Incremental Schema Matching

Philip A. Bernstein
Microsoft Research, USA
Phil.Bernstein@microsoft.com

Sergey Melnik
Microsoft Research, USA
Sergey.Melnik@microsoft.com

John E. Churchill
Microsoft Corp., USA
echurch@microsoft.com

Abstract

The goal of schema matching is to identify correspondences between the elements of two schemas. Most schema matching systems calculate and display the entire set of correspondences in a single shot. Invariably, the result presented to the engineer includes many false positives, especially for large schemas. The user is often overwhelmed by all of the edges, annoyed by the false positives, and frustrated at the inability to see second- and third-best choices. We demonstrate a tool that circumvents these problems by doing the matching interactively. The tool suggests candidate matches for a selected schema element and allows convenient navigation between the candidates. The ranking of match candidates is based on lexical similarity, schema structure, element types, and the history of prior matching actions. The technical challenges are to make the match algorithm fast enough for incremental matching in large schemas and to devise a user interface that avoids overwhelming the user. The tool has been integrated with a prototype version of Microsoft BizTalk Mapper, a visual programming tool for generating XML-to-XML mappings.

1 Introduction

Many data management tasks, such as data translation, data integration, and data warehousing, are driven by schema mappings. *Schema mappings* describe the relationships between the elements of two schemas, e.g., between XML types and relational tables. Schema mappings can be specified directly using data transformation languages such as SQL, XQuery, and XSLT, or using a visual representation in a graphical tool.

A number of commercial visual programming tools are available that help an engineer to produce mappings, such as Altova MapForce [1], BEA WebLogic Workshop [2], IBM WebSphere [8], Stylus Studio [14], and Microsoft BizTalk Mapper [5]. For example, BizTalk Mapper allows an engineer to specify a mapping between two XML schemas using a set of graphical primitives (e.g., links and functoids, illustrated in the screenshot in Figure 1) that describe how elements of the source schema should be mapped to elements of the target schema. Once complete, the mapping is compiled into an XSLT script that transforms source XML documents into target XML documents. The script can be used, for example, to translate XML messages between e-business applications.

Schemas used in enterprise applications often contain thousands of elements. The engineers who build mappings for such schemas are skilled IT professionals who usually possess detailed knowledge of the application domain and are proficient in specifying complex data transformations. Nevertheless, mapping development is costly and labor-intensive, often needing time measured in person-months.

Anecdotal feedback we have received from users suggests that today's commercial schema mapping tools have two major shortcomings. First, basic visualization and navigation tasks, such as finding which schema elements are linked to each other, are seriously impaired when the schemas and mappings get large. Advanced visualization techniques such as the ones in [12] are making their way into the tools to improve usability in such scenarios.

Second, an engineer needs automated assistance to understand and discover the semantic relationships between schema elements. *Schema matching* is a technique for providing such assistance by generating candidate correspondences between schema elements from which the engineer can choose. This problem becomes much harder as the schemas and mappings become large [11].

Schema matching has drawn significant research attention. Since the first survey of schema matching was published in [10], dozens of new schema matching algorithms and prototypes have been suggested in the database and artificial intelligence literature. However, despite years of intensive research, hardly any of the commercial mapping tools incorporate schema matching techniques.

A characteristic feature of past approaches to schema

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the VLDB copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Very Large Data Base Endowment. To copy otherwise, or to republish, to post on servers or to redistribute to lists, requires a fee and/or special permission from the publisher, ACM.

VLDB '06, September 12-15, 2006, Seoul, Korea.
Copyright 2006 VLDB Endowment, ACM 1-59593-385-9/06/09

matching is that they attempt to calculate the set of correspondences between all schema elements in a single shot. Invariably, the result presented to the engineer includes many false positives, especially for large schemas. This is partly inherent, because matching algorithms are heuristic. But it is also partly due to the one-shot approach. For example, when deciding whether to match an address record in the source schema to a 'shipping address' or 'billing address' in the target, the algorithm needs to commit to a particular choice, possibly misguiding its computation of other matches. Such false positives require a lot of manual clean-up. This is aggravated by the absence of user-defined mapping annotations, which makes it nearly impossible for the user to track which match candidates she has validated. As a result, the engineer feels overwhelmed, tempted to discard all of the schema matcher's suggestions and start from scratch. Thus, the single-shot approach to schema matching appears to be a key barrier to user acceptance.

In collaboration with the BizTalk Mapper team, we have been investigating alternatives to single-shot schema matching. We developed a tool that supports the mapping engineer in an interactive, non-intrusive way and mitigates the negative impact of false positives. The tool suggests candidate matches for a selected schema element and allows convenient navigation between the candidates.

The ranking of match candidates uses two kinds of heuristics. First, it leverages known schema-based matching techniques, such as lexical similarity, element types, and schema structure. Second, it exploits the history of the user's prior matching actions to bias the ranking computation toward the schema regions that are likely to be relevant to the selected element. For example, if a source element E's neighbors have all been mapped to the same region of the target schema, it is likely that E will map to that region too. Taking the existing matches and the user action history into account makes the process of mapping creation interactive and personalized. This is a new matching technique to the best of our knowledge.

The technical challenges are to make the match algorithm fast enough for incremental matching in large schemas and to devise a user interface that avoids overwhelming the engineer. To address the former, the matching is done in two phases, first pruning the search space and then performing a more expensive rank computation. The user interface challenge is tackled in part by providing convenient keyboard navigation and highlighting.

The tool has been integrated with a prototype version of Microsoft BizTalk Mapper, which incorporates some of the advanced visualization techniques proposed in [12]. It was demonstrated to customers at the Microsoft Professional Developers Conference [7] and received strong positive feedback. A video presentation of the tool and a customer blog is available from Channel9 on MSDN [6].

Our broader research agenda is to build a model-management system that helps engineers develop metadata applications more effectively using high-level operations on models and mappings [9]. The schema matching task is abstracted as the model-management operator Match [3].

The remainder of this paper describes the scope of the demonstration and highlights some of its technical details.

2 What is Demonstrated

We demonstrate the tool using a real, partially-completed mapping from a customer in a financial domain. The schemas represent loan applications, one using a relatively flat representation and another deeply nested with numerous repeating fragments.

BizTalk Mapper's user interface is split into three vertical panes. The two schemas are displayed in the left and right panes. The engineer uses graphical elements in the middle pane (e.g., lines, cells, functoids, drop down boxes) to describe how elements of the left schema should be mapped to elements of the right schema.

The system can be best understood via a usage scenario (see Figure 1 for a sample screenshot):

1. The user selects an element E of one of the two schemas by highlighting it (e.g., CoSigner1MonthlyAutoPayment in the left pane of the figure). The user then presses a 'hotkey', such as SHIFT, to tell the system to generate candidate matches. The system invokes the schema-matching algorithm to calculate the best candidates and displays a small number of them on the screen as lines from E to the candidate elements. The best candidate according to the system's calculation is highlighted (e.g., the AutomobileMonthlyPayment element in the right pane, one of whose ancestors is the CoSigner element).
2. The user scrolls through the candidates by using the up-arrow and down-arrow keys on the keyboard. The candidates are displayed in rank order of goodness. The first press of down-arrow moves the selection to the second-best candidate according to the system's calculations (e.g., another AutomobileMonthlyPayment element, this time in the context of Borrower). The next press of down-arrow goes to the third best, etc.
3. After the user has selected the candidate C that is the desired match for E, she presses another key, such as ENTER, to cause that selection C to become part of the mapping. That is, the line from E to C becomes permanent and the lines for matches of E to other candidates disappear.
4. The tool now moves to the next element after E. Moreover, if the hotkey key is still pressed, the system immediately calculates the candidate matches for this new selection, as in (1). Thus the user can match one element after another rather quickly, with very few keystrokes and no mouse movement.

Notice that in step (1), the auto-match feature is not intrusive. The user presses hotkey to see if the system produces useful matches. If not, she simply releases the hotkey and the suggested candidate matches disappear.

In step (1), the system displays only a small number of matches. In step (2), the user can select those matches one

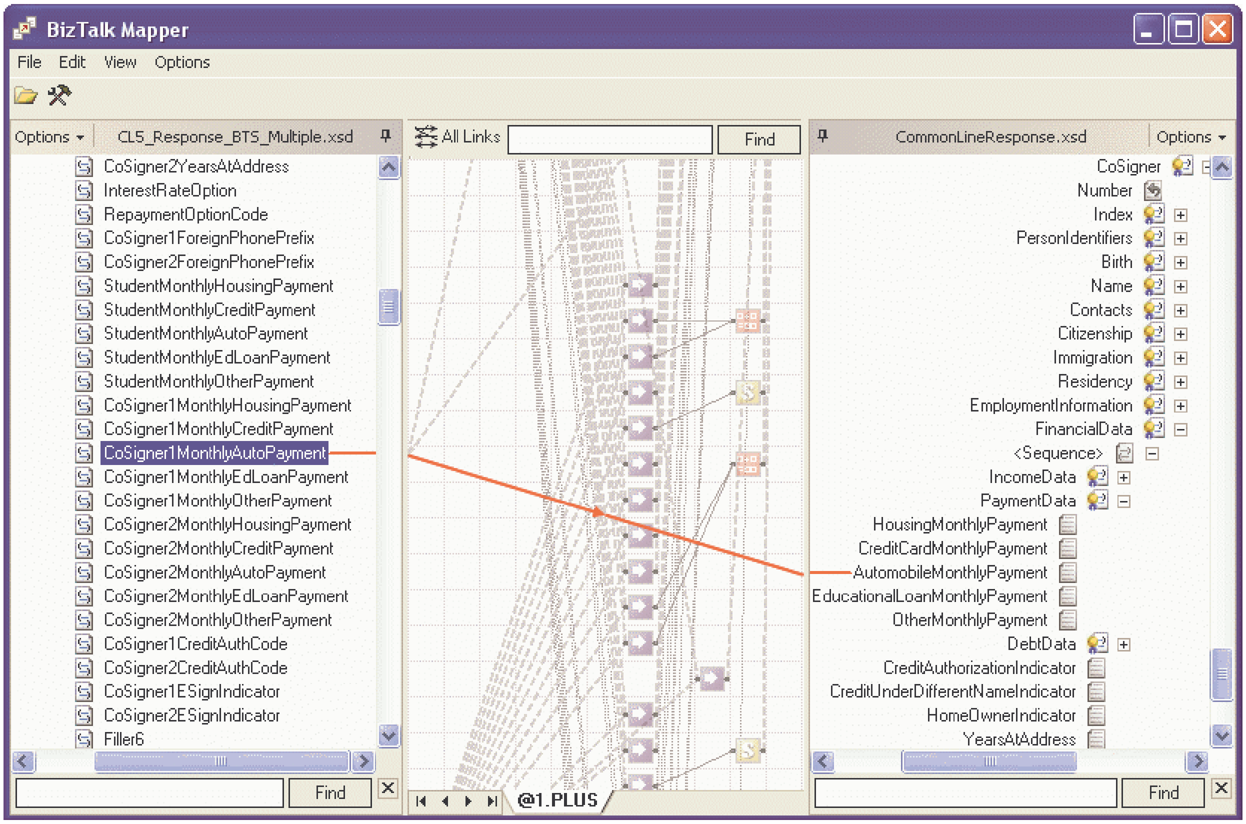
BizTalk Mapper
File
Edit
View
Options
Options
CL5_Response_BTS_Multiple.xsd
All Links
Find
CommonLineResponse.xsd
Options
CoSigner2YearsAtAddress
InterestRateOption
RepaymentOptionCode
CoSigner1ForeignPhonePrefix
CoSigner2ForeignPhonePrefix
StudentMonthlyHousingPayment
StudentMonthlyCreditPayment
StudentMonthlyAutoPayment
StudentMonthlyEdLoanPayment
StudentMonthlyOtherPayment
CoSigner1MonthlyHousingPayment
CoSigner1MonthlyCreditPayment
CoSigner1MonthlyAutoPayment
CoSigner1MonthlyEdLoanPayment
CoSigner1MonthlyOtherPayment
CoSigner2MonthlyHousingPayment
CoSigner2MonthlyCreditPayment
CoSigner2MonthlyAutoPayment
CoSigner2MonthlyEdLoanPayment
CoSigner2MonthlyOtherPayment
CoSigner1CreditAuthCode
CoSigner2CreditAuthCode
CoSigner1ESignIndicator
CoSigner2ESignIndicator
Filler6
CoSigner
Number
Index
PersonIdentifiers
Birth
Name
Contacts
Citizenship
Immigration
Residency
EmploymentInformation
FinancialData
<Sequence>
IncomeData
PaymentData
HousingMonthlyPayment
CreditCardMonthlyPayment
AutomobileMonthlyPayment
EducationalLoanMonthlyPayment
OtherMonthlyPayment
DebtData
CreditAuthorizationIndicator
CreditUnderDifferentNameIndicator
HomeOwnerIndicator
YearsAtAddress
Find
@1.PLUS
Find

Figure 1: Screenshot: best match candidate (right) is an element in the context of CoSigner (as opposed to Borrower)

at a time. If the user has selected the last of the displayed candidate matches and then presses down-arrow, the system displays the next best match and selects it. Thus, the system does not clutter the screen with too many candidate matches, yet gives the user the opportunity to see more candidates if she wants to.

The combination of steps (3) and (4) enable a user to walk through all the elements of one schema, matching each one in turn, without ever having her hands leave the keyboard to use a pointing device (e.g., mouse or trackpad). That is, she first uses the pointing device to select the first element of the schema. After that, she uses her left hand to press and release hotkey, and her right hand to use the arrow keys to select the best candidate. If she likes one of the candidates, then by pressing ENTER she both records her selection and moves to the next element. If she does not like any of the candidates, then she releases the hotkey with her left hand, uses down-arrow to move to the next element to consider, and then presses hotkey again to see candidate matches for this next element. And so forth.

In the research literature, if C is calculated to be the best candidate to match E, and E is calculated to be the best candidate to match C, then the match is called a 'stable marriage,' because neither element prefers another match over the one it is currently assigned. Since the match calculation is heuristic, there is still no guarantee that this stable marriage is the correct match that the user desires. Still, reversing the match direction can help a user validate candidates. Navigating back and forth between the source and target schema can also help a user understand the structure of unfamiliar schemas and tease out their semantics more quickly. The navigation between schemas is supported as follows. In a variation of the above scenario, after step (2), while still pressing the hotkey, the user presses the left-arrow or right-arrow key (depending on which schema contains E), thereby moving the selection to the currently-selected candidate element C in the other schema. Since the hotkey is still depressed, the system calculates the best matches (say E1, E2 and E3) of C to elements of E's schema. Now, the user can decide whether the candidate elements (E1, E2, and E3) are better choices than E to match with C. For example, switching from source element E = CoSigner1MonthlyAutoPayment to target element C = AutomobileMonthlyPayment in Figure 1 produces the candidates E1 = CoSigner1MonthlyAutoPayment, E2 = CoSigner2MonthlyAutoPayment, E3 = StudentMonthlyAutoPayment, etc.

In the demonstration, we illustrate how the individual heuristics implemented in the tool help to identify good candidate matches. We show the robustness of the lexical matching component; e.g., it finds AmtAvaiForReinstatement as a candidate match for ReinstatementAvailableAmount. We illustrate how contextual information encoded in element names is leveraged to disambiguate the context of candidate matches using the nesting structure. We demonstrate how prior user actions and established

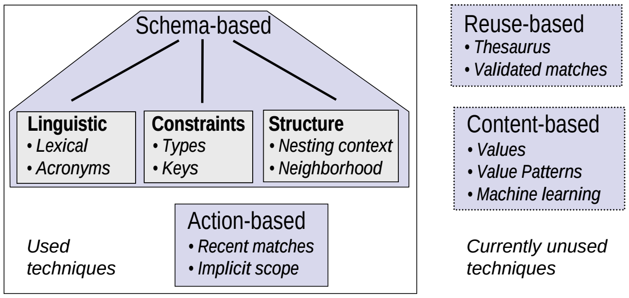

Figure 2: Schema matching techniques used in the demonstrated tool

matches bias the subsequent choice of match candidates.

3 Matching and Tuning

Schema matching heuristics can be roughly classified as shown in Figure 2. In the demonstrated system, we use a combination of schema-based and action-based techniques. The schema-based techniques that we exploit are based on those presented in [4]. For example, the lexical component tokenizes the element labels based on camel case, detects abbreviations, etc. Reuse-based and content-based techniques are currently not exploited.

The ranking of match candidates is obtained using a weighted formula that combines the similarity computed by the individual techniques, e.g., lexical similarity, structural similarity, etc. Each matching technique uses several configuration parameters. To fine-tune these parameters and the weighted formula, we utilize an approach similar to that of [13]. We built a tuning component that takes as input a collection of validated mappings and attempts to find a configuration of weights and parameters that maximizes the accuracy of the automated matcher for the given mapping collection. In the demonstration, we use a parameter setting fine-tuned for an e-business mapping collection.

The rank computation is performed in two modes, on-the-fly and cached. The demonstrated matching tool runs in the on-the-fly mode; using no precomputed state reduces memory requirements and simplifies testing. It is very fast even for large schemas due to pruning of candidate matches prior to ranking. The cached mode is used for tuning the configuration parameters. Since tuning requires thousands of matching runs across multiple schemas, the efficiency gain through caching is noticeable.

Currently, our system supports finding element-to-element links only and does not take into account the network of functoids in the middle pane of the BizTalk Mapper. The functoids are used for specifying more complex data restructuring, built-in functions, and custom computation. Discovering complex data transformations is currently beyond the reach of the state-of-the-art schema matching techniques.

4 Acknowledgements

We thank John Ballard, Uday Bhaskara, Vincent Celie, Patric McElroy, Alvaro Miranda, Prasad Sripathi Panditharadhya, George Robertson, and Andy Wu for their support and feedback on design issues and implementation.

References

[1] Altova MapForce. http://www.altova.com/products_mapforce.html.

[2] BEA WebLogic Workshop. http://www.bea.com/framework.jsp?CNT=index.htm &FP=/content/products/workshop/.

[3] P. A. Bernstein. Applying Model Management to Classical Meta Data Problems. In *Proc. CIDR*, 2003.

[4] P. A. Bernstein, S. Melnik, M. Petropoulos, C. Quix. Industrial-Strength Schema Matching. *SIGMOD Record*, 33(4):38–43, 2004.

[5] Microsoft BizTalk Server 2004. BizTalk Mapper. http://msdn.microsoft.com/library/en-us/introduction/htm/ebiz_intro_story_jgtg.asp, 2004.

[6] J. E. Churchill. BizTalk's Sexy New XSLT Mapper. http://channel9.msdn.com/ShowPost.aspx?PostID=127918.

[7] J. E. Churchill, P. McElroy. Future Directions: Beyond BizTalk Server 2006. Breakout Session DAT319, The Microsoft Professional Developers Conference, Los Angeles, 2005.

[8] C. Lau. Developing XML Web Services with Websphere Studio Application Developer. *IBM Systems Journal*, July 2002.

[9] S. Melnik, P. A. Bernstein, A. Halevy, E. Rahm. Supporting Executable Mappings in Model Management. In *Proc. ACM SIGMOD*, 2005.

[10] E. Rahm, P. A. Bernstein. A Survey of Approaches to Automatic Schema Matching. *VLDB Journal*, 10(4):334–350, 2001.

[11] E. Rahm, H.-H. Do, S. Massmann. Matching Large XML Schemas. *SIGMOD Record*, 33(4):26–31, 2004.

[12] G. G. Robertson, M. P. Czerwinski, J. E. Churchill. Visualization of mappings between schemas. In *CHI '05: Proc. of the SIGCHI conf. on Human factors in computing systems*, pages 431–439, New York, NY, USA, 2005. ACM Press.

[13] M. Sayyadian, Y. Lee, A. Doan, A. Rosenthal. Tuning Schema Matching Software using Synthetic Scenarios. In *Proc. VLDB'05*.

[14] Stylus Studio. http://www.stylusstudio.com/.