Visual Interfaces for Model Mappings
Large Mapping Visualization

António Carlos Higgs Bento Valério Painha

Dissertation submitted to obtain the Master Degree in
Information Systems and Computer Engineering

Jury
Chairman: Prof. João Paulo Marques da Silva
Supervisor: Prof. José Luís Brinquete Borbinha
Members: Prof. Manuel João Caneira Monteiro da Fonseca

May 2012
Acknowledgments

This dissertation has been an immense and fruitful journey, and there are many people I need to thank for helping me steer it to good end. Professor José Borbinha, for his support and decisive orientation in the most dire moments of this work, pushing me to always take one step further. Professor Hugo Manguinhas, XMApper’s project leader, for his many advices and amazing work. Also, all the participants who willingly gave up their free time to test my solution, and whose feedback was invaluable. My friends at IST Taguspark, for being counselors, workmates and party buddies, all in one shiny package. And, lastly but certainly not least, all my family, and in particular my mom and dad, for being my unswerving safe harbor, whenever bad tides push me astray.
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. But 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, which works as a standalone web-based application and as integrated module of the Metadata Registry (MDR). The MDR is an information system designed to store and maintain, in a controlled environment, the range of information models used within an organization, and how these relate to other models.

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
Resumo

O mapeamento entre modelos de informação é um importante tópico nas áreas de investigação sobre a integração de dados e interoperabilidade, as quais procuram conseguir a reutilização de dados (ou seja, informação) em diferentes contextos. Para isso recorrem frequentemente a ferramentas visuais para criar mapeamentos, pois esta é a maneira mais simples e rápida de definir as relações necessárias entre dois modelos e criar o modelo de mapeamento correspondente. Mas as actuais abordagens adoptadas pelas ferramentas de mapeamento visuais, para visualizar os mapeamentos entre dois modelos, têm dificuldades em funcionar correctamente quando os modelos ou os mapeamentos entre eles são de grandes dimensões.

Este trabalho propõe então sete novas, ou melhoradas, técnicas de visualização (chamadas de paradigmas de interface) para abordar este problema, as quais pretendem simplificar a visualização, permitindo ao utilizador trabalhar efectivamente com modelos e mapeamentos muito maiores. Esta nova abordagem foi integrada no protótipo XMApper, uma ferramenta de mapeamento visual, o qual funciona como uma aplicação web autónoma e como um módulo integrado no Metadata Registry (MDR). O MDR é um sistema de informação desenhado para guardar e manter, num ambiente controlado, toda a gama de modelos de informação utilizados dentro de uma organização, e como estes se relacionam com outros modelos.

Foi efectuado um estudo com utilizadores para avaliar os paradigmas de interface propostos. Os resultados levaram à conclusão de que a maioria dos novos paradigmas são úteis e efectivos, quando aplicados à interface de uma ferramenta de mapeamento visual. A principal contribuição deste trabalho é a demonstração de novas maneiras de apresentar efectivamente informações sobre mapeamentos muito complexos.

Palavras-chave

Ferramentas de Mapeamento Visuais; Técnicas de Visualização; Visualização de Informação Complexa; Interfaces Visuais; XMApper
# Contents

1. Introduction .......................................................................................................................... 1
   1.1. Problem and Motivation .................................................................................................. 2
   1.2. This Work as a Concept Model .................................................................................... 3
   1.3. Objectives ....................................................................................................................... 4
   1.4. Main contributions ......................................................................................................... 5
   1.5. Dissertation outline ....................................................................................................... 5
2. Related Work ......................................................................................................................... 7
   2.1. Visual Mapping Tools ..................................................................................................... 8
      2.1.1. MapForce ............................................................................................................... 8
      2.1.2. BizTalk Mapper ...................................................................................................... 9
      2.1.3. Stylus Studio XML Mapper .................................................................................. 10
      2.1.4. Clio Schema View ................................................................................................ 11
      2.1.5. COMA++ .............................................................................................................. 12
      2.1.6. Anatomy of a Visual Mapping Tool’s Interface ..................................................... 13
   2.2. Interface Paradigms: Large Mappings Issues ................................................................. 14
      2.2.1. Mapping Board Improvement ............................................................................... 15
      2.2.2. Schema View Improvement .................................................................................. 16
      2.2.3. Navigation ............................................................................................................ 17
      2.2.4. General Improvement ........................................................................................... 18
      2.2.5. Analysis and Discussion ....................................................................................... 19
   2.3. Development Technology ............................................................................................... 21
      2.3.1. Prototype + Scripty2 ............................................................................................... 21
      2.3.2. jQuery ................................................................................................................... 22
      2.3.3. Ext JS ..................................................................................................................... 22
      2.3.4. MooTools .............................................................................................................. 23
      2.3.5. Dojo ....................................................................................................................... 23
      2.3.6. ZK .......................................................................................................................... 23
      2.3.7. GWT – Google Web Toolkit ................................................................................... 24
      2.3.8. Vaadin ................................................................................................................... 25
      2.3.9. Analysis and discussion ......................................................................................... 25
   2.4. Summary .......................................................................................................................... 27
3. The Problem .......................................................................................................................... 29
   3.1. Context of the Problem ................................................................................................... 30
      3.1.1. About Models ....................................................................................................... 30
      3.1.2. About Visual Modeling ......................................................................................... 31
3.1.3. About the Mapping of Models .................................................................31
3.1.4. About Visual Mapping Tools .................................................................32
3.1.5. The MDR and its Mapper .................................................................33
3.2. Reformulate the Problem: Large Mapping Visualization .........................34
3.3. Problem's Requirements ........................................................................36
   3.3.1. Business Requirements .................................................................36
   3.3.2. Graphic Interface Requirements ..................................................37
3.4. Summary .................................................................................................37
4. Proposed Solution .........................................................................................38
   4.1. Proposed Solution ..................................................................................39
      4.1.1. Mapping Board Improvement .......................................................39
      4.1.2. Schema View Improvement .........................................................40
      4.1.3. The sum of all parts .....................................................................42
   4.2. Development Scenario ..........................................................................43
      4.2.1. The GWT as a Web Development Framework ..............................43
      4.2.2. Platform – The XMApper ............................................................43
   4.3. Implementation .......................................................................................45
      4.3.1. [SP1.1] Connection Visibility ......................................................45
      4.3.2. [SP1.2] Connection Render ..........................................................46
      4.3.3. [SP1.3] Connection Argument Visibility ......................................46
      4.3.4. [SP2.1] Quick Filters ..................................................................47
      4.3.5. [SP2.2] Tree Element Connection Status ......................................48
      4.3.6. [SP2.3] Sorting ..............................................................................49
      4.3.7. [SP2.4] Connection Statistics ......................................................50
   4.4. Summary ...............................................................................................51
5. Evaluation ....................................................................................................53
   5.1. Methodology .........................................................................................54
      5.1.1. Prototype versions ........................................................................54
      5.1.2. Participants ...................................................................................55
      5.1.3. Tasks .............................................................................................55
      5.1.4. Test map ......................................................................................56
      5.1.5. Final survey ..................................................................................57
      5.1.6. Main study procedure .................................................................57
      5.1.7. Statistics study procedure ............................................................57
   5.2. Results and Discussion .........................................................................57
      5.2.1. Task times ....................................................................................57
5.2.2. Satisfaction data .................................................................................59
5.2.3. Usability issues ..................................................................................61
5.3. Summary .................................................................................................61
6. Conclusions .................................................................................................63
6.1. Future Work ..............................................................................................68
References .......................................................................................................69
Appendices .......................................................................................................72
A. Glossary of Concepts ..................................................................................73
B. Glossary of Acronyms ...............................................................................74
C. The XMappe – User Manual Preview – Getting Started .........................75
D. User Study: Final Survey ............................................................................84
E. Visualizing Schema Mappings: Tags ..........................................................85
List of Figures

Figure 1. Concept map depicting this dissertation’s work ................................................................. 4
Figure 2. Screenshot of Altova’s MapForce ......................................................................................... 9
Figure 3. Screenshot of BizTalk Mapper ............................................................................................. 10
Figure 4. Screenshot of StylusStudio XML Mapper .......................................................................... 11
Figure 5. Screenshot of Clio’s Schema View ....................................................................................... 12
Figure 6. Screenshot of COMA++ ...................................................................................................... 13
Figure 7. The tree principal areas of BizTalk Mapper’s interface ......................................................... 14
Figure 8. Overview of the use cases commonly found on Visual Mapping Tools ......................... 32
Figure 9. Concept map depicting this work on the MDR’s context ..................................................... 33
Figure 10. Example of a complex mapping between two models ....................................................... 34
Figure 11. Example of a confuse mapping in BizTalk’s Mapper .......................................................... 35
Figure 12. Screenshot of the XMApper base interface ...................................................................... 44
Figure 13. Example of connection’s rectilinear render ...................................................................... 46
Figure 14. Example of connection’s direct render ............................................................................. 46
Figure 15. Example of connection arguments on selected connection .............................................. 47
Figure 16. Screenshot depicting the interface with no filters applied ................................................ 48
Figure 17. Screenshot depicting the interface with filters applied (on the source schema) ......... 48
Figure 18. Example of a tree sorted alphabetically ............................................................................ 50
Figure 19. Example of a tree sorted by Connection Status ............................................................... 50
Figure 20. The statistics window with the pie chart with selectable slices ....................................... 51
Figure 21. The map used in the user study experiments .................................................................... 56
Figure 22. Average task times for each of the three prototype versions ........................................... 58
Figure 23. Average times by prototype version and task ................................................................. 58
Figure 24. Average task times for prototype version without Statistics (Version C) and with Statistics (Version D) for the two studied types of task .................................................. 59
Figure 25. Average user satisfaction ratings for the three prototype versions studied ................ 60
Figure 26. XMApper interface’s overview ......................................................................................... 75
Figure 27. The different panels ......................................................................................................... 76
Figure 28. The two schema panels: (a) the source schema panel; (b) the target schema panel .......... 76
Figure 29. Schema panel header and toolbar ..................................................................................... 77
Figure 30. Schema panel drop-down menu ....................................................................................... 77
Figure 31. Source Schema XML Element .......................................................................................... 78
Figure 32. Target Schema XML Element .......................................................................................... 78
Figure 33. Source schema XML String Attribute ............................................................................... 78
Figure 34. Target Schema XML Attribute........................................................................78
Figure 35. Mapping board panel ..................................................................................79
Figure 36. Mapping board header and toolbar .................................................................80
Figure 37. Mapping board ................................................................................................80
Figure 38. Screenshots of the Properties Panel: (a) Properties Panel when an element is
selected; (b) Properties Panel when inactive (no selection). ..............................................81
Figure 39. Properties panel header ..................................................................................82
Figure 40. The metadata tab ............................................................................................82
Figure 41. The toolbox ......................................................................................................83
List of Tables

Table 1. Results for the evaluation of the mapping tools and their support for the interface paradigms .................................................................20
Table 2. Comparison of characteristics from the different Ajax frameworks. .....................26
Table 3. Summary of the proposed solution. ..................................................................42
Table 4. The connections possible visibilities and their trigger conditions. .........................45
Table 5. Tree element representation depending on connection and selection situation. .......49
Table 6. Two of the task sets used in the main study .......................................................55
Table 7. The two task sets used in the statistics study .....................................................56
Table 8. Average user satisfaction ratings for the 3 versions of the mapper. Questions are the original from the survey, but higher ratings indicate higher satisfaction for all questions. ....60
Table 9. Results for the evaluation of the mapping tools and their support for the interface paradigms ........................................................................67
Table 10. The schema elements available identification icons ........................................79
1

Introduction
1. Introduction

Today, information technology (IT) plays a central role in our world, being all around us even when we’re unaware of its existence. Information is always being stored and shared between electronic devices, from our laptops and mobile phones to bank ATMs and electronic train tickets. Structuring and organizing all this information has become a vital need for IT organizations, which not only manage it, but often share it between themselves.

This modern scenario has given rise to the ever-evolving field of information modeling, whose focus is to structure and organize, i.e. model, information to make the tasks of managing, sharing and working with information a simpler job. But modeling isn’t an easy task, and requires a great deal of vision by the people doing it. Visually representing information models has become a common way to address this issue, leading to the advent of visual modeling tools, i.e. software that allows modelers to do their job through a computer, with all the advantages they bring, using visual representations of the information and its modeling operations. Also, it’s common to find in the interface of such tools specific techniques for representing information. Specific, because they help the tool’s interface to adapt to unusual situations, created when trying to display information with unusual characteristics (e.g. large quantities, complex organization, special words). For the sake of simplicity, these visualization techniques are here on out referred to as interface paradigms.

A more recent interest in the representation of models is coming from the data integration and interoperability research areas, which look at modeling also as a way to represent how different models may relate to each other, so that data (i.e. information) can be combined and reused in different contexts. This process of defining relationships between two different models is commonly referred to as mapping, and the product of such process as mapping information, gathered within a mapping model. However, 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 goal of making it easy for a designer to establish mappings [2].

A current problem with such applications is that they are unable to cope with the growth in size and complexity of both the information models and the mappings that can be established between them.

1.1. Problem and Motivation

Currently, information modeling can be used to describe rather extensive domains, requiring proportionally extensive information models to describe them. This all transcends to the mapping between models process. In such a scenario, one can state the problem for this
dissertation with a question: how will a visual mapping tool deal with such extent models while, at the same time, represent the specificity of the relationships that can be established between them? To find the answer to this question, this dissertation will evaluate current visual mapping tools, bearing these questions in mind, and propose new interface paradigms that can improve them.

The motivation of this dissertation is to contribute for the development of a visual mapping tool to be integrated in the Metadata Registry (MDR), a project under development by INESC-ID\(^1\), and to improve visual mapping tools interface in general. The MDR is 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 (more information in Section 3.1.5.)

### 1.2. This Work as a Concept Model

The concept map showed in Figure 1 is a succinct presentation of this work and what it aims to achieve. The highlighted concepts say it all: this dissertation proposes to find a set of interface paradigms (the solution) addressing visualization problems that arise when trying to work with large mappings within visual mapping tools. As will be seen later, these problems can affect the Schema View (where the models to be mapped are represented), the Mapping Board (where the mapping operations are done), and the navigation of the whole mapping workstation. This work will focus efforts in developing paradigms that approach problems with the Schema View and Mapping Board, although some of these will be no doubt inherently important for the navigation itself.

---

\(^1\) [http://www.inesc-id.pt/](http://www.inesc-id.pt/)
1. Introduction

Figure 1. Concept map depicting this dissertation's work

1.3. Objectives

The overall aim of this work is to contribute to the development of visual mapping tools’ interfaces, by developing and evaluating one or more interface paradigms that improve the current approaches regarding the visualization of large mappings.

In detail, the objectives of this project are to:

1. Evaluate the interface paradigms and representation techniques used by current tools, projects and research in the field.
2. Assess and compare various web development frameworks to determine the one best suited to develop and implement the solution.
3. Create a prototype for the MDR’s mapper, with an interface similar to that of current visual mapping tools.
4. Develop and implement one or more interface paradigms that allow the user to work more efficiently with models and mappings of large dimensions.
5. Validate that the proposed interface paradigms will have a positive impact on user’s performance and satisfaction.
Objective 1 and 2 correspond to the related work and state of the art research and analysis. Objective 3, 4 and 5 are the main core of this work. The new interface paradigm(s) will be developed and implemented on a prototype, the XMApper (for more info see Section 4.2.2), which will be used as testbed for evaluation. Results will then be analyzed and their success measured according to the fulfillment of these objectives.

1.4. Main contributions

1. Development of the XMApper’s interface.

2. Five new interface paradigms, and two improved ones, for working with larger models and mappings.

1.5. Dissertation outline

The remaining chapters of this document are organized as follows. Chapter 2 is a compilation of this dissertation’s Related Work, starting with an overview of some of the most prominent visual mapping tool and their interface, followed by a listing and analysis of their interface paradigms related to the Large Mappings Visualization predicament. Finishing the related work is a report on the research made for finding the ideal development technology for this work, and the results of that research.

Chapter 3, The Problem, begins by giving a more in-depth contextualization for this work's problem, and follows by reformulating it with more detail. The chapter ends with the listing of the guide requirements for the proposed solution.

In chapter 4, the Proposed Solution for the dissertation’s problem is presented and explained, followed by a quick introduction of its development scenario and a detailed section about the implementation of each of solution’s components.

The report for the Evaluation of the proposed solution, a user study, is made in chapter 5. The followed methodology is explained in detail and, afterwards, the achieved results are presented and discussed.

Finally, chapter 6 presents the final Conclusions of this work, wrapping up all that has been said and presenting the final thoughts about the developed solution, the obtained results and the future work.
Annexes A and B are the glossaries of the most important concepts and acronyms. Note that the concepts in the glossary are found within the text written in bold for a better reference. Also annex is (C) a small preview of the XMApper User Manual\textsuperscript{2}, (D) the user study’s background and satisfaction survey, and (E) a small specification draft of a technique that ended up not being used for this work (Tags).

Related Work
2. Related Work

This chapter presents some of the current applications for model mapping (Section 2.1) and evaluates their interfaces, gathering interface paradigms for analysis (Section 2.2). Finally, the last section presents a small evaluation of the current technology for web applications development, aiming to determine the one best suited for developing and testing a web-based visual modeling tool.

2.1. Visual Mapping Tools

Visual mapping tools are software applications which can be used to visually establish mappings between models, defined within a schema (usually in XML\(^3\)). In such programs, mappings are specified by allowing the users to make connections between visual representations of source and target schemas’ entities. The mappings 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 mapping function application. More information about visual mapping tools can be read in Section 3.1.4.

The current section will present some of the more prominent visual mapping tools and the current developments made for evolving their interface.

2.1.1. MapForce

MapForce\(^4\) is a visual tool, developed by Altova\(^5\), for any-to-any data mapping and integration, allowing the defining of mappings between any combination of not only XML, but also database, flat file, EDI, Excel, XBRL and web services formats, using a design interface, with visual data processing functions and various tools for mappings navigation. The MapForce’s “Design Pane”, the mapper’s main window, is free dragging zone where multiple source and target schemas may be represented as well as function boxes to represent data processing functions (in Figure 2, the big center panel).

\(^3\) http://www.w3.org/XML/

\(^4\) http://www.altova.com/mapforce.html

\(^5\) http://www.altova.com/
2.1.2. BizTalk Mapper

The *BizTalk’s Mapper*\(^6\), included as a module in Microsoft’s BizTalk Server – which becomes active when a new map is added to a BizTalk project or when an existing map is opened – is another visual editor that allows the creation of a map between XML schemas by expressing their correlation graphically. Recent advances, by Microsoft’s interface development teams [1], have introduced a lot improvements in the last version of the mapper included with BizTalk Server 2010. BizTalk mapper uses a three panel configuration with the source schema represented in the leftmost, the mapping grid in the center and the target schema in the rightmost panel (Figure 3).

---

\(^6\) [http://www.microsoft.com/biztalk/](http://www.microsoft.com/biztalk/)
2. Related Work

2.1.3. Stylus Studio XML Mapper

Stylus Studio XML IDE provides developers and data integration specialists with a comprehensive toolset for managing XML data transformation and aggregation. One of those tools is the XML Mapper\(^7\), a many-to-one schema mapper, designed to simplify XML data integration projects by providing visual XML mapping tools which operate on standard XSLT or XQuery files. The mapper's main interface layout consists of three windows, from left to right: an XML source document window, the mapping window and the target document window (Figure 4). The editing environment may also split horizontally in two panels, the top being the visual XML mapping graphic and the bottom displays the underlying source code, being both automatically synchronized.

\(^7\) http://biztalkmessages.vansplunteren.net/

\(^8\) http://www.stylusstudio.com/xml_mapper.html
Figure 4. Screenshot of StylusStudio XML Mapper

2.1.4. Clio Schema View

Created by IBM’s research department, Clio\(^9\) is a system for managing and facilitating the complex tasks of heterogeneous data transformations and integration [4], and its main feature is the semi-automatically definition of mappings from one or more source schemas to a target schema (these schemas can be of any combination of relational databases and XML data). However, Clio permits corrections to be made by the user, and to facilitate this process, it makes use of a graphical user interface to show information [5]. In this Schema View mode, users see a visual representation of the schemas including any generated information (the connecting lines in Figure 5). This view may be used to edit and further augment the mappings. Late reports claim that Clio is on its way to become a fully-fledged industrial tool [6].

\(^9\) http://www.almaden.ibm.com/cs/projects/criollo/
2. Related Work

![Figure 5. Screenshot of Clio's Schema View](image)

2.1.5. COMA++

Created 10 years ago at the University of Leipzig, COMA++\(^{10}\) is a matching tool for generic schemas still under development. COMA (COmbining MAthchers) main function is, has its name suggests, schema and ontology matching, i.e. deriving correspondences between the elements or concepts of two or more data models, such as XML schemas or formal ontologies [7]. The graphical interface (Figure 6) offers a variety of interactions, allowing the user to influence in the match process in many ways, like edit a mapping or manual reuse of previous match results.

\(^{10}\) [http://dbs.uni-leipzig.de/Research/coma.html](http://dbs.uni-leipzig.de/Research/coma.html)
2.1.6. Anatomy of a Visual Mapping Tool’s Interface

The interface of a visual mapping tool varies from tool to tool. Taking a look at the interface screenshots of the previously presented mapping tools (Figure 2 to Figure 6), one crucial part stand out as common to all: the representation of the source and target schemas as hierarchical trees, where each node represents a schema element and navigation is possible by expanding and collapsing their nodes. This is called the Schema View.

Also common to the three first tools, is a mapping area where the mappings (represented by lines connecting the elements of the map) and the mapping functions (the small rectangle boxes in MapForce and the small square boxes in the BizTalk and Stylus Studio’s mappers) are placed by the user designing the map. This is called the Mapping Board. Another common feature, although hidden in all previous examples except in MapForce’s screenshot (Figure 2) is the Function Toolbox or Function Library panel (the far-left panel in Figure 2), which contains all the functions possible to add and use in the current mapping.
2. Related Work

These are the most essential elements of a visual mapping tool’s interface. Figure 7 shows the same picture as Figure 3, but now with a color scheme to help identify these four principal areas (see legend).

Figure 7. The tree principal areas of BizTalk Mapper’s interface

Legend:

- Yellow – The button to show the collapsed Function Toolbox.
- Green – The Schema View of the source schema.
- Red – The Mapping Board (with a mapping function and mapping connections)
- Blue – The Schema View of the target schema.

2.2. Interface Paradigms: Large Mappings Issues

As previously discussed, interface paradigms are representation techniques that are applied to an application’s interface with the goal of displaying information with unusual characteristics, which creates representational problems for the interface, thus viewing problems for the user. In the scope of this dissertation, such problems come from the representation of large mappings. This section starts by listing and describing some of the relevant interface paradigms applied by the mapping applications presented in the last section (and their recent research), identifying their advantages. The paradigms will be tagged by
application and divided into four targeted areas: Mapping Board and Schema View Improvement (with visibility, organization, space and other optimizations regarding the Mapping Board and the model trees); Navigation (mappings navigation methods) and General Improvement (with optimizations that affect multiple areas, including Mapping Board and trees, at the same time).

**Note:** Although the term ‘functoid’ is only really used by the BizTalk Mapper, in this work it will be used to refer to all mapping function boxes, either belonging to BizTalk Mapper, MapForce or Stylus Studio XML Mapper. Also, abbreviations used: MapForce (MF), BizTalk Mapper (BTM), Stylus Studio XML Mapper (SS XML Mapper or simply SSXM).

2.2.1. **Mapping Board Improvement**

- **Auto scroll functoids** (BizTalk Mapper [8] [9]) - This is a BizTalk Mapper function that was recently researched, it auto scrolls functoids into view (and aligns them) when a connection to those functoids is selected, so the user can view the whole mapping. The scrolling is animated so the user doesn’t get disoriented.

- **Bendable links** (BTM—research [1] [10], Clio [11], COMA++ [12]) – Clio and COMA++ render connections as curvy lines instead of straight from source-target, and in their scenario with no mapping this helps differentiate connections in such a small rendering space. The BizTalk Mapper approach is different, only bending the links when these pass behind a functoid, which would have made the user unsure if the link connected to the functoid or not. This feature is still under research and isn’t included in the latest version of BTM.

- **Fit in canvas** (Sylus Studio XML Mapper [13]) – The SS XML Mapper allows the user to widen the panes displaying the source and target documents, which implies reducing the size of the Mapping Board canvas in the process. To address this issue they introduced the “Fit in Mapper Canvas” function, which, with the click of a button, redraws the mapping diagram in whatever space is currently available to the Mapping Board canvas.

- **Functoid suppression** (SSXM [13]) – When reloading an old mapping, on the SS XML Mapper, existing functoids disappear leaving a simple line connecting the elements from source to target schema. Each of these simple lines only has a small circle in the middle, which, on hovering, shows a small description of the mapping operations done in that connection.

- **Functoids ‘invalid’ state** (BTM [14]) – BizTalk Mapper places an extra indicative icon on top of a functoid when its current state is ‘invalid’. This allows the user to know the state of the functoid with no space intrusion, also helping cleaning unnecessary functoids.

- **Functoids as small single-cell boxes** (BTM [14], SSXM [13]) – BizTalk Mapper and SS XML Mapper have a similar approach when representing mapping functions on the Mapping
2. Related Work

Board. They do so as simple small boxes with only an icon indicating its function and some attachment ports where the links can connect. The input and output of the functions is later configured via properties panel and until then the **funtoid** will remain in an 'invalid' state.

**[P1.7] Grid pages** (BTM [14]) – In BizTalk Mapper the Mapping Board can have multiple grid pages, which basically means there can be multiple Mapping Board instances. The user can isolate different aspects of the mapping in different pages, and then view and work with those pages separately. Pages are represented as tabs and can be added, renamed, deleted moved between and within them, and so on.

**[P1.8] Links visibility by tree node** (SSXM [13], COMA++ [12]) – A simple function in COMA++ that, when a schema element gets selected, hides all links that don’t connect to that element allowing a cleaner view and also the possibility of showing other link related info. In SS XML Mapper’s case the schema elements have a right-click context menu, which allows for the same functionality or the opposite, i.e. hide all links connecting to this element.

**[P1.9] Positioning of connection’s annotations** (MapForce [15]) – MapForce allows the showing of annotation’s text alongside the connections on the design panel. More importantly it gives many configuration possibilities for this feature, like placing above or below the link representation or not showing them at all, which can help on connections-clogged mappings.

**[P1.10] Show relevant links** (BTM [8] [9], SSXM [13]) – This is a toggling function that allows the user to clean the Mapping Board view by changing the visibility of board connections that are not relevant to the actual view of the mapping (board plus model trees). The approach of BizTalk Mapper and SS XML Mapper is slightly different. Both completely hide the connection when neither extremity element is visible (BTM has the possibility to only blur out), but while the SS XML Mapper also hides when only one extremity is visible (showing a green arrow on the visible element), BTM still shows the link, but it’s represented blurred and dashed.

**[P1.11] Visual alignment guides** (MF [15]) – To help user orientation and Mapping Board organization on MapForce’s free-placement design pane, alignment guides appear when a component is being dragged on the panel, showing leveling possibilities with other components.

**[P1.12] Visual function builder** (MF [15]) – MapForce’s visual function builder is a separate editing area that allows the user to combine multiple **funtoids** into one that returns a single result, optimizing space and reusing of that combination.

2.2.2. Schema View Improvement

**[P2.1] Auto scroll trees** (BTM [8] [9]) – Similarly to the auto scroll for **funtoids**, this is a function recently implemented in the BizTalk Mapper, which auto scrolls a tree element into
view when a connection to it is selected. The animated transition is also present, so the user doesn’t get disoriented.

**[P2.2] Expand and collapse trees** (MF [15], BTM [14], SSXM [13], Clio [11], COMA++ [12]) – In all of the analyzed visual mapping tools, the representation of source and target schemas in expand/collapsible tree panels was omnipresent. The tree’s hierarchical order is a faithful reflection of a XML document’s organization, while expanding and collapsing is its naturally associated method for view simplification and navigation.

**[P2.3] Sibling coalescence** (BTM [8] [9]) – In order to address the problem of non-relevant information being displayed in the schema tree view, BizTalk Mapper’s research team introduced a method for coalescing sibling nodes deemed not relevant at the moment. Relevance is based in two simple factors: whether a schema element or any of its descendants has a link, and the selected and highlighted schema elements. Unconnected, unselected and un-highlighted siblings of a connected element will be coalesced into a single node with a down-arrow icon, while the linked sibling will remain visible. Hovering over one of these nodes will produce a tooltip that describes what has been coalesced. Clicking the down-arrow icon will make the coalesced nodes visible and the down-arrow an up-arrow, which, upon clicking, will coalesce the nodes once again.

### 2.2.3. Navigation

**[P3.1] Overview** (MF [15], BTM [14]) – MapForce’s overview window allows users to visualize an entire data mapping project in a global view, and zoom in on specific areas as needed. Also, while scrolling through the project itself, the overview indicates the design panel view position within its map, helping user orientation when navigating large maps. The BizTalk Mapper’s version, called ‘grid preview’, is very similar, but the only components it shows are functoids present in the Mapping Board.

**[P3.2] Panning mode** (BTM [8] [9]) – BizTalk Mapper’s ‘Panning Mode’ toggle allows the user to activate the option of navigating the Mapping Board grid by dragging on the board itself, producing an intuitive way to navigate that can help orientation in large mapping diagrams.

**[P3.3] Search function** (MF [15], BTM [8] [9], SSXM [13]) – Simple search function is deemed very valuable for navigating in MapForce, BizTalk Mapper and SS XML Mapper. MF and SSXM allow searching of source and target schemas to instantly navigate to a specific node, while BTM adds to this the possibility of also searching the Mapping Board components (functoids and links), with options allowing searching for almost any text, from name labels to annotations and even custom made functoid scripts.

**[P3.3.1] Search: Auto tree view collapse** (BTM-research [1] [10]) – This is research technique, not included in the last version of BizTalk Mapper, which enable the automatic
2. Related Work

collapsing of nodes without hits when using the mapper’s search engine to find nodes in the source or target schemas, enabling the quick location of search results.

[P3.3.2] Search: Interactive scrollbar highlighting (BTM-research [1] [10]) – Another research feature, not implemented in the latest version, is the ‘Interactive Scrollbar Highlighting’ that presented interactive colorful tick marks in the scrollbar when a search was made, showing where search hits occurred and scrolling the schema to an element when its corresponding tick mark was clicked. This optimizes searching in larger-than-view schemas.

[P3.6] Zoom grid (BTM [8] [9]) – BizTalk Mapper also allows the user to control the zoom level of the Mapping Board grid. This can be used has a quick global view orientation, very much like ‘Overview’, with similar benefits, and the addition of letting the user isolate-by-zooming specific areas of a larger diagram,

2.2.4. General Improvement

[P4.1] Background color (COMA++ [12]) – COMA++ changes the model trees’ background color to indicate to the user that edit mode is enabled. This is a very clean way to indicate simple important information to the user without any space constrains.

[P4.2] Empathizing selected objects (BTM [8] [9]) – BizTalk Mapper employs an object empathizing technique, which highlights a selected object (schema element, functoid or connection) in two steps: first there is highlight propagation from that object, which means that all links are followed in both directions, and every schema element, link, and functoid that is relevant to the selected item is highlighted as well. For complex maps that might not be sufficient due to the density of links, so as a complement technique all non-highlighted links and functoids are de-empathized: links are drawn greyed out and functoids with 30% transparency, effectively placing them on the background.

[P4.3] Information popups (MF [15], BTM [14]) – Information popups, or tooltips, can be very useful in visual mapping tools, providing a good way for the user to have easy-to-access information with virtually no space needed to show it. MapForce applies this technique in certain parts of the mapping connections where the user can view additional information, such as the mapping target item(s) or its datatype. On the other hand, BizTalk Mapper uses tooltips on functoids, providing quick access to user inputted comments and annotations.

[P4.4] Link color (COMA++ [12]) – Latest versions of COMA++ vary the color of the links accordingly to a similarity rating (which can go from 0 to 1.0), going from green to bright red. Similar to the background color changing, this automatically gives link classifying information to the user, without any intrusion in terms of space.
No Mapping Board (Clio [11], COMA++ [12]) – Not exactly an interface paradigm, but worth noticing nevertheless, Clio and COMA++ opted (no doubt due to their focus in match making) for not having any Mapping Board to place links and functoids. Links still exist, but connect the directly adjacent source and target schema representations, and to replace functoids the user can directly attach transformation functions to the links. This contribute to a simpler mapping representation, but links cluttering still happens and, more importantly, the user loses the ability of looking to a specific mapping and immediately see its conversion information and quickly edit it if needed.

Show\hide information toggles (MF [15]) – Toggle buttons, or options, can be used for efficiently configure a view, regarding the information provided versus the space available ratio. MapForce gives two example of this, making use of two quick to access toggle buttons, one that hides\shows schema datatypes for each element or attribute and another that does the same with displaying a functoid library name on its header.

Analysis and Discussion

Table 1 lists all identified interface paradigms on the studied tools, and by looking at it, some brief observations can be made. Overall, there is a distinct difference between MapForce and the BizTalk’s Mapper when compared to the other three applications, as they have much more paradigms. Although being also a professional tool, Stylus Studio XML Mapper comes in second in this aspect, while the prototypes Clio and COMA++ come last, a reflection of their research focus not being the interface. Nevertheless, SS XML Mapper has a very considerable quantity of interface paradigms affecting the Mapping Board and its elements.

Also by counting the number of paradigms, one can observe that the Mapping Board, and the representation of its elements, has been subject to the implementation of various optimization techniques when compared to the Schema Views, which can be due to the Schema View’s simpler nature, but is a potential improvement spot nevertheless.

Regarding Navigation paradigms, the Overview and specially the search function are adopted by the most prominent applications, while the versatility of the Information Popups lands the same classification on the wider General Optimization area.

Still, 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.
## 2. Related Work

<table>
<thead>
<tr>
<th>Interface Paradigms</th>
<th>MapForce</th>
<th>BizTalk Mapper</th>
<th>SS XML Mapper</th>
<th>Clio</th>
<th>COMA++</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auto scroll functoids</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bendable links</td>
<td>X¹¹</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Fit in canvas</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Functoid suppression</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Functoid 'invalid' state</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Functoid as small single-cell boxes</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Grid pages</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Links visibility by tree node</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Positioning connection annotations</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Show relevant links</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Visual alignment guides</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Visual function builder</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Auto scroll trees</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Expand and collapse trees</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Sibling coalescence</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Overview</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Panning mode</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Search function</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Zoom grid</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Background color</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Empathizing selected objects</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Information popups</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Link color</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>No Mapping Board</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Show/hide information toggles</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

¹¹ Still in research.

Table 1. Results for the evaluation of the mapping tools and their support for the interface paradigms.
2.3. Development Technology

Web 2.0 is becoming a popular concept, referring to the recent wave of interactive web applications, which intend to help people collaborate and provide interactive services, not just static HTML. This became possible, in part, by means of the Asynchronous JavaScript and XML (or AJAX\(^{12}\)) technology. In reality, AJAX represents several technologies, each flourishing in its own right, coming together in powerful new ways [16]. Ajax incorporates: standards-based presentation using XHTML\(^{13}\) and CSS\(^{14}\); dynamic display and interaction using the Document Object Model (DOM\(^{15}\)); data interchange and manipulation using XML and XSLT\(^{16}\); asynchronous data retrieval using XMLHttpRequest\(^{17}\); and JavaScript\(^{18}\) binding everything together. In comparison with those old web applications constrained by HTML, Ajax based applications bring user interface and functionality with rich, responsive, intuitive, interactive and dynamic features entirely inside the browser without plug-in or other software required [17]. Besides rendering HTML and executing script blocks, the browser plays a more active role in processing HTTP requests and responses in these applications. Instead of traditional “click, wait and refresh” user interaction, these rich internet applications show better performance and web experience since they could add or retrieve users’ requests asynchronously without reloading web pages [18].

Next in this section is an evaluation of current JavaScript technology for web applications development, aiming to determine the framework best suited for this work’s purposes.

2.3.1. Prototype + Scripty2

Prototype\(^{19}\) is a JavaScript framework that aims to ease the development of dynamic web applications. In 2007, it was regarded to be the most widely used AJAX framework and its main quality was the compatibility towards different browsers. Even so, Prototype is one of the few JavaScript frameworks that does not include user interface (UI) components out-of-the-box.

\(^{13}\) http://www.w3.org/TR/xhtml1/
\(^{14}\) http://www.w3.org/Style/CSS/
\(^{15}\) http://www.w3.org/DOM/
\(^{16}\) http://www.w3.org/TR/xslt20/
\(^{17}\) http://www.w3.org/TR/XMLHttpRequest/
\(^{18}\) http://www.javascript.com/
\(^{19}\) http://www.prototypejs.org/
2. Related Work

Instead, all these elements are included on its sister library, Script.aculo.us (or, as the latest version is known, Scripty2).

Scripty2 comes with a wide range of UI effects and behaviors. Effects such as: highlighting, morphing, shaking and sliding. Furthermore, Scripty2 also provides support for drag-and-drop, sliders and auto-completers. On the downside, when comparative to other frameworks, in Scripty2 all the UI elements design is relieved to the developer, providing no standard skin to work from.

2.3.2. jQuery

Unlike Prototype, the jQuery\textsuperscript{20} JavaScript library brings in its core a few basic UI components. Some of these are similar to those of Scripty2, such as sliding and fading. However, for more advanced effects, it’s required to install the jQuery UI library, which, besides the extra effects, also packs interactivity functions like drag-and-drop, resizing and sorting. Unlike Scripty2, jQuery UI has some widgets which make the development of attractive interfaces a lot easier. Some of these widgets are: accordion, auto-complete, date picker, dialog, progress bar, slider and tabs. These UI components are completely themeable and jQuery provides an extensive selection of themes that can be used to blend a component with the particular style of a website or web application.

2.3.3. Ext JS

The Ext JS\textsuperscript{21} framework, developed by Sencha, was originally created has a Yahoo! User Interface extensions library. Unlike the Prototype and jQuery frameworks, Ext JS already includes widgets on its core install, most notably some very powerful grid controls with support for inline editing, pagination, filtering, grouping, summaries, buffering, and data binding. These widgets have a professional look-and-feel and are completely customizable. Other widgets include tabs, charts, dialog windows, trees, drag and drop support and layout managers. All this features are elements that can be seen on the common operative system and, to illustrate just that, one of the Ext JS examples is indeed a Web Desktop\textsuperscript{22}.

\textsuperscript{20}http://jquery.com/
\textsuperscript{21}http://www.sencha.com/products/js/
\textsuperscript{22}http://xant.us/ext-ux/lib/ext-3.0.0/examples/desktop/desktop.html
2.3.4. MooTools

At first glance, the MooTools framework can be compared with jQuery. Like it (and Prototype) it does not include any base UI components or widgets, packing only a limited amount of effects. Just like jQuery, you have to install an extension to get more advanced features, the MooTools.More.js. However, and unlike jQuery, that extension not only adds UI components, but also provides additions to the array, date, hash, string and element classes. Besides the additional effects and widgets, the MooTools.More.js also brings support to the drag-and-drop feature and controls such as accordion, sortable HTML tables, scrollbars and tooltips. Like Scripty2, however, it is needed to implement the design of all these controls.

2.3.5. Dojo

Dojo is also a JavaScript framework that eases up the development of web applications interfaces. Its library provides many layout and form controls. Dojo is based on extensible widget model, using JavaScript to configure the widgets and define different behaviors and interface styles.

2.3.6. ZK

ZK is an event and component-driven widgets framework to create rich graphical user interfaces for web applications [19]. The ZK main elements are an Ajax-based event-driven architecture, a vast array of XML User Interface (XUL) and XHTML components and its own markup language, called ZK User Interface Markup Language (ZUML).

All this elements set up a framework that allows the development of web applications in Java, with little programming and none JavaScript knowledge. The developers design the pages for their applications with XUL/XHTML components and manipulate them upon events triggered by en user’s activity. It’s similar to the programming model used for creating GUI-based desktop applications. ZUML’s purpose is designing simplification.

ZK takes the so called server-centric approach, in which the content synchronization of components and the event pipelining between clients and servers are automatically done by the engine and Ajax plumbing codes are completely transparent to web application developers.

---

23 http://mootools.net/
24 http://www.dojo.org/
25 http://www.zkoss.org/
2. Related Work

Consequently, the end users get similar engaged interactivity and responsiveness of a desktop application, while the programmers experience development with a similar simplicity to that of desktop applications [20].

2.3.7. GWT – Google Web Toolkit

GWT\(^{26}\) is an emerging technology that helps the integration of the existing and efficient tools for Java development into the interactive web world. The main feature of GWT is the ability to compile Java programming code into a combination of JavaScript and HTML files, ready for browser execution. By programming in Java, GWT allows the applying of current software engineering techniques in an area that lacks solid development tools \(^{21}\).

Using a compiler to generate JavaScript, creates an opportunity to optimize the code that otherwise would be difficult \(^{22}\). The GWT architecture employs efficient and reusable solutions to resolve recurring Ajax development problems, like asynchronous remote procedure calls\(^{27}\), history management, bookmarking, internationalization and cross-browser portability. Unlike ZK, GWT is client-centric, with all the UI processing being made in the user-side of the application.

Some of GWT strong points are:

- **GWT Java-to-JavaScript Compiler** – Translates and optimizes code from Java programming language to JavaScript.

- **GWT Development Mode** – Allows programmers to run GWT applications in development mode (the application runs on the JVM\(^{28}\) without compiling to JavaScript).

- **GWT Web UI class library** – A set of custom interfaces and classes for creating widgets. Allows extension using external libraries like Ext-GWT\(^{29}\), Smart GWT\(^{30}\) or GWT Pleso\(^{31}\).

- **GWT Designer** – A GWT plug-in, which allows the easy creation of an application GUI using a visual designer with drag-and-drop capabilities. The Java code for the GUI is automatically generated.

\(^{26}\) [http://code.google.com/webtoolkit/](http://code.google.com/webtoolkit/)


2.3.8. Vaadin

Vaadin\(^{32}\), like ZK, is a server-sided AJAX web application development framework that enables developers to create user interfaces with Java. It provides a library of ready-to-use interface components and the possibility of creating new ones. Vaadin focuses on the ease-of-use, re-usability, extensibility and meeting the requirements for large enterprise applications development.

Similar to ZK, Vaadin’s key idea in the server-driven programming model is that it allows developers to forget the web, making the programming much like it would be if developing a Java desktop application. On the other hand, for the client-side of the application, Vaadin uses GWT for rendering the resulting web page. Vaadin’s default component set can be extended with custom GWT widgets and themed with CSS.

2.3.9. Analysis and discussion

Table 2 summarizes the relevant characteristics, for the task at hand, of the researched frameworks for an easier comparison. Its analysis gives closure when deciding which framework(s) is suited for the task at hand. As a start, the frameworks can be divided by those that use JavaScript and those that use Java, with the later gathering a small advantage due to Java’s more common use and ample support. Next, looking at some core requirements for this dissertation project, MooTools and ZK are the first to go down. The inexistence of a XML parser, the ZK’s hard widget extension and the MooTools small widget collection are major drawbacks. And although ZK makes it back with easier coding and a softer learning curve, its low server-centric overall performance takes it out of the race when compared to its client-centric counterparts.

Also with low performance is the Prototype+ Scripty2 combo, which together with a low documentation and a medium sized widget collection also makes it an unsuitable option. A similar read happens with Vaadin, although with more widgets (due to its GWT component) and a slightly better documentation. The jQuery and ExtJs frameworks have better documentation and lower learning curve, but still lack in term of performance and multi-language support which is crucial for an interface to be used by professionals with different languages.

\(^{32}\) http://vaadin.com/home
2. Related Work

<table>
<thead>
<tr>
<th></th>
<th>Prototype+Scripty2</th>
<th>MooTools</th>
<th>jQuery</th>
<th>Ext JS</th>
<th>Dojo</th>
<th>GWT</th>
<th>ZK</th>
<th>Vaadin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cross Browser Support33</td>
<td>High</td>
<td>High</td>
<td>High</td>
<td>High</td>
<td>High</td>
<td>High</td>
<td>Medium</td>
<td>High</td>
</tr>
<tr>
<td>Multi-Language Support34</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Widget Extension35</td>
<td>Easy</td>
<td>Easy</td>
<td>Easy</td>
<td>Easy</td>
<td>Easy</td>
<td>Easy</td>
<td>Hard</td>
<td>Easy</td>
</tr>
<tr>
<td>Widget Collection33</td>
<td>Medium</td>
<td>Small</td>
<td>Medium with plugin</td>
<td>High</td>
<td>High</td>
<td>High with plugin</td>
<td>High</td>
<td>High with plugin</td>
</tr>
<tr>
<td>XML Parser36</td>
<td>With Plugin</td>
<td>No</td>
<td>With Plugin</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Ease of Use (API)33</td>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
<td>Low</td>
<td>Medium</td>
<td>High</td>
<td>Medium</td>
</tr>
<tr>
<td>Learning Curve33</td>
<td>Medium</td>
<td>High</td>
<td>Low</td>
<td>Medium</td>
<td>High</td>
<td>Medium</td>
<td>Low</td>
<td>Medium</td>
</tr>
<tr>
<td>Documentation33</td>
<td>Low</td>
<td>Medium</td>
<td>High</td>
<td>Medium</td>
<td>Low</td>
<td>High</td>
<td>High</td>
<td>Medium</td>
</tr>
<tr>
<td>Community Support33</td>
<td>Medium</td>
<td>Large</td>
<td>Large</td>
<td>Large</td>
<td>Small</td>
<td>Large</td>
<td>Medium</td>
<td>Medium</td>
</tr>
<tr>
<td>Source Language36</td>
<td>JavaScript (JS)</td>
<td>JS</td>
<td>JS</td>
<td>JS + HTML</td>
<td>Java</td>
<td>XML + Java</td>
<td>Java</td>
<td></td>
</tr>
<tr>
<td>Overall Performance37</td>
<td>Low</td>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
<td>High</td>
<td>High</td>
<td>Low</td>
<td>Low</td>
</tr>
</tbody>
</table>

Table 2. Comparison of characteristics from the different Ajax frameworks.

This takes the analysis to a final round with Dojo versus GWT. Both of them have very good overall performances and pack essential features for the project development, so this will come down to support and accessibility. Dojo takes it bad in these areas, with low documentation available and a harder time using the application programming interface (API). Furthermore, its learning curve is one of the highest of the table and the community support is the lowest. On the other hand, GWT is also not an optimum choice when it comes to API’s ease of access, but it shines with the double plus of high documentation and a thriving community support that

33 http://www.slideshare.net/hsplmkting/top-javascript-frameworks-comparison
enabled the creation of many libraries to provide additional widgets and functionalities. Bottom line, GWT will demand the investment of more time, but allows for a more robust, and easier-to-debug, solution.

2.4. Summary

This chapter started by presenting some of the more prominent visual mapping tools and their interface research. MapForce 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 Mapper – a module included in BizTalk Server bundle – is another visual editor that allows mapping between XML schemas, and has had a lot of recent research and development dedicated to its interface; The XML Mapper – a part of Stylus Studio’s XML IDE – is a many-to-one visual schema mapper dedicated to a XML environment; Clio 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 correction and information viewing; and finally, the COMA++, 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 these 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; and the function Toolbox, where all the available mappings functions are stored.

Next, the concept of interface paradigms was revisited: representation techniques that are applied to an application’s interface with the goal of displaying information with unusual characteristics, which creates representational problems for the interface. For this dissertation’s scope, the interface paradigms will focus in the representation of large mappings. Based on the previously identified parts of a visual mapping tool’s interface, the various interface paradigms regarding this problem were separated into four different areas: Mapping Board Improvement, Schema View Improvement, Navigation and General Improvement.

Proceeding, there was a study to show the interface paradigms present in the previously introduced visual mapping tools, listing many unique ones and other that were present in more than one application. From this analysis some thoughts and ideas were drawn, like the dominance of paradigms dedicated to Mapping Board improvement and the value of some navigation techniques present in many of the applications.
2. Related Work

Finally, to create the new interface, where any new paradigms developed in this dissertation 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, ZK 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 generally allow for a more robust, versatile and easier-to-debug solution.

The next chapter will be making a more in-depth reintroduction of this dissertation's problem, preceded by a more detailed contextualization and ending with the enumeration of the problem's requirements.
3

The Problem
3. The Problem

This chapter starts by making some contextualization of the problem followed by its more in-depth reintroduction.

3.1. Context of the Problem

To better formulate this dissertation’s target problem some contextualization is required.

3.1.1. About Models

When the word ‘model’ is used, most likely, it will bring to mind physical models such as those that are constructed to depict new buildings, cars or other artifacts. Such models are a precursor to actually building the artifact “for real”. However, the use of the word goes beyond physical models. Using conceptual models is a common and important practice in many fields of science. Geology has realistic geologic models, important for the oil and gas industry and seismic prevention; Biology has ecosystem models important for simulations and environment protection; Economy has models for economic activity forecasting and action planning; Businesses have revenue models for sharing and learning of market strategies; and so on. Models are mechanisms of communication, but each model shows only a part of the totality of information about the subject, as too much information in a model makes it difficult to comprehend. In a general sense definition, a model is anything used in any way to represent anything else – it uses ideas to represent something concrete – and are employed to help us know and understand the subject matter they represent. Models are made of various elements that depend on the type of the model, so model entities is the general term that captures all the basic primitives of a specific model.

As described before, models can be used in several areas for different purposes. On the scope of this dissertation, the focus will be on models that represent information managed by information systems. These models are commonly used in describing activities such as business processes, manufacturing processes and software development and are referred to as information models. The information model is a representation of concepts, relationships, constraints, rules and operations to specify data semantics for a chosen domain of discourse [23]. In the current technological stage, the need for information models is growing and their main advantage is that they can provide sharable, stable and organized structure of information requirements for a domain context [23].

With the advent of information modeling, came the need to represent information in a way that it could be stored and exchanged between information systems, resulting in the definition of data models. Used to organize data, the data model captures the cardinality and referential integrity rules needed to ensure that the data is of good quality for the users. More specifically,
a data model can be seen within an information system as having three main roles: getting data in, integrating data and getting data out.

3.1.2. About Visual Modeling

A model can have different kinds of representation; it might be imagined in the mind or described in words, but no better way exists to achieve its sharing and communication-enabling purposes, than to present a model using a visual representation. The activity of conceptualizing objects and systems of interest is commonly known as modeling, while the same activity using a visual conceptual representation is visual modeling.

In today’s businesses, information technology (IT) plays a central role. Not only is it integrated into most enterprises’ core business processes, but can also be a major source of innovation. With the evolution of the IT arose the need to model and visually represent information systems, i.e. to find visual representations of information models. This originated the recent boom of visual modeling, leading to its widespread use not only in software design, but also in non-software disciplines such as systems engineering and in the business domain [24].

3.1.3. About the Mapping of Models

More recently, the data integration and interoperability areas have shown a rising interest in model representation, looking at modeling as a way to represent how different models may relate to each other, so that their data, and thus information, can be combined and reused in different contexts. This process is commonly known as mapping between models.

Given the proliferation of data models (like XML Schemas38), often containing very similar information, it is sometimes necessary to transform document instances from one schema to another. This involves mapping entity names, changing structure, and transforming content values (e.g. format of dates or units of measure). The mapping between models, or data mapping, is the process of creating entity mappings between two distinct data models, and the product of this process, the mapping model, is a set of rules and techniques used to modify one model in order to get another model [25]. Data mapping is used as a first step for a wide variety of integration and interoperability tasks, as it helps to determine how data can be combined and reused in different contexts.

38 http://www.w3.org/XML/Schema
3. The Problem

3.1.4. About Visual Mapping Tools

Mappings between models can be defined visually by using specialized tools called visual mapping tools. Designing mappings can be a very time-consuming process due to the high heterogeneity of the typically independently developed schemas and many visual mapping tools have been developed to make this task easier [2]. These tools provide a way for users to visually specify the mappings by connecting entities defined in the source model with entities from the target model. This associations might not be direct and require the use of specific operations (like mathematical, logical, type conversion, string operations and so forth) to convert entities from source to target schema format. A well accepted way of doing this consists of using little operations boxes, or functoids, which are placed between connections in the Mapping Board and encapsulate information about a determined mapping operation. After the mapping is done, some tools are able to compile these user-specified mappings into an executable component (e.g. a XSLT template, a SQL query, a Java or C++ program) which can afterwards run over the instances of source models to transform them into instances conformant with the target models.

![Figure 8. Overview of the use cases commonly found on Visual Mapping Tools](image)

The basic functions a visual mapping tool can perform are (Figure 8): a mapping tool can be able to load the data models into the view, two at a time. Also, when two models are loaded the tool can allow editing the mappings between them, using adequate mapping operators, as
well as saving the editions. A **visual mapping tool** can allow the visualization of mappings, through the adequate representation, as well as allowing the navigation of those mappings. Finally, it can allow loading of existent **mapping models**, being that loading a **mapping model** should automatically load the two affected models into view.

### 3.1.5. The MDR and its Mapper

As mentioned before, the work developed in the context of this dissertation will be integrated into the Metadata Registry’s mapper. The conceptual map showed in Figure 9, helps locating and comprehending the integration of this dissertation’s problem within the MDR problem’s context.

![Concept map depicting this work on the MDR's context](image)

Figure 9. Concept map depicting this work on the MDR’s context

The MDR is a system that stores and maintains metadata (which defines data models) and the mapping rules (which define **mapping models**), which make the transformation between data models possible. For this purpose, the MDR will need a GUI comprised of two modules: the MDR Modeling Tool for data model management, and the MDR’s mapper for **mapping models** management. These interface modules implement specific **interface paradigms** according to their tasks, which enable a better visualization of the data and **mapping models** they need to represent.
3. The Problem

Note: The MDR’s mapper is designed to also work as an independent module and was given the working title of ‘XMApper’.

3.2. Reformulate the Problem: Large Mapping Visualization

How will a visual mapping tool deal with such extent models while, at the same time, represent the specificity of the relationships that can be established between them? – from Section 1.1.

As previously discussed, the goal of this paper will be the research, idealization and testing of one or more interface paradigms to address the problem for visual mapping tools: Large Mapping Visualization.

![Diagram of complex mapping between two models](http://digmap2.ist.utl.pt:8080/xmapper-0.2/)

Figure 10. Example of a complex mapping between two models

Visual modeling tools are in constant evolution to keep up with the pace of the new developments in information representation and information modeling, as information domains’

---

39 Prototype available at: [http://digmap2.ist.utl.pt:8080/xmapper-0.2/](http://digmap2.ist.utl.pt:8080/xmapper-0.2/)

40 From Altova’s blog at [http://blog.altova.com](http://blog.altova.com)
detail is constantly expanding, and so is the model representation of those domains. This causes new problems for creators of **visual modeling tools**, one of them being finding a way or ways, i.e. **interface paradigms**, to scale the representation of the new models and their features (content, size, complexity…) with a specific modeling purpose in mind (model editing, model mapping, model navigation…). Although this extends to just every type of information modeling, here the focus will be in the problems it poses to visual mapping between models and, of course, **visual mapping tools**. The problem with the current **visual mapping tools** solutions is that they do not scale well to large schemas or large maps, yet that’s exactly what business are demanding. Figure 10 (an example of a mapping between two different **XML schemas**, using Altova’s MapForce) is an example of such failure, where the details of interest, like what connects to what or what operates on what, are lost in a maze of complexity. And that is not an uncommon case (Figure 11, an example using the BizTalk Mapper). When the schemas become larger or the mappings increase in complexity or number, it’s a real challenge to give a satisfying representation of the whole ensemble. New **interface paradigms** are needed to answer that challenge.

![Figure 10](image)

**Figure 10.** An example of a mapping between two different XML schemas, using Altova’s MapForce.

![Figure 11](image)

**Figure 11.** Example of a confuse mapping in Biztalk’s Mapper

To more easily address the issue, it will be split into two smaller parts: representing (through trees) Schema Views with a large number of elements; and representing Mapping Boards with a

---

3. The Problem

A large number of functoids and connections. Reformulating the problem, how can visual mapping tools represent mappings (1) between schemas with a high number of elements or (2) with a high complexity?

So, there are schemas with a high number of elements and overloaded Mapping Boards. From here it's possible to pinpoint a set of specific problem points, and to each project the new interface paradigms. Schemas with high number of elements, represented by trees:

- It's hard to visualize just specific elements (or sets of elements) of interest (which can even be spread throughout the tree).
- It's hard to know what's already mapped and what's not.

Overloaded Mapping Boards:

- Lack of space for new mappings.
- It's hard to select, visualize and understand already made mappings.

The Mapping Board's overload will be no doubt directly connected to the schemas with high number of elements, but it also depends on the complexity of the needed mappings (the necessity of a high number of functoids and connections for each element's mapping) and the dimensions of the Mapping Board itself, i.e. the number of elements it can hold without compromising their visibility.

3.3. Problem’s Requirements

This section gathers the main business requirement for de XMApper as a schema mapping application and its interface requirements.

3.3.1. Business Requirements

The XMapper business requirements related to this dissertation’s problem are derived from the use cases of visual mapping tools identified in Section 3.1.4.

[Requirement1]. View Mappings: It must be possible to view the mappings, including identifiable presentations for the models and the mapping operations.

[R2]. Edit Mappings: It must be possible to edit the mappings. Operations for map editing must be explicit in identification and functioning.

[R3]. Navigate Mappings: It must be possible no navigate all of the mapping, including the models and mapping board.
3.3.2. Graphic Interface Requirements

The designed interface paradigms must take into consideration Nielsen’s 10 usability heuristics [26], thus it’s possible to define some requirements that the designed interface paradigms must take into consideration:

[R4]. Visibility: The system status must be visible. Instructions for use of the system must be visible or easily retrievable. Irrelevant information must take the background and not take visibility from the relevant.

[R5]. User Friendly: The system must allow to user to always be in control, be consistent, follow real-world conventions, adapt to the advanced user and provide help and documentation.

[R6]. System Stability: The system must minimize error-prone conditions, helping users to avoid, recognize, diagnose and recover from them.

3.4. Summary

This chapter started by talking about the importance of models, information modeling and the natural appearance of visual modeling, to give a more in-depth context to this dissertation’s problem. It also talked about the more recent weight given to mapping between models for data integration and interoperability areas, how it can be a very time consuming activity and, thus, the origin of visual mapping tools, which are specialized tools for users to visually define these mappings. It was then shown how this project fits in the Metadata Repository’s context and how it will interact with its mapper.

Next, the dissertation problem was reformulated, being divided into two parts for a better approach: how can visual mapping tools represent mappings (1) between schemas with a high number of elements or (2) with a high complexity? This led to the conclusion that there are two main areas of the visual modeling tools with problems, which the new interface paradigms should address: Schema Views of trees with high number of elements and overloaded Mapping Boards, each with a subset of specific problems.

Finally, based on the analysis made (on visual modeling tools) for the context of the problem and in Nielsen’s 10 usability heuristics, some business and graphic interface requirements were listed to serve as guidelines for the solution design and prototype development.

The next chapter will describe the idealized solution, the development setup for the prototype, details about its implementation and decisions made.
Proposed Solution
This chapter starts with the idealized solution. Next the development setup is described, followed by the last section with implementation details.

4.1. Proposed Solution

This section presents the designed interface paradigms to tackle this dissertation’s set of target problems, identified at the end of Section 3.2, which were divided into two areas: Mapping Board Improvement and Schema View Improvement. Designed paradigms shall also conform to the problem’s requirements identified in Section 3.3. The objective will be to combine a set of new interface paradigms and hopefully come up with a solid set that represents a useful solution for the identified challenge. Paradigms referenced as [SP1.1] and [SP2.2] are improvements on already existing ones, while the other four are new proposals.

Note: Paradigms [P1.6], [P2.2] and [P4.3] as well as variations of [P4.1] (for filters), [P4.4] (for connection status) and [P4.6] (for cardinality show/hide configuration in the Schema View) were also implemented in the XMApper prototype.

4.1.1. Mapping Board Improvement

Recap of overloaded Mapping Boards problem points (from Section 3.2):

- Lack of space for new mappings.
- It’s hard to select, visualize and understand already made mappings.

[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 (see [P1.10] in Section 2.2.1) 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).
4. Proposed Solution

[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 functoids (does it come from that functoid? Or does it come from another source and it’s just crossing the functoid’s space?). Techniques like bendable links (see [P1.2] in Section 2.2.1) 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 connection targets.

[SP1.3] Connection Argument Visibility – Minimizing the space occupied by the functoid’s visual representation is an effective way of simplifying the view of a Mapping Board (as shown by BizTalk Mapper and Stylus Studio XML Mapper in [P1.6]), but doing so often makes it impossible or impractical to have the functoid inputs and outputs directly shown on the functoid representation. A possible way to address this dilemma, and thinking of what MapForce does with its connection’s annotations (see [P1.9] in Section 2.2.1), is to show the inputs and output of the functoids near the docking point where the connections meet the functoids 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 advanced stages when many connections are already made. To mitigate this problem the mapping tool user should be given the option to show or hide all arguments, and by default the mapper should only show arguments on the currently selected connection(s).

4.1.2. Schema View Improvement

Recap of schemas with high number of elements, represented as trees, problem points (from Section 3.2):

- It’s hard to visualize just specific elements (or sets of elements) of interest (which can even be spread throughout the tree).
- It’s hard to know what’s already mapped and what’s not.
Quick Filters – Inspired by the thought of joining the paradigms of information toggling and information suppression like sibling coalescence (see [P4.5] and [P2.3] in Section 2.2), quick filtering is a simple way to address both problem points related to the Schema Views. 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 the other 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.

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 inform if that element is connected or not, but this technique can be expanded as well as its purposes. The connection line in the element can be further customized in terms of color (like in [P4.4] from Section 2.2.4 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.

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, separating (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.

Connection Statistics – Statistics present different views of what is first perceived in a situation. Showing trees 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:
4. Proposed Solution

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.

4.1.3. The sum of all parts

Following the problem points identified at the end of Section 3.2 (Reformulate the Problem) it is possible to compose the following table to summarize this dissertation’s proposed solution.

<table>
<thead>
<tr>
<th>Problem's Points</th>
<th>Solution's Interface Paradigms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schema View</td>
<td>• Quick Filters</td>
</tr>
<tr>
<td>Visualize specific elements of interest.</td>
<td>• Sorting</td>
</tr>
<tr>
<td>Identify what's mapped and not.</td>
<td>• Quick Filters</td>
</tr>
<tr>
<td></td>
<td>• Sorting</td>
</tr>
<tr>
<td></td>
<td>• Tree Element Connection Status</td>
</tr>
<tr>
<td>Mapping Board</td>
<td>• Connection Visibility</td>
</tr>
<tr>
<td>Lack of space.</td>
<td>• Connection Argument Visibility</td>
</tr>
<tr>
<td>Select, visualize and understand</td>
<td>• Connection Render</td>
</tr>
<tr>
<td>existing mappings.</td>
<td>• Connection Visibility</td>
</tr>
<tr>
<td></td>
<td>• Connection Argument Visibility</td>
</tr>
</tbody>
</table>

Table 3. Summary of the proposed solution.

Starting with the issues that originate from schemas with high number of elements, quick filters and sorting were designed to address the problem of visualizing specific elements of interest in such crowded trees. Quick filters will shine in this area, because it effectively clears the Schema View of unwanted clutter, but sorting can also be very valuable by re-organizing the tree giving priority to what is needed. Thus, both of these paradigms will also inherently help with the main job of identifying what's mapped and what's not (with the right filters and sorters) and joining them will be the ability to see the connection status of each of the tree elements, which is specifically designed with this last problem point in mind. Finally, one might notice, that connection statistics are not represented in this table as it was opted so, but it could in fact be
listed to address all of the Schema View’s issues, because it is intended to be a gateway joining mapping statistics with any other interface paradigm.

As for the Mapping Board’s issues, connection visibility and connection argument visibility are the designed choices for solving the problem of lack of space in Mapping Boards overloaded with mapping elements, because they bring the ability to simply make unwanted/unneeded elements disappear. Because of this mapping simplification, they are also important in visualizing and understanding existing mappings, an area where there is also the connection render technique, which brings advantages in selecting existing mappings.

As last note, a mention that there were more options introduced, designed or only thought of, which were cut for simplicity and for lack of time to implement and test. The initial specification of one of them, ‘Tags’, can be found in Annex E.

4.2. Development Scenario

This section will quickly cover the development scenario of this work: the tools that were used and the base XMApper’s interface, where the new interface paradigms were implemented.

4.2.1. The GWT as a Web Development Framework

After the analysis made to web development frameworks in Section 2.3, GWT was selected as being the one best suited for creating a new interface for the XMApper, and testing new interface paradigms upon it. To present a more attractive look-and-feel to the interface, the open-source GWT widget extension Ext-GWT\(^\text{42}\) (or GXT) was also used, which enabled the faster creation of the web interface, using GXT’s widgets that can be easily changed according to the user’s needs.

The modularity of GWT’s component architecture allowed not only the creation of personalized components, but also to easily test them whether individually or together.

4.2.2. Platform – The XMApper

As mentioned before, XMApper\(^\text{43}\) is the name of the visual mapping tool being developed as a standalone web application and a module for Metadata Registry (MDR). The standalone

\(^{42}\) http://www.sencha.com/products/extgwt

\(^{43}\) Prototype available at: http://digmap2.ist.utl.pt:8080/xmapper-0.2/
4. Proposed Solution

version is capable of opening locally stored mappings, allowing the user to select schemas and mapping models directly from its file system. The MDR module provides MDR users with visual means to map between model schemas currently stored (or to be stored) in the registry. The solution of this dissertation will be implemented in the XMApper and integrated in its development cycle. Evaluation efforts will be conducted in the XMApper standalone version.

Figure 12 shows the base state of the current XMApper interface. From left to right, there is the source Schema View, the Mapping Board, the target Schema View and the properties/toolbox column. Following the most popular visual mapping tools, the user creates and edits mappings by means of drag-and-dropping components' connectors (little connection-starting arrows, which appear near a selected component), and places functoids by selecting them first in the toolbox and then insert them in the desired spot on the drawing board (with double-click).

![Figure 12. Screenshot of the XMApper base interface](image)

One very interesting feature of the prototype has to do with the Mapping Board itself, as it was constructed using a Scalable Vector Graphics (SVG) library for GWT, developed in the scope of the XMApper's project with the help of another library, the lib-gwt-svg. This library provides the necessary integration between SVG objects and GWT, allowing the creation of a SVG drawing board within a GWT panel. By making the Mapping Board one of such SVG

---

44 http://www.w3.org/Graphics/SVG/
45 http://code.google.com/p/lib-gwt-svg/
boards, was possible to create and shape the Mapping Board components through SVG, while assessing and controlling them just like any other GWT component.

More details about the main functions of the XMApper can be found in Annex C and the latest version of its User Manual can be found online\(^4^6\).

### 4.3. Implementation

This section describes how each interface paradigm was implemented, the relevant details, the decisions made and why.

#### 4.3.1. [SP1.1] Connection Visibility

For the implementation of different visibilities for the connections, it was first determined their representation and triggering conditions (Table 4).

<table>
<thead>
<tr>
<th>Condition</th>
<th>Connection Visibility</th>
<th>Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 connection ends visible</td>
<td>Black solid line</td>
<td>![Solid Line]</td>
</tr>
<tr>
<td>1 connection ends visible</td>
<td>Gray dashed line</td>
<td>![Dashed Line]</td>
</tr>
<tr>
<td>0 connection ends visible</td>
<td>The connection is totally hidden</td>
<td>N/A</td>
</tr>
</tbody>
</table>

*Table 4. The connections possible visibilities and their trigger conditions.*

The connection’s visibility has 3 possible states: both ends of the connection are visible, then the connection is drawn in a solid black line; only one end is visible, then the connection is drawn in a dashed gray line; and no ending of the connection is visible, in which case the connection is completely hidden from view.

A connection ending can be of two types: functoid or schema element. This determines the ways the ending can be visible. To determine if a schema element is visible, its model tree is asked if either the element’s parent is collapsed or if the element is not visible according to the current scroll. In the functoid’s case the Mapping Board is asked, which can determine if the functoid isn’t visible according to de current scroll.

4. Proposed Solution

Implementation-wise, the connections themselves are separated SVG objects and, as such, it is possible to control their style by configuring their SVG properties.

4.3.2. [SP1.2] Connection Render

Two types of rendering for connections were implemented, one that took an indirect path from the starting to the ending point using horizontal and vertical rectilinear lines and a simple path building algorithm, and another that simply build a direct line from start to finish point.

As said before, the connections are SVG objects, and the renders are basically functions that given the connection’s data (like starting and ending point) use it to draw an SVG path according to its specification. This simple architecture allows for the render to be changed on-the-fly during runtime via the preferences interface, and refresh all the connections representations for and immediate effect on the drawing board. In Figure 13 and Figure 14 it’s possible to see the two renders giving different perspectives to the same situation.

![Figure 13. Example of connection’s rectilinear render](image)

![Figure 14. Example of connection’s direct render](image)

4.3.3. [SP1.3] Connection Argument Visibility

The connection’s arguments visibility was implemented as an integrated part to the connection’s representation. As such and when needed, a SVG text object will be added to the appropriate position either at starting or finishing end of the connection, with information about the needed connection’s argument.

Through the preferences interface the user can control how frequently the connection arguments are displayed, between three possible choices: always show, never show and show only on selected connections (the default option, seen in Figure 15). Also worth noting, the connection’s arguments are derived from the functoid input or output to which the connection is linked, thus only the connection’s ends that link to a functoid will show arguments.
4.3.4. [SP2.1] Quick Filters

As said before, the main idea of was to provide a fast and simple way for the user to drastically change the view of the Schema View, with a pre-defined intent in mind. To achieve this, toggle buttons groups were chosen for their simple method of operation and the ability to embed them directly into the header of the Schema View panel. As of now the prototype implements two of these toggles: the first is the “main view”, i.e. show all model elements with no filter applied, and the second is the “show connected”, which configures the tree view to show only the connected elements of the model (and their respective parents for path context). Unfortunately, at the time of this writing, the filter “show unconnected” wasn’t successfully implemented. Figure 16 show an example mapping without any filter applied. Notice that, although the source tree elements are expanded to see if it was possible to have all the mapping displayed on-screen, one of the connections still leads to an off-screen element.

![Connection Arguments Diagram](image)

**Figure 15.** Example of connection arguments on selected connection

![Screenshot](image)

**Figure 16.** Screenshot depicting the interface with no filters applied

On the other hand, in Figure 17 the “Show Connected” filter is toggle on (notice the toggle change in the header) in the source tree and now, with all the elements expanded (some
4. Proposed Solution

Elements don’t have connections, but they’re the parents of elements that do, it is possible to see all connected elements on-screen in a very space-efficient view.

In this last picture it’s also possible to see another characteristic of the quick filters. As a way of telling the user that the view is filtered, besides the simple change in the toggle buttons, the background color of the tree is changed to a light green, the color representing the “filtered” status for this prototype’s Schema Views.

![Figure 17. Screenshot depicting the interface with filters applied (on the source schema)](image)

4.3.5. [SP2.2] Tree Element Connection Status

In the developed prototype, two features were added to the normal representation of the tree elements to present the different connection status of the elements. To the line representation of the connection, already used in mappers like the BizTalk Mapper and Stylus Studio XML Mapper, was added an underscore to the text of the element. The idea is that even if the connection line disappears (due to changes in the connection’s visibility or maybe filtering options, for example) the underscore will remain to show the user that element is already connected. Table 5 lists all the combinations currently implemented on the prototype.

Worth noting is that all the examples in Table 5 are taken from a source tree. The target tree representation is identical, but with the addition of an arrow at the end of the connection line, representing the end of the connection, in the cases of a direct connection to the element (has can be seen for instance in Figure 13 and Figure 14).
<table>
<thead>
<tr>
<th>Situation</th>
<th>Connection Line</th>
<th>Underscore</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unconnected</td>
<td>None</td>
<td>None</td>
<td>[1..1] category</td>
</tr>
<tr>
<td>Connected</td>
<td>Solid and black</td>
<td>Solid and black</td>
<td>[1..1] category</td>
</tr>
<tr>
<td>Connected and selected</td>
<td>Solid and yellow</td>
<td>Solid and black</td>
<td>[1..1] category</td>
</tr>
<tr>
<td>Children connected</td>
<td>Dashed and black</td>
<td>Dashed and black</td>
<td>[1..1] category</td>
</tr>
<tr>
<td>Children connected and selected</td>
<td>Dashed and yellow</td>
<td>Dashed and black</td>
<td>[1..1] category</td>
</tr>
<tr>
<td>Children connected but</td>
<td>None</td>
<td>Dashed and black</td>
<td>[1..1] category</td>
</tr>
<tr>
<td>connection line not visible</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5. Tree element representation depending on connection and selection situation.

Implementation-wise both the underscore and the connection line are made by configuring the borders of an HTML element via the use of CSS styles. The underscore’s element is the text element itself, while the connection line required the introduction of its own HTML element, which would vary in size according to the needed connection line size, in the HTML template for the tree’s elements.

### 4.3.6. [SP2.3] Sorting

Implementing the Alphabetical sorting of the trees on the prototype wasn’t too complicated as GXT already supports it for their tree panels, which are the base for the prototype’s Schema Views. It was only a matter of assuring the sorting HTML element was the schema element’s text field. Now for the Connection Status sorting, the same sorting system already present in GXT was used, but a variation was made allowing modification of the sorting conditions. The new sorting condition specified the priority of each tree element according to their number of connections, i.e. a first element would have priority over a second if, when subtracted the number of connections of the second element to the first’s, the result would be positive. On the other hand, if the result is negative, the second element would have priority. If the result is zero, priorities would be equal unless one of the elements has nested connections and the other hasn’t, which gives priority to the element with nested connections.

49
4. Proposed Solution

To access the sorting options in the prototype, the user can use the dropdown menu present in each of the mapping tree's header, which contains a sorting sub menu. In Figure 19 there is an example scenario of a tree that is originally alphabetically sorted, then having its sorting changed to “by Connection Status” (Figure 19).

![Figure 18. Example of a tree sorted alphabetically](image18.png)

![Figure 19. Example of a tree sorted by Connection Status](image19.png)

4.3.7. [SP2.4] Connection Statistics

Following the original idea of having statistics as a possible mean of navigating or change the view over the Schema Views, a small sections of statistics was implemented in the prototype. It is accessible in each tree dropdown menus, in the “Statistics” option. Through there, it is possible to consult some mapping statistical data relative the trees and view a simple graph about it.

Figure 20 shows a simple pie chart with the ratio of connected to unconnected elements present in the tree. The pie graph’s slices are interactive and, in this case, clicking the “connected” slice will apply the quick filter “Show Connected” while clicking the “unconnected” slice will apply the sorting “By Connection Status” ascending. These were the “effects” chosen, but this technique could synergize with any other effect (depending on the situation, of course).
Note: In Figure 20 many of the Statistics fields didn’t show the correct data, because it still wasn’t implemented.

![Figure 20. The statistics window with the pie chart with selectable slices](image)

As for the implementation, the graphs are made using a Flash library supported by GXT, and then linked with other existing interface paradigms.

4.4. Summary

This chapter presented the full extension of the proposed architecture, with detailed information about each of its components. It started with the proposed interface paradigms being divided between two areas and then discussed in detail. The first area was the Mapping Board Improvement and the proposed paradigms were: Connection Visibility, Connection Render and Connection Arguments Visibility. The second area was Schema View Improvement and the proposed paradigms: Quick Filters, Tree Element Connection Status, Sorting and Connection Statistics.

Next, the Development Scenario was presented were, after Section 2.3 analysis, the chosen development technology was reaffirmed and new details introduced, and also an

---


48 [http://teethgrinder.co.uk/open-flash-chart/](http://teethgrinder.co.uk/open-flash-chart/)
4. Proposed Solution

An overview of the prototype platform – the XMApper – was given along with a description of its base interface.

Finally, implementation details about all the proposed interface paradigms were presented, dwelling a little more into how each of the new paradigms was implemented using the development technology and how each is accessed through the prototype's GUI.

The next chapter will start by detailing the methodology followed to test the solution. After that, the results are reported and analyzed.
5 Evaluation
5. Evaluation

This chapter starts by explaining the evaluating methodology followed to test the developed work. The chapter ends with the results and their analysis.

5.1. Methodology

5.1.1. Prototype versions

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 task sets. Aside from that, the versions ranged from one with no new interface paradigms (baseline version) to other with all paradigms simultaneously activated (full version). This provided the ability to incrementally investigate the influence of each set of paradigms in comparison to the baseline version (plus \[SP2.2\]) of the XMApper’s interface.

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 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\(^{49}\) design.

\(^{49}\) http://mathworld.wolfram.com/LatinSquare.html
5.1.2. 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 at degree in Multimedia and/or Information Systems. Although most participants didn’t have previous experience with the XMApper due to its recent nature (only 3 had tried it before), 60% had experience using visual mapping tools.

5.1.3. 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 functoids 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 visualizations. Two of the tasks sets used in the main study are shown in Table 6 as an example. The two task sets used for the statistics study are shown in Table 7.

<table>
<thead>
<tr>
<th>Task Set A</th>
<th>Task Set B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Go to the first “NOT” functoid (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” functoid (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 functoid’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 functoid’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 6. Two of the task sets used in the main study
5. Evaluation

<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 source schema.</td>
</tr>
</tbody>
</table>

Table 7. The two task sets used in the statistics study

5.1.4. Test map

The map that was used in the experiment is shown in Figure 21. The schemas used were MODS 3.0\textsuperscript{50} (source) and ESE 3.3\textsuperscript{51} (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.

![Figure 21. The map used in the user study experiments](image)

The aspect ratio of the window used for the study, as shown in Figure 21 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

\textsuperscript{50} Metadata Object Description Schema - http://www.loc.gov/standards/mods/v3/

\textsuperscript{51} Europeana Semantic Elements - http://www.europeana.eu/schemas/ese/
Mapping Board and limited window size, helped simulating the behavior required for a more traditional aspect ratio on larger schemas and a larger map.

5.1.5. Final survey

After all the prototype tests were done, each participant completed a background and satisfaction survey that can be seen in Annex D.

5.1.6. Main study procedure

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.

The sessions started with a brief presentation of the prototype, the different versions and features. The users were then directed to their starting version and given a first task. After completing a task set in that version they would move to the next one (except in Version C, when they did the statistics study) until they had tested all three versions. Comments and task time where noted during each test and clarifications about the method used to complete the task were asked. After the prototype testing the users completed the final survey to record background and satisfaction data.

5.1.7. Statistics study procedure

Whenever a participant concluded the main task set for the prototype Version C, the statistics study would follow. This consisted of first realizing two tasks on Version C and then two similar tasks on Version D. The task set order alternated from participant to participant. Times and comments were noted for later analysis.

5.2. Results and Discussion

5.2.1. Task times

The post-hoc analysis of all obtained task times allowed the plotting of the chart shown in Figure 22, representing the average task times by version and their standard deviation(σ). The prototype Version A (base version plus [SP2.2]) was significantly higher than that of Version C (all paradigms except [SP2.4] Connection Statistics), 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 22.
5. Evaluation

Interaction between prototype version and tasks reveals that task 1 was harder than the others, and task 3 and 4 were especially difficult when using prototype Version A. The average task time by task and prototype version is shown in Figure 23.

Figure 22. Average task times for each of the three prototype versions.

Figure 23. Average times by prototype version and task.
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 participants didn’t even use them. Some of them noted that this was due to the lack of signalization of the applied effects and/or that they were already used to apply the effects via their normal interface. Average task times for the statistics study are shown in Figure 24 (Standard deviations – C1: \(\sigma=7.6\); C2: \(\sigma=5.2\); D1: \(\sigma=4.4\); D2: \(\sigma=5.1\)).

![Average Task Times with and without Statistics](image)

**Figure 24.** Average task times for prototype version without Statistics (Version C) and with Statistics (Version D) for the two studied types of task.

5.2.2. 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 [27]. 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 25, with the averages of the user satisfaction ratings and their standard deviation (\(\sigma\)). 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
5. Evaluation

Standard deviations were low for all versions (A: $\sigma=0.8$; B: $\sigma=0.4$; C: $\sigma=0.5$), representing a good accuracy for the results. The overall average ratings for each prototype version can be seen in Figure 25, while all of the satisfaction data is included in Table 8 (Note: the questions are the original from the survey, but, for this table, higher ratings always indicate higher satisfaction).

![Average User Satisfaction Ratings](image)

**Figure 25.** Average user satisfaction ratings for the three prototype versions studied.

<table>
<thead>
<tr>
<th>Survey Question</th>
<th>Version A</th>
<th>Version B</th>
<th>Version C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Difficulty of finding related elements?</td>
<td>3,3</td>
<td>4,9</td>
<td>6,2</td>
</tr>
<tr>
<td>2. Overall ease of use?</td>
<td>4,4</td>
<td>5</td>
<td>5,9</td>
</tr>
<tr>
<td>3. Discoverability of features?</td>
<td>4,6</td>
<td>4,2</td>
<td>5,1</td>
</tr>
<tr>
<td>4. Mental load?</td>
<td>3,8</td>
<td>4,7</td>
<td>5,6</td>
</tr>
<tr>
<td>5. Temporal demand?</td>
<td>3,1</td>
<td>4,4</td>
<td>6,3</td>
</tr>
<tr>
<td>6. User performance?</td>
<td>5,2</td>
<td>5,3</td>
<td>6,3</td>
</tr>
<tr>
<td>7. Overall satisfaction?</td>
<td>3,4</td>
<td>4,7</td>
<td>6,5</td>
</tr>
<tr>
<td>8. Frustration level?</td>
<td>3</td>
<td>4,3</td>
<td>6,2</td>
</tr>
</tbody>
</table>

**Table 8.** Average user satisfaction ratings for the 3 versions of the mapper. Questions are the original from the survey, but higher ratings indicate higher satisfaction for all questions.
Looking at Table 8, 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.

5.2.3. 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 change render 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.

5.3. Summary

This chapter presented the user study conducted to test the proposed solution of this dissertation. It started by explaining the methodology followed in various steps. First the four tested versions of the XMApper prototype were presented:
5. Evaluation

- **Version A** is the baseline version plus the [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].

Then the test population was described in detail, followed by the presentation of the study's tasks, composed of 4 task sets with 4 tasks for the main study and 2 task sets with 2 tasks for the statistics study. Next, the test map and final survey were presented. After this, two sections detailing the procedure followed in the main study and the statistics study, as it was settled that Version D of the prototype would be tested in a different fashion of the other three, due to its need of having the other interface paradigms already available.

The Results and Discussion section was divided in three parts. The first part presented the task times of the study, concluding that the average task time for the prototype Version C was considerably lower than that of the other versions, and some tasks of the study required a longer time to finish than others. The second part showed the results of the user satisfaction survey, concluding that the XMApper's Version C was preferred in every question, although the Discoverability of Features question arose some concerns. Finally, in the third part, some usability issues observed during the tests were listed. These issues are good indications of how to improve the prototype in the future.

The next chapter will finalize this dissertation by wrapping up what has been said until here, clarify the contributions and limitations of this work, and state what can be done in the future.
Conclusions
6. Conclusions

A good **visual mapping tool** will show all the information available in its models and mappings. Due to the constant expansion of the information domains, their model representation is constantly expanding, and so are the mappings created between these new models. An excellent **visual mapping tool** will show all the information available in its models and mappings in a simple fashion, even if dealing with complex models and mappings.

In the Related Work chapter of this work, the more prominent **visual mapping tools** and their interface research was introduced. MapForce 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 Mapper – a module included in BizTalk Server bundle – is another visual editor that allows mapping between XML schemas, and has had a lot of recent research and development dedicated to its interface; The XML Mapper – a part of Stylus Studio's XML IDE – is a many-to-one visual schema mapper dedicated to a XML environment; Clio 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 correction and information viewing; and the COMA++, 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; and the function Toolbox, where all the available mappings functions are stored.

Addressing this dissertation’s problem requires a special type of visualization techniques that were named **interface paradigms**: representation techniques that are applied to an application’s interface with the goal of displaying information with unusual characteristics, which creates representational problems for the interface, thus viewing problems for the user. In the scope of this dissertation, such problems come from the representation of large mappings. Based on the previously identified parts of a **visual mapping tool's** interface, the various **interface paradigms** regarding this problem were separated into four different areas: Mapping Board Improvement, Schema View Improvement, Navigation and General Improvement.

Studying the **interface paradigms** present in the previously mentioned **visual mapping tools** (MapForce, BizTalk Mapper, Stylus Studio XML Mapper, Clio and COMA++), led to the listing of many unique ones and other that were present in more than one application. From this analysis some thoughts and ideas were drawn, like the dominance of paradigms dedicated to
Mapping Board improvement and the value of some navigation techniques present in many of the applications.

Any new paradigms developed in this dissertation would be tested in a prototype called the XMApper, so a search was conducted to find the right JavaScript framework to develop a new interface for it. Many solutions were analyzed and compared: Prototype + Scripty2, MooTools, jQuery, Ext JS, Dojo, GWT, ZK and Vaadin. GWT (or Google Web Toolkit) was the elected choice, because, although it would demand the investment of more time, it would also allow for a more robust, versatile and easier-to-debug solution.

Next, this dissertation’s problem was reanalyzed, starting by a much more in-depth view of its context, which told about the importance of models, information modeling and the natural appearance of visual modeling. It also talked about the more recent weight given to mapping between models for data integration and interoperability areas, and how it can be a very time consuming activity. Thus, **visual mapping tools** were born, which are specialized tools for users to visually define these mappings. The Metadata Registry (MDR) will implement one of such tools, which has been given the working title of ‘XMApper’, and will be the testbed for this work’s developed solution.

The dissertation problem was then reformulated, being divided into two parts for a better approach: **how can visual mapping tools represent mappings (1) between schemas with a high number of elements or (2) with a high complexity?** This led to the conclusion that there are two main areas of the **visual modeling tools** with problems, which the new interface paradigms should address: Schema Views of schemas with high number of elements and overloaded Mapping Boards, each with a subset of specific problems.

The Proposed Solution chapter presented the full extension of the solution’s architecture, with detailed information about each of its components. The proposed **interface paradigms** were divided between two areas and then discussed in detail. The first area was the Mapping Board Improvement and the proposed paradigms were: Connection Visibility, Connection Render and Connection Arguments Visibility. The second area was Schema View Improvement and the proposed paradigms: Quick Filters, Tree Element Connection Status, Sorting and Connection Statistics. Next, the Development Scenario was presented were GWT was reaffirmed the chosen development technology, and an overview of the prototype platform – the XMApper – was given along with a description of its base interface.

Implementation details for all the proposed **interface paradigms** were then presented, dwelling a little more into how each of the new paradigms was implemented in the prototype and how each is accessed through the GUI.
6. Conclusions

A usability study, run with potential XMApper users, systematically investigated the various interface paradigms additions to the prototype’s interface. The study participants were monitored while performing pre-defined task sets in four different versions of the interface, and afterwards completed a satisfaction survey.

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 paradigm 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.

Table 9 overviews the interface paradigms implemented in the XMApper, in comparison to the other studied visual mapping tools.
<table>
<thead>
<tr>
<th>Interface Paradigms</th>
<th>Map Force</th>
<th>BizTalk Mapper</th>
<th>SS XML Mapper</th>
<th>Clio</th>
<th>COMA ++</th>
<th>X MAPper</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auto scroll functoids</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bendable links</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fit in canvas</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Functoid suppression</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Functoid 'invalid' state</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Functoid as small single-cell boxes</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Grid pages</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Links visibility by tree node</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Positioning connection annotations</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Show relevant links</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Visual alignment guides</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Visual function builder</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Connection Arguments Visibility</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Connection Render</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Auto scroll trees</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Expand and collapse trees</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Sibling coalescence</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Quick Filters</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sort</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Connection Statistics</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overview</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Panning mode</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Search function</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Zoom grid</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Background color</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Empathizing selected objects</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Information popups</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Link color</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>No Mapping Board</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Show/hide information toggles</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 9. Results for the evaluation of the mapping tools and their support for the interface paradigms.
6. Conclusions

6.1. Future Work

There is still much to do improving 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.

Due to time issues the scrolling of the Mapping Board wasn’t implemented and thus the hiding of functoids couldn’t be tested for the [SP1.1] Connection Visibility paradigm. Similarly, the “Show unconnected” filter couldn’t be implemented and tested on time for the writing of this document.
References
References


Appendices
A. Glossary of Concepts

**Functoids** – Little operations boxes which are placed between connections in the mapping board and encapsulate information about a determined mapping operation (like mathematical, logical, type conversion, string operations and so forth). Functoids are used to convert entities from source to target schema format.

**Information Models** – Models that represent information managed by information systems. Are a representation of concepts, relationships, constraints, rules and operations to specify data semantics for a chosen domain of discourse.

**Interface Paradigm** – Are specific techniques for representing information, because they help an application’s interface to adapt to unusual situations, created when trying to display information with unusual characteristic (e.g. large quantities, complex organization, special words).

**Mapping Model** – Is a precise specification of how an instance over one schema, called the *source* schema, is to be translated into an instance over a second schema, called the *target* schema. It’s the product of mapping between models.

**Visual mapping tools** – A visual modeling tool with the goal of making easier the process of creating a mapping model between two information models.

**Visual modeling tools** - Software that allows modelers to do their job through a computer, with all the advantages they bring, using visual representations of the information and its modeling operations.
B. Glossary of Acronyms

AJAX – Asynchronous Javascript And XML
BTM – BizTalk Mapper
CSS – Cascading Style Sheets
DOM – Document Object Model
EDI – Electronic Data Interchange
GUI – Graphic User Interface
GWT – Google Web Toolkit
GXT – Ext GWT
HTML – HyperText Markup Language
HTTP – HyperText Transfer Protocol
IT – Information Technology
JS – JavaScript
MDR – MetaData Registry
MF – Altova’s MapForce
SSXM or SS XML Mapper – Stylus Studio XML Mapper
SVG – Scalable Vector Graphics
XHTML – eXtensible HyperText Markup Language
XML Mapping Application
XML – eXtensible Markup Language
XSLT – eXtensible Stylesheet Language
C. The XMApper – User Manual Preview – Getting Started


XMApper prototype available at: http://digmap2.ist.utl.pt:8080/xmapper-0.2/

2. Getting Started

This section will give you a starting overview of the application’s interface and, afterwards, a more detailed examination of each of its components.

2.1. Interface overview

In Figure 26 is main view of the application’s interface, while in Figure 27 that interface is divided by its different components.

![Figure 26. XMApper interface’s overview](image)
Figure 27. The different panels

Legend:
- Source Schema Panel
- Mapping Board Panel
- Target Schema Panel
- Properties Panel
- Toolbox Panel

2.2. Source and Target Schema Panels

The Schema Panels hold the visual representation of the source and target schemas, organized in a hierarchical tree view.

Figure 28. The two schema panels: (a) the source schema panel; (b) the target schema panel.
2.2.1. The header and toolbar

Figure 29. Schema panel header and toolbar

Legend:
1. Panel name
2. Clean schema filters
3. Show connected filter
4. Expand all
5. Collapse all
6. Drop-down menu

2.2.2. The drop-down menu

Figure 30. Schema panel drop-down menu

Allows access to the features:
- Hide/show cardinality
- Sort schema elements
- Filter schema elements
- Auto expand/collapse schema elements by
2.2.3. The schema elements

The XML Element representation components:
1. Expand/Collapse box
2. Icon
3. XML Element text
4. XML Element cardinality

The XML Attribute representation:
1. Icon
2. XML Attribute text
3. XML Attribute cardinality
2.3. Mapping Board

The Mapping Board is the editable area where you can create and view your mappings. This is done by means of mapping links (represented by arrowed lines) and mapping functions (represented by small boxes).
2.3.1. The header and toolbar

Legend:
1. Panel name
2. Unsaved content
3. Preferences menu
4. Save mappings (mapping model)
5. Build XSLT Script

2.3.2. The mapping board

Figure 36. Mapping board header and toolbar

Figure 37. Mapping board
Legend:
1. Mapping links from the source schema
2. Mapping link to the target schema
3. Mapping functions boxes (also known as functoids)
4. Selected mapping function
5. Mapping links creator (also known as connector)
6. Empty cells
7. (All the light blue cells) Border cells, no functoid placement possible

2.4. Properties Panel

The Properties Panel allows you to see further information about any component of the Mapping Board (i.e. functions, constants, conditions and connections). It is divided in two parts, accessible by two different tabs: first, there are the details (which tab's name is the selected component's name), with general information about the component; second, there is the metadata part, which can hold user inputted data about the component.

The Properties Panel becomes active (see Figure 38.a) when an element of the Mapping Board is selected.

![Properties Panel](a)

![Properties Panel](b)

Figure 38. Screenshots of the Properties Panel: (a) Properties Panel when an element is selected; (b) Properties Panel when inactive (no selection).
2.4.1. The header tools

Figure 39. Properties panel header

Legend:
1. Hide/show properties and toolbox panels
2. Hide/show properties panel

2.4.2. Details and Metadata tabs

The properties panel has two different tabs:
- **Details tab** – which changes its name accordingly to the selected board element and shows its properties.
- **Metadata tab** – which shows an editable metadata fields, “Name” and “Description” (Figure 40).

The details tab presents different properties for four different mapping board elements:
- Functions (Figure 38)
- Conditions
- Constants
- Mapping links

Figure 40. The metadata tab
2.5. Toolbox

The Toolbox is where you can access all the available mapping functions. It is divided by various toolsets each representing a different library of functions.

![Figure 41. The toolbox](image)

Legend:
1. Hide/show toolbar panel
2. Close/open toolset
3. Mapping function selection buttons
4. Closed toolsets
5. Open toolset

This is a sample of the XMAPper User Manual. Full version available at:
D. User Study: Final Survey

This survey is also available in Portuguese.

1. BACKGROUND

Age: Click here to enter text.  Sex: Click here to enter text.

Years of computer experience (aprox.): Click here to enter text.

Qualifications: Click here to enter text.

Do you have a degree in Multimedia?  Yes☐ No☐
Do you have a degree in Information Systems?  Yes☐ No☐
Do you have any previous experience with schema mapping tools?  Yes☐ No☐

2. SATISFACTION

Rate from 1 (Very Low) to 7 (Very High):

<table>
<thead>
<tr>
<th>Question</th>
<th>Base Version</th>
<th>+ Mapping Board Features</th>
<th>+ Schema Trees Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Difficulty of finding related elements?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. Overall ease of use?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3. Discoverability of features?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4. Mental load?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5. Temporal demand?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6. User performance?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7. Overall satisfaction?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8. Frustration level?</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Any final comments?  Click here to enter text.
E. Visualizing Schema Mappings: Tags

1. Idea and Advantages

Labeling the visual elements of a mapping model visual representation (i.e. Schema Elements, Functoids and Connections), by applying tags to them. This way multiple elements of different types can be grouped so common visualization operations (like filtering, highlighting or other property changing operations) can be applied to all of them simultaneously. Another important aspect, an element can have multiple assignments (same element can belong to multiple tags), thus introducing flexibility in configuring and switching between views. There are simple user-created tags, which enable user-configured views, and system tags, invisible to the user, which the system automatically creates and uses.

Tags can be placed automatically (ex. “Apply tag to all this element’s context elements” or pre-tag all the children of a schema element with their parent’s tag) or manually (to create user configured views). This tag system was inspired by similar tag systems employed in other areas, particularly email and blog management (Ex. Google Gmail’s Labels).

2. Implementation Structure

To implement tagging for the elements of a mapping model, a simple management system was designed. One Tag Manager manages all the operations related to Tags and Taggable elements, creating, registering, keeping taggables-by-tag and tags-by-taggable relations and executing tag groups’ operations.

2.1. Tag Manager

The tag manager is an one instance class that controls all the tag related operations. It controls tag creation and deletion as well as registering and unregistering of taggable elements. Furthermore, the main responsibility of a tag manager is to control group-wide operations of taggable elements grouped by a common tag.

Examples of group-wide operations:

- hideItems(Tag t)
- showItems(Tag t)
- showItemContext(Taggable item)

52 The context of an mapping element is represented by all the mapping elements that have that element’s specific tag.
2.2 Taggable Interface

The *taggable* interface is implemented by every element of a schema mapper we want to have *tags*. It declares the general methods and properties methods of a *taggable* element.

**General Methods:**
- `getTypeName()`
- `registerTaggable()`
- `unregisterTaggable()`

**Properties Methods:**
- `hideTaggable()`
- `showTaggable()`

The general methods are critical methods for the *taggables*, while the properties methods implement the behavior methods needed to implement group behavior of a *tag* group (Ex. hide all elements with the tag “X”).

### 2.3. Tag

Tags are basically a text label to classify an element. They are associated with *taggable* elements, grouping them so that the tag manager can control them as a whole. The main property of a tag is its title and if this property doesn’t exist the tag isn’t valid.

### 3. Tag Visual Representation

The generic visual representation for a tag has two kinds of identification: the title text and the color of the tag. The title text is unique to each tag and is what differentiates them ultimately, while the color is a visual aid for quick recognition of a tag you’re expecting to see.

On another level, the color of the tag could reflect on the properties of the group of elements it affects. For instance, choosing the option to highlight all the elements with a tag, would highlight them with the tag’s color.

The tags could be visually represented in the elements properties window or in a small popup/mouse-over menu, in small boxes with the title text written upon a colorful background or just the title written in the tag’s color.