When is a langauge not a language?

- When it's well-typed
- *Despite being somewhat unfashionable, type systems keep software partially safe, through the prevention of undefined execution sequences, ideally before program execution begins.*

Untrapped errors

- *When this is not possible, the difference between detected and undetected type errors is still critical.*
  - *var x = 3 / 0; // fail – divide by zero*
  - *var y = p.name; // fail – no such label*
  - *var z = lookup(p, "name"); // success*
    - *No allowance by programmer*
    - *May lead to wrong answer - undetected*

A database person type

- type person is { name : string, age : int }

```
function f ( var p : person ){
        … p.name …
        … p.address …
        … p.ape …
}
```

A web person

- Can not conform to a rigid type definition
  - Autonomy
    - Independence
    - Disagreement
  - Evolution
  - Pre-existing data
  - (Internationalisation)
- But: probably has a name, maybe age…

A web person type

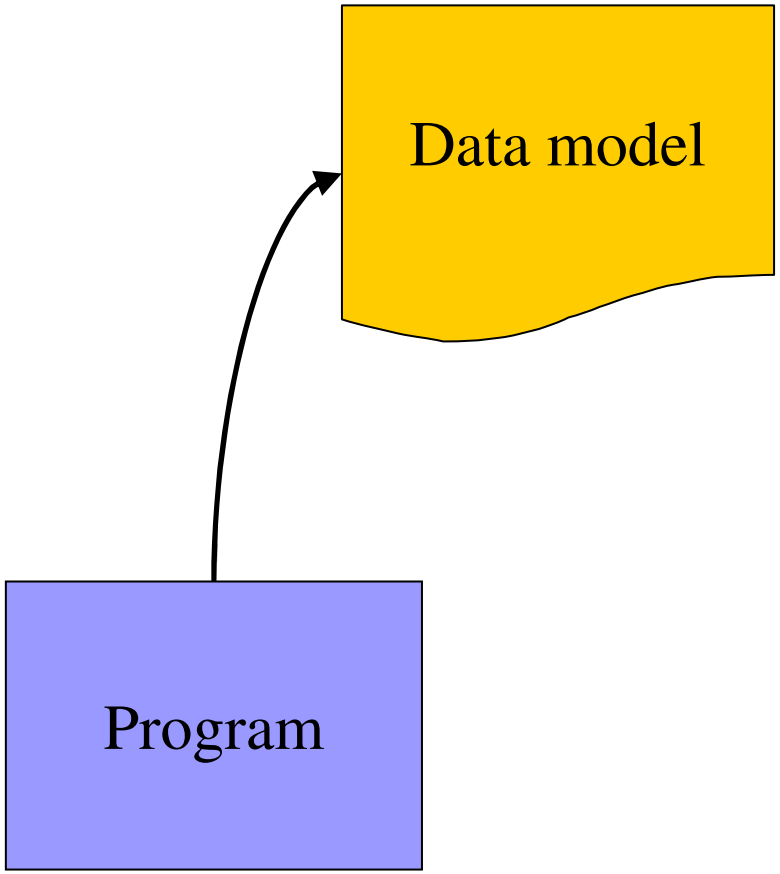
- typeof person is { name, age?, other stuff? }
  typeof name is String | {sname, fname}
  typeof other stuff is anything

  function f (var p : person){
  … p.name …
  … p.address …
  … p.ape …
  }

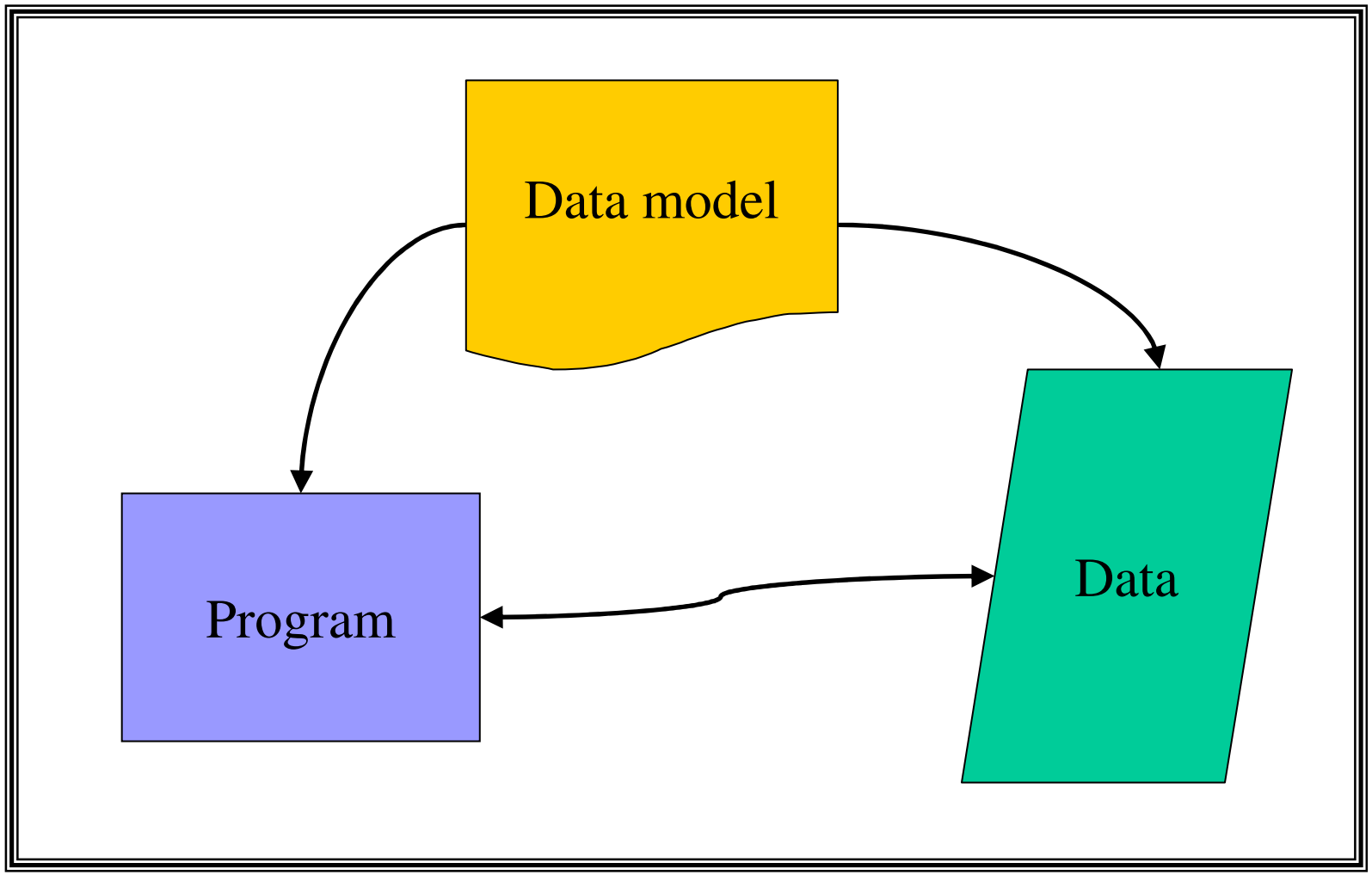
The web-type alchemist's dream

- A loosely agreed, flexible data model
  - May change with evolution
  - May only be partial wrt actual data
  - May be partially inaccurate wrt actual data
- Programs should work correctly
  - No untrapped error can occur
    - Ideally trapped before execution commences
  - As the data model changes
  - Also world peace would be nice

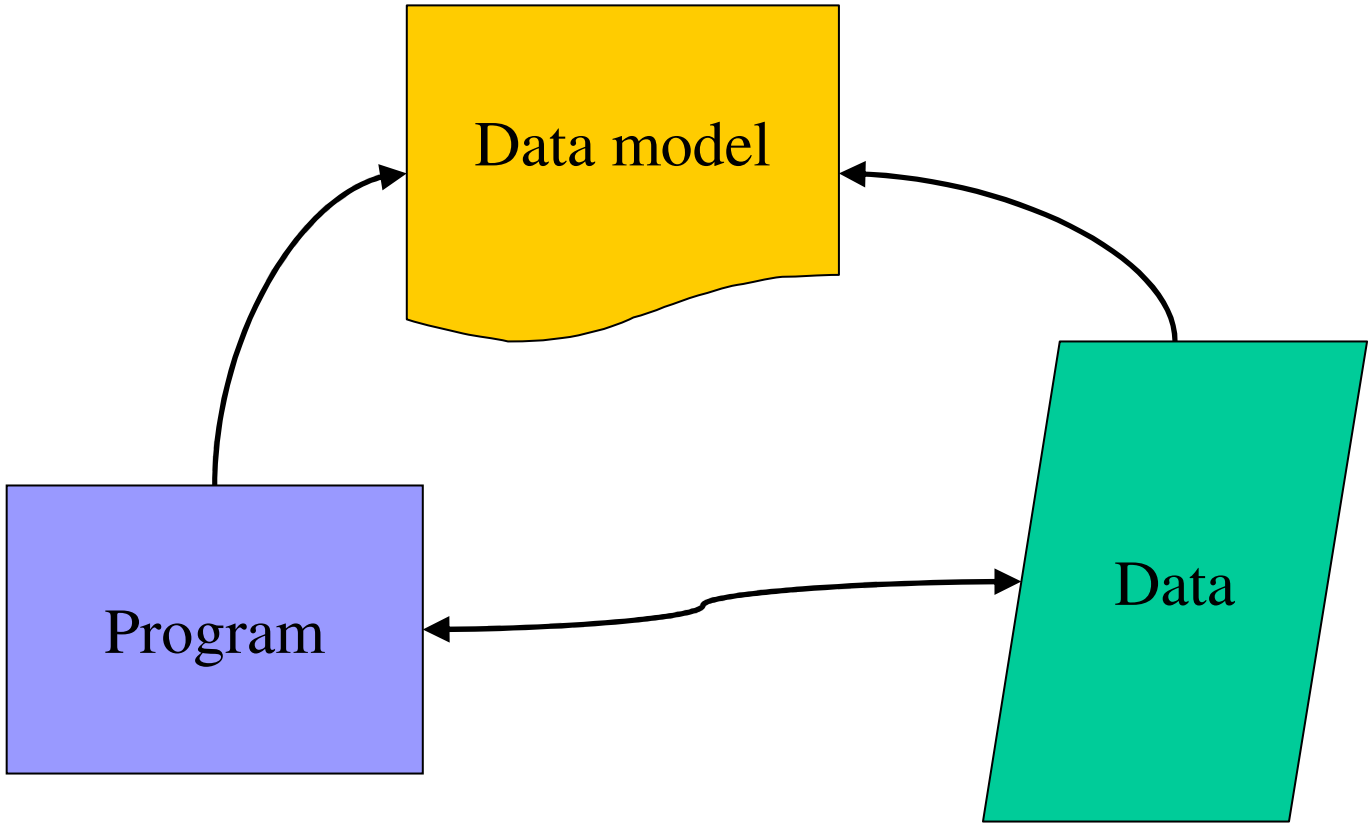
The programmer's task



Type safety - enforced



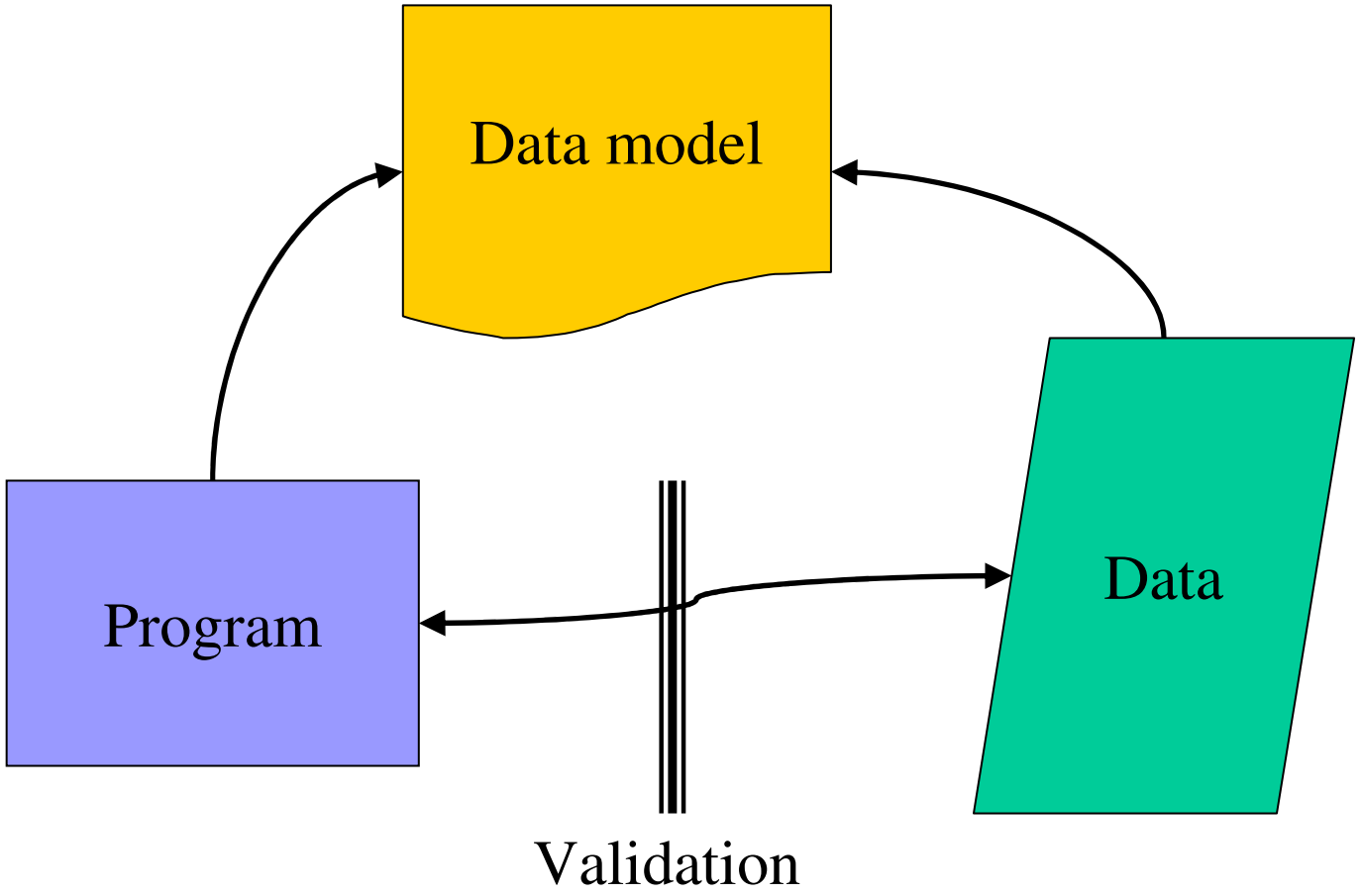
Type safety – by convention



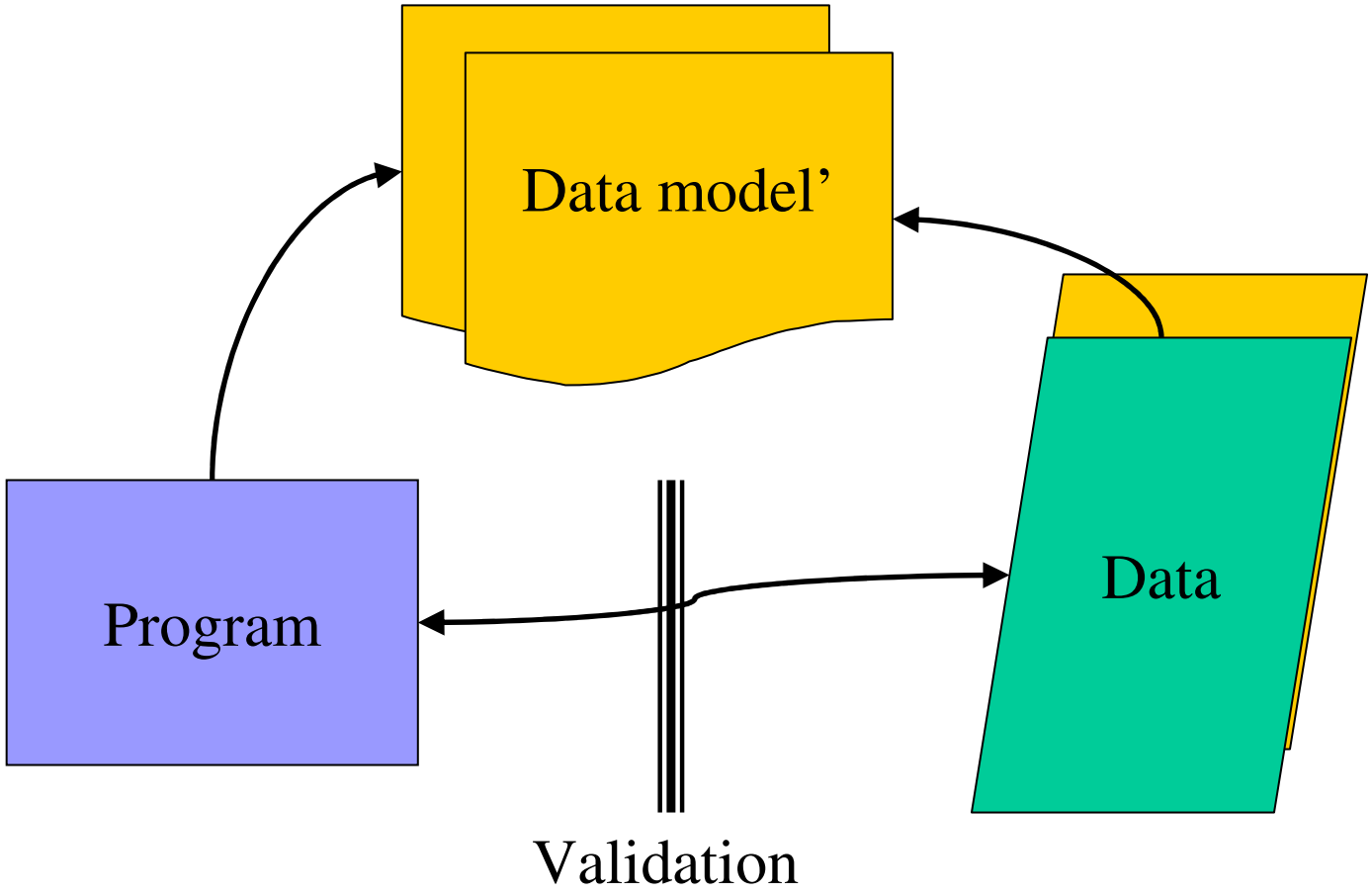
Safety through convention

- More flexibility
  - Should work if structural attributes are correct
  - Should still work if changes occur…
    - … which don't affect the meaning of the code
- Less safety
  - To achieve this flexibility, something must give
    - What, and how much?

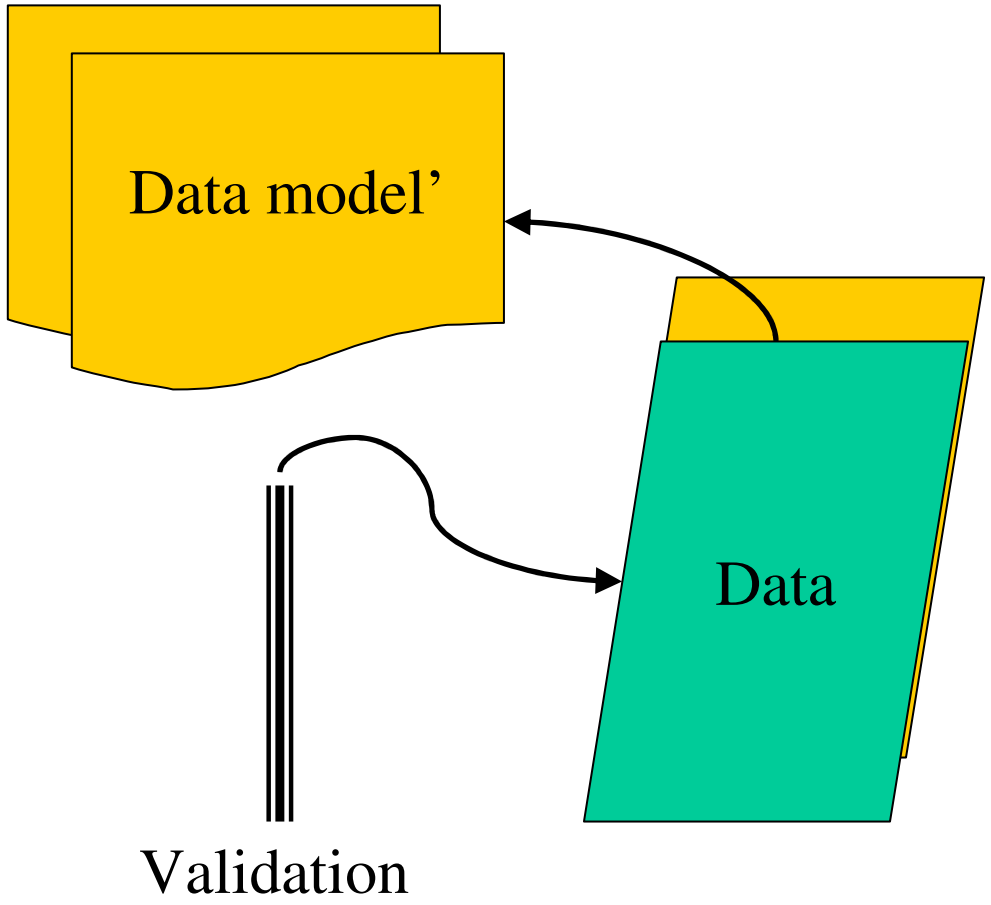
More type safety – by convention



Less type safety – by convention



Less type safety – by convention



“Identity” conventions

- URI
  - UUID
    - 094067-0396097546-3002497-34082087
    - persons.richard.cis.strath.ac.uk
  - URL
    - http://www.cis.strath.ac.uk/~richard/persons.htm
- xmlns
  - label to URI binding
- Weak identity – doesn’t capture equality!

Idioms of well-typing

- Interpretation
  - Program behaves itself, adjusting to context
- Generation
  - Data and program stubs generated from data model
- Projection
  - Data matched against most general program type

Interpretation

- SAX, DOM, XSLT
- Data is presented to programmer as a tree
  - Mechanically typed as such
- Type safety requires programming skills
  - Semantics and context are programmer's task
  - Attribute identifiers are treated as strings
    - Many errors are not statically detectable

miniDom interface

- class miniDomTree{
  miniDomTree(URL document)
  miniDomTree firstChild()
  miniDomTree nextSibling()
  String incomingEdge() // no direct model
  String elementText()
  }

miniDOM code

```
• void printNames( miniDomTree t ){
      if( t.firstChild != nil )
            { printNames( t.firstChild() ) }
      if( t.nextSibling != nil)
            { printNames( t.nextSibling() ) }
      if( t.incomingEdge() == “name” )
            { print( t.elementText() ) }
  }
```

miniSax interface

- interface miniSax{
    void openElement(String elementName)
    void closeElement(String elementName)
    void text(String theText)
  }

  class SaxEngine{
    void SaxEngine(URL document)
    void runEngine(miniSax whatToDo)
  }

miniSAX code

- class namePrinter implements miniSax{
  private Boolean inName = false

  void openElement(String elementName)
  if(elementName==“name”)
  { inName = true }

  void closeElement(String elementName)
  if(elementName==“name”)
  { inName = false }

  void text(String theText)
  if(inName){ print(theText) }
  }

XSLT

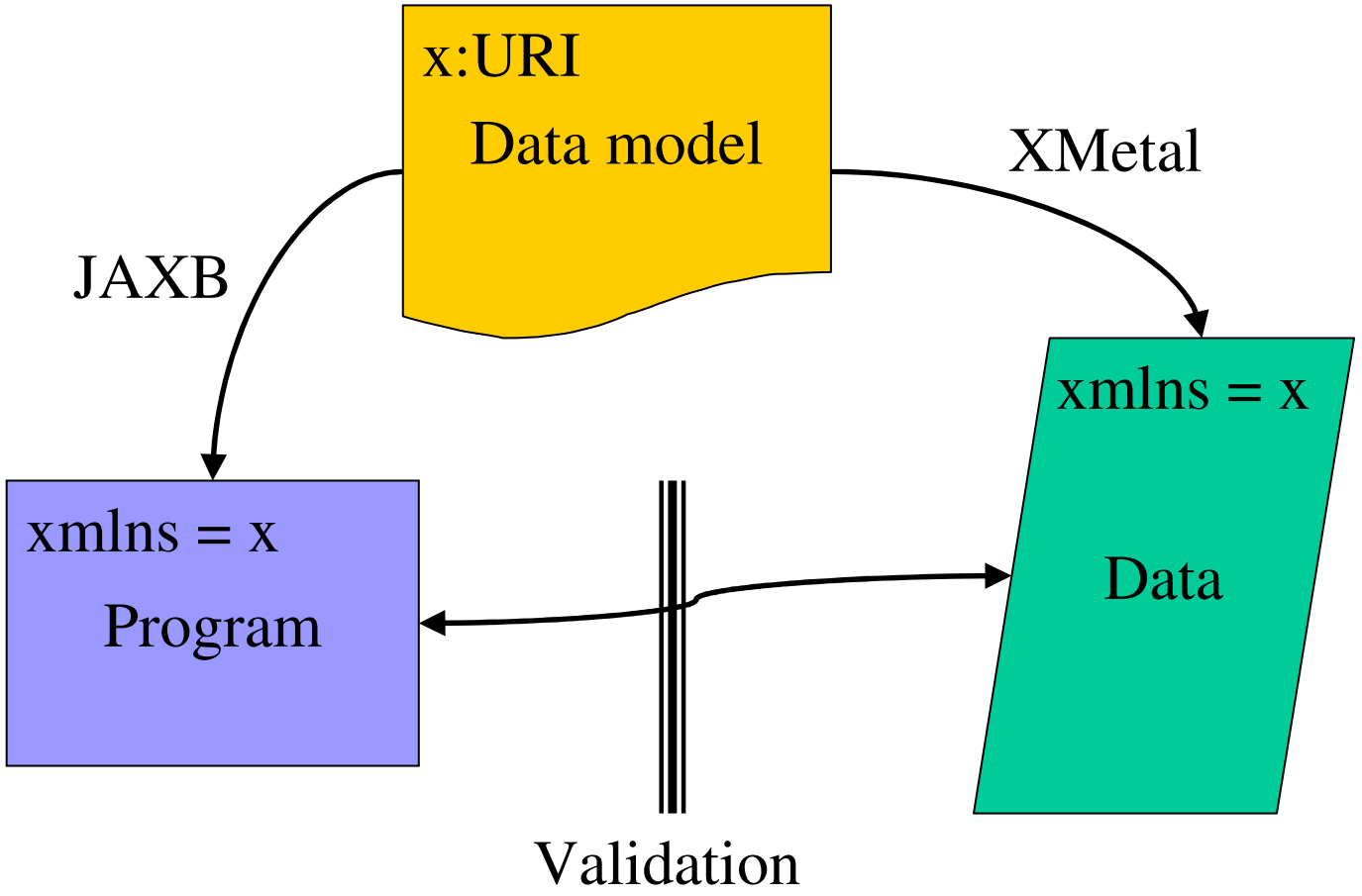
- <xsl:template match=“person">
  <xsl:value-of
  select=“name"/>
  </xsl:template>

All of the above…

- … lead to …

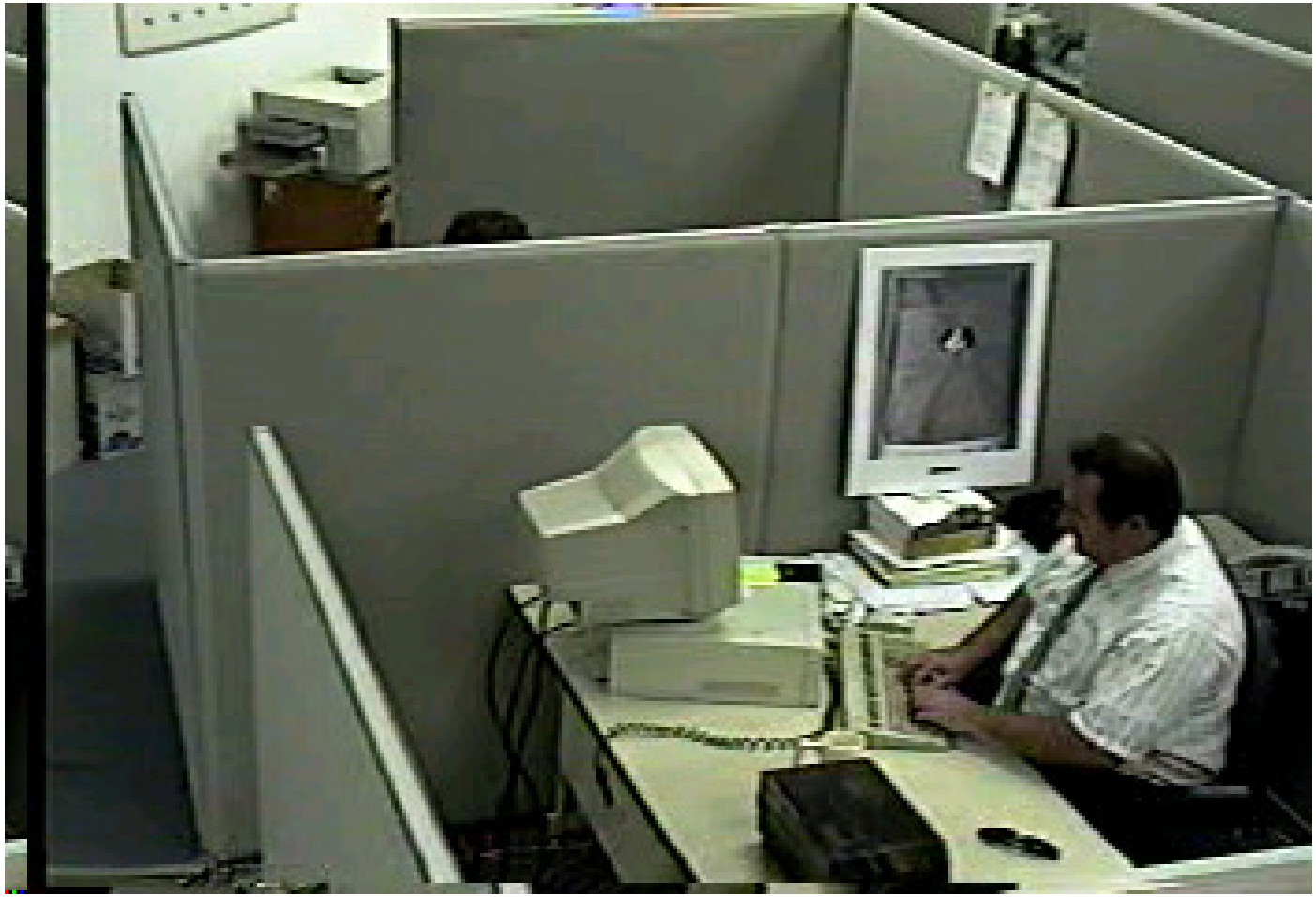
Untrapped type errors!

Generation


x:URI
Data model
JAXB
XMetal
xmlns = x
Program
xmlns = x
Data
Validation

Restrictive: can’t change schema

Leading to…



Projection

- Ultimate cool!!!