A User Interface for Computer-Based Message Translation*

Meera M. Blattner** Lawrence T. Kou

University of California, Davis, and University of California, Davis
Lawrence Livermore National Laboratory Davis, CA 95616
Livermore, CA 94550

ABSTRACT

This paper describes a generic message translator that enables a computer user without programming knowledge to specify a program for translating between two different computer-based message systems. Thus messages formatted for use in one system can be automatically translated by the computer into the proper format for use in the other system. The computer user works interactively with the computer to specify the translating program, making use of the appropriate visual paradigms for the message structure.

1. INTRODUCTION

A computer-based message system allows the user to create, send, and receive messages via a computer network. Such systems were originally developed by vendors or computer centers for their own specific applications. Subsequent improvements in network technology have greatly increased the potential for communication between disparate systems, but incompatible formats make this a problem. In recent years there have been various attempts to standardize message system formats [1-3]. However, many existing systems do not follow recommended standards, and those systems that do follow them may contain nonstandard, vendor-defined data fields.

The problem of message translation equates to the more general one of generic data translation. By a "data translator" we mean a translator whose input is data and whose output is part or all of that data in some restructured form; a data translator is "generic" if it has the ability to produce many different types of data structures.

As part of our work we studied military communication systems, which transmit messages from one communication node to another using formats that were specified many years ago. These message systems describe real-world events such as the movements of men and equipment, messages that are more closely related to information found in databases than to expressions in programming languages or communication protocols. They use a variety of data structures, and thus there is a need to translate these messages from one format to another so that different installations can communicate more readily with each other. The results of our work apply not only to military communication systems, however; they apply as well to all computer-based message systems except those specifically excluded by Ref. [1].

2. BACKGROUND

Data structures used for generic message translation are varied and depend upon real-world information. The problem of storing and retrieving real-world information has been studied in the context of database management systems. However, it is important to realize that database management systems are designed to retrieve information rather than to serve as translators between systems. Report generators can be thought of as having a restructuring objective, but the basic objective is still querying and retrieving data rather than translating. Object-oriented techniques were used by Breazeal et al. [4] to reformat data in heterogeneous bibliographic citation databases to a standard form. Since citation structure has many similarities to the message structure we analyze in this paper, the techniques are similar, but a visual interface was not used in the work on citation databases. In our earlier work [5] we used examples as input to a data conversion tool and combined it with a visual interface to simplify its use. Subsequently [6], we introduced the idea of generic message translation with a visual interface to simplify the programming process.

Special languages for data restructuring have been examined by Kitagawa et al. [7], Navathe and Fry [8], Shu et al. [9], and More [10]. The work that most closely captures the translation efforts we describe in this paper was done by Shu [11] in FORMAL. In most business data processing applications the business form is a natural interface between the user and the data. FORMAL is a visual, nonprocedural, forms-oriented data manipulation language implemented at IBM Los Angeles Scientific Center on an experimental basis. FORMAL provides the user with a visual means of translating data from one form to another, although it does not address the diversity of data structures used in our GMT.

To enable the nonprogrammer to create the specifications for a translation, we use a visual interface in GMT. Visual interfaces are more easily understood when the problem domain is one of manipulating visual objects, such as in many engineering CAD tools. Jacob [12]

** Also with the Department of Biomathematics, M. D. Anderson Cancer Research Hospital, University of Texas Medical Center, Houston, Texas.

0073-1129/89/0000/0043$01.00 © 1989 IEEE
points out that difficult problems arise when a visual language attempts to provide an interface for more abstract ideas, such as time sequences, conditional statements, and other objects that depend upon metaphor for their interpretation. We had the task of creating our own visual metaphors for a description of a translation. Work done in simple graphical visual programming languages, such as Pict [13], provided us with many insights into suitable visual representations of objects that require manipulation.

3. MESSAGES AND FIELDS

A message is a unit of communication from an originator to a recipient. It is composed of data elements arranged into fields. Fields may be organized into subfields as well. Fields can be described in terms of their syntax (format and structure) and their meaning (semantics). Messages are designed to be read by machines, so the beginning and end of each field must be clearly demarked in some way. One way is to allot the field a fixed length, such as ten spaces; another is to end the field with a delimiter. Parsing to determine the beginning and end of each field, as in an LR grammar, is not excluded, but it will not affect the material we present in this paper. We will use as our examples messages that have either fixed length or specific delimiters (or both). Hence a message is a string of data fields. In order to understand messages we must understand the rules governing field organization.

Definition: A message $M = (S, D)$, where $S$ is a message schema and $D$ is an instantiation of that schema (the data). A message schema also has two parts: the message expression and the message specification. The message expression is formulated by the rules given below, while the message specification is a frame associated with each data field. To explain what we mean, we give some illustrations.

Example 1: Below is a message—that is, the actual data that might be used in a message:

```
truck /200 / 5-25-88: A526-3/ Pennyworth; Chicago/ Maxibuck; Los Angeles/ chairs@$150ea /tables@$500ea/ B555-2/ Crunchy; Milwaukee/ Tastee; Los Angeles/ dogfood@$25bx/ catfood@$35bx/
```

The message expression for the above example is as follows:

```
shipment-information: carrier /total#items /date: /shipper; receiver-address/receiver-address /item-name /cost/
```

Example 2: Below is a message with a somewhat different organization:

```
```

The message expression for the above message is as follows:

```
shipper-list: [transportation] transportation /State registration/ state registration /date: /id# /no. of items] #items /source] [destination] destination /item-names /number at value
```

Notice that the two messages have similar meanings—they both describe shipment information. The information in the fields is also the same with two exceptions: the first message, shipment-information, contains the total number of items, while the second message, shipper-list, lists the number of items being shipped by each source (shipper) as well as a state registration. However, we can get the information in the first from the second. Shipper-list appears to be missing source and destination addresses. The source and destination fields are composed of subfields that are the source name and source address and the destination name and destination address. To translate source to shipper and shipper-address the source must be split into two fields—the first translated to shipper and the second translated to shipper-address. A similar division is made for the destination, which is translated to receiver and receiver address. To translate shipper-list into shipment-information is impossible without more information concerning the number of items with each ID and a state registration. The second message, shipper-list, has tags in front of the data fields. These tags are literals; in other words, they appear in every message of this type. We have put the tags in italics to differentiate them from the data.

Each message schema has a name. The names of the message schemas in Examples 1 and 2 are shipment-information and shipper-list. Field names are designated by bold type and they are replaced by data to form a message. As an example, carrier is replaced by truck in the actual message. The items not in bold remain constant, and the v is used before a field that may be repeated an arbitrary number of times, and the v is used as an "or." The rules for composition are as follows:

1. A field name may be a message expression.
2. A literal may be a message expression.
3. If $X$ is a message expression, then $Xv$ is a message expression, where a and b are literals or blanks.
4. If $X$ and $Y$ are message expressions, then $XvY$ is a message expression.
5. If $X$ and $Y$ are message expressions, then $XY$ is a message expression.
6. If $X$ is a message expression, then $X[a]$ is a message expression.
7. $X[a]$ is a message expression.
8. Parentheses are used to indicate order of operations and groupings. They may be dropped if the order and groupings are clear—parentheses are not to be interpreted as literals.

The definition of a message expression is similar to that of a regular expression; however, the rules for manipulating these expressions are different—more like summation. For this reason, we chose not to use the * for repetition. The message expressions in Examples 1 and 2 are preceded by the underlined message name. We will use this convention to easily identify the message expression.

Fields may contain subfields; for example, the date field may have subfields of month, day, and year. The organization of the subfields is similar to that of fields: each field may have subfields
that form an expression. Fields that have no subfields have a frame that gives the specification of the data. The data structures that are used in fields are discussed in Section 5.

Modern message formats actually consider the field to be a collection of components that include the specification of the field as well as the data field itself. Typically, the data element is described in terms of a component, an identifier, a length, a qualifier with additional field information, a property list of components, and, finally, the data element itself. The great advantage of message formats with components that contain specification fields is their flexibility in accommodating many message types. Our particular application does not have this component list preceding the data element. However, our completed system will accommodate both types of message systems, those with and those without specification fields.

The message fields may be classed into originator fields, recipient fields, message handling fields, etc. The body of the message is often encapsulated text. In the present paper, this particular division is not our primary concern. It is the data structures within the field and operations required for their translation that we investigate, whether they occur in the body, headings, or other locations.

4. THE USER INTERFACE

The purpose of the GMT is to enable a user to construct a message translator without using programming skills. This can be done in various ways:

1. Commands and message specification can be entered visually with direct manipulation, menus, and forms. A visual user interface requires learning relatively few operation sequences. Prompts and menus assist the user in learning and recalling operations that require user input. Some operations are easier to specify in pictures than in words. The GMT uses visual techniques that seem to be the most natural expression of the operation to be done, in order to minimize the time required to learn the system.

2. A large number of modules are available for reorganization and translation. After the user specifies the desired translation, the GMT provides a module to do it.

The two capabilities mentioned above have been the essence of recent progress in software tool development. The spreadsheet was one of the first successful applications of this new technology.

Spreadsheet programs were so successful because the operations were limited and well understood, and the visual interface was natural to the application. However, as software designers attempt to use these methods with other and more general applications, problems arise. Neither the visual paradigms nor the operations may be clearly understood. We believe these techniques can be used in the field of data conversion. This paper is an attempt to clarify the nature of the operations used in a class of data conversion problems as well as to identify suitable visual paradigms for them.

Figure 1 shows the basic user interface for the GMT. The source message is displayed in the upper one-third of the figure, and the target message is displayed in the lower one-third. The operations pertaining to the translation process take place in the middle. The user may request to see an example of the data set for the source or target messages. In Figure 1 the user has selected the option of viewing an example of the target message.

The field names are displayed in boxes. The message name is in an oval. Our solution to the problem of a visual representation of repeated fields was to create an image of stacking sheets of paper. The user can see a display of a sample of the data for the message expression on the screen, a specification (the associated frame for a selected field), or a history of the operations used. A menu of the operations used for field reorganization or field reformatting pops up if the operations box is selected.

The GMT has two modes of operation: one is for the association of two message types, and the other is for the creation of a new message type or modification of an existing message type. In the first case, the translation is accomplished by bringing up two existing message schemas M1 and M2. It is the task of the user to establish a mapping from the fields of M1 and M2. Nothing in the specifications of the fields of M1 and M2 will be changed—the user creates the rule used to associate the fields of M1 with those of M2. Then the user identifies a function that formats the data that will be entered into those fields. Once the rules for the mapping have been completed, the GMT creates a translator to map a message with the schema of M1 into a message with the schema of M2. Figure 2 shows an example of this kind of association between fields. In the second case, where a new message type is created, the user creates a copy of an existing message schema M3 and uses operations that modify the schema and the specifications. This becomes a new or updated message M4. There are no translators that remain in the system to change data formatted as M3 into data formatted as M4; M4 will be stored as a new message schema. Figure 3 shows the creation of a new message type.

5. THE OPERATIONS FOR FIELD REORGANIZATION

Message translation is divided into two distinct tasks: the reorganization of fields, and the reformatting or translating of data structures within fields. The reorganization of fields changes the message expression, while the reformatting within a field changes the contents of the frame assigned to the data field. When the message expressions are reorganized, the data frames associated with field names are not examined, nor are they changed (with the exception of deletion or insertion) when an empty frame is added to the data. The subfields may be restructured then by split and glue operations (see below).

There are eleven operations used to reorganize sets of fields:

1. SWITCH (X,Y) exchanges expressions X and Y, so that (X Z Y) becomes: (Y Z X).
2. DISTRIBUTU (X, Y) distributes expressions according to the rules:
   - a. X(\(U v Z\)) becomes (X(U v Z)), where Y = U v Z.
   - b. X \(\{Y\}\) becomes \(\{X(Y)\}\).
3. FACTOR (X, Y) factors X from the expression Y, so that:
   - a. Y(\(X(U v X Z)\)) or (\(X U v Z\)) becomes \(X(U v Z)\).
   - b. Y is \(\{Z\}\) or \(\{X(Z)\}\) becomes \(\{X(Z)\}\).
4. MATCH (X, Y, Z), where Y is a subexpression of X and Z, creates one expression \(X(Z1 v Z2)\), where Z1 is with Y removed.
   - a. X(Z1 v Z2) becomes \(X(Z1 v Z2)\), where Y is removed from Z1 and Z2.
   - b. \(X(Z)\) becomes \(\{X(Z)\}\).
5. DIVIDE (X, Y, Z) creates two expressions \((X Y) (X Z)\), where X may be empty.
   - a. \((X Y Z)\) becomes \((X Y) (X Z)\).
Other field manipulation operations will be provided, such as:
6. DELETE (F) deletes a field with the name F.
7. INSERT (F) inserts a field with the name F (if F is new, a blank specification is added).
8. SPLIT (F, G, H) separates one field F into two fields G and H.
9. GLUE (F, G, H) creates one field F from two fields G and H.
10. COPY (F) copies field F and inserts it into the designated location.
11. REPLACE (F, G) replaces F with G. If G is new, a blank specification is added.
The first five operations can reorganize any message M1 into a message M2 with the same semantic information and whose fields differ from M1 by a change of name. The operations are in pairs, where one operation is the inverse of the other, and SWITCH is an inverse of itself. MATCH may not be an inverse for DIVIDE under certain conditions (similar to creation of a lossy join). We have intentionally left out literals, such as delimiters and syntactic sugar. The reason is that the GMT maintains a message template with literals inserted in their correct position. If M1 is translated into M2, the GMT has a format specification for M2 and the position of the literals is known in relation to the field names. The only information the GMT must have from the user is the semantic relationship between the fields in

![Figure 2. The field association list.](image)
Figure 3. An application of the match operation to a new message format.

M1 and M2. The only relationship allowed in the first five operations is that field F in M1 is the same as field G in M2.

Example 3: In the example (date) (supplier item cost), before distribution, we have:

(5/8/88 Farm1 tomatoes $20 bushel) (5/20/88 Farm2 potatoes $15 bushel) (Farm3 squash $10 bushel)

After distribution we have:

(5/8/88 Farm1 tomatoes $20 bushel) (5/20/88 Farm2 potatoes $15 bushel) (5/20/88 Farm3 squash $10 bushel)

Note that the association of the right date is kept with the supplier, item, and cost.
5. A VISUAL INTERFACE FOR FIELD REORGANIZATION

When message schemas M1 and M2 already exist, the translation begins by asking the user to map the field names of M1 to those of M2. This can be done by filling in a form for the corresponding names, or by clicking on corresponding names successively with a mouse. The form version is shown in Figure 2 with shipper-list and shipment-information as examples.

The user can either type in the name of the corresponding field from shipment-information or click on the field displayed in the lower one-third of the screen. There is one repeated field (#items) that is collected into an aggregate field (total#items). The computational program fragment for the aggregate field operation is inserted into the translator at the time it is compiled. The state-registration field in the source does not appear in the target message, hence the field association list has been marked to delete this field in the target message.

Figure 3 shows an application of the match operation in the definition of a new message format. Shipper-list is the source message in Figure 3. The target message is the same as shipper-list except that the match operation has been applied to id-number. The operation was executed by selecting the two id-number fields in the source and the match operation from the field operations menu.

7. REFORMATTING OPERATIONS FOR THE TRANSLATION OF FIELDS

In the section above we laid out the rules for the manipulation of fields and discussed the semantic equivalence of messages with reorganized fields. In this section we discuss the translation of a field to another format. Hence we may translate date in shipment-information to date in shipper-list, but now the date is formatted differently. In one message the date may appear (instantiated) as May 25, 1988, and in another message the date may appear as 5/25/88.

Translator objects are responsible for the translations. Each translator has a domain, a range, and a technique. The domains are strings of symbols; for this reason, we do not include arrays, geometric forms, or other types of multidimensional structures. Translators are of the following types:

User Defined Associations

In a user-defined mapping, the user points to elements in the domain and points to elements of the range to make the association. Pointing operations are usually performed when the domain and range of the mappings are finite. The user interface displays both the elements of the domain and the range to allow this association. An example of this type of mapping that occurs in messages is in the date field. A month may be "Jan" in M1, while it may be "1" in M2. An association of this type is often made on a case-by-case basis, using a catalog, directory, or authority list. For the most part, it is the user's knowledge of the real world that allows him or her to make this association.

Under certain conditions, user-defined associations may be made with a large or infinite domain as well. In the next section we show how this can be done. We call these "fine-to-coarse translations." This type of translation may be specified by a computational rule as well.

Associations that Use a Computational Rule

Elements from the domain of a field may have to be computed before they can be associated with the corresponding field in another message. Three types are used:

a. Numeric translators: These translators do a simple mathematical calculation. The operations are similar to those provided on an inexpensive calculator. An example is: \( Y = 2X^2 + 3Z + 1 \). Many standard functions are included, such as absolute value, degrees/radians, and miles/kilometers.

b. String translators: These translators understand words and characters. The operations are insert character, delete character, upper case, lower case, count characters, right-justify, left-justify, and truncate. Some additional text editing operations may be added as the need arises.

c. Fine-to-coarse translations: These translations are also called "many-to-one mappings." Common numeric examples are truncation, round-off, and the conversion of real to integer (with truncation). Another example is the assignment of integers to the integers mod n.

Associations that Use Aggregate Operations

An aggregate is a list of elements, each from a different field and possibly from different domains. Translators that operate on aggregates operate on sets of data elements.

a. The aggregate numeric operations are the average, max, min, count, and sum. Since every domain has a definition of magnitude and relative order, these operations may be applied to every aggregate list.

b. String-based aggregate operations are those thought of as global text editing operations, where lower case, upper case, and substitute are examples.

c. Aggregates also map multiple fields to multiple fields. These operations include order, keep duplicates, and eliminate duplicates.

8. A VISUAL INTERFACE FOR REFORMATTING OPERATIONS

Visual Interfaces for User-Defined Associations

Two familiar visual images we have of the operation of mapping elements from one finite set to another finite set are graphical representation of the mapping relationship (nodes and arcs) or tabular representation. We chose the tabular form for GMT. Figure 4 shows the prototype display for this type. Many, if not most, fields in messages are of this form. For example, transportation might specify in its field a choice of air, surface, or sea, while carrier might use plane, light truck, heavy truck, or auto. In this example, an obvious mapping problem exists which the user, with his or her knowledge of the world, must solve. If carrier were to be mapped to transportation, the solution is simple: plane is mapped to air, and light truck, heavy truck, and auto are mapped to surface.

Fine-to-coarse translations from a continuous domain to a finite range can be done as shown in Figure 5. The domain is shown as the
real line, and the user sweeps portions out with the cursor to be mapped to an element in the range. If the domain cannot be represented by a real line or a circle (for numbers mod n), then the user must apply a computational rule.

**Visual Interfaces for Mappings Using a Computational Rule**

Our prototype implementation uses Fortran-like mathematical expressions. A list of operations is given to the user. The image of a hand-held calculator displayed upon the screen is the one that seems to be the most appropriate to the computation of mathematical expressions. A syntax-based parser for expressions using the basic mathematical operations can monitor the user's input and check for correctness.

**Aggregate Operations**

At the present time, the fields that are the domain of the aggregate function are selected from the source message, and the aggregate function itself is selected from a menu.

Figure 4. Tabular mappings.

Figure 5. User-defined associations in fine-to-coarse translations.
Composite Operations

It is important to allow compounding of the operations, else the translations will be too limited. Our compounding rules allow sequential and conditional applications of operations. We have not had the need for iteration as yet.

9. DISCUSSION

In this paper we have examined the question of whether data restructuring and format translation are feasible using a visual interface. We believe that data translation cannot be done automatically without user intervention because of the large amount of information required about the real world. Although the user must intervene in the translation process he or she need not have programming skills. The problem appears to be threefold: (1) reorganizing data fields, (2) reformatting data within the fields, and (3) choosing appropriate visual paradigms that suggest the nature of the translation task to the user. We have seen that (1) is a straightforward and well-understood task, with message structure defined as in Section 3. The operations we provide for field reorganization allow a message expression to be reorganized into another semantically equivalent message. As for (2), reformatting data within fields, we have isolated a number of broad classes of translations that accomplish the tasks we require. A more theoretical understanding of the nature of the translation process is required in order to identify the nature of all translations or to construct a set of tools to serve as a foundation for generic translation. The third task, finding visual paradigms for translation, is the least understood theoretically. Our visual paradigms correspond to the broad classes of translations we have identified. We have not tried to measure how well a visual paradigm corresponds to the task of translation; we have only applied our intuitive sense of "rightness" in selecting these paradigms. Some more objective measure is probably required.

We have not discussed the problem of translating messages with somewhat dissimilar semantic content. Although this is a major problem in data conversion of any type, it must be the subject of another paper.

Acknowledgments: John Carlson and Douglas Daniel, Lawrence Livermore National Laboratory, also participated in this project. Thomas Longstaff assisted in the design of the user interface.

10. REFERENCES