Visual Interfaces for Model Mapping – Large Mapping Visualization

António Painha, Hugo Manguinhas, José Borbinha
INESC-ID, Rua Alves Redol 9, Apartado 13069, 1000-029 Lisboa, Portugal
IST – Department of Information Science and Engineering, Instituto Superior Técnico, Lisbon Technical University, Portugal
(antonio.h.painha, hugo.manguinhas, jlb}@ist.utl.pt

Abstract. The mapping of information models has become an important part of the data integration and interoperability research areas, which aim to enable the reuse of data (i.e. information) in different contexts, and often resort to visual mapping tools as the simplest and fastest way to define the needed relationships between two different models and create the desired mapping model. However, the current approaches adopted by visual mapping tools, to visualize the mappings between two models, have difficulty to work properly when the models or the mappings between them become large. This work then proposes seven new, or improved, visualization techniques (called interface paradigms) to address this problem, which aim to simplify the viewing, making it possible for the user to effectively deal with much larger models and maps. This novel approach was integrated into the XMApper prototype, a visual mapping tool. A user study was conducted to evaluate the proposed interface paradigms, and results led to the conclusion that most of the new paradigms are very useful and effective, when applied to the interface of a visual mapping tool. The primary contribution of this work is a demonstration of new ways to effectively present highly complex mapping information.

Keywords: Visual Mapping Tools; Visualization Techniques; Visualization of Complex Information; Visual Interfaces; XMApper.

1 Introduction

Currently, information modeling can be used to describe rather extensive domains, thus requiring proportionally extensive information models to describe them. This all transcends to the mapping between models process, where, for complex models and mappings, defining a mapping model is very difficult [1]. One current, well-received solution to this problem are visual mapping tools, which are visual modeling tools with the purpose of making it easy for a designer to establish mappings [2]. An existing problem with such applications is that they are unable to cope well with the growth in size and complexity of both the models and the mappings that can be
established between them. Figure 1 is an example of such failure. In this MapForce’s example, details of interest are lost in a maze of complexity.

![Figure 1. Example of failure to scale well for large mappings.](image)

The motivation for this work comes from our efforts to develop a Metadata Registry (MDR), an information system, with a web-based interface, designed to store and maintain in a controlled environment the range of information models used within an organization, and how these models relate to others [3]. As a result, the MDR promotes a common understanding of the information managed within an organization and assists organizations in the sharing and exchanging of mutually agreed information. To achieve this, the MDR requires a visual mapping tool with a web interface that can cope with the diversity and complexity of each domain. This tool is an under development prototype named ‘XMApper’.

This work proposes with five new visualization techniques, or interface paradigms, and two improvements over existing ones to approach the issues of large mappings visualization. This paper starts by making a small overview of the researched related work, followed with the description of the new interface paradigms and, after that, a report of results of the user study that verifies the their usefulness, usability and effectiveness.

## 2 Related Work

Visual mapping tools are software applications which can be used to visually establish mappings between models, defined within a schema (usually in XML¹). In such programs, mappings are specified by allowing the users to make connections between visual representations of source and target schemas’ entities. The associations can range from very simple, such as a direct line between two elements, to more complex, such as an association of two or more elements through a graphical box that denotes a function application. In the ambit of this work, many visual mapping tools were studied. MapForce² is a commercial tool, developed by Altova, allowing the definition of mappings between many formats, like XML or EDI, using a visual design interface; Microsoft’s BizTalk Mapper³ – a module included in BizTalk Server bundle – is another visual editor that allows mapping between XML schemas,

---

¹ [http://www.w3.org/XML/](http://www.w3.org/XML/)
and has had a lot of recent research and development dedicated to its interface [1]. The XML Mapper\(^4\) – a part of Stylus Studio’s XML IDE – is a many-to-one visual schema mapper dedicated to a XML environment; Clio\(^5\) is an IBM’s prototype system for managing and facilitating data transformations and integrations, with the particular focus of the semi-automatic definition of the mappings, and includes a graphical interface, the Clio Schema Viewer, to allow user corrections and information viewing; and finally, the COMA++\(^6\), another research prototype – developed at the University of Leipzig – that aims to study automatic schema and ontology matching, and possesses a graphical interface, which allows the user to view and influence the matching process in many ways. The interfaces of all this tools have some common parts: the Schema View, which represents the models usually through a hierarchical tree; the Mapping Board, which is where all the mappings are made and visually represented through mapping links an function boxes; and the function Toolbox, where all the available mappings functions are stored. Therefore, the interface paradigms (representation techniques that are applied to an application’s interface with the goal of displaying information with unusual characteristics, which create representational problems for the interface) for the representation of large mappings (this work’s problem) will be separated into four different areas: Mapping Board Improvement, Schema View Improvement, Navigation and General Improvement.

A study was conducted to list the interface paradigms present in the previously introduced five visual mapping tools, revealing many unique ones and other that were present in more than one application. This study drawn from various sources (more precisely [1] [4] [5] [6] [7] [8] [9] [10]) and from its analysis came some thoughts and ideas, like the dominance of paradigms dedicated to Mapping Board Improvement and the value of some navigation techniques present in many of the applications. Also, many areas remain problematic. On the Mapping Board, correct identification of connections, without selecting them, remains the biggest issue, while in the Schema Views the main problems are finding and viewing specific information about the models and relevant information about the mappings.

Finally, to create the new interface, where any new paradigms developed in this work shall be tested, a search was conducted to find the right JavaScript framework for the job. Many solutions were analyzed and compared: Prototype + Scripty2, MooTools, jQuery, Ext JS, Dojo, GWT and Vaadin. GWT (or Google Web Toolkit) was the elected choice, because, although it will demand the investment of more time (due to a harsher learning curve and ease of use), it will allow for a more robust, versatile and easier-to-debug solution.

### 3 Proposed Interface Paradigms

The study of the problem and the analysis of current visual mapping tools interfaces, led to the idealization of seven new interface paradigms, divided between Mapping

---

\(^4\) http://www.stylusstudio.com/xml_mapper.html
\(^5\) http://www.almaden.ibm.com/cs/projects/criollo/
\(^6\) http://dbs.uni-leipzig.de/Research/coma.html
Board Improvement and Schema View Improvement. Paradigms [SP1.1] and [SP2.2] are improvements on already existing ones, while the other four are new.

3.1 Mapping Board Improvement

[SolutionParadigm1.1] Connection Visibility – Current iterations of mappers are starting to implement paradigms that aim to clean the view of the Mapping Board by manipulating the visibility of the shown connections with user toggled functions. This work’s approach to this paradigm is to implement a similar paradigm as being the mapper natural behavior. The idea is to transmit the user the impression that if both end of the connection aren’t visible then that connection doesn’t matter at the time (is hidden), or if only one end is visible then the connection appears out of focus (greyed-out and dashed) just to inform the user where the other end can be found (by following the line). The definition of visibility status for a schema element has also been improved: an element isn’t visible not only when it is scrolled out of the view, but also when its parent element is collapsed. This is important because it means that connection cluttering will be reduced even further, by removing focus from connections that go to elements hidden by collapsed parents (but with the other end of the connection still visible) and even hiding them completely (if the other end is also hidden or scrolled out of view).

[SP1.2] Connection Render – A mapping overloaded with connections will often lead to user confusion when he tries to identify overlapping connections or even connections overlapped by function boxes (does it come from that function box? Or it has another source and it’s just crossing the function box’s space?). Techniques like bendable links have tried to address this problem, but still with limited success. This dissertation proposes to allow the user to change how the connection is rendered at any time, i.e. if he’s having problems comprehending links with the current connection shape he shall be able to change it on-the-fly and try with a different one. For now, two complimentary approaches shall be tested: a rectilinear connection render, which will plot a path from A to B with only horizontal and vertical lines; and a direct connection render, which will go directly from A to B. The first method is cleaner, but may cause confusion with overlapping connections, while the second might be messier, but simpler to (in most cases) identify the connections’ targets.

[SP1.3] Connection Argument Visibility – Minimizing the space occupied by the mapping functions’ visual representation (usually a small box) is and effective way of simplifying the view of a Mapping Board (as can be seen in BizTalk Mapper and Stylus Studio XML Mapper), but doing so often makes it impossible or impractical to have the function inputs and output directly show on the its representation. A possible way to address this dilemma, and thinking of what MapForce does with its connection’s annotations, is to show the inputs and output of the function boxes near the docking point where the connections meet them, and call them simply connection arguments. This way the connection arguments will only show when needed, i.e. when a connection is effectively using it, instead of being always shown in the Mapping Board. Although being a nice and clean method in the early stages of a mapping, this technique can also originate much cluttering of the Mapping Board on
Visual Interfaces for Model Mapping – Large Mapping Visualization

3.2 Schema View Improvement

[SP2.1] Quick Filters – Inspired by the thought of joining the paradigms of information toggling and information suppression, like sibling coalescence, quick filtering is a simple way to address the problems that come from a Schema View with a high number of elements. The idea is to have quick access, by toggles, to some filters that allow the user to isolate important elements or show views of interest. One of these filters will be the ability to hide all unconnected elements, so it’s possible to easily know what the current state of the mapping is, and other the inverse, i.e. show only unconnected elements, with the objective of knowing what isn’t mapped. Furthermore, other filters like “hide all but the user selection” would certainly be worth investigating.

[SP2.2] Tree Element Connection Status – Having visual clues of the current connection status of the tree schema elements in the tree element representation itself, can be of precious aid to navigate and assert the current state of a mapping. Currently, some mappers (like Biztalk’s) already use a solid line and a small icon in the tree element, to represent if that element is connected or not, but this technique can be expanded as well as its purposes. The connection line in element the can be further customized in terms of color and shape (e.g. solid or dashed) to transmit more information to the user (like nested elements connections or connection visibility) without needing extra space. Furthermore, a small underscore under an element’s text will also transmit more information about an element’s (or its children’s) connection status.

[SP2.3] Sorting – Doing the right sorting of the tree elements can be another way to quickly find what you are looking for in a heavy populated tree. Two types of sorting seem to be relevant when trying to find a specific tree element: Alphabetical and by Connection Status. The simple alphabetical sorting is a valid option, being a common and intuitive method the users are used to deal with, which can be employed to find elements by their name. On the other hand, there is the more mapping-specific Connection Status ordering, which, as the name says, consists of ordering the tree elements by their number of connections, isolating (in the same tree level) unconnected elements from the connected without having to remove one of the groups. The main drawback of sorting the tree elements is changing the ‘natural’ order of the schemas, which could cause some disorientation issues.

[SP2.4] Connection Statistics – Statistics present a different view of what is first perceived in a situation. Showing schemas’ connection statistics, like the ratio of connected and unconnected elements of a schema, can be helpful for a mapping user to know aspects such as the completion of the current mapping model. Also, those statistics could be used to directly navigate or change views in trees or in the Mapping Board. The idea of this paradigm is to simply convert the statistic data to graphs and having the different portions of the graph hyperlinked to the action wished to be
applied upon the elements of the mapping (Example: clicking the “connected” portion of a pie graph showing the ratio of connected schema elements, would filter all the connected schema elements in the mapper). This way the user will not only be informed about the mapping, but will also have a quick way to navigate it.

4 User Study

As the developed solution aimed to change the users experience while using the XMApper’s interface, a user study was conducted to evaluate its usability and usefulness. To this end, different versions of the prototype standalone version were built. For all versions, even the baseline one, the [SP2.2] Tree Element Connection Status was active due to its simple nature and valuable help in defining the experiment’s tasks. In total there were four versions of the XMApper prototype in this study: Version A is the baseline version, with no new interface paradigms (besides [SP2.2]). Version B is version A plus the Mapping Board Improvement paradigms. Version C is version B plus the Schema View Improvement paradigms, except the [SP2.4] Connection Statistics. Version D is the final version, with all of the version C paradigms plus the [SP2.4].

The [SP2.4] paradigm’s synergy with the other paradigms led to the choice of creating a final version to collect data about it specifically. This version was tested separately, with a specific task set, and compared only to Version C.

So, the main study consisted in all participants experimenting with the first three versions of the interface with a small study of Version D being conducted right after every test of Version C. To control for order effects, the order in which participants experienced each of the three main versions of the mapper was counterbalanced using a Latin Square design. After all the prototype tests were done, each participant completed a background and satisfaction survey.

Participants

The data for this study was collected from a population of 10 male computer users. From the analysis of the background questionnaire, the participants had an average of 25 years (ranging from 24 to 26), 16 years of computer experience and 70% had a degree in Multimedia and/or Information Systems. Although most participants didn’t have previous experience with the XMApper due to its recent nature, 60% had experience using visual mapping tools.

Tasks

For the main study, four task sets with four tasks each were devised, involving finding elements and information in the source and target schema, their related function boxes and connections. For the statistics study, two task sets of two tasks each were created, involving finding and counting schema elements. An effort was made to keep the task sets isomorphic so that the participants experienced similar tasks as they viewed each version of the interface. To ensure that no one task set was accidentally more difficult that the rest, however, they were rotated through

---

http://mathworld.wolfram.com/LatinSquare.html
visualizations. Two of the tasks sets used in the main study are show in Table 1 as an example. The two task sets used for the statistics study are show in Table 2. All sessions were conducted online (the four different versions of the prototype were online) in video conference, with a single participant at any one time and lasted on average 50 minutes. Because the tests were performed online, the Properties Panel on the prototype’s interface was disabled, so that the users couldn’t name a related element before they navigated to it.

<table>
<thead>
<tr>
<th>Task Set A</th>
<th>Task Set B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Go to the first “NOT” function box (from above) and find out what it’s connected to in both the source and target schemas.</td>
<td>1. Go to the first “AND” function box (from below) and find out what it’s connected to in both the source and target schemas.</td>
</tr>
<tr>
<td>2. Go to the last connected element of the target schema and find out what it’s connected to in the source schema.</td>
<td>2. Go to the first connected element of the source schema and find out what it’s connected to in the target schema.</td>
</tr>
<tr>
<td>3. Go to the second connected element of the source schema and find out the name of the function box’s argument it is connected to.</td>
<td>3. Go to the second to last connected element in the target schema and find out the name of the function box’s argument it is connected to.</td>
</tr>
<tr>
<td>4. Determine which root element from the target schema has fewer connections.</td>
<td>4. Determine which element from the source schema has more connections.</td>
</tr>
</tbody>
</table>

Table 1. Two of the task sets used in the main study.

<table>
<thead>
<tr>
<th>Extra Task Set A</th>
<th>Extra Task Set B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Determine how many connected elements the source schema has.</td>
<td>1. Determine how many connected elements the target schema has.</td>
</tr>
<tr>
<td>2. Tell me, which is the first connected element in the target schema.</td>
<td>2. Tell me, which is the last connected element in the target schema.</td>
</tr>
</tbody>
</table>

Table 2. The two task sets used in the statistics study.

![Figure 2. The map used in the user study experiments.](image)
Test Map
The map that was used in the experiment is shown in Figure 2. The schemas used were MODS 3.0 (source) and ESE 3.3 (target). Although the used mappings were fictional and specially created for the tests, special care was taken to maintain them valid, respecting the data types. The aspect ratio of the window used for the study, as shown in Figure 2 example, was chosen so the Schema View maximum visible height was equal to the Mapping Board height. In this prototype the Mapping Board scrolling wasn’t implemented. This reduced space of the Mapping Board and limited window size, helped simulating the behavior required for a more traditional aspect ratio on larger schemas and a larger map.

4.1 Results and Discussion
Task times
The post-hoc analysis of all obtained task times allowed the plotting of the chart shown in Figure 3, representing the average task times by version and their standard deviation (σ). The average task time for the prototype Version A (base version plus [SP2.2]) was significantly higher than that of Version C (all paradigms), being 28.4 (σ=3.2) and 17.9 (σ=4.0) seconds respectively. The prototype Version B (baseline plus [SP1.1], [SP1.2] and [SP1.3]) with an average task time of 22.1 seconds (σ=3.2), stood in the middle of the other two. These results are shown in Figure 3.

As for the statistics study, task 1 was devised to study the impact of having a quick access to item count and task 2 was devised to validate the idea of having interactive charts. While the time for the first task was reduced to less than a third of the original, showing the importance of having a profile with general information of the schemas, the second task results were suboptimal and, in fact, having interactive charts proved to be slower in some occasions and many users didn’t even use them.

---

Figure 4. Average times by prototype version and task.

Figure 5. Average task times for prototype version without Statistics (Version C) and with Statistics (Version D) for the two studied types of task.

Satisfaction Data

A user satisfaction questionnaire was completed by the participants at the end of the session. To improve the methodological rigor, some statements were asked in a favorable way toward the prototypes tested and some were phrased in a negative manner. Responses were collected using a 7-point Likert scale with 1 = Very Low and 7 = Very High [11]. In order to improve readability, questions which required a lower response to reflect a positive satisfaction were flipped prior to analysis (e.g. if the user rated a question with 1, meaning the highest possible value, it was flipped to 7). A post-hoc analysis of the satisfaction ratings allowed the plotting of the chart shown in Figure 6, with the averages of the user satisfaction ratings and their standard deviation (σ). The Version C of the prototype was rated significantly higher than each of the other versions, and Version B was slightly higher rated than Version A. The standard deviations were low for all versions (A: σ=0.8; B: σ=0.4; C: σ=0.5), representing a good accuracy for the results. All of the satisfaction data is included in Table 3 (Note: the questions are the original from the survey, but, for this table, higher ratings always indicate higher satisfaction).
Looking at Table 3, the overall higher satisfaction of Version C is noticeable not only directly in the results for question 7, but also in the answers of the other questions, being Version C the only with ratings above the 6 points. Besides the overall satisfaction question (Question 7), these higher ratings were also obtained in the performance (Q.6) and frustration (Q.8) of the user, which are directly related, as well as in the questions about the time consume (Q.5) and difficulty of finding related elements (Q.1), which are also related between them. When compared to these, the discoverability of features (Q.3) scored poorly, barely passing the 5 points, and is an issue to look into in future. A final note to the satisfaction rating of the user performance (Q.6), which was not only one of the highs scoring in Version C, but also the highest scoring in A and B, showing that although most users didn’t have previous experience with the XMApper, they felt secure about how they used it during the tests.

Usability issues
Some usability issues were observed during the study, which need to be addressed in future designs. For instance, in Version C and D, many users used the Show Connected filter and then the Expand All button in quick succession, as this has
proven to be a fruitful combination in many situations. Some users suggested that the option to change the connection render should be more accessible (currently in the preferences menu), and maybe due to this, a few users would just apply as the first thing every time it was available. In addition, one user felt like reaching the sorting options for the Schema Views was also not intuitive, and suggested that the most useful (or even all) of the sorts were moved to the toolbar of the Schema Views, like the filters. Two users said that the direct render (for the connections) should be active by default, as it was simpler to identify the different connections. Several users had problems determining the number of connections of a root element in the base version, and some of the more experienced with mapping tools resorted to deleting or editing connections to count them, instead of trying to identify the overlapped connections. One user suggested having an option that tells you how many connections a singular element of the map has. Some users also reported that the background color of the schema elements when filters are applied (light green) make it hard to tell which element was selected (light blue) because of the color similarity.

5 Summary and Future Work

Addressing the visualization issues created by large mappings was the goal of this work. To that end, seven new, or improved, interface paradigms were designed and then tested in a user study. The study results revealed a significant time advantage for using the new paradigms over the baseline version. In addition, user satisfaction ratings corroborated those performance results, with the new interface versions receiving significantly higher ratings than the prototype’s baseline version. Comments from study participants assured the validity and usefulness of the new interface paradigms, but still some usability issues were observed and should be addressed in future designs. Overall results show that, although being clearly a work in progress, the new paradigms are valid in helping the Visualization of Large Mappings, even if some better than others:

[SP1.1] Connection Visibility − Made distinguishing connections and knowing their related elements visibility and easier job for the users; Improved task times.

[SP1.2] Connection Render − Users liked the ability to change renders, but it should be more accessible to be effective, and have a more varied render selection.

[SP1.3] Connection Argument Visibility − The option to “show on selected connections” was considered a great improvement over the “show always”, and the configuration options were almost unnecessary.

[SP2.1] Quick Filters − The paradigms that produced the most positive feedback, because of its dramatic view change and ease of toggling on and off. Users liked the prospect of having new different filters.

[SP2.2] Tree Element Connection Status − Helped the users all through tests, although many had problems in grasping the difference between the meaning of a solid and dashed underscore.

[SP2.3] Sorting − Also a popular paradigm, the best one to count connections, but less used than the quick filters due to its more hidden location.

[SP2.4] Connection Statistics − The value of having information about the map was proven, but the use of the charts to apply effects to the view needs work.
There is still much to do to improve the new interface paradigms. Some examples: more connection renders; expand statistics to have Mapping Board statistics and work with Mapping Board interface paradigms; more quick filters can be included in tree drop down menu, and choosing those that can be accessed by toggling should be possible. Test participants also suggested many interface changes for the paradigms, especially in their accessibility. Some of the already existing interface paradigms, that weren’t implemented in the XMApper, could work well together with the new ones. More development and testing are required. Also, due to time issues, the scrolling of the Mapping Board wasn’t implemented and thus the hiding of function boxes couldn’t be tested for the [SP1.1] Connection Visibility paradigm. Similarly, the “Show unconnected” filter couldn’t be implemented and tested in time for the writing of this paper.

6 References