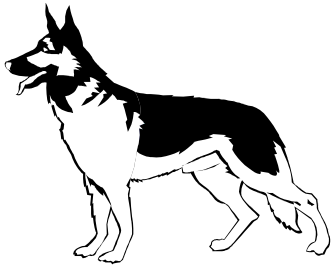
# The correct and true way

- InfoSet is a nice data modelling domain
- XML is a transient concrete syntax for it
  - (note neither of these statements is true…)

# Why now?

- We can process it!
  - Most database technology is about how to avoid more than a few hard disk hits
    - On data that nowadays fits in processor cache!
- Data isn't much bigger than it used to be (?)
- Change to lose restrictive models
  - And share data globally

# Context switch…



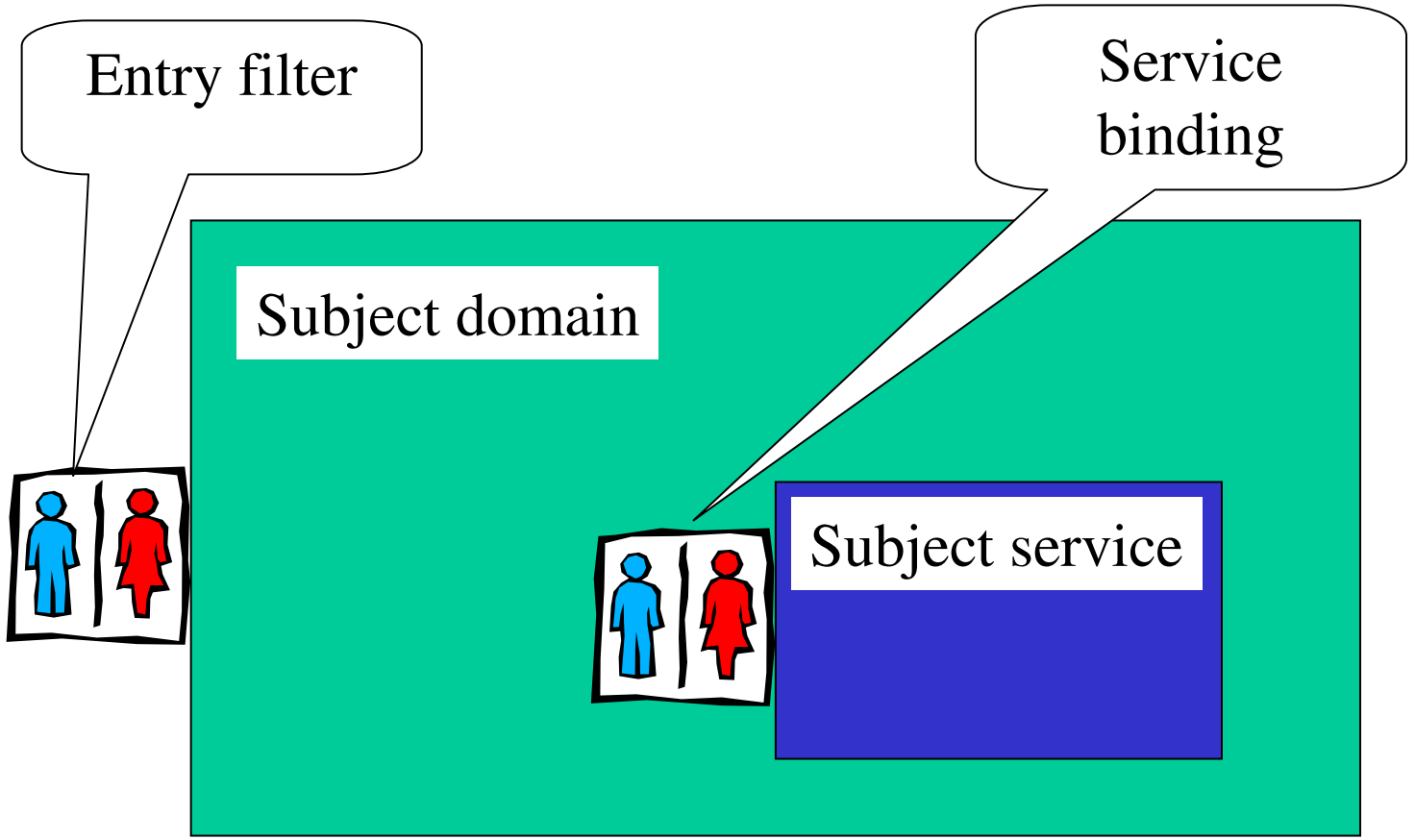
# Convention vs enforcement

- Safety versus Flexibility
- In the olden days…
  - … we liked enforcement
    - Maximum safety
    - Minimum flexibility
      - So who cares?
- Now, convention is king
  - Assume everyone obeys the rules

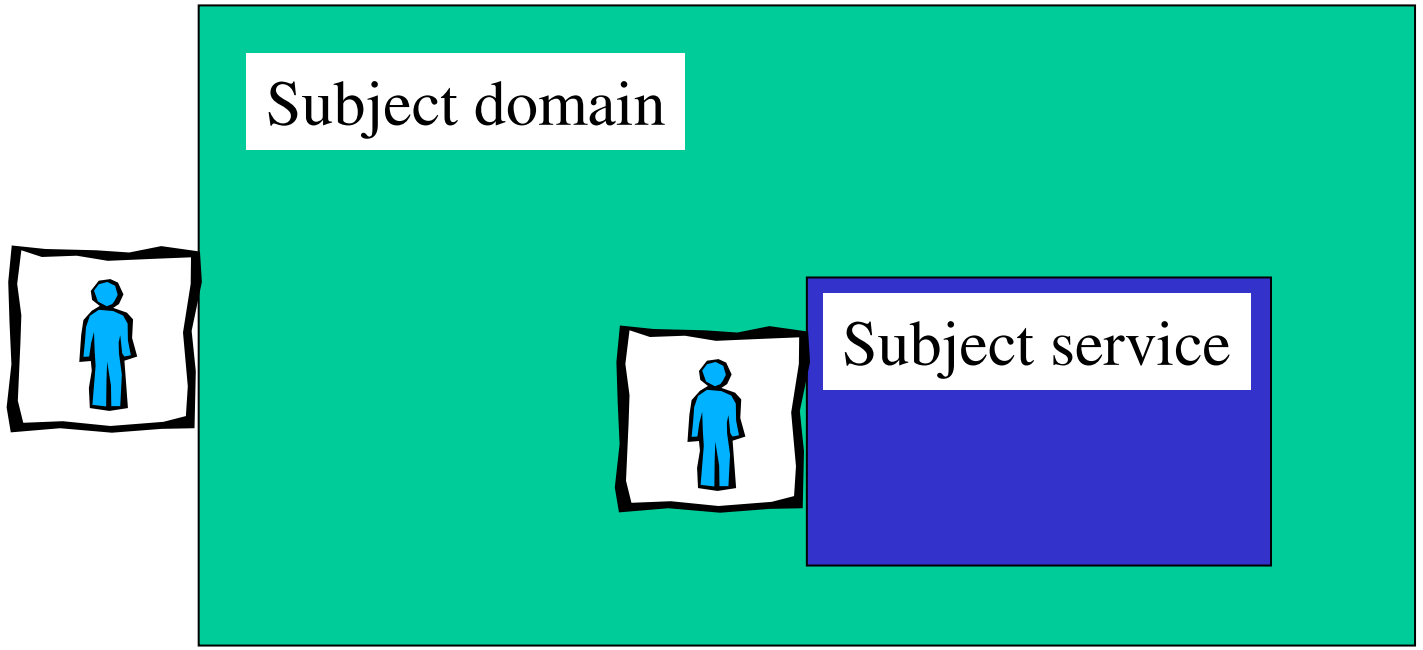
# Enforcement

# Convention

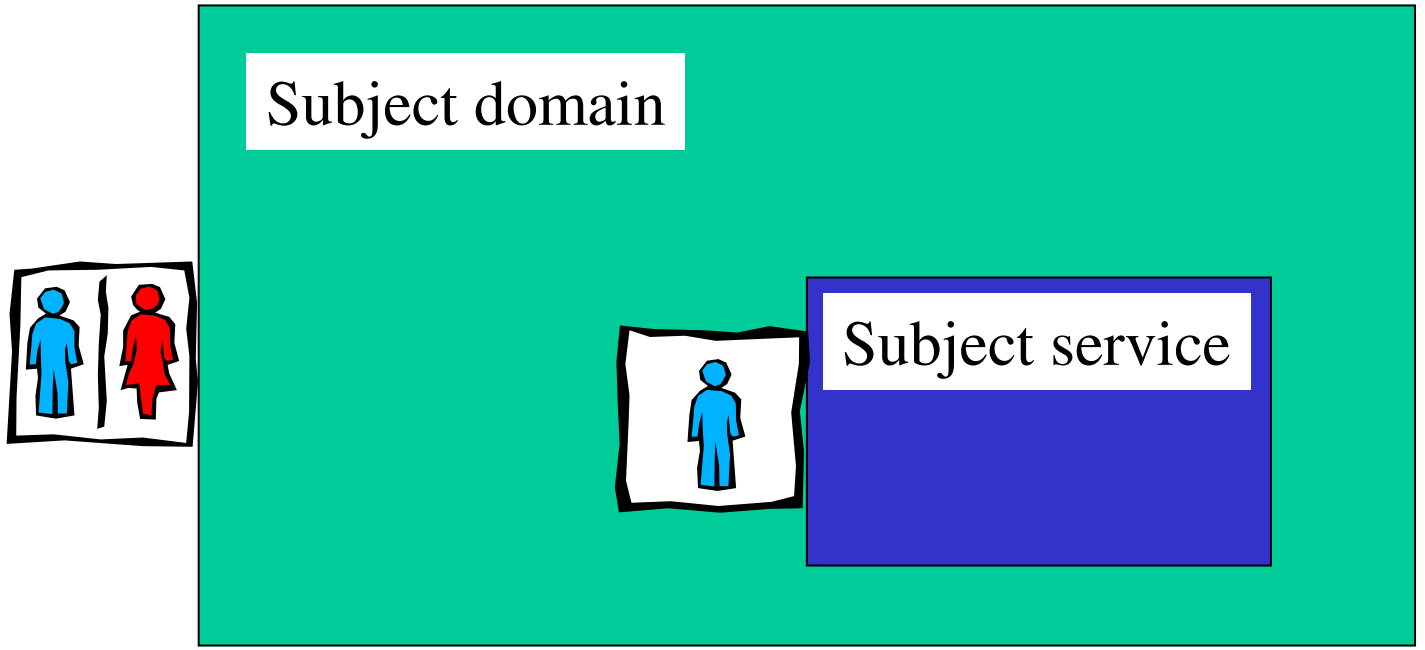
# Generic Model



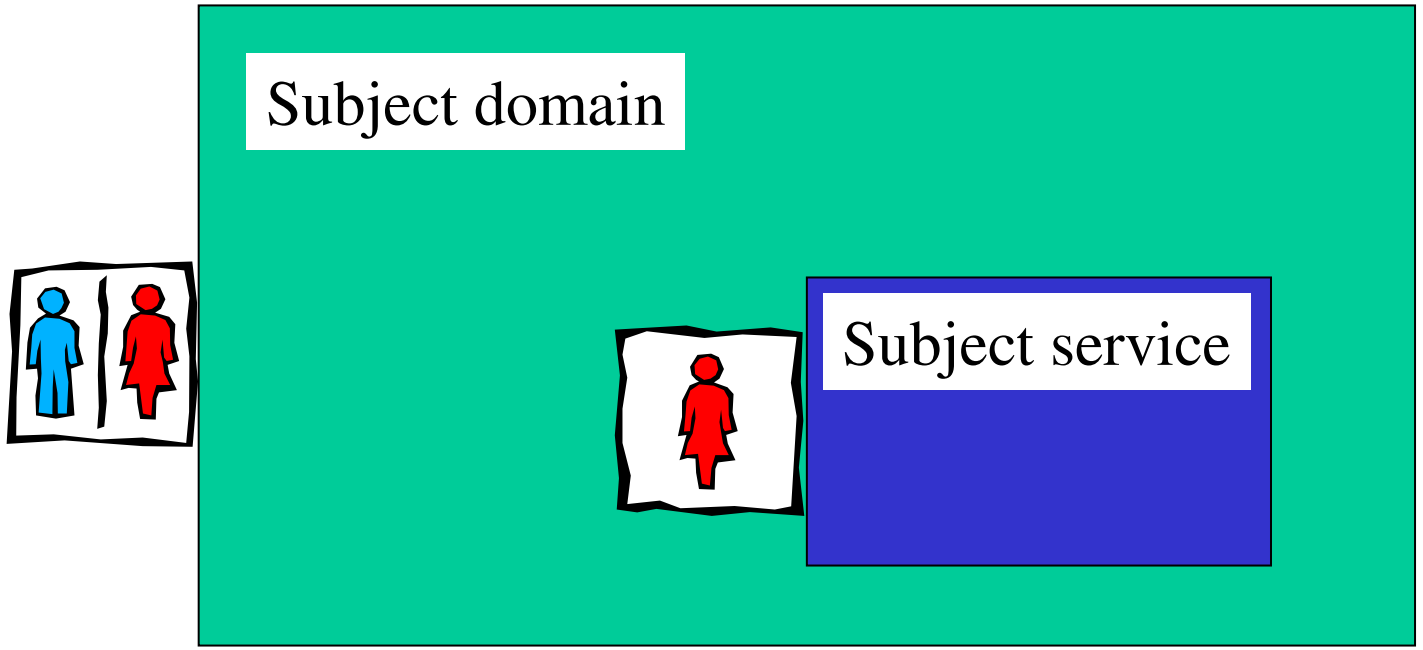
# Enforcement Model



# Convention Model



# Convention Model – failure!



# Program model - enforced

- Eg Java

```
Class Person{ person stuff }
Class BluePerson is Person{
      void resuscitate() }

Var Person p
p.resuscitate()                    // won't compile

try{(BluePerson)p.resuscitate() }
catch{ /* p isn't blue, do something else */}
```

# Program model - convention

- Eg JavaScript

```
var p

try{ p.resuscitate() }
catch{ /* p isn't blue, do something else */}
```

# Program model – bad use of convention

- Eg JavaScript

```
var p

p.resuscitate()
```

# Analysis

- Flexible scenario requires same programmer effort in both cases
  - Good solution is equally complex
- JavaScript example is more likely to have unguarded failures
  - No static error detection because of flexibility…
- Java example can only be written in static context of class definition
  - The killer for global autonomy

# Context: the Web

- Entry by convention:
  - DNS
  - http
  - HTML
  - … and now - XML
- No bars on entry
- Only works if most stick to conventions
  - “standards”
    - In reality, loose sets of rules with a working core

# The result

- XML gives a good enough modelling context that…
- … if everyone obeys the conventions, we can share data over the web …
- …allowing a useful mix of autonomously provided data collections.

# The future…