Supporting information mapping in Health Informatics via integrated message transformation

Mugridge, W.[a], Grundy, J.[a], Hosking, J.[a] Kendall, P.[d]

[a] *Dept of Computer Science, University of Auckland, Auckland, New Zealand*
[b] *Orion Systems Ltd, Mt Eden, Auckland, New Zealand*

Abstract

In order to provide an effective overall health information system, a number of separate systems used by different providers must typically be integrated. However supporting the exchange of data between these disparate health information systems often requires complex data transformation from one system's data formats to another's. We describe a novel data mapping specification tool, domain-specific language and mapping engine that greatly simplifies building such integration infrastructure. Our system allows health systems integrators to specify correspondences between information messages generated by one system to messages that another system consumes. A special mapping language is used to express these correspondences and is run by a mapping engine to effect data transformation. Input and output messages can be expressed in XML or EDI formats and a separate message exchange system is used to communicate between the data source and data target health information systems. We describe our mapping system approach, key elements of its architecture and experiences in commercializing our basic research to produce a successful new product.

Keywords:

Systems integration, Electronic Data Interchange, XML, messaging systems

Introduction

Health information systems increasingly need to be integrated in order to maximize patient information availability, reduce redundancy and inaccuracies in information shared between systems and to facilitate timely action by medical and support staff. For example, a treatment provider's Information System might describe a patient, hospital visit information, patient treatment and treatment costs. A health insurer or funding organisation requires this information to record the treatments, costs and reimbursements. However, typically these systems use different ways of representing similar information, due to various design choices made during the systems' construction. Each system typically provides different data formats to encode their information, and in order to integrate the systems, one format must be mapped (translated) into another.

One of the most common ways of supporting system integration in the health industry is the use of messaging, where information from one system is packaged into a stand-alone message and transmitted to another system. Many health information systems use Electronic Data Interchange (EDI) messaging (Huemer and Tjoal 1999; Emmelhainz 1990; McLure and Moynihan, 1995). More recently, eXtensible Markup Language (XML) documents and web services (Cheung et al, 2000; Estes 2001) have been used to represent information, providing more generalised and easier to implement and extend messaging technologies than EDI solutions (Liou et al, 2000; Wallin, 1999; Sokolowski, 1999). However, many message-based systems use different sets of EDI and XML message formats. In order to support message-based information exchange between systems using a different message dialect, message transformation must take place (Spencer, 2000; Morgenthal, 2001; Lincoln et al, 1999). For example, a health provider must supply an insurer/funder system with its expected message format, or the funder must translate the provider message(s) into its own message-based protocol. Similarly, data sent back to the provider from the funder must be appropriately converted. Often these messages are very large and translation between them requires complex algorithms and code (Grundy et al, 2001). Current approaches to supporting message-based system integration require much programming using conventional implementation languages, or utilise proprietary tools or over-general transformation scripting approaches.

We describe a new approach that uses a domain-specific data transformation language, visual specification environment and a data mapping engine to support complex message-based health information system integration. We firstly present a motivation for this work, focusing on patient treatment data exchange. We then outline our message mapping system and how it allows health systems integrators to much more easily specify data transformations between complex EDI and XML messages. A special purpose data mapping language implements these transformation specifications and a mapping engine runs them efficiently and in conjunction with a message scheduling system. We report on our experiences developing a commercial message mapping product, Symphonia Message Mapper™ from basic research in this problem domain.

Motivation

Consider the problem of integrating multiple health information systems such as those of a provider (hospital, GP, pathlab etc) and funding agency (health insurer or government funding body). Figure 1 illustrates this problem domain. The provider system needs to communicate information to the insurer. The provider extracts data from its database and formats one (or more) messages that describe the data it wants to exchange with the insurer. The provider then encodes this message into a form that can be transmitted to the insurer over a computer network. The insurer system receives the message(s) and extracts the data it requires. It may then update its own database and may send one or more response messages to the provider (for example, if the provider issues a request for data from the insurer).

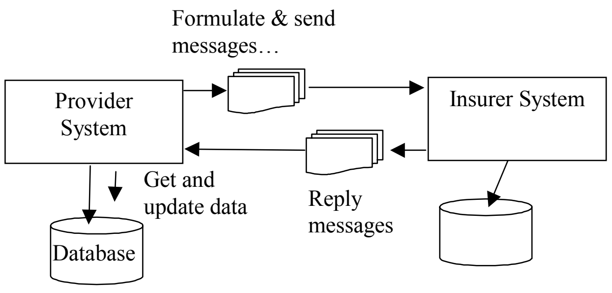


Figure 1. Integration example.

Very often the systems that need to exchange health information use incompatible formats for this data (Liou et al, 2000; McLure and Moynihan, 1995; Sokolowski, 1999). This situation arises due to systems being designed with quite different data exchange requirements; due to different design decisions by the system developers; or due to the adoption of competing standards for encoding information for exchange. Integrating health informatics systems often means supporting complex data transformation from one system's message formats to another set of formats. This transformation can be very complex.

Several current approaches exist to handling this message-based system integration problem. Most EDI and many XML-based messaging technologies provide a set of computer program function libraries that programmers use to encode and decode messages for particular EDI and XML message protocols (Wallin, 1999; Lincoln et al, 1999). Developers use these functions to read a message in the source system format, then write program code to construct a new message in the target system format, and then use a function to generate the physical target system message. Programmers thus implement all message mappings manually which is time consuming, error-prone and hard to maintain (Grundy et al, 2001). Some message mapping systems have been developed (Aditel Corp, 2001), but these typically suffer from using a low-level representation of mappings that can not handle complex transformations. Some Message-Oriented Middleware systems provide message integration tools, such as MQ Integrator™ (IBM Corp, 2001). These provide limited abstract message translation facilities, very often requiring low-level programming of translations. XML-based message encoding and message translators have been produced (Spencer, 2000; XML.org 2001; Cheung et al, 2000). These include XSLT, Seeburger's data format and business logic converter (Seeburger, 2001) and eBizExchange (OnDisplay Corp, 2001). These systems typically use XSLT (XML Style Sheet-based Translations) which suffers from a lack of expressive power and modularity (especially for complex hierarchical mappings) and most tools only partially support visual mapping and XSLT script generation. Some Enterprise Application Integration products, such as Vitria BusinessWare™ (Vitria Technology Inc, 2001), BizTalk™ (Goulde, 1999) and the Universal Translation Suite (Data Junction Corp, 2001) provide message translation support for database, message and XML-encoded data. However most of these solutions are limited to simple record structures and are relatively difficult to use.

Health Data Mapping Requirements

To better support complex message-based health information systems integration, Orion Systems Ltd wanted to develop a better approach to EDI and XML message mapping. Orion had developed the Symphonia™

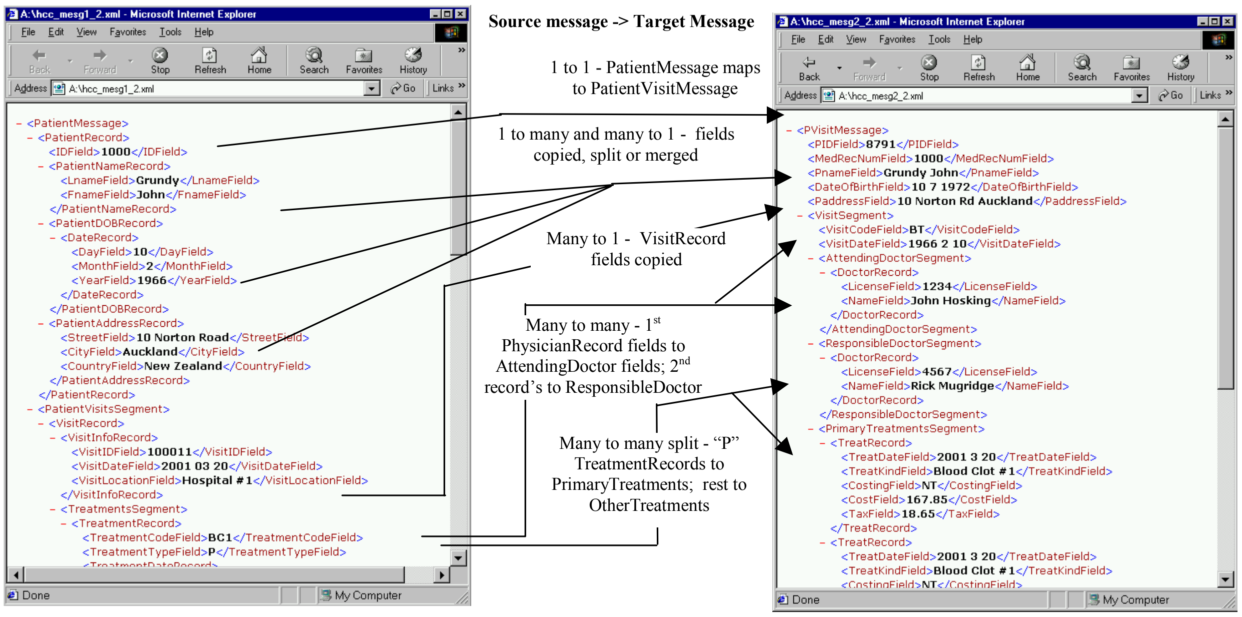
suite of message encoding and decoding function libraries, and wanted to develop a message mapping system to make it easier for health systems integrators to implement translations of messages from one system's formats to another's.

Message mapping is not a simple problem. To illustrate the nature of the problem, Figure 2 shows two example messages representing health informatics data (shown in XML message formats in the IE 5.5 web browser). The left message encodes patient treatment information using a "deep" structural hierarchy (Patient->Visits->Treatments). The right message encodes (mostly) the same data, but uses a flatter format. To translate the messages we need to apply a variety of field-, record-, segment- and record collection-level translations between these two representations of the patient treatment data.

To translate the right into the left, we apply mappings to convert the flat structure into the deeper hierarchical one. Several fields and collections must be merged or split e.g. the patient name, dates and address merged. A number of formulae, some dependent on source message content, need to be applied e.g. the treatment cost and treatment units. Some structures in the messages repeat, such as the list of treatments required, and these may be organized in quite different ways e.g. a list of Treatments in the left message is grouped into Primary and Other treatments in the right message.

This example transformation problem is reasonably typical of many of the EDI and XML mappings we have encountered. Most EDI and XML messages are much larger however, often with hundreds of segments, records and fields. This means developers need high-level support for expressing and managing their message mappings. A message mapping system should ideally:

- Allow developers to extract message format information from existing system's meta-data, such as EDI message encodings and XML Document Type Definitions.
- Allow developers to visualize message structures, which are typically predominantly hierarchical in nature.
- Support the specification of correspondences between elements in the source message and elements in the target message. As illustrated, these can be 1 to 1, 1 to many, many to 1 or many to many relationships.
- Allow developers to use expressions and control constructs specific to this message mapping domain i.e. provide higher-level constructs such as collection selection, iteration, filtering, and so on that traditional computer programming languages don't support directly.
- Compile message mapping specifications to a form that can be very efficiently run to map potentially thousands of EDI and XML messages a minute for large health informatics system deployment.


A:\hcc_mesg1_2.xml - Microsoft Internet Explorer
Source message -> Target Message
1 to 1 - PatientMessage maps to PatientVisitMessage
1 to many and many to 1 - fields copied, split or merged
Many to 1 - VisitRecord fields copied
Many to many - 1st PhysicianRecord fields to AttendingDoctor fields; 2nd record's to ResponsibleDoctor
Many to many split - "P" TreatmentRecords to PrimaryTreatments; rest to OtherTreatments
A:\hcc_mesg2_2.xml - Microsoft Internet Explorer

```
- <PatientMessage>
  - <PatientRecord>
      <IDField>1000</IDField>
    - <PatientNameRecord>
        <LnameField>Grundy</LnameField>
        <FnameField>John</FnameField>
      </PatientNameRecord>
    - <PatientDOBRecord>
      - <DateRecord>
          <DayField>10</DayField>
          <MonthField>2</MonthField>
          <YearField>1966</YearField>
        </DateRecord>
      </PatientDOBRecord>
    - <PatientAddressRecord>
        <StreetField>10 Norton Road</StreetField>
        <CityField>Auckland</CityField>
        <CountryField>New Zealand</CountryField>
      </PatientAddressRecord>
    </PatientRecord>
  - <PatientVisitsSegment>
    - <VisitRecord>
      - <VisitInfoRecord>
          <VisitIDField>100011</VisitIDField>
          <VisitDateField>2001 03 20</VisitDateField>
          <VisitLocationField>Hospital #1</VisitLocationField>
        </VisitInfoRecord>
      - <TreatmentsSegment>
        - <TreatmentRecord>
            <TreatmentCodeField>BC1</TreatmentCodeField>
            <TreatmentTypeField>P</TreatmentTypeField>
```

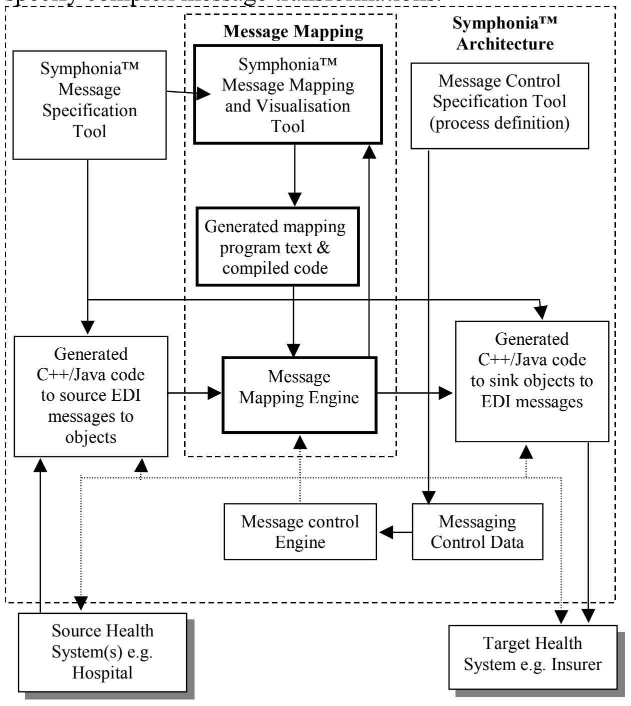
```
- <PVisitMessage>
    <PIDField>8791</PIDField>
    <MedRecNumField>1000</MedRecNumField>
    <PnameField>Grundy John</PnameField>
    <DateOfBirthField>10 7 1972</DateOfBirthField>
    <PaddressField>10 Norton Rd Auckland</PaddressField>
  - <VisitSegment>
      <VisitCodeField>BT</VisitCodeField>
      <VisitDateField>1966 2 10</VisitDateField>
    - <AttendingDoctorSegment>
      - <DoctorRecord>
          <LicenseField>1234</LicenseField>
          <NameField>John Hosking</NameField>
        </DoctorRecord>
      </AttendingDoctorSegment>
    - <ResponsibleDoctorSegment>
      - <DoctorRecord>
          <LicenseField>4567</LicenseField>
          <NameField>Rick Mugridge</NameField>
        </DoctorRecord>
      </ResponsibleDoctorSegment>
    - <PrimaryTreatmentsSegment>
      - <TreatRecord>
          <TreatDateField>2001 3 20</TreatDateField>
          <TreatKindField>Blood Clot #1</TreatKindField>
          <CostingField>NT</CostingField>
          <CostField>167.85</CostField>
          <TaxField>18.65</TaxField>
        </TreatRecord>
      - <TreatRecord>
          <TreatDateField>2001 3 20</TreatDateField>
          <TreatKindField>Blood Clot #1</TreatKindField>
```

Figure 2. An example of two health message formats to be mapped between.

- Provide an integrated solution: a development environment supporting visualization of message structures and expressions using a domain-specific programming language tailored to the message mapping domain and being run by an efficient message mapping engine.

Overview of Our Approach

We have developed an integrated solution to the message mapping problem outlined in previous sections. Figure 3 illustrates the key parts of this system. The Symphonia™ message specification tools are used to specify EDI and XML message formats. These generate Java and C++ code to read and write encodings of such messages for communication over networks. They also generate message format meta-data. A visual message mapper tool reads these format specifications and allows developers to specify complex message transformations.



Message Mapping
Symphonia™ Architecture
Symphonia™ Message Specification Tool
Symphonia™ Message Mapping and Visualisation Tool
Message Control Specification Tool (process definition)
Generated mapping program text & compiled code
Generated C++/Java code to source EDI messages to objects
Message Mapping Engine
Generated C++/Java code to sink objects to EDI messages
Message control Engine
Messaging Control Data
Source Health System(s) e.g. Hospital
Target Health System e.g. Insurer

Figure 3. Orion Symphonia™ System Architecture.

This tool generates code expressed in a special-purpose, "domain-specific" computer programming language. This is an executable form of the message mapping specifications that is compiled and is run by a mapping engine. The Symphonia™ message co-ordination tool allows developers to specify when and how messages are sourced from one system, given to the mapping engine for transformation and sent to a target system. Without such tools very large programs have to be hand-coded to map complex health system messages.

In the following sections we briefly overview the visual message mapping tool, our special purpose mapping language and the operation of our mapping engine, to illustrate how these support complex message transformation implementation.

Visual Mapping Tool

The purpose of our visual mapping tool is to provide an environment for specifying inter-message mappings using a visual language tailored for this task. Message structures are visualized using a hierarchical representation of the source and target messages. A third hierarchical representation represents the correspondences between source and target message elements. These provide the mapping specifications that are run to translate between messages.

Figure 4 shows some example mapping specifications for the patient treatment example. Diagram (1) shows a representation of some of the patient record field mappings. On the left is the source hierarchical PatientRecord message structure, and on the right the target flatter PvisitMessage. Correspondences between source and target message elements are expressed in the oval mapping nodes in the center. Nodes labeled <-> are bi-directional field copying, -> are unidirectional formulae. Nodes labeled map are mapping functions. Diagram (2) shows a mapping function called to select "primary" treatment records from the PatientTreatments record collection, this function being used by the highlighted map node in diagram (3). Diagram (4) shows examples of conditional mappings, where source record content is used to determine the mapping formulae used.

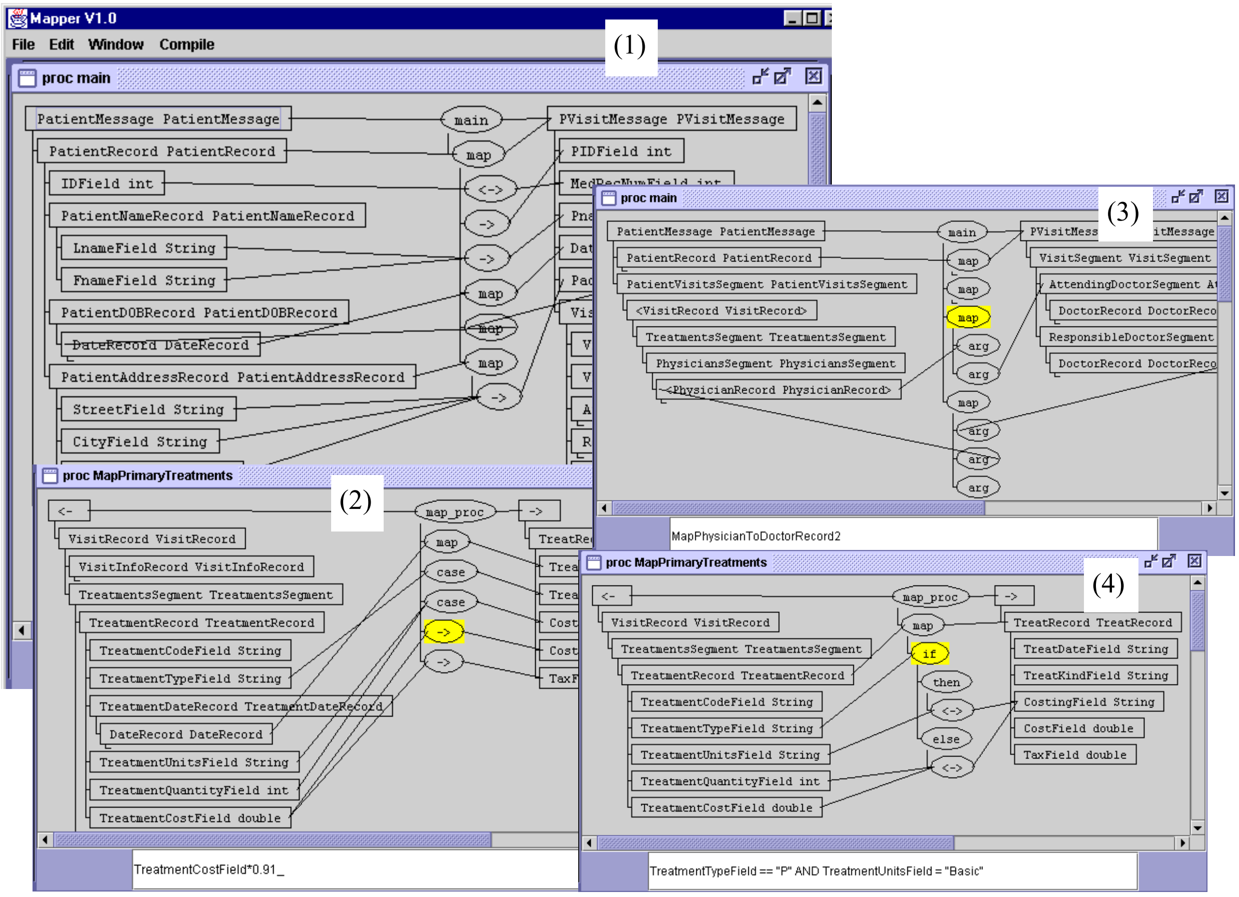
Domain-Specific Mapping Language

The visual mapping tool generates an executable message transformation specification expressed in a special-purpose textual mapping language. We developed this language to provide high-level constructs to assist the expression of message mappings that conventional programming languages are unsuited to. It includes special-purpose constructs such as declarative collection iteration, selection and construction.

Figure 5 shows part of the generated mapping code describing how to translate PatientMessage data to PVisitMessage data. The message formats are declared as types at the beginning of the mapping specification. Note that these types can be encoded using an XML document or various EDI messages or even comma-separated value file, Excel™ spreadsheet data or database table data. At run-time the particular encoding mechanism is associated with the mapping specification by our message mapping engine. Mapping constructs include bi-directional copying (<->), unidirectional formulae (->), conditional and guarded execution (if and case), and map functions (map). Map functions can take collections as arguments and construct and return collections as results, using functional language execution.

Mapping Engine

Our message mapping engine uses compiled byte code from the special purpose textual mapping language to automate the transformation of EDI and XML messages. Figure 6 illustrates the basic process of message transformation. A source message from a health information system is given to the mapper by the Symphonia™ message controller. This is decoded into a source message data structure by code generated by the Symphonia™ message designer (1). The Symphonia™ message controller then requests that the mapping engine apply the transformation from source to target message (2). The mapping engine runs the compiled mapping specification hierarchically, running each mapping function and then each of its constituent mapping constructs and functions in turn (3). Note that the source message records and fields can be read in any order by the mapping specification, and the target message can similarly be constructed in any order, its values put into a target message data structure (4). When the mapping process is complete, a physical target message is constructed from target data structure, by generated encoding classes (5). The controller then passes this message to the target health information system.


Mapper V1.0
File
Edit
Window
Compile
(1)
proc main
PatientMessage PatientMessage
PatientRecord PatientRecord
IDField int
PatientNameRecord PatientNameRecord
LnameField String
FnameField String
PatientDOBRecord PatientDOBRecord
DateRecord DateRecord
PatientAddressRecord PatientAddressRecord
StreetField String
CityField String
main
map
<->
->
map
PVisitMessage PVisitMessage
PIDField int
MedRecNumField int
(2)
proc MapPrimaryTreatments
<-
map_proc
->
VisitRecord VisitRecord
VisitInfoRecord VisitInfoRecord
TreatmentsSegment TreatmentsSegment
TreatmentRecord TreatmentRecord
TreatmentCodeField String
TreatmentTypeField String
TreatmentDateRecord TreatmentDateRecord
DateRecord DateRecord
TreatmentUnitsField String
TreatmentQuantityField int
TreatmentCostField double
case
(3)
proc main
PatientMessage PatientMessage
PatientRecord PatientRecord
PatientVisitsSegment PatientVisitsSegment
<VisitRecord VisitRecord>
TreatmentsSegment TreatmentsSegment
PhysiciansSegment PhysiciansSegment
<PhysicianRecord PhysicianRecord>
arg
VisitSegment VisitSegment
AttendingDoctorSegment
DoctorRecord DoctorRecord
ResponsibleDoctorSegment
(4)
if
then
else
TreatRecord TreatRecord
TreatDateField String
TreatKindField String
CostingField String
CostField double
TaxField double

MapPhysicianToDoctorRecord2

TreatmentCostField*0.91_

TreatmentTypeField == "P" AND TreatmentUnitsField = "Basic"

Figure 4. Some message mapping examples from our visual mapping specification prototype tool.

Implementation and Future Research

We built a proof-of-concept textual mapping language compiler, mapping engine and visual mapping specification environment to determine the approach described in this paper would be feasible. This demonstrated that complex message mapping specifications could be substantially visually designed and that our special-purpose mapping language provided appropriate high-level structural transformation constructs. We carried out two evaluations of this prototype: one assessing the performance of the mapping engine for through-put of message transformations, the other assessing the usefulness of the visual specification techniques used (Grundy et al, 2001).

Orion Systems then developed a commercial version of the message mapping system. This was implemented using similar interface and architectural approaches as the rest of the Symphonia™ product suite to ensure good integration between these products. A number of complex message transformations have been specified and generated using the Symphonia Message Mapper™, and the system has been deployed by several major health systems providers to facilitate message-based systems integration.

```
type T_Patient = struct {
  int PK_PatientID;
  T_PatientName PatuentName;
  optional String MedicalRecordNumber;
  optional String MothersMaidenName;
  String DateOfBirth;
  String PSex;
};

type T_PatientName = struct {
  int FK_PatientID;
  String FirstName;
  String LastName;
};

…
type PVisitMessage = struct {
  int ExternalID;
  String InternalID;
  String PName;
  String Sex;
  int DOB;
  String MMaidenName;
};

…

map main(<-PatientMessage pm,
         -> PvisitMessage[] pvs) {
  pvs[0].ExternalID <-> pm.PK_PatientID;
  MapPatientName(pm.PatientName,pvs[0].Pname);
  …
  mapPrimaryVisits(select(i from
   pm.PatientVisitSegment[*] where
          i.VisitRecord.VisitType = "P"),
       pvs[0].PrimaryVisitsSegment);
  …
}
map mapPatientName(<-T_PatientName name,
             -> String pname) {
       pname <- name.LastName+", "+name.FirstName;
}
map  mapPrimaryVisits  (<-T_PatientVisit  pv,  ->PVisit
pvs) {
  …
}
```

Figure 5. Mapping Language Examples.

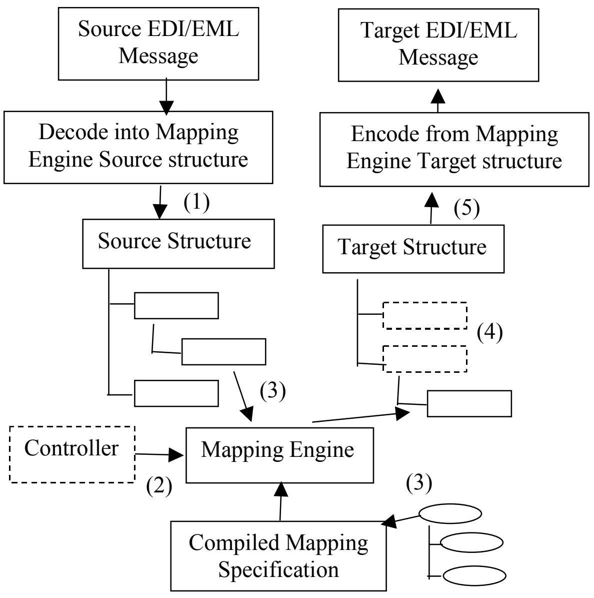


Figure 6. Message mapping engine processing.

A number of extensions to this message mapping approach and toolset are possible. We are working on a more concrete visualization metaphor to allow non-programmers to effectively specify complex message mappings using form-based representations of source and target messages and a drag-and-drop, "form copying" metaphor to specify element correspondences. This aims to make message mapping specification and generation feasible for non-programmer system integrators in domains where volatile message transformation specifications are needed. The mapping system was designed to support transformations between transient message objects i.e. non-persistent data. However, the use of persistent data as source and/or target "message data", such as database tables, is being investigated. Currently this can be supported using XML query and datagram messages to SQL Server databases, but this mechanism is cumbersome and the transformation specifications do not take into account the persistent nature of the data to be transformed. Further enhancements to both the visual mapping language and textual language to allow easier specification of common mapping constructs, and to provide better visualization of complex mappings to developers, is being carried out in response to user feedback.

Summary

Integrating health informatics systems is challenging, with many systems using different message protocols to support interoperation. We have developed a new system to support the specification of complex message transformations. This incorporates a visual language for expressing hierarchical structure mappings, a special-purpose textual programming language to implement these mappings, and a run-time mapping engine to perform message transformation. A commercial product has been developed from this research by Orion Systems Ltd as part of their Symphonia™ messaging suite.

References

Aditel Corp. ETS for Windows™, www.aditel.be, viewed June 2001.

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 CS Press, 2000, pp.152-157.

Data Junction Corp, Universal Translation Suite™ General Information, www.datajunction.com, viewed May 2001.

Emmelhainz, M.A. *Electronic Data Interchange: A Total Management Guide*, Van Nostrand Rein-hold; New York; 1990.

Estes, D. Disciplined XML, *EAI Journal*, Jan. 2001, www.eaijournal.com.

Goulde, M.A. Microsoft's BizTalk Framework adds messaging to XML. *E-Business Strategies & Solutions*, Sept. 1999, pp.10-14.

Grundy, J.C., Mugridge, W.B., Hosking, J.G., Kendall, P. Generating EDI Message Translations from Visual Specifications, In *Proceedings of 2001 IEEE International Conference on Automated Software Engineering*, San Diego, CA, Nov 26-29 2001, IEEE CS Press.

IBM Corp, MQ Series Integrator, www.ibm.com, viewed May 2001.

Liou, D.M., Huang, E.W., Chen, T.T. and Hsiao, S.H. Design and implementation of a Web-based HL7 validation system. *Proceedings 2000 IEEE EMBS International Conference on Information Technology Applications in Biomedicine*, IEEE CS Press, 2000, pp.347-352.

Lincoln, T., Spinosa, J., Boyer, S., Alschuler, L. HL7-XML progress report. In Proceedings of XML Europe '99, Alexandria, VA, USA, 1999, pp.733-736.

McLure, M.L, Moynihan, J.J. Organizing for EDI (healthcare industry). *Healthcare Financial Management*, vol.49, no.1, Jan. 1995, pp.90-93.

Morgenthal, J.P. XML: The New Integration Frontier, *EAI Journal*, Feb. 2001, www.eaijournal.com.

OnDisplay Corp, CenterStage eBizXchange, www.ondisplay.com, viewed May 2001.

Seeburger Corp, SEEBURGER data format and business logic converter, www.seeburger.de/xml/, viewed May 2001.

Spencer H. XML standards for data interchange. *Imaging & Document Solutions*, vol.9, no.9, Sept. 2000, pp.15-17.

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.

Sokolowski, R., Expressing health care objects in XML, In *Proceedings of the 1999 Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, IEEE CS Press, 1999 pp, 341 -342

Vitria Technolgy Inc, Vitria BusinessWare White Paper, www.vitria.com, viewed May 2001.

Wallin, G. A new look at EDI healthcare. *Health Management Technology*, vol.20, no.5, June 1999.

XML.org, XML and XSLT, www.xml.org, viewed May 2001.

Author details for correspondence

Associate-Professor Rick Mugridge
Department of Computer Science
University of Auckland
Private Bag 92019
Auckland
New Zealand.
rick@cs.auckland.ac.nz
www.cs.auckland.ac.nz/~rick