Copyright 2002 IEEE. Published in the Proceedings of 2002 IEEE International Conference Human-Centric Computing, Arlington, VA, Sept3-6 2002. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works, must be obtained from the IEEE. Contact: Manager, Copyrights and Permissions / IEEE Service Center / 445 Hoes Lane / P.O. Box 1331 / Piscataway, NJ 08855-1331, USA. Telephone: + Intl. 732-562-3966.

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

Yongqiang Li[1], John Grundy[1, 2], Robert Amor[1] and John Hosking[1]

[1]Department of Computer Science and [2]Department of Electrical and Electronic Engineering, University of Auckland, Private Bag 92019, Auckland, New Zealand
{john-g, trebor, john}@cs.auckland.ac.nz

Abstract

Many systems require data transformation – the conversion of complex data from one format to another. Most current approaches require programming, scripting or use abstract visual specifications and are targeted to programmers, not business analysts or other end users. We describe a data transformation specification tool that uses a concrete visual metaphor based on the concept of copying data from one business form to another. We describe the visualisation of complex business data in a form that matches the cognitive needs of non-programmer business analysts and the specification of data transformations using our form copying metaphor. A prototype environment is described along with a cognitive dimensions evaluation of our end user visual language.

Keywords: end user computing, data integration, data transformation, business to business E-commerce, form-based data visualisation

1. Introduction

Organizations need to exchange data in order to support business-to-business (B2B) E-commerce [1, 6, 17]. For example, suppliers need to receive information about customers, product orders and invoices from customers. Very often supplier systems use a different format to represent this information than do their customers' systems and customer systems formats differ from each other. B2B data can be exchanged via distributed object APIs (CORBA, DCOM, .NET), EDI messages, XML documents, SOAP messages and custom data formats [17, 18, 23]. However, in this paper we ignore such transport-level issues and focus on the specification of the transformation needed to convert data from one business model to another. Quite complex data transformations are often required: source data model fields may be copied, split, merged and recalculated. Source data structure may be transformed into quite a different target structure (hierarchies flattened or built, collections merged, split or filtered and so on) [9].

Most current approaches to specification and implementation of B2B data transformations are programmer-centric. They include program-based transformations [21], script-based transformations using XSLT or similar languages [22, 19], or specifications that use abstract data representations such as tree or entity-relationship structures [10, 16]. None of these approaches are very suitable for non-programmer end users to use. However, most of the knowledge about what data means in one business and how it can be converted to another business's required data format is held by business analysts – almost always non-programmers [3].

We describe a prototype data transformation specification tool that is based on the most common business model and operation of data transformation: copying data from one business's form to another business's form, either in hard-copy or on a computer screen. This business form copying metaphor is realised by visualising complex data models as concrete representations mimicking the layout and composition of real hard-copy or electronic-copy business forms. A business analyst can modify automatically generated layouts to more accurately represent their hard copy form formats. This end user then connects fields and groups of fields on one form to fields and groups on another, thereby specifying how data is to be "copied" from the source data format to the target format.

We describe the motivation for this work in the following section using a typical B2B example. We survey related research on data transformation specification and then describe our approach which allows business analyst end users to specify such transformations. We illustrate the use of our prototype tool in visualising data models as business forms, allowing modification of form layout, and the specification of both simple and complex transformations

by form copying operations. We present an evaluation of our approach using the cognitive dimensions framework and summarise our contributions and possibilities for future research.

2. Background

There is an increasing demand for B2B electronic co-operation [1, 3, 6]. Despite great efforts to develop standards for representing business data and processes, most such "E-business" requires data transformation from one business's data format to another's [3, 6, 9, 10]. Business analysts are the people who are responsible for understanding and developing business processes and procedures. They have the knowledge of what one business's data structure and semantics mean and how this can be mapped onto another business's data. However, currently it is programmers who typically implement all business-to-business data exchange mechanisms. Figure 1 illustrates a typical example scenario. In this example, two businesses are exchanging various combinations of customer, product, order and invoice data.

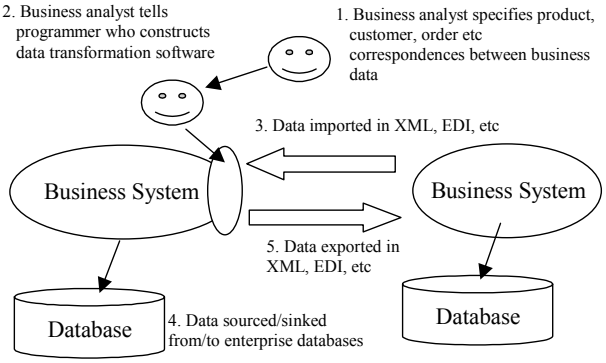


Figure 1. Data integration for E-business.

In today's business world, this process is inefficient and typically bottlenecks on the programmers – analysts continually want to have their business systems integrate with new systems or have new data integration and transformations supported, but these often take considerable time to describe to programmers and the software developed and tested. Communication with the programmers can also be problematic as they often don't fully understand the business context, meaning and use of data from both source and target systems.

Ideally business analysts themselves could specify data transformations and have required software generated. Programmers need to be involved developing particular technology interfaces for sourcing and sinking business data, but most new systems are architected to do this. What is required is flexible, end-user data transformation specification and generation support. Key requirements of such a system include:

- Support for importing and then visualising data formats from available business system meta-data.
- A representation for business data that is easy for business analysts to understand and use and which closely matches their own model of business data.
- A mechanism for specifying inter-business data correspondences that again matches analysts' current model of business data exchange. This should support complex data correspondence specification.
- Automatic generation of data transformation implementations with no end user involvement.

Current data transformation specification techniques are programmer-centric. Many systems use custom-coded transformation modules or components that take considerable design, implementation and testing by expert programmers [6, 21]. Many systems have moved to scripted solutions which support easier evolution of transformations [16, 10, 22]. Spreadsheet-style metaphors are used by some systems where target data is expressed in terms of source data formulae. These approaches don't leverage visual data transformation specification and don't generate transformation implementations that use common technology solutions. A common transformation script implementation solution is the XML Stylesheet Layout Transformation (XSLT) technology [4, 14, 19, 22]. A number of XSLT generators use visual data representation and transformation specification techniques [22, 10]. Many database and message passing systems also provide similar visual data transformation tools [11, 6]. These are almost always based on entity-relationship or tree hierarchy renderings of data. These are not a representation many business analysts use, especially when designing and documenting required business data exchange.

Programming-by-demonstration (PBD) systems utilise user interactions to deduce task specifications which are then automated or partially automated [5]. Many examples of such systems exist and have been applied to a wide range of problem domains. Examples include MetaMouse [13] supporting learned and repeatable CAD and word processing tasks, Masuishi's report generator [12] supporting production of reports from relational tables, and Sugiura's Internet Scrapbook [20] which automates the assembly of web pages. A key to the success of many PBD systems is the use of real-world metaphors by which users demonstrate actions and computer applications reflect learned operations to users.

3. Our Approach

We have developed an approach to providing business analysts with a data transformation specification tool by using a common concrete metaphor.

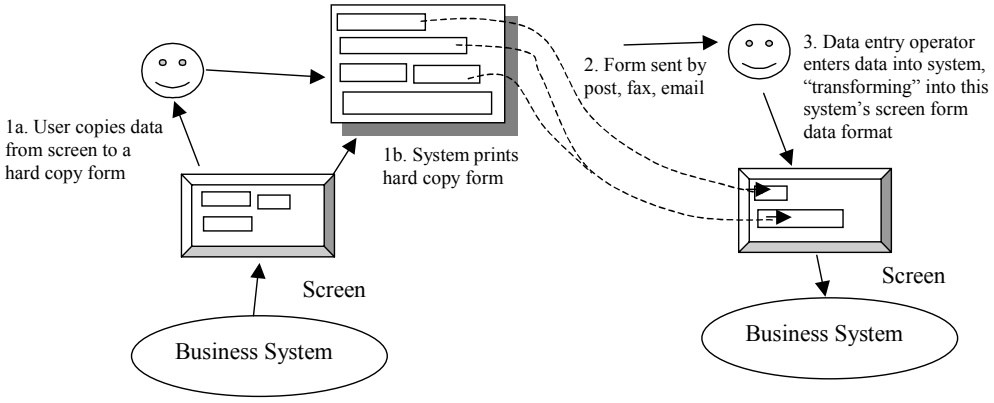


Figure 2. Business form copying metaphor.

How do businesses currently exchange data using non-computer means? They use business forms – one organization generates a form e.g. an order, and sends it to another (via post or fax, but also possible by email or other data communication technology e.g. HTML, EDI, CORBA or XML messaging). The receiving organization copies data from the form into its own form-based format, usually realised by a computer program screen. Usually this is done by data entry personnel (if the source business form is received in hard-copy form or email). If received electronically, data transformation programs written by programmers do this transformation. Figure 2 illustrates the hard-copy business form-based data exchange process.

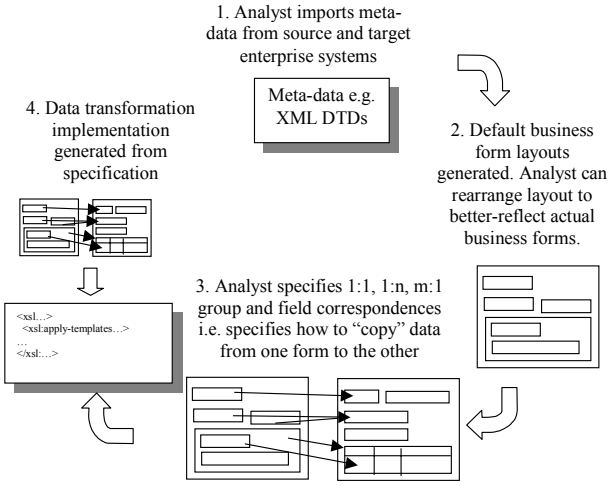


Figure 3. Form-based data integration process.

Our approach to supporting end-user data transformation specification is to adopt this "business form copying" metaphor, as outlined in Figure 3. Essentially business analysts demonstrate how data is to be copied from elements in a source form to elements in a target form, generating executable data transformations. Meta-data is extracted from enterprise systems (1) that describes source and target data models. A default "business form" visualisation is generated for each data model (2) which the business analyst can modify to better-conform to the physical business forms (whether in hard-copy or computer screens) they use. The analyst then specifies correspondences between fields and groups of fields on the source and target forms using direct manipulation (3). These correspondences and associated formulae are used to generate a data transformation implementation such as XSLT scripts, Java programs or 3rd party data mapping tool code (4).

4. Visualising Data as Business Forms

In this and the following section we illustrate our data transformation specification technique using a prototype tool and a business-to-business data exchange example. The business analyst first imports meta-data from a data source enterprise system and from a data target enterprise system. Such meta-data can be extracted from APIs (e.g. CORBA or COM APIs), XML DTDs or Schema, component meta-data, or system-specific means. Our prototype tool currently uses XML DTDs (Document Type Definitions), which specify the detailed structure of XML-encoded business data. In the example we use here, a business analyst is defining data mapping transformations to support one business exporting a customer order for books (Customer, Order, OrderLine and Book product data) to another system that needs the same information but represents it in quite a different way.

Firstly the analyst imports the structure of the encoded source and target system data, from either XML DTD files (specifying source and target document structure) or from example data encoded in XML files. Figure 4 (a) shows part of the source DTD file used in this example.

Note the business analyst doesn't look directly at this, programmers supply these or the data is extracted from an enterprise system's meta-data API.

```
<?xml encoding="UTF-8"?>

<!ELEMENT person
      (name,email*,url*,
  orders)>
<!ATTLIST person id ID #REQUIRED>

<!ELEMENT name ((family,given)>
<!ELEMENT family (#PCDATA)>
<!ELEMENT given (#PCDATA)>
<!ELEMENT email (#PCDATA)>

<!ELEMENT url EMPTY>
<!ATTLIST url href CDATA 'http://'>

<!ELEMENT orders (order+)>

<!ELEMENT order (date,item+)>
<!ELEMENT date (#PCDATA)>

<!ELEMENT item (book,qty,price)>

…
```

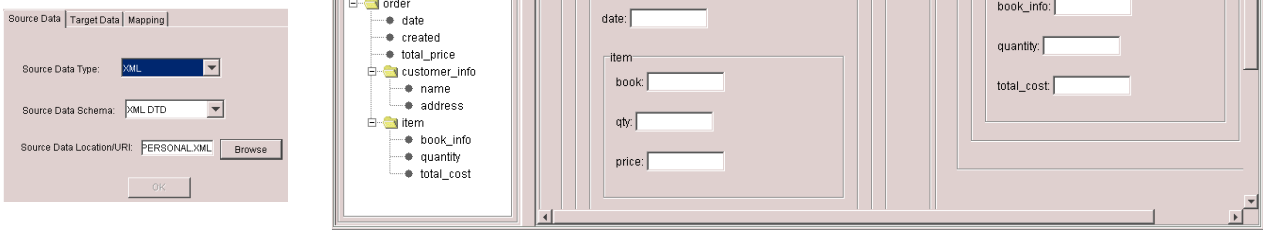


Figure 4. (a) XML data structure and (b) visualising imported meta-data as business forms.

Default business form visualisations of these source and target data models are generated. The form generation process uses information about the grouping of records in the data model to organise fields into groups. Repeating groups in the XML DTDs are realised as tables or multi-form groups. Field and group labels are drawn from the XML field tags. If extracting data from an XML Schema (richer than a DTD), field datatype and possible field size can be estimated. Figure 4 (b) shows the source and target structures as trees on the left, and on the right two default business form layouts generated by the tool.

In this example, the source is an XML message containing customer information, under which is grouped a number of order records and then book order items. The target is another XML message which is a list of order records, containing order information, customer information and book order item information. Some fields in the target require source fields to be merged, reformatted, recalculated, or split.

The default grouping of fields, field labels, field data types, field width, field alignment, and field placement may not match the layout the business analyst wants to work with. Similarly, the generated form will not have any annotations (lines, boxes, images) the actual hard copy or computer-based forms the analyst works with have. The analyst can interactively modify the generated form layout to move fields, re-label fields, change field data types, width and height, and re-group combinations of fields. This allows a closer approximation of the real business forms to be realised and for the analyst to use in the form field correspondence specification process.

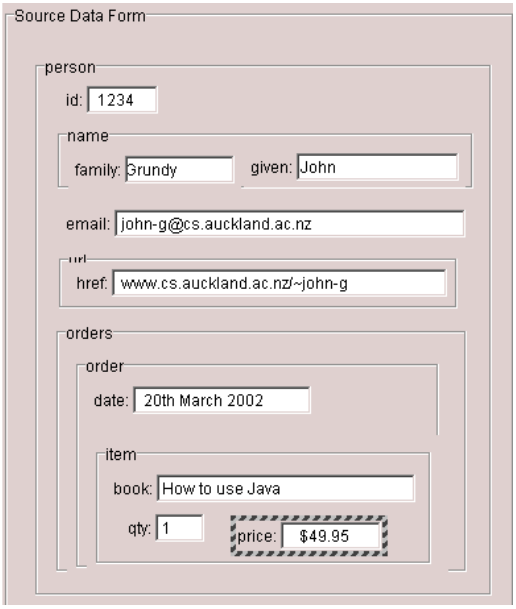


Figure 5. Rearranged source business form example.

Figure 5 shows some modifications made to the generated source business form visualisation by the analyst. These changes include moving fields (e.g. first and last name fields), resizing fields (e.g. email, last name), re-grouping fields e.g. id and resizing groups (orders, order items). In this example we have also asked the mapping tool to display example data from an XML data file in the fields of our business form visualisation. These can be used by analysts to help them determine appropriate field correspondences and formulae when specifying mappings.

5. Specifying Data Mappings

5.1. Demonstrating Mappings

We use a real-world programming-by-demonstration metaphor to support the specification of source and target data structure correspondences. The business analyst interactively drags and drops connections between one or more fields or field groups in the source business form to one or more fields or field groups in the target form to specify data transformation mappings. This mimics the "copying of data from one business form to another" approach to data integration utilised in many business systems and organizations. Data mappings range from simple one-to-one copying of field values and applying of formulae to field values to complex iteration over structures, filtering structure elements, and applying transformations to each item in the structure to produce a target data set. The complexities of such mappings are discussed in detail in [9]. Here we illustrate how our form-based mapping tool allows business analysts to specify such data mappings interactively using the form field copying visual metaphor.

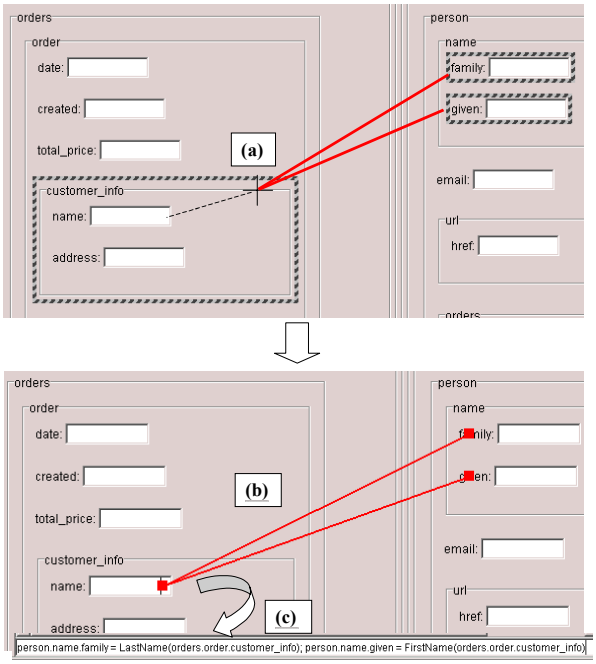


Figure 6. The form field copying metaphor.

Figure 6 shows how the field copying metaphor is realised. A business analyst identifies one or more source form fields or groups of fields that correspond to one or more target form fields or groups of fields. The analyst interactively selects source fields, dragging the mouse to corresponding target fields (a). Multiple source or target fields can be selected and linked, supporting one to many, many to one and many to many correspondence specification. Groups of fields can be linked, indicating sub-structure correspondences. Fields from one or more source form groups can be linked to fields in one or more different target form groups. Repeating fields or groups can be linked, with selection over source fields supporting filtering of data to target fields and groups. Once connected, a correspondence visualisation between the fields is displayed (b). In addition, the analyst can specify formulae that translate source field(s) into target field(s) values (c).

5.2. Specifying Simple Data Mappings

Initially a business analyst wanting to specify data mappings between the book order messages illustrated in Figure 4 will specify basic field copying and merging formulae, along with simple 1-to-1 group correspondences. Again, when doing this, the idea is that the analyst is demonstrating how to "copy" data in one (or more) fields from the source business form to a corresponding field or fields in the target business form.

The complexity of mappings that can exist form a continuum over several axes from extremely basic through to uncomputable. Examples of basic mappings

are one-to-one copies where field types, names, and data constraints are identical. Mappings of this form are not common and can be specified automatically. Of slightly greater complexity are mappings where field names or the base type differs. In many cases these types of mappings can be specified simply by equivalences between fields.

In Figure 7(a) examples of this can be seen in the mappings from source order item 'qty' and 'price' to the target item quantity and total_cost fields. With these fields the business analyst has simply selected the source field and dragged the mouse to the target field, indicating a one-to-one copy. In a similar manner the 'person' record (group of fields) has been connected to the 'customer_info' target record. This indicates that whenever a customer_info record and its constituent field values are required in the target data structure, its values are copied from the corresponding person fields in the source structure.

The types of fields in a one-to-one mapping can of course take differing forms and require some formulae to ensure a correct mapping. In Figure 7 (a) the mapping from 'date' shows such a type conversion requiring reformatting of the data type.

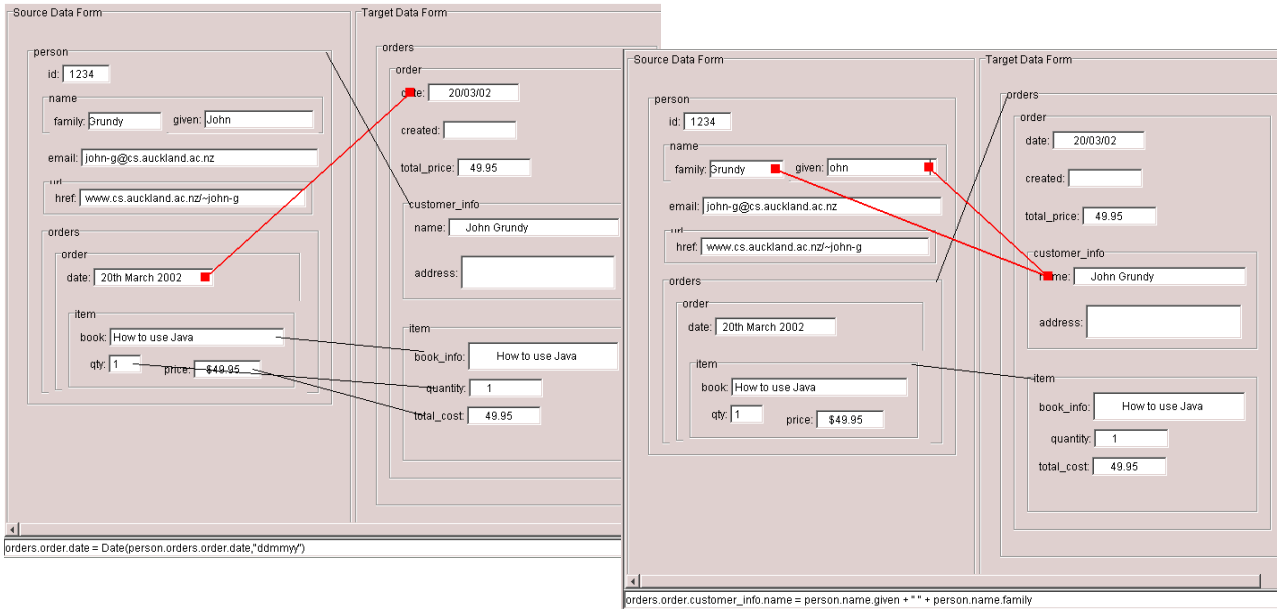


Figure 7. Some examples of (a) one-to-one field and group copying and (b) one-to-many and many-to-many copying.

The business analyst has connected the order date fields in the source and target business forms, but the target requires a different date format. A date conversion function is used to implement this type conversion. Such functions are provided by our tool to support data conversions and data extraction from single field values.

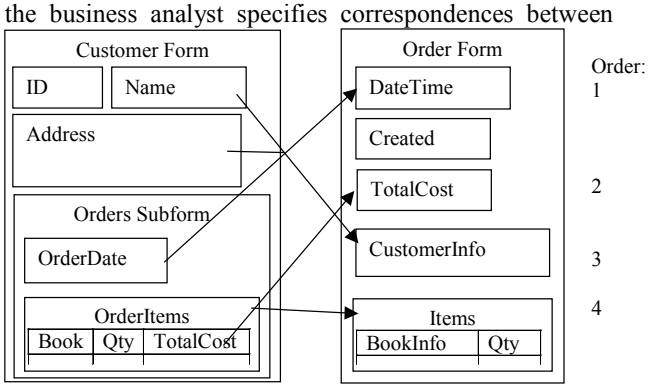
5.3. Complex Mappings

As a mapping moves from one-to-one to higher arities the complexity of the transform usually increases. Several fields may be required to determine the value of one field (m:1 mapping) as shown in Figure 7 (b) where the 'family' and 'given' fields of the 'name' entity are required to calculate the 'name' field in the target form. The business analyst has selected the two source fields and then dragged to the target field, creating a 2 to 1 linkage. A formula is provided by the analyst to indicate how the target value is arrived at with the two source field values as inputs. In this example, the two name fields are merged with a separator.

These m:1 mappings are usually of a simpler form that their inverse (1:n mapping). In the previous name example one can see that some form of pattern matching or parsing would be required to ensure the inverse mapping, and there are many simple 1:n mappings which can not be fully determined (e.g. from an area value to width and depth values).

These examples show field-based mappings, but there are many other complexities to consider. One common complex mapping form is that between a field and a structure or vice-versa. Further complexity is usually encountered when considering structure to structure mappings of various arities. When considering n:m mappings, of any combination of fields and structures, there are usually conditional constraints which help determine which objects take part in the mapping specifications.

In Figure 7 (b), an order record (group of fields) in the source form maps onto a whole target form (group). This implies a single customer with multiple orders in the source form will generate multiple target "forms" (target order data records). In this example, multiple source order items map onto the same number of target order items. In many domains, more complex structural mappings exist e.g. selection from source group to form target, m:1 or 1:m structural transformations, and so on. In our mapper the business analyst specifies correspondences between one or more source fields or groups and one or more target fields or groups. They then may need to qualify the correspondence indicating selection from source repeated groups by indexing or filtering. They may need to qualify the target by specifying collection indexing or construction. In our tool the analyst currently makes use of provided functions acting on source field(s) or group(s) to produce result values for target field(s) or group(s).




```
<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>
…
```

Figure 8. Examples of (a) form element correspondences and (b) generated XSLT transformation script.

Mappings which are complex to describe are often associated with differing notions of identity. Where a source schema has differing criteria for an object's uniqueness versus that found in a target schema then the specification of the required mapping often becomes cumbersome. For example, in CAD (Computer-Aided Design) systems point objects are used to identify geometric parameters of higher-order objects. In some systems the combination of an object and a point is the key, whereas in other systems a point's location is itself the key. This means that some CAD systems can have multiple point objects existing at the same co-ordinates and others may not. Describing a mapping between point objects of these differing systems is currently difficult. While the mapping between point objects is simple to describe, the conditions under which a point is created is not as easy to describe with a visual notation.

6. Design and Implementation

We have built our proof-of-concept form-based data mapping tool using Java's Swing GUI API and JAX XML parsing API. Our tool allows users to import meta-data from XML-encoded data files or from XML Document Type Definition files. It is expected that these would be provided via Enterprise system meta-data APIs or message APIs in a fully realised data mapping system. This meta-data is used to generate a basic form layout with simple heuristics used to generate form elements and element groupings using Java Swing components. These include generating labels and text fields for simple-valued leaf nodes in the XML DOM to generating form groups for complex XML record nodes. The user can move components within and between form groups and can add or delete groups to modify the form layout. The use of Swing layout managers constrains the degree of form layout that can be achieved in our prototype tool, but use of Swing allowed us to build and maintain a prototype tool with minimum effort. Layout support could be augmented in the future with absolute-position layout if necessary.

Analysts link fields using drag-and-drop between form components. We used the Java Swing component event-passing mechanism and transparent panel overlays to intercept user interaction with the generated form components, to implement drag-and-drop between native Swing form components. We use the form component setText() etc functions to put data from XML data files into displayed form components to support mapping-by-example data for users.

The algorithm for generating data transformation implementations traverses the target form data structure, constructing translation code based on the links into the target form elements. This is illustrated in Figure 8 (a). We used this approach as we have adopted an XSLT-based transformation script as the data mapping implementation for the prototype. This is stream-based and needs to build the target data set (an XML data file) using in-order traversal. Part of a generated XSLT

transformation script resulting from this process is shown in Figure 8 (b). The first part of this script extracts the OrderDate from the source, then generates a creation date/time for the target order. It then sums the total cost for the source order items to produce a target order total cost. Customer information is extracted from the source, and then for each source order item record a corresponding target item record generated.

7. Discussion

The cognitive dimensions framework [7] provides a way to assess a wide variety of visual language properties. We have carried out a cognitive dimensions assessment of a visual form-based mapping tool. We summarise our results below.

Abstraction Gradient. The key abstraction used in our system is the business form, a concrete visual metaphor comprising primitive form elements (labels, text fields, check boxes, etc) and groups of primitives. These abstractions map onto meta-data elements, though the user can create further abstraction groups if required. Links between fields represent formulae converting source data item(s) and group(s) to target data item(s) and group(s).

Closeness of mapping. Our form-based data transformation tool uses a concrete metaphor – the business form – to support data mapping specification. Its visual representation thus maps directly onto business analyst's (the end users) cognitive model of their problem domain. The purpose of allowing generated form layout modification is to support even closer mapping allowing analysts to tailor the generated layout to be closer to the actual screen and hard-copy business form layouts they are familiar with.

Consistency. Both source and target form representations use the same visual form elements. All inter-form element links are rendered the same way. The latter presents a potential problem in that discriminating between simple and complex mappings may be desirable.

Diffuseness/Terseness. Compared to more abstract approaches to representing data transformation, our form-based data mapping tool employs a more verbose visual language that can include elements not directly used in the mapping process e.g. business form layout groups, labels, lines and boxes and images. In contrast, mapping specifications using meta-data renderings such as trees and entity-relationship diagrams seldom include elements not directly used in the meta-data mapping specification. The use of a concrete form-based metaphor in our approach necessitates a less terse notation to support the desired visual metaphor.

Hard Mental Operations. For end users, hard mental operations are greatly reduced by having a visualisation close to their cognitive model of inter-business data exchange.

Hidden Dependencies. All inter-form dependencies are explicitly represented as form element and group links. Within forms, element groupings are all explicitly represented. The mapping formulae associated with inter-form links hide the detail of dependencies between the source and target fields.

Role-Expressivenes. Concrete representations are used for all form elements that denote their role. Enclosure of elements by groups provides an additional role specification, that of the elements' relationship to others in the group.

Secondary Notation and Escape from Formalism. The business analyst can reorganise automatically generated form layouts, creating their own cognitively meaningful business form representation using form element layout, appearance and grouping. Our tool supports the use of this secondary notation relating to form element layout as it is cognitively important to the user and has meaning in terms of the grouping of form elements. The form layout and appearance has no effect on generated mapping code but regrouping or retyping form elements does. No unstructured form annotations are currently supported though this may be a useful addition allowing end users to make notes against forms and form element links.

Visibility and Juxtaposability. The form-based mapper has explicit inter-form element links providing good visibility, but the links between form elements and element groups to the underlying meta-model is hidden. When the user modifies form layout e.g. by adding or rearranging grouping, this linkage is blurred and is not visible in the visual form-based visualisation nor tree-based structure views. Two views are supported in our tool: a concrete form-based visualisation and tree-based structure visualisation, which are viewed side-by-side. Sub-views are currently supported using the tree-based view to select a portion of the form for display, but multiple views displayed simultaneously are not currently supported

8. Future Work

We are planning an empirical assessment of our tool using several experienced business analysts and business-to-business data transformation implementers. This will assess the suitability of our proposed approach for both the target end user community and programmers involved in the implementation of data exchange solutions. We plan to extend our mapping tool to support more complex structural mappings and their visualisation , to assist users in understanding many-to-many repeating field mappings. As we utilise a business form paradigm there may already exist a mapping from the underlying data representation through to the presented form. We will be considering the impact these existing mappings have in terms of the business mappings that can be supported. Semi-automated field copying, using example data elements in the source

and target fields to identify same-valued copies, will be added together with a spreadsheet language to replace the formulae used to express field-level value copying, splitting and merging. This will provide analysts with a familiar visual metaphor to express these (sometimes complex) formulae. We are also planning to experiment with extracting field parsing formulae from a query-by-example interface, allowing field splitting 1-to-many mappings to be deduced from these example data manipulations. We are planning to implement code generators for Java-based, Rimu-based [9] and other data mapping technology implementations from our form-based mapping specifications. We also plan to investigate the use of MS Access™ and Internet Explorer™-implemented forms and scanned business forms to provide concrete form layout for connecting. This will allow analysts to import form data from these "concrete" form implementation technologies and specify data mappings using these actual layouts, linking concrete form elements to imported meta-data elements.

9. Summary

Business analysts are the domain experts in business-to-business data mapping domains. They know the meaning of complex data sets to each business and the required data transformations (or "copying") required to support inter-business data exchange. We have developed a proof-of-concept data mapping prototype that uses a concrete "business form copying" metaphor to support data transformation specification. Business analysts, who are not programmers, import meta-data into our tool, which constructs user-modifiable business form layout from this data. Users connect form fields to specify how data is "copied from one business's form to another's". This data copying-based specification is used to generate, with no user intervention, a complex data mapping transformation implementation.

References

1. Aleksy M, Schader M, Tapper C. Interoperability and interchangeability of middleware components in a three-tier CORBA-environment-state of the art. Proceedings Third International Enterprise Distributed Object Computing. Conference, IEEE CS Press. 1999, pp.204-13. Piscataway, NJ, USA.
2. Alonso G, Fiedler U, Hagen C, Lazcano A, Schuldt H, Weiler N. WISE: business to business e-commerce. Proceedings Ninth International Workshop on Research Issues on Data Engineering: Information Technology for Virtual Enterprises. RIDE-VE'99. IEEE CS Press, 1999, pp.132-9. Los Alamitos, CA, USA.
3. Blackham, J., Grundeman, P., Grundy, J.C., Hosking, J.G. and Mugridge, W.B., Supporting Pervasive Business via Virtual Database Aggregation, In Proceedings of Evolve'2001 – Pervasive Business, Sydney, Australia, March 15-16 2001, DSTC Press.
4. Cheung, D., Lee, S.D., Lee, T., Song, W., Tan, C.J. Distributed and scalable XML document processing architecture for E-commerce systems. In *Proceedings of the Second International Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems*. IEEE, 2000, pp.152-157.
5. Cypher, A, (ed) *Watch what I do: Programming by demonstration*, Cambridge, Mass, MIT Press.
6. eXcelon Corp, eXcelon B2B Integration Server White Paper, www.exceloncorp.com.

7. Goulde, M.A. Microsoft's BizTalk Framework adds messaging to XML. *E-Business Strategies & Solutions*, Sept. 1999, pp.10-14.
8. Green, T.R.G and Petre, M, Usability analysis of visual programming environments: a 'cognitive dimensions' framework, *Journal of Visual Languages and Computing* 1996 (7), pp.131-174.
9. Grundy, J.C., Mugridge, W.B., Hosking, J.G. and Kendall, P. Generating EDI Message Translations from Visual Specifications, In Proceedings of the 2001 IEEE Automated Software Engineering Conference, San Diego, CA, 26-28 Nov 2001, IEEE CS Press.
10. Huemer, C. and Tjoa, A.M. Meta Messages in Electronic Data Interchange (EDI), In *Proceedings of the Third IEEE meta-data conference*, April 1999.
11. IBM Corp, *MQ Series Integrator*, www.ibm.com.
12. Masuishi, T., Takahashi, N., A Reporting Tool Using Programming by Example for Format Designation, in Liebermann H (ed) *Your Wish is my Command*, Morgan Kaufman, 2000.
13. Maulsby D, and Witten, I H, MetaMouse: an instructable agent for programming by demonstration, in Cypher A (ed) Watch what I do, MIT Press, 1993.
14. Morgenthal, J.P. XML: The New Integration Frontier, *EAI Journal*, Feb. 2001, www.eaijournal.com.
15. OnDisplay Corp, www.ondisplay.com.
16. Seeburger Corp, www.seeburger.de/xml/.
17. Senn JA. The evolution of business-to-business commerce models: the influence of new information technology models. Proceedings of International Workshop on Advance Issues of E-Commerce and Web-Based Information Systems. IEEE CS Press, 1999, pp.153-8. Piscataway, NJ, USA.
18. Sessions, R. COM and DCOM: Microsoft's vision for distributed objects, John Wiley & Sons 1998.
19. Spencer, H. XML standards for data interchange. Imaging & Document Solutions, vol.9, no.9, Sept. 2000, pp.15-17.
20. Sugiura, A, Web Browsing by Example, in Liebermann H (ed) *Your Wish is my Command*, Morgan Kaufman, 2000.
21. Swatman, P.M.C., Swatman, P.A., Fowler, D.C. A model of EDI integration and strategic business reengineering. *Journal of Strategic Information Systems*, vol.3, no.1, March, 1994, pp.41-60.
22. XML.org, *XML and XSLT*, www.xml.org.
23. Wu, E. A CORBA-based architecture for integrating distributed and heterogeneous databases. Proceedings Fifth IEEE International Conference on Engineering of Complex Computer Systems, IEEE CS Press, 1999, pp.143-52. Los Alamitos, CA, USA.