Legacy System Integration using a Grammar-based Transformation System

Guido Menkhaus and Urs Frei

Abstract. *Enterprise information system management is the operation of different corporate databases, applications, and more and more often of integration and interoperability of legacy systems, acquired through mergers and acquisitions. These legacy systems produce structured or semi-structured data that add to the vast amounts of data that a company generates every day. This data needs to be communicated between heterogeneous systems within the same company and eventually beyond the company's walls. Transformations of communicated data are required to enable companies to tightly integrate their systems into a cohesive infrastructure without changing their applications and systems. This article presents a transformation system that uses a grammar-based approach to provide direct integration of applications and systems at the data level. Sequences of transformations allow flexible and effective exchange of data between heterogeneous systems resulting in a single information network.*

Keywords. Legacy System, Transformation, Enterprise Information System

1. Introduction

Most established companies have acquired legacy systems through mergers and acquisitions. The systems were developed independently of each other and very often they do not align with the evolving IT infrastructure. Still, they drive day-to-day business processes. Replacing the legacy application with new solutions might not be feasible, practical or cost a considerable amount of time. However, immediate integration might be a requirement for a strategic project, such as supply chain management or e-business [7, 15].

This article presents a legacy system data integration middleware that allows flexible and effective transformation of data between heterogeneous systems. Our data integration middleware provides a transformation system in which transformation sequences are described based on the grammar of the format of the source and the target data. It provides direct integration of applications and systems at the data level.

The remainder of the article is structured as follows: The motivation and the requirements of this work are discussed in Section 2. Section 3 provides a brief overview about related work and Section 4 presents transformation systems. Section 5 illustrates a use-case scenario for legacy data integration. The architecture of the system is presented and discussed in Section 6. Section 7 concludes the article with a brief talk about our future research directions and work.

2. Motivation and Requirements

For companies to stay competitive, it has become important to interconnect seamlessly their databases, applications and legacy systems into a coherent IT infrastructure. Integrated, flexible and extensible enterprise information systems allow providing services to the maximum efficiency. However, heterogeneous systems including legacy systems acquired through mergers and acquisitions often do not easily integrate with other enterprise-wide applications. They become barriers to agility and innovation [1]. These systems use different communication protocols and produce data in proprietary format. We need to transform the data, control that data and ensure that the transformation from one format to another is correctly carried out.

Transforming data is usually done by writing custom programs [6]. However, if either the format of the source data or the target data changes, if new requirements emerge, the custom programs need to be rewritten. Adapting to frequent changes results in high maintenance costs. To integrate legacy system data we need transformation systems that provide the following features:

1. *Adaptation:* The way data is processed and stored is diverse and might be subject to changes. If the format of the source data or the target data in a transformation sequence changes, quick adaptation to the transformation sequence is essential to sustain system interconnection.

2. *Control:* When data is transformed while communicated between two systems, the target system might require the data to change, to be enriched, filtered, and modified.

3. *Format Guarantee:* The transformation sequence guarantees that the data results in a specified format. The specified structure of the target data is produced, because the transformation is generated based on the grammar describing this structure.

We present a grammar-based transformation system, in which the transformation sequence is generated originating from a set of grammars describing the target data structure of each transformation step. Semantic controls need to be programmed manually. The system provides means to integrate them into the transformation sequence. Adaptation is accomplished by respecifying the grammars describing the data structures.

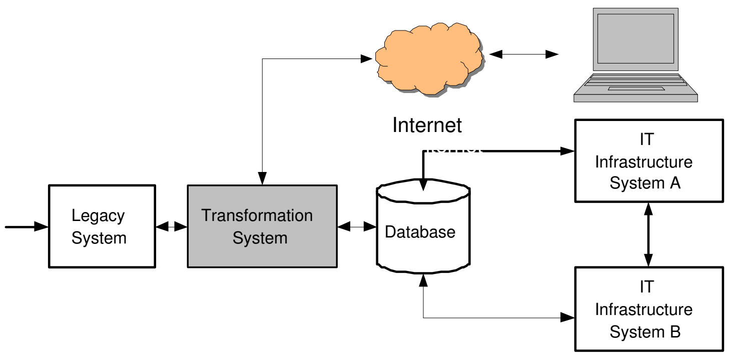

Figure 1: **Application scenario: Integration of legacy system**

3. Related Work

This section provides a brief overview about the related work. It also aims at giving a short view on legacy system modernization and integration techniques.

Software system evolution activities can be divided into three categories: maintenance, modernization, and replacement [18]. System maintenance supports the evolution of the system according to the business needs but has its limitations, since maintenance does not include major structural changes. Modernization involves extensive and pervasive changes, requiring a significantly greater effort than during maintenance activities. Replacement is necessary if the system can not keep up with business requirements [4]. Legacy systems are therefore characterized as ”information systems that significantly resist modernization as part of an evolution towards delivering business solutions”[18, 2].

The most proven solution to legacy integration and modernization is legacy system wrapping [20]. It can be subdivided into wrapping presentation modules, functionality and data.

Carr presents a technique for user interface modernization, where new user interfaces wrap old, text-based interfaces [3].

Wrapping functionality for utilization of legacy system in a distributed environment to cope with new requirements that the Internet has introduced, is provided by Yoshioka *et al*. They use CORBA and DCOM technology to wrap the functionality and to provide network-centric communication. However, the legacy system needs to provide the appropriate level of abstraction, be modular and fine-grained enough to allow wrapping the business rules and functions [1]. The ERCOLE project provides a process that describes how to wrap legacy applications with OO systems. The process involves encapsulation, reengineering and concepts for the co-existence of objects with legacy application functionality [9].

Wrapping legacy data involves the addition of an extra layer or bridge to provide transparent access to the legacy system. Ontology Works provides an ontology and bridges to map data from legacy information systems onto the ontology [12]. DataMirror [5] provides an engine for bi-directional data transformation, exchange and integration. The engine incorporates build-in functionality and promises that zero-programming is required for application integration.

As described, many researchers have been active to develop diverse approaches allowing integration of legacy system at the data level. However, not many have been conducted to build grammar-based systems that allow transformation of data and flexible adaptation of changing data structures.

4. Short Overview of Transformation Systems

Transformation systems transform elements of a source language into elements of a target language. The source and the target language can be very different from one another [19]. Partsch and Steinbruggen divide transformations into two types of processes [13]: *procedural* and *schematic*. Procedural transformations specify semantic rules that can be applied globally to the entire source data. They have applicability terms that contain semantic conditions that are not easily decidable by inspecting the syntax. Procedural transformations include consistency checks and analysis tasks. Schematic transformations are syntax-oriented; their applicability conditions can be verified by checking a portion of the syntax. They make local changes to the source data. It should be noted that global, procedural transformations can be accomplished by schematic processes, but that the required transformation might become arbitrary complex. Therefore, complex rules are better expressed applying procedural rather than schematic transformations [19].

Partsch and Steinbruggen classify transformation systems into manual, semi-automatic, and automatic transformations [13].

- *Manual:* In manual transformation systems, the user chooses from a predefined set of transformations those, which the user wants to apply to the source language. Manual transformation systems provide an environment that puts the user in the position to use transformations more effectively than the current program-

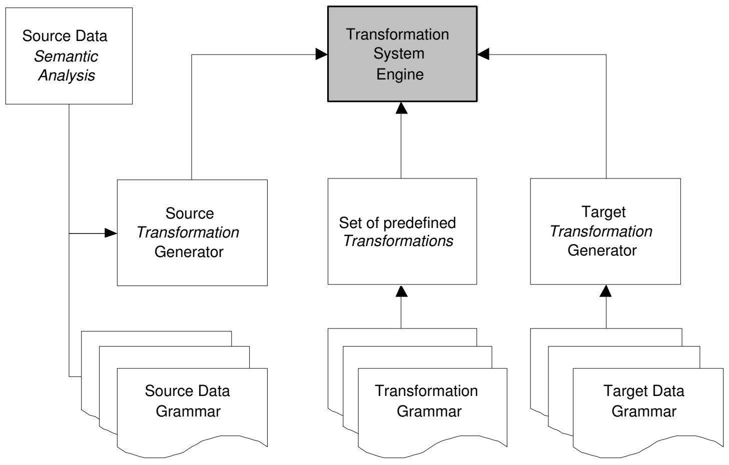
Source Data
Semantic Analysis
Transformation System Engine
Source *Transformation* Generator
Set of predefined *Transformations*
Target *Transformation* Generator
Source Data Grammar
Transformation Grammar
Target Data Grammar

Figure 2: **Transformation System Architecture.**

ming paradigm that requires s programmer to manually code a transformation.

- *Semi-automatic:* The objective of semi-automatic transformation systems is to automate the process of transforming and to minimize the intervention of the user. Although the major decisions will still be made by the user.
- *Automatic:* The intent of automatic transformation system is to fully automate the transformation process.

The class of problems that can be solved using manual transformation systems is the largest, since most transformation solutions require insight in the problem domain and decision taking that is beyond what automation techniques can do. Semi-automatic systems need a restricted problem domain where difficult decisions about transformation configuration do not occur and transformations can be generated automatically. Most limitations are in the automatic transformation system class, where the system selects on the basis of a knowledge base the transformation sequence. However, the system can only be as good as the programmer has designed the knowledge base.

In this paper, we present a transformation system that is semi-automatic. The automatic part of the system is schematic-based and syntax-oriented. The procedural part of the transformation consists of semantic analysis and actions, which are applied to the entire source data, which need to be programmed manually, since this part requires insight into the problem domain.

5. Application Scenario

The transformation system was designed as middleware for the integration of legacy systems. This section outlines in brief a practice area, which demonstrates the value of our transformation system.

Supply chain management helps companies in controlling the flow of information and goods within their network of suppliers and customers by providing a full view on what happens in the network [7, 15]. But before extending operation management beyond the company's wall and integrate companies' suppliers and customers into a single information network, the company's own operations must run smoothly towards cooperation and collaboration. This involves the integration and interoperability of different corporate databases, applications, and more and more often of legacy systems. These legacy systems produce structured or semi-structured data. This data needs to be communicated between heterogeneous systems within the same company and eventually beyond the company's walls (Figure 1). Transformations of communicated data are required to enable companies to tightly integrate their systems into a cohesive infrastructure without changing their applications and systems.

In our application scenario, the legacy system produces data. This data is checked and verified, transformed into an internal and intermediate XML format, and finally imported into a central database and prepared for publishing on the Internet.

6. Architecture of Transformation System

The architecture of the transformation system is illustrated in Figure 2. The transformation system runs a sequence of transformations, in which source data complying with a source data grammar is transformed into target data described by the target grammar. The schematic part of each transformation sequence is generated using parser and transformation generating systems. The procedural aspect is manually programmed and integrated in the schematic part.

Each transformation in a sequence consists of three intermediate subtransformations, which are driven by:

- Source Data Grammar
- Configuration
- Target Data Grammar

Source Data Grammar-based Transformation The source data grammar based transformation (SD) is decomposed into an analysis and a synthesis part.

Analysis: The analysis consists of the lexical, syntactic, and semantic analysis. It breaks a complex structure into elementary pieces.

1. *Lexical Analysis:* The lexical analysis is done using a scanner component. The scanner is generated on the basis of a lexical analysis specification of the source data and produces a sequence of tokens. A token is a syntactically structures symbol, whose structure is described in the lexical analysis specification.

2. *Syntactic Analysis:* The sequence of tokens produced by the scanner is forwarded to the parser, which verifies the structure of the source data against the source data grammar. We use an attributed grammar for structured and semi-structured data.

3. *Semantic Analysis:* The semantic analysis checks local and global context conditions. It checks conditions that can not or are hard to be verified using a syntactic analysis. The results influence subsequent semantic analysis steps.

Synthesis: The synthesis is a process that proceeds from elementary to complex structures. In the transformation phase a complex target document is constructed from simple elements.

1. *Transformation:* The transformation converts the data that has passed the syntactic and semantic analysis into an internal, intermediate format.

We use CoCo/R [11] as transformation tool. The lexical analysis specification is described by regular expressions. The attributed grammar of the source data is defined in EBNF. Attributed grammars were introduced by Knuth in [8] to formalize the semantics of context-free languages. They describe in their original form dependencies between attributes of symbols, originating from the lexical analyzer. However, attributed grammars can be seen as a dynamic description of a process, i.e. as a syntax-directed algorithm. The structure of the source data determines the order of the global semantic analysis and the local transformations.

Configuration-based Transformation The configuration driven transformation system (CD) contains a set of CD types with associated configurations for different target data formats. The CD transformation bridges the source and target transformations. It decouples the source data grammar from the target data grammar so that the two can vary independently. This avoids a binding between the associated transformations and allows flexible adaptation in case of a modification or extension of the source and the target data grammar, respectively.

Target Data Grammar-based Transformation The target data grammar based (TD) transformation is generated from the target data grammar. It takes the data from the CD transformation and generates data in the target grammar format. Since the transformation is produced from the target grammar, the transformation system guarantees that the data results in the specified format.

6.1. Legacy System Integration

In the application scenario we take the data from the legacy system and

1. Import the data into a central database.

2. Prepare them to be published on the Internet.

The transformation system applies a sequence consisting of two transformations. For the import into the database we convert the legacy data into XML format while verifying the data during the SD subtransformation. The second transformation parses and processes the data before importing it into the database. For publishing on the Internet we substitute the second transformation with a XSLT transformation.

6.2. Token-XPath Matrix

The first transformation converts the legacy system's proprietary data format into an intermediate format (Figure 3).

In the SD subtransformation, the parser is generated from an attributed grammar. The semantic verification (in our application scenario suppression of duplicate data entries in the source data) is manually programmed and integrated into the generated parser as is the following CD subtransformation via a callback style.

The CD subtransformation determines where data, originating from a token produced by the scanner component and semantically checked and converted during the semantic analysis, is inserted into the resulting XML document, serving as an intermediate data format in the transformation sequence. This is performed applying a Token-XPath-Assigment matrix (TXPA matrix) $M_{TX} = T \times X$, which consists of the tokens symbols T of the source data grammar and the target data grammar XML elements, expressed as XPath elements X. The target grammar is presented as a XML Schema. The target grammar driven subtransformation is generated using JAXB [16], which generates a suite of hierarchical classes that produces an XML document complying with the XML Schema. This suite of classes is subsequently used by the CD and the TD transformation. To accomplish the construction of the XML document using the hierarchical suite of classes we use reflection to determine the class representing the root element. An object of the class is instantiated using a

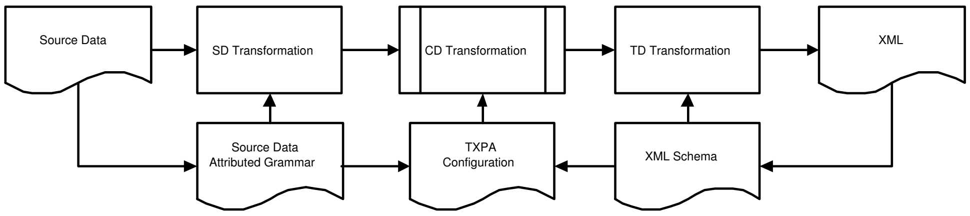

Figure 3: **Transformation from legacy data to XML.**

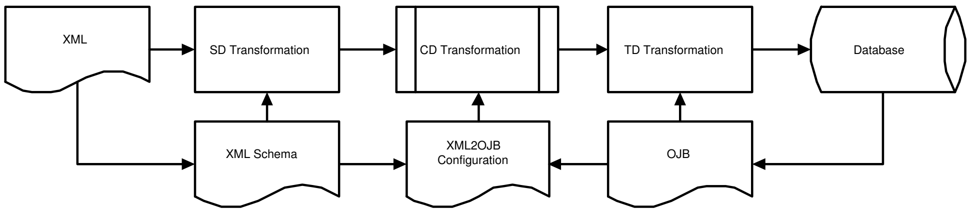

Figure 4: **Import from XML into a database.**

factory. The factory is analyzed for objects that are required to be present for the resulting XML document being grammatically correct. These objects are being instantiated dynamically, filled with the corresponding data and integrated in the XML document. This is done using a set of heuristics in order to find the correct methods and classes.

The intermediate (CD) subtransformation decouples the source and the target grammar driven subtransformation. If the source or the target grammar is modified or the semantic analysis changes, only the TXPA matrix needs to be adapted. This makes the transformation system flexible and robust in the case of changes.

6.3. XPath-Database Configuration

The second transformation imports the data from the XML document into a database (Figure 4). Most databases allow importing XML data or comma-separated value lists. However, data can only be inserted into a single table, and most often this data requires further processing such as splitting the data and distributing the data among several database tables.

The SD transformation is accomplished employing an XML parser. The CD and the TD transformations use OJB [17]. OJB generates a set of classes on the basis of a database design allowing transparent persistent mapping of objects against relational databases. It allows storing objects, or part of an object in relational databases, and reading data from a relational database into the generated object structure. The grammar-oriented transformation needs to rework the data from an XML into an OJB object representation. The OJB object structure is then imported into the database.

The objective of the CD transformation is to remain independent from the grammar of the source XML document and the target configuration of the database. We need to take into account the following requirements:

- Specification of a mapping between XML elements and OJB objects.
- Instantiation of OJB objects creating a new dataset.
- Relations between the OJB objects.
- Processing of duplicate datasets. Duplicates are already filtered out in the first transformation. However, at this stage we cannot detect duplicates, which might occur during the reordering of the data in the second transformation, nor can we detect duplicates that are already in the database.
- Declaration of an import sequence to prevent primary key violation.

We have developed XML2OJB, a mapping from XML documents to OJB object structure [10]. It allows flexible, adaptable, and independent import of arbitrary structured XML data into arbitrary database table configuration. The XML2OJB configuration is divided into five parts.

- *Target Definition:* This section defines the target objects that are imported into the database.
- *Source Definition:* This element declares where to find the necessary information in the XML source document.
- *Duplicate Record Definition:* The Duplicate Record section specifies the element that functions as autokey in the database. The specification of an autokey is necessary to avoid duplicate entries in database tables.
- *Workflow Definition:* The ImportSequence section determines the sequence in which OJB objects import their data into the database. Together with the assembly section they specify a control flow when and where data is inserted into the database. A Repeat element specifies the start of a new data record in the XML document. The CreateObject element defines the objects that are required to be instantiated and an Insert element specifies where the data is set in created the OJB objects.

The TD transformation consists of importing the set of OJB classes into the database. The process is configured using a specific OJB configuration file.

6.4. Internet Publishing

Maverick [14] is framework that is build around the Model-View-Controller principle. It is intended to publish data on the Internet and to process incoming requests. Maverick is composed of commands. Each command consists of a set of views and a controller. The controller connects to the model and decides which view renders data from the model. The view then executes a XSLT transformation and publishes the data.

7. Conclusion

We have presented a transformation system that manages sequences of transformation. Each transformation is made up of three individual subtransformations, which are grammar-based. The objective is to decouple the source and target grammar transformation by using an intermediate transformation. Semantic analysis and the configuration of the intermediate transformation require domain knowledge and therefore this task is done manually.

The TXPA matrix maps a sequence of tokens onto XML elements. The XML2OJB configuration maps XML elements to OJB objects, which can be imported into a relational database. Communicating data via the Internet is accomplished applying a XSLT transformation and the Maverick framework. The TXPA matrix and the XML2-OJB transformation proved to be successful due to their flexibility. The architecture of the transformation system represents a viable solution for rapid legacy systems integration requiring frequent reconfiguration and maintenance.

Future work will focus on extending the set of predefined transformations. We will continue working on fault tolerance and error recovery within a single transformation.

References

[1] H. A. Aminian. The Legacy System Dilemma: Making the Right Choices. Technical report, Insurity, 2003.

[2] M. Brodie and M. Stonebraker. *Migrating Legacy Systems Gateways, Interfaces and the Incremental Approach*. Morgan Kaufman, 1995.

[3] D. Carr. Web-enabling legacy data when resources are tight. *Internet World*, 1998.

[4] S. Comella-Dorda, K. Wallnau, R.C. Seacord, and J. Robert. A Survey of Legacy System Modernization Approaches. Technical Report CMU/SEI-2000-TN-003, Carnegie Mellon University, Pittsburgh, PA, USA, 2000.

[5] DataMirror. Managing Your Data the XML Way: Data Transformation, Exchange and Integration, 2001.

[6] P. Fiore. Data Warehousing. *Evolving Enterprise*, 1(1), Spring 1998.

[7] R. Hieber. *Supply Chain Management. A Collaborative Performance Measurement Approach*. vdf Hochschulverlag, Zürich, Switzerland, 2002.

[8] D.E. Knuth. *Mathematical System Theory 2*, chapter Semantics of Context-Free Languages, pages 127 – 145. D.E. Knuth, 1968.

[9] A. De Luzia, G.A. De Lucia, A.R. Fasolino, and P. Guerra. Migrating Legacy Systems towards Object-Oriented Platforms. In *International Conference on Software Maintenance (ICSME)*, 1997.

[10] G. Menkhaus and U. Frei. Transformation-oriented Middleware for Legacy System Integration. In *International Conference of Enterprise Information Systems*, Porto, Portugal, April 2004.

[11] H. Mössenbeck. A Generator for Fast Compiler Front-Ends. Technical Report Report 127, Institut für Computersysteme, ETH Zürich, 1990.

[12] Ontology Works. Integration of Legacy Information Systems. Technical report, Ontology Works, 2002.

[13] H. Partsch and R. Steinbruggen. Program Transformation Systems. *ACM Computing Surveys*, 15(3), September 1983.

[14] Sourceforge. Maverick, 2003.

[15] M. Stör, N. Birkeland, J. Nienhaus, and G. Menkhaus. IT Infrastructure for Supply Chain Management in Company Networks with Small and Medium-sized Enterprises. In *Proceedings of the 5th International Conference of Enterprise Information Systems*, volume 4, pages 280 – 287, Angers, France, April 2003.

[16] SUN Microsystems. Java Architecture for XML Binding (JAXB), 2003.

[17] The Apache DB Project. Object/Relational Bridge (OJB), 2003.

[18] N. Weiderman, H. Nelson, J.K. Bergey, and D.B. Smith. Approaches to Legacy System Evolution. Technical Report CMU/SEI-97-TR-014, Carnegie Mellon University, Pittsburgh, PA, USA, 1997.

[19] V.L. Winter. Program Transformations in HATS. In *Proceedings of the Software Transformation Systems Workshop*, California, USA, 1999.

[20] M. Yoshioka, T. Sudo, A. Yoshikawa, and K. Sakata. Legacy System Integration Technology for Legacy Application Utilization from Distributed Object Environment. *Hitachi Review*, 47(6):284 – 290, 1998.