A data mapping specification environment using a concrete business form-based metaphor

Yongqiang Li, John Grundy, Robert Amor and John Hosking
Department of Computer Science
University of Auckland, New Zealand
Outline

- Motivation for our work
- Need for data mapping systems
- Existing approaches to data mapping
- Our approach: “form-based copying”
- Overview of prototype environment
- Examples of specifying mappings
- Data transformation code generation
- Future work
- Summary
Motivation

• Data transformation required in many problem domains:
  - Message-based systems integration
  - Database integration
  - ERP system integration
  - Document exchange

• Existing approaches to specifying data mappings:
  - Programming, scripting (textual)
  - Tree-based visual linking
  - Custom “visual language” (programmatic)
Existing Tool Example

- Common approach to specifying mappings
- Tree based schema and mapping functions
- Programmers are users
- Complex mapping language underneath
Context for Our Work

• Enterprise systems integration
• Need to be able to specify complex data transformations
• Want END USERS to be able to specify complex data mappings…
Our Approach

• Aim: to support end-user specification of complex mappings
• Users = “business analysts”
• Want to generate mapping implementations
• Most businesses manually implement such data exchange manually via “copying” from one business form to another (hard-copy or on a computer)
• We wanted to support this “form-copying” metaphor in an end-user oriented mapping specification tool...
**Process**

1. Analyst imports meta-data from source and target enterprise systems

4. Data transformation implementation generated from specification

2. Default business form layouts generated. Analyst can rearrange layout to better-reflect actual business forms.

3. Analyst specifies 1:1, 1:n, m:1 group and field correspondences i.e. specifies how to “copy” data from one form to the other

**Meta-data e.g. XML DTDs**
Example of Form-based Visualisation of Documents
Rearranging Form Layout

Resize

Add sub-structure

Rearrange layout
Specifying Mappings
Code Generation...

Customer Form
- ID
- Name
- Address

Orders Subform
- OrderDate

Order Form
- DateTime
- Created
- TotalCost

Order Items
- Book
- Qty
- TotalCost

Orders Subform
- BookInfo
- Qty

Customer Info

Order:
1

XSLT transformation
script generation

```xml
<xsl:template match="/">
  <Order>
    <Number>…</Number>
    <DateTime><xsl:value-of select="/Order[1]/Order/Date"/></DateTime>
    <Created><xsl:value-of select="date:to-string(date:new())"/></Created>
    <TotalCost><xsl:value-of select="sum(//OrderItem/TotalCost)"/></TotalCost>
    <xsl:variable name="customer_id" select="/Order/OrderItem[1]/CustomerSID"/>
    <CustomerInfo><xsl:apply-templates select="//Customer[@id = $customer_id]"/></CustomerInfo>
    <Items><xsl:apply-templates select="//OrderItem"/></Items>
  </Order>
</xsl:template>
```

...
Future Work

• Lots of kinds of mappings – need to distinguish e.g.
  - Record -> record
  - Field -> multiple fields
  - Fields -> one field
  - Records -> select to one record
  - Multiple records -> multiple records

• Programming by example specification of these plus field splitting/merging

• “Sub-form”-based mappings for “functions”

• Visual formulae for target field/record values
Summary

• Data mapping systems required in many problem domains
• Current approaches very programmatic
• Want to support business analysts specifying complex data mappings
• Used form-based copying metaphor within a prototype environment to do this
• Automatic mapping code generation
• Extending in various ways to improve usability/visualisation of complex mappings
References