Applied Computer Science, vol. 14, no. 4, pp. 93–108
doi:10.23743/acs-2018-32

Submitted: 2018-09-06
Revised: 2018-12-01
Accepted: 2018-12-16

Business standards, Interoperability, Canonical Data Models,
Graphs, Graph Databases, Graph Transformations

Leszek JASKIERNY [0000-0002-9451-8569]*

REVIEW OF THE DATA MODELING STANDARDS AND DATA MODEL TRANSFORMATION TECHNIQUES

Abstract

Manual data transformations that result in high error rates are a big problem in complex integration and data warehouse projects, resulting in poor quality of data and delays in deployment to production. Automation of data transformations can be easily verified by humans; the ability to learn from past decisions allows the creation of metadata that can be leveraged in future mappings. Significant improvement of the quality of data transformations can be achieved, when at least one of the models used in transformation is already analyzed and understood. Over recent decades, particular industries have defined data models that are widely adopted in commercial and open source solutions. Those models (often industry standards, accepted by ISO or other organizations) can be leveraged to increase reuse in integration projects resulting in a) lower project costs and b) faster delivery to production. The goal of this article is to provide a comprehensive review of the practical applications of standardization of data formats. Using use cases from the Financial Services Industry as examples, the author tries to identify the motivations and common elements of particular data formats, and how they can be leveraged in order to automate process of data transformations between the models.

* AGH University of Science and Technology, Faculty of Electrical Engineering, Automatics, Computer Science and Biomedical Engineering, Department of Computer Science,
e-mail: leszekj@agh.edu.pl

1. INTRODUCTION

Although from a business perspective data models are created to support particular business objectives, from the practical perspective they must be focused on particular outcomes and selected groups of the end users. Both, technical complexity of the model and level of details included in the data model are dependent on the reason why the model was created and on its target audience. Looking from this perspective, models can be divided into sub-categories such as: conceptual models, logical models, and physical models. From a technical perspective, models can be defined as structured models, requiring strict definition of the details of the model, and non-structured (conceptual) models, built from more generic, loosely coupled objects. Conceptual models are created using various modeling techniques that don't enforce formal validation of the model. Structured logical models are created using modeling techniques supporting formal validation of the model. A good example of such a model is a UML class diagram

Finally, physical models can be represented by various schema definitions, e.g. XML Schema Definition (XSD), that specify how to formally describe the elements in an Extensible Markup Language (XML) document.

Such a formal definition is mandatory for documents being exchanged between applications. Further specifications, such as Web Service Description Language (WSDL), are built on top of XSD, to specify functional (business related) and non-functional (IT systems related) aspects of the data exchanged between applications.

2. REVIEW OF DATA FORMAT STANDARDIZATION METHODS AND TECHNIQUES

The following chapter provides an overview of data modeling languages and standards.

2.1. Data Format Description Language

Data Format Description Language (DFDL), published as an Open Grid Forum Proposed Recommendation in January 2011, is a modeling language for describing general text and binary data in a standard way. A DFDL model (or schema) allows any text or binary data to be read (or "parsed") from its native format and to be presented as an instance of an information set.

Unlike XSD, which is usually created to model an existing business environment, DFDL takes a practical approach, building a model based on the existing data structures where, in some cases, source data might not be well formatted.

An information set is a logical representation of the data contents, independent of the physical format. For example, two records could be in different formats, because one has fixed-length fields and the other uses delimiters, even though they contain exactly the same data and are both represented by the same information set.

2.2. Dataset Structure Definition

Dataset Structure Definition (DSD) describes how information in a specific dataset is structured. Knowledge of the structure is important, because it allows desired information to be filtered out and very precisely limited specific dimensions based on selected criterions. In addition to DFDL, DSD provides a mechanism to model, not only particular data items, but also complex data structures.

2.3. UML

Unified Modeling Language is a general-purpose modeling language, intended to help visualize the design of a system. Data models, as part of the overall design of a system, can be visualized using Class Models. Class Models significantly differ from business conceptual Entity-Relationship Diagrams because of their focus on object-oriented design. Class models can be used at any stage of the modeling process, i.e. to create conceptual models, logical models and physical models. Although Class Models have several drawbacks they set a foundation for other notations, e.g. XSD.

2.4. XML

Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable (Thompson & Lilley, 2014). The design goals of XML emphasize simplicity, generality, and usability. It is a textual data format with strong support via Unicode for different human languages (Bray, Paoli, Sperberg-McQueen, Maler & Yergeau, 2008).

2.5. XML Schema Definition (XSD)

XSD, a recommendation of the World Wide Web Consortium (W3C), specifies how to formally describe the elements in an Extensible Markup Language (XML) document. Such a formal definition is mandatory for the documents being exchanged between applications. Further specifications, such as Web Service Description Language (WSDL), are built on top of XSD to specify functional (business related) and non-functional (IT systems related) aspects of the data exchanged between applications.

3. INDUSTRIAL DATA MODELS

The following chapter provides an overview of the standards from the Financial Services Industry, influenced mainly by the commercial organizations, such as Society for Worldwide Interbank Financial Telecommunication (SWIFT), who provides a network that enables financial institutions worldwide to send and receive information about financial transactions in a secure, standardized and reliable environment. Over the past several years, SWIFT provided strong commercial support, leading to the creation of widely adopted messaging standards, e.g. ISO 15022 and ISO 20022. The following examples aim to provide an overview of those standards, in order to identify common characteristics driving particular vendors and particular industry solutions.

3.1. SWIFT MT

SWIFT allows financial and non-financial institutions to transfer financial transactions through a financial message (“SWIFT”, 2018).

While SWIFT started primarily working with the simple payment instructions, it now sends messages for wide variety actions including security transactions and treasury transactions. Nearly 50 percent of SWIFT traffic is still for payment-based messages, 43 percent now concern security transactions, and the remaining traffic flows to treasury transactions.

SWIFT can be considered as a true pioneer of the global standardization of data formats on the industry level. All the messages are built to comply with various needs and regulations, while maintaining core compatibility across the industry.

The main benefit of SWIFT messages is their business content, while standardization of the data format is important, the main challenge is in ensuring that message’s business content meets the following criterion:

- Universal core business entities are shared by the whole industry,
- Extendable, specific extensions are built on top of the core components of the model in order to preserve compatibility, while maintaining customer-specific features,
- An open model has to be prepared for future extensions and changes, while maintaining its core compatibility.

By setting-up the above foundations, SWIFT made an important step towards industrialization of the data exchange processes. SWIFT messages consist of five blocks of data including three headers, message content, and a trailer. Message types are crucial to identifying content. All SWIFT messages include the literal “MT” (Message Type). This is followed by a three-digit number that denotes the message category, group and type.

3.2. ISO 20022

CIOs and IT architecture experts from leading banks and other major financial institutions, along with independent software vendors, systems integrators, and SAP, have joined in a collaborative effort to shape the future of enterprise service-oriented architectures (enterprise SOA) in the banking industry. This joint effort – known as the SAP industry value network (IVN) for banks – was launched in September 2005 and had 35 members ("ISO 20022", 2018). Although the main focus of IVN was creation of the reusable services, creation of the appropriate data model was necessary in order to assure data consistency and interoperability of those services. SWIFT, as one of the strong contributors to IVN for Banking, took over work on the definition of a comprehensive data model that would cover main activities of the banks. This effort greatly enhanced the previous standardization efforts of SWIFT, mainly related to payments and asset transfers. The ISO 20022 standard is described in the document "ISO 20022 Financial Services – Universal financial industry message scheme".

3.3. OASIS Universal Business Language (UBL)

UBL, the Universal Business Language, defines a royalty-free library of standard XML business documents supporting digitization of the commercial and logistical processes for domestic and international supply chains such as procurement, purchasing, transport, logistics, intermodal freight management, and other supply chain management functions (Holman, 2018). UBL can be thought of as a language that allows disparate business applications and trading communities to exchange information along their supply chains using a common format. UBL also provides the opportunity to end the debate over standards for business document formats that has discouraged the adoption of new technologies for conducting business in the digital age.

4. PROPRIETARY DATA MODELS

Despite high level of standardization, particular organizations (banks, insurance companies, payment providers, etc.) still tend to use proprietary data formats that meet particular business requirements and allow them to distinguish from their competitors.

Because standardization processes are expensive, there is always a need for strong commercial support in order to produce tangible results. Commercial institutions are funding developments of internal data models, that are focusing on specific business requirements and following the particular technical limitations of a given organization (McKnight, 2014). Those models (such as ISO20022) are often used as a central model, spanning other data models used in the organization (e.g. models defined by particular software systems) and can be referenced as Canonical Data Models (CMD) (Roman, 2006).

Proper construction of the CMD is especially important from the perspective of ensuring efficient and flawless Data Transformation Processes. A systematic approach to CMD creation, resulting in the highly optimized data model, can be done using the following steps:

- Review the Domain Models already used in the Organization,
- Prepare a common data dictionary, including the main business and technical datatypes,
- Define business areas and main groups of data, together with a dependency matrix between those areas,
- Review other relevant industry models, e.g. SWIFT (ISO 15022 and ISO 20022),
- Map business requirements on the CMD templates created during past projects,
- Build versioning mechanisms directly into the model (CMD),
- Extend the meta-data related to the CMD, e.g. define defaults for particular datatypes, iterate and enhance the model with business owners.

Some of the commercial models have been wrapped up and packaged in the form of the templates that are distributed on a commercial basis. An example is ADRM Software – the leading independent provider of large-scale industry-specific data models. ADRM Enterprise Data Models provide the reference for related industry business areas, data warehouses and data mart models (“ADRM Software”, 2018). The ADRM Enterprise Data Model incorporates integrated data requirements taken from the best-practices developed by organizations from particular industries, combined within a single model consisting of approximately 400 entities and 2,500 attributes. It is the essential data model for strategic planning, communicating information requirements across the organization, developing integrated systems and organizing data in the Business Area, Data Warehouse and Data Mart models. Each Enterprise Data Model is built upon a common core of entity building blocks, which contributes those same common entities for the construction of business area, data warehouse, data and application models.

A common set of core entities enables the related models to be consistent and extendable across the industries. The same common core of entities is used wherever applicable in other industries, thus providing a means of integrating data across different industries or lines of business. Such a data model has several common design characteristics:

- Industry-specific design,
- Comprehensive business area coverage,
- Is fully-attributed,
- Provides complete and detailed definitions,
- Is semantically clear and easy-to-understand,

– Reflects current industry data best-practices,
– Provides flexible and extendible design,
– Utilizes industry-standard data whenever possible,
– Can be presented using large format graphic representation,
– Supports a wide audience of interests,
– Is integrated with Business Area, Data Warehouse and Data Mart models.

5. BANK-MF PROPRIETARY MESSAGE FORMAT OF BANK X

Bank-MF (or Bank Message Format) is used as an example of a proprietary data format; the following example is not directly related to any particular implementation, but rather summarizes common elements of various implementations of proprietary data models.

Note: Data Model and Message Format are used here as synonyms; this is because focus of this article is on the data models used for the exchange of information between various IT systems. Usually this kind of data exchange is done using messages built upon technical formats (such as XML or JSON) and business data models.

In order to understand the complexity of the model, let's analyze the following diagram that presents the components of the sample solution: The Lego bricks represent messages compatible with Bank-MF data model (or Canonical Model), while wooden bricks and Post-It notes represent proprietary data formats, derived from particular IT systems.

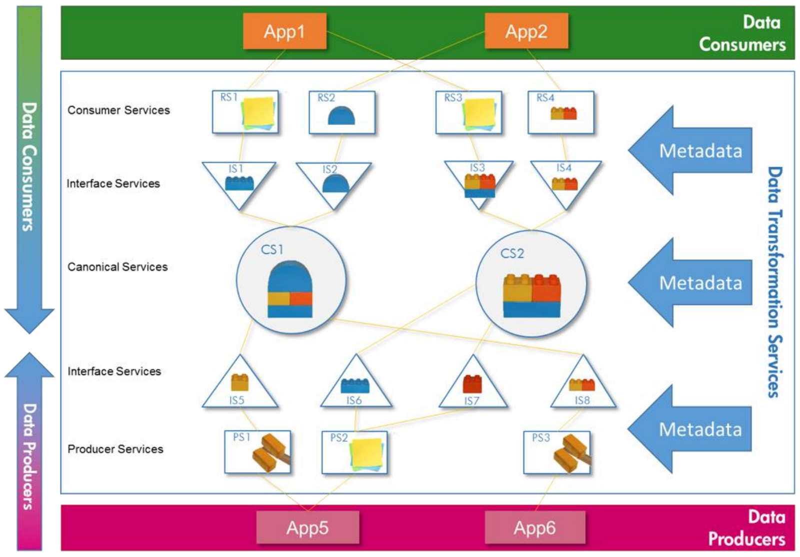

Fig. 1. Data transformation via Canonical Model

Although particular proprietary models are representing the same business data, there is a need for additional knowledge (referenced here as Metadata), in order to translate data between particular formats. The most important observation is that a canonical model is central to all other models, i.e. trans-formation between any pair of the proprietary models always goes via a Canonical Data Model. In order to facilitate mediation between a large number of proprietary models, Bank-MF has to meet the same criteria, as defined in a foundation of the SWIFT MT messages: universality, extensibility and openness.

The sample model referenced in this article was defined as a complex XSD schema, with clearly defined common components, basic data types, complex data types and orchestration of the messages into complex structures, representing particular business objects.

6. CHARACTERISTICS OF THE GRAPH MODELS

Graph theory is the study of graphs, which are mathematical structures used to model pairwise relations between objects. A graph in this context is made up of vertices, nodes, or points, which are connected by edges. A graph may be un-directed, meaning that there is no distinction between the two vertices associated with each edge, or its edges may be directed from one vertex to another (Angles & Gutierrez, 2008). The edges may be directed or un-directed.

In case of this article, we will focus on directed graphs, defining a clear relation between a parent node and a child node. In addition, each node of a graph will be equipped with a set of the attributes, defining proprietary characteristics of this node.

7. TRANSFORMATION OF THE DATA MODEL TO ITS GRAPH REPRESENTATION

As described earlier, there is large number of various data modeling standards and patterns common in multiple commercial implementations. In order to enable automated transformation of the data model, first we need to perform initial transformation between the actual definition of the model and reference definition, expressed in form of the graph. Alongside this transformation, we are also trying to understand the business context of given data model. Understanding the context is mandatory, because proper transformation has to be based on the business context of the data, not only on its technical implementation (Sleger, 2010).

In this article, we will attempt to provide a generic prescription for how to map any proprietary data model on the formal definition of the graph. The following chapter describes formal methods and techniques used to transform structured (e.g. XSD), or unstructured (e.g. NoSQL) data models into graph representations.

Further on, this process will be called universalization (or initial transformation) of the data model, i.e. transformation of a given model to its universal form. Normalized models would be compatible in terms of formal definitions, such as definition of the dependencies, list of attributes, etc.

Unlike the highly automated Data Model Transformation Process described later in this article, Initial Transformation might require signification support from the Subject Matter Expert (SME) knowledgeable about the business context of the model being transformed. The ultimate goal is to make the transformation process automated to its maximum possible extent. In order to achieve a high level of automation, we need to build a knowledge base containing already processed patterns and templates.

Presented below is an example from the financial industry.

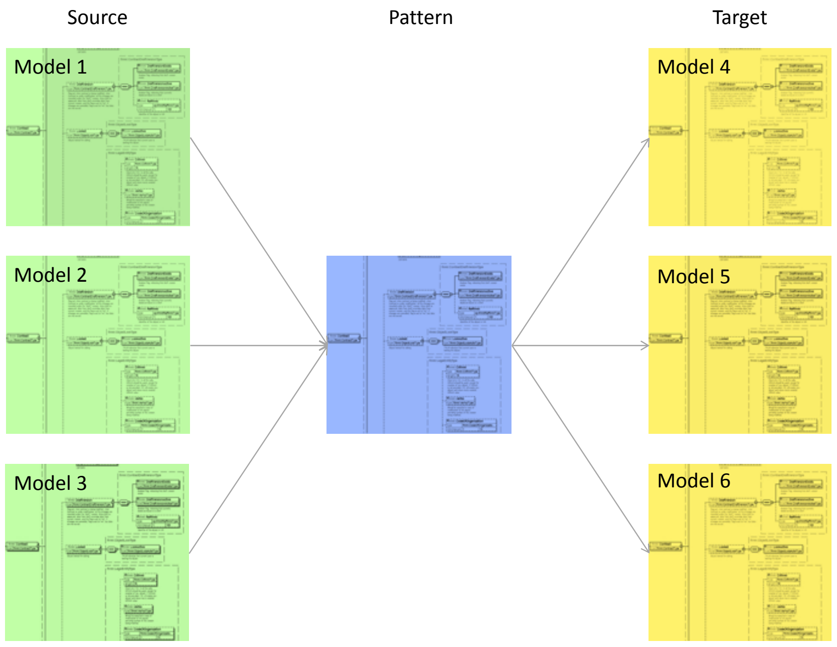

Fig. 2. Building a pattern from linked data models

In this example, the type of the data model can be recognized by detecting the dependencies between data elements in the model and by detecting the element names. The following list of dependencies: Contract_Acc_Currency can be matched against patterns and can by qualified as a financial data model.

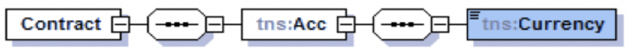

Fig. 3. Example of data dependencies

Knowing that, we can significantly limit the number of patterns that will be used for further analysis of the model. Following on that, we can use a limited dictionary to match the names of the attributes with the patterns. Once this simple model is analyzed and confirmed by the SME, it will be added to the knowledge base of patterns. In the future, similar patterns will be recognized automatically.

Initial transformation is a multi-step process, designed to extract important business information from the source model (S) and make sure that none of the business information is lost. At the same time, non-business information (e.g. headers, routing information, signatures, etc.) might be removed from the source model before its normalization.

The following graphic illustrates the process described below:

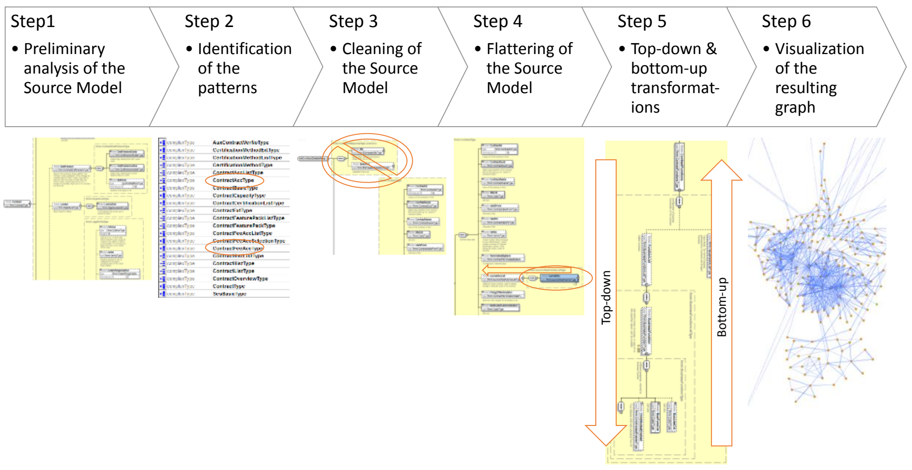
Step1
• Preliminary analysis of the Source Model
Step 2
• Identification of the patterns
Step 3
• Cleaning of the Source Model
Step 4
• Flattering of the Source Model
Step 5
• Top-down & bottom-up transformat-ions
Step 6
• Visualization of the resulting graph
Top-down
Bottom-up

Fig. 4. Transformation of the Data Model into Graph representation

Step 1. The first step leads to the identification of the major artifacts; this step can be performed by simple counting of the attributes used in the model. E.g. a large number of occurrences of the complex data type called 'account' suggests that this might be one of the key attributes of the model. Following on this process, we can build the list of attributes that are most popular in the source model.

Step 2. During the second step of universalization, we are looking for given patterns and pre-defined elements (see the example earlier in this paragraph). The goal of this process is to associate initial transformation with the right set of dictionaries and patterns. In other words, we need to understand the overall business context of the data model in order to apply the right set of the transformations.

Step 3. The third step of data model universalization is focusing on cleaning-out unnecessary elements from the model, especially technical artifacts. The importance of this step would differ significantly between various models. Technical (non-functional, non-business) elements of the data model could be either entirely removed, or replaced with standard place-holders, in order to simplify further processing while not losing any important artifacts. This stage of the process might require manual input from the expert.

Step 4. The next step is about 'flattening' of the model. Highly normalized data models tend to be extremely complex and hard to visualize. Simplification focuses on the removal of unnecessary groupings, while maintaining the original business meaning of the model.

Step 5. Now comes an important question: where to start the 'universalization' process? Two possible choices are: top-down or bottom-up. The first option focuses on the overall meaning of the model, where each node is further enriched by the definition of its child elements. The bottom-up approach focuses on the details that could be further composed into the meaningful entities. Although the top-down option is preferred, certain element of the bottom-up approach is necessary in order to preserve the right amount of detail. Graph notation is a very good way to explain the dependencies within the model, and this is why the top-down approach seems to be a better choice.

Step 6. The final stage of universalization allows us to generate a consistent graph-view of the source model. Using a graph-view, we can easily understand the overall complexity of the model, its bottlenecks, cross-dependencies and the overall structure.

7.1. Preparation for the transformation of the data model

The following example presents the transformation of the physical data model (that could be represented by Data Definition Language) to its graph equivalent. Transformation between the flat structure and graph have been performed using metadata generated from the Reference Data Model.

The following changes have been applied as a part of initial transformation of the model:

1. Element names have been changed, according to the data stored in the dictionary.
2. Basic data structures have been created from the flat structure, based on the metadata stored in the dictionary.

Customer	
First_name	Str(20)
Middle_Initial	Str(1)
Last_name	Str(30)
Cust_ID	Str(14)
Street	Str(35)
House	Str(6)
State	Str(2)
City	Str(30)
Cust_type	Int

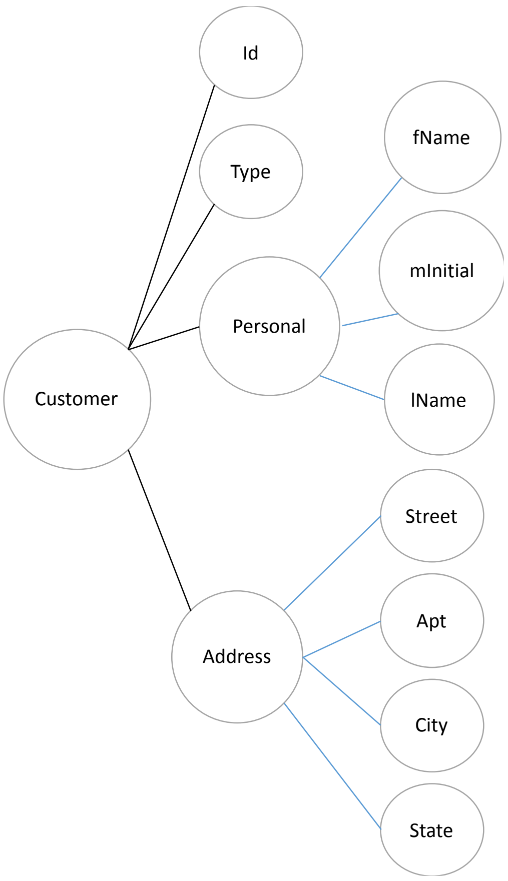

Fig. 5. Visualization of the dependencies in the data model

The resulting graph doesn't represent Reference Data Model yet, but is using the naming convention and basic structures, compatible with the template. Further work would be required to perform a full transformation of the source model to the Reference Data Model.

7.2. Data Model Transformation Processes

Transformation means that two Data Models, that are compatible on a business level, can be equipped with a set of the rules that enable transformation between those models (Kotulski, 2013). Those two models are further referred to as a Source Model (S) and a Target Model (T). Transformation between these models can be expressed as:

$$S \rightarrow T$$

The assumption is that both models S and T, are already "normalized" to the graph-representation.

In order to prove that the models are compatible (i.e., can be transformed), we need to analyze each graph, looking for the productions (results of graph's transformations) that have to be used to generate this graph. This set of productions, for the source model, can be defined as:

SL → SR

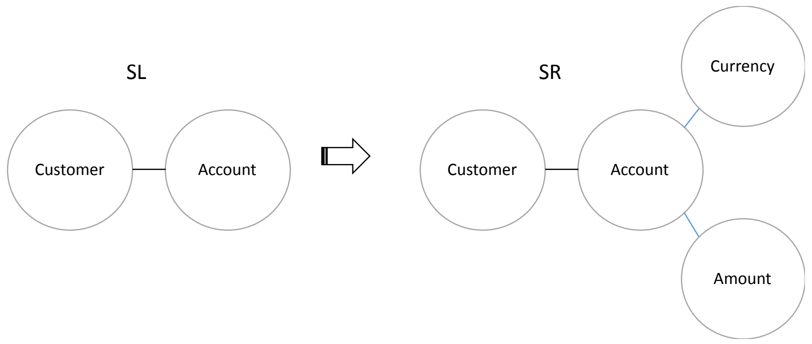

Fig. 6. Graph production on the source model

Similar analysis would be performed on the target model. The set of productions for the target model can be defined as:

TL → TR

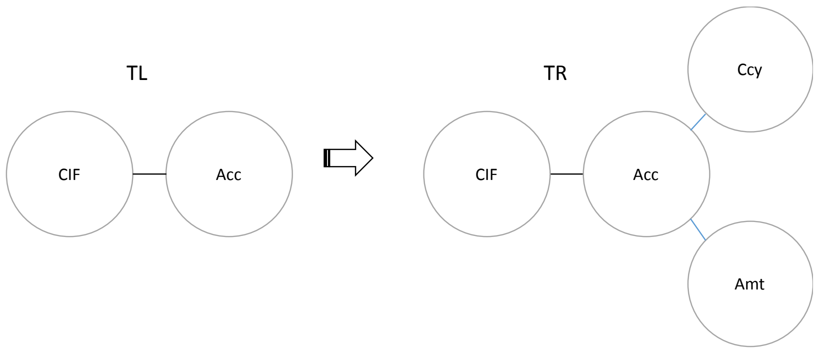

Fig. 7. Graph production on the target model

In the above example, we can consider productions SL→SR and TL→TR as compatible, provided that we can map the names of the attributes between the models. Such a mapping needs to be created by an expert during the first transformation of given model and recorded as meta-data that can be used for future automated transformations.

The simple example above presents transformation from S → T. At this point, we can assume that there is another model (e.g. X, Y, Z), compatible with S and T, but having different structures and using different names of the attributes. These models can be based e.g. on country regulations and local languages. In order to make the model universal, we would need to look for compatible productions, using a pre-defined set of meta data. Such a mapping would require each pair of the models to be compared and individually transformed, i.e.:

$$S \rightarrow X,\ S \rightarrow Y,\ S \rightarrow Z,\ X \rightarrow T,\ Y \rightarrow T,\ Z \rightarrow T.$$

In order to simplify this process, we can introduce an intermediary model that can be used as a source and target for each model being analyzed for suitability for transformation. The intermediary model would be called a Pattern Model (P). Provided that there are a large number of models, this approach would significantly limit number of potential model → model transformations. Resulting in a list of transformations that would look like the following:

$$S \leftrightarrow P,\ X \leftrightarrow P,\ Y \leftrightarrow P,\ Z \leftrightarrow P.$$

The following example demonstrates two productions, resulting in different target models:

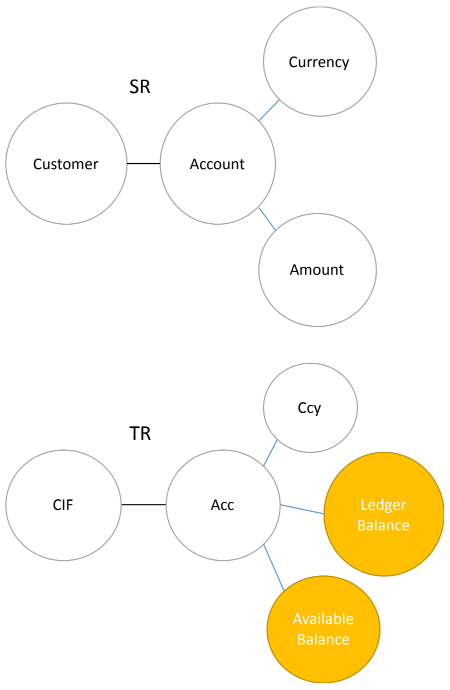

Fig. 8. Example of productions, resulting in different target models

8. PRACTICAL APPLICATIONS OF THE DATA MODEL TRANSFORMATIONS

Automation of the Data Model Transformation Processes becomes a critical success factor in multiple commercial solutions. Listed below are practical examples where a “higher than current” level of automation would significantly improve both timing and cost efficiency.

8.1. Integration Projects

The EU’s Payment Services Directive (PSD) was originally published in December 2007. The recently proposed PSD2 introduces Trusted Third Party Account Access, which are represented by two acronyms: Third Party Payment (TPP) under the Access to Accounts (XS2A) rule. Besides legal and operational discussions, XS2A has significant impact on how Banks will enable TPP providers to access their resources (Skinner, 2015).

Proprietary data formats would require each payment provider to commit to the API exposed by particular banks. On the other hand, banks my want particular payment providers to easily access their resources; in this case, banks may need to adopt the API exposed by the TPP.

As discussed in the previous chapters, particular data formats might significantly vary, while the amount of information is very similar between particular parties.

8.2. Advanced Analytics

The advent of Big Data, supported by virtually unlimited storage capacities available at the low cost, results in very large amounts of data stored by particular organizations. Unlike typical database solutions, Big Data doesn’t require data formats to be precisely defined and followed. It is important to collect a lot of data and use proper analytical tools to make meaningful use of that data.

8.3. Automation of Test Data Preparation

Automated testing is one of the key components of currently emerging DevOps. The term “DevOps” is a compound of “development” and “operations”, and is a movement or practice that emphasizes the collaboration and communication of both software developers and other information-technology (IT) professionals, while automating the process of software delivery and infrastructure changes (Cortet, 2014). An important part of DevOps is the automation of testing processes. While in traditional testing approaches, test data are prepared together with test scripts; an agile approach, combined with DevOps, requires quick access to the test data for new developments, where actual data doesn’t yet exist. On the other hand, test data can be created based on some other data, that are similar from a content perspective but are completely different from format perspective. Once again, the automated transformation of such a data would significantly improve performance of the projects.

9. CONCLUSIONS

Transformation of a data model to its formal graph representation is necessary, in order to allow further automation of the transformation process. Provided that each of the source data models go through this transformation, the resulting models will share common set of characteristics, such as attribute names and basic data structures. Resulting high level of automation of the transformation process would provide significant savings, both in terms of time and money, compared to similar work performed by a human expert.

REFERENCES

ADRM Software. (2018, August 1). *Business Area Data Models*. Retrieved from http://www.adrm.com/data-model-business-area.html

Angles, R., & Gutierrez, C. (2008). *Survey of graph database models*. New York, USA: ACM.

Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., & Yergeau, F. (2008). *Extensible Markup Language (XML) 1.0* (Fifth Edition). Retrieved from https://www.w3.org/TR/xml

Cortet, M. (2014). *Access to the Account (XS2A): accelerating the API-economy for banks?* Retrieved from https://innopay.com/blog/access-to-the-account-xs2a-accelerating-the-api-economy-for-banks

Holman, K. (2018). *OASIS Universal Business Language (UBL) TC*. Retrieved from https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ubl

ISO 20022. (2018, August 1). *Universal financial industry message scheme*. Retrieved from https://www.iso20022.org

Kotulski, L. (2013). *Rozproszone transformacje grafowe*. Kraków, Poland: Wydawnictwo AGH.

McKnight, W. (2014). *IBM Industry Data Models in the Enterprise*. Retrieved from https://www-01.ibm.com/software/data/industry-models/

Roman, D. (2006). *Canonical Data & Process Models for B2B Integration*. Retrieved from http://ceur-ws.org/Vol-170/paper3.pdf

Skinner, Ch. (2015). *How will Banks organise themselves to manage APIs built for PSD2/XS2A?* Retrieved from http://thefinanser.com/2015/11/how-will-banks-organise-themselves-to-manage-apis-built-for-psd2-xs2a.html/

Sleger, G. (2010). *Data Transformation Mapping – Can it be Automated?* Retrieved from https://www.cleo.com/blog/data-transformation-mapping-can-it-be-automated

SWIFT. (2018, August 1). *Financial messaging services*. Retrieved from https://www.swift.com/about-us/discover-swift/messaging-standards

Thompson, H. & Lilley, C. (2014). *XML Media Types, RFC 7303*. Retrieved from https://tools.ietf.org/html/rfc7303