

# Proposed architecture for ETL workflow generator\*

Matija Novak, Dragutin Kermek, Ivan Magdalenić

University of Zagreb

Faculty of Organization and Informatics

Pavlinska 2, 42000 Varaždin

{matija.novak, dragutin.kermek, ivan.magdalenic}@foi.hr

**Abstract.** *The most important process, on which most time is spent, when building data warehouses is the Extract Transform Load (ETL) process. Automation of such process is an expected thing to do. In this work the focus is on automation for generating ETL workflows which is currently not very well researched. The idea is to build systems which model the ETL process based on semantics and suggest the needed transformation and mappings for automatic generation of ETL workflows. In this work an architecture is presented to automatically integrate the system which generates mappings and transformations based on ontologies and the traditional ETL tools. The architecture of the prototyped system is message based which enables parallel processing.*

**Keywords.** ETL, data warehouse, architecture, generator, ontologies, message based

## 1 Introduction

Extract Transform Load (ETL) process [1] is the most important process of data warehouse. Around 70% of resources and time when building data warehouse go to ETL process. ETL is used to extract data from different sources, transform the data which includes cleaning the data and align the data between different sources in a common way and load the data into data warehouse. [2] A lot of research is done in the field of ETL, some of the main research topics are: how to do ETL efficiently [3, 4], how to optimize it [5, 6], how to model ETL at conceptual and logical level [7, 8], how to automatize the whole process [9, 10], how to test it [11, 12] and so on.

In this work the focus is on the ETL automation. There are various aspects of ETL automation like: automation to recognize needed mapping and transformation, automation for ETL testing, automation for generating ETL workflows, etc. Our focus is on automation for generating ETL workflows and the idea is to use systems which model ETL process based on semantics and suggest the needed transformations and mappings for automatic generation of ETL workflows. Publications that described such systems (e.g. [13, 14, 15]) are focused on suggesting information about the needed

transformations and mappings, but the usage of this information is not researched enough. During literature review we did not find the use of this information for automatic generation of ETL workflows.

In this work we plan to use the information about the needed transformations and mappings, got from previous systems, and use it in traditional ETL tools automatically. To be able to do that in this work an architecture of an ETL workflow generator is proposed and a prototype was built using Apache Camel [16]. ETL workflow generator is a tool that uses the information about transformations and mappings and generates ETL workflows for existing traditional ETL tools.

While in this work we will focus on ETL in the domain of data warehouses in the future we could expand this idea to be used in operational applications. As Poess et al. say [12]: “Recently, ETL was replaced by the more comprehensive acronym, data integration (DI).” They define DI as follows: “DI describes the process of extracting and combining data from a variety of data source formats, transforming that data into a unified data model representation and loading it into data store.” [12] While ETL has been used for building data warehouses, DI can be used for building data warehouse and for synchronizing data between operational applications. Also, recently some ideas have arisen like content driven ETL process [10]. This is because of the large amount of different data on the web that is not all structured and needs to be somehow integrated, but this is also out of the scope of this work.

The work is structured as follows. Section 2 gives an overview of related work in the field of ETL process. Section 3 describes the ETL workflow generator, shows the proposed architecture of ETL workflow generator with description of needed patterns for implementing such system. Section 4 gives the conclusion and future work.

## 2 Related work on ETL process

Various researches deal with optimization, modelling and automation of ETL process. Vassiliadis made in 2009 a systematic review [17] of work in the field of ETL technology where he describes what are the conceptual and logical modelling problems, what are the

\*This paper is published and available in Croatian language at: <http://ceciis.foi.hr>

problems within each stage of ETL and he gave review of some prototype tools. From his work one can see that the ETL process can be viewed on three levels:

- Conceptual - highest level of abstraction, at this level all data sources are treated equally and not distanced by the source type, because it is unknown whether this is a file or a database or something else. Transformations are usually data union, intersection, difference, filter, join, etc. Also for data warehouse it is unknown what kind it will be, star schema or snowflake. Articles like [18] try to optimize and make it easy the conceptual modelling of the ETL process.
- Logical - at this level we know exactly what kind of source is it, what type of data warehouse (star schema or snowflake) but it is still unknown whether a relational database is used as data warehouse (DW) or something else.
- Physical - physical implementation of the process.

Most researches today are focused on conceptual and logical level while the physical level was researched before (e.g., Kimball[19]).

For modelling at conceptual and logical level there are two main approaches:

- Approach Based on unified modelling language (UML) [20, 21, 22]
- Semantic approach [14, 15, 23]

Each of these approaches has their benefits and problems. UML has the benefit that it is based on standard modelling language that is well-accepted. The problem with UML is that one must follow the restrictions that came with this language. Other approaches have the benefit that they can represent ETL process without any restrictions that come with previously defined generic language, but they might not be well accepted by others. [15]

Naiqiao et al. [24] propose “*a semantic-aware data generator for ETL workflows, where ETL workflow activity semantics are transformed to symbols and a set of constraints over them, and concrete data sets are derived by solving constraints.*” In this work we will focus on semantic approach using ontologies because it gives the possibility to reason over data and adding knowledge to the whole system. Usually web ontology language (OWL) is used for defining the ontologies. As Abelló et al. say [25]: “*... the use of an OWL ontology, instead of a global schema, provides a formal model on which automated reasoning mechanisms may be applied.*”

Researches that focuses on using ontology based ETL process are [13, 14, 23, 26]. What is not covered in these articles is the implementation of the ETL process. Implementation in the articles only gives the info what should be done but it is not explained how

this is used in the ETL process itself. In article [14] the idea is to perform SPARQL queries directly on the data that are normally done in data warehouses. The problem with this method is that it is slower than traditional ETL in the meaning of execution of the process.

This work focuses on automatize the mappings between source and destination and defining transformations that should be applied on the data using ontologies. So the idea is to use the speed of traditional ETL and optimizations that were made there and take the advantage of the knowledge and information got by semantic modelling. ETL workflow generator is suggested to transform this semantic knowledge into traditional ETL tools rather than to build a new ETL tool or framework as it is done in previous research like in [27, 28].

### 3 ETL workflow generator

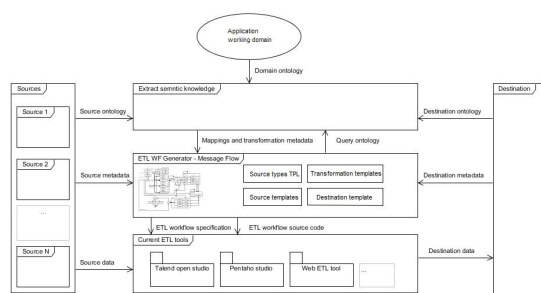
As already said there have been articles from Skoutas like [13, 23] that have described how to model the ETL process from ontologies. In short the process is to create ontology for every source, create domain ontology and ontology of the destination and then querying the created ontologies. One can get the mappings between destination and source, and what transformations need to be performed. On the other hand there are existing ETL tools that are fast and useful (like Talend Open Studio [29] or Pentaho Data Integration [30]) but ETL workflow still must be created manually. Our idea is to create ETL workflow generator which will convert the knowledge got from systems about needed transformations and mappings and generate specification needed for traditional tools.

In the next subsections are presented: high level architecture, detailed message flow of ETL workflow generator and description of used design patterns.

#### 3.1 Proposed architecture

In Figure 1 high level architecture is presented and it can be seen that ETL workflow generator communicates with following parts:

- Sources — present the data sources like databases, files, web, etc.
- Destination — presents the data warehouse to which data should be loaded, here can be different models like star schema or snowflake;
- Extract semantic knowledge — represents the existing systems which can tell what mappings and transformations need to be performed based on source, destination and domain ontology;
- Current ETL tools — represents existing ETL tools like Talend Open Studio, Pentaho, Web ETL tool, etc. Each of this systems has its own specification based on which it performs the ETL process;



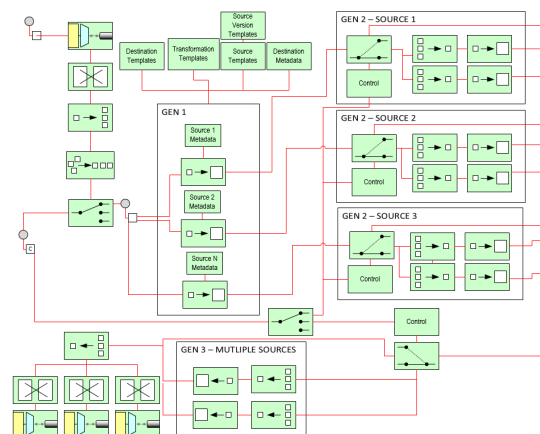
**Figure 1:** High level architecture of ETL workflow generator

- ETL workflow generator — generates specification for existing ETL workflow system based on mappings and transformation that it gets. In the next subsection detailed description of ETL workflow generator is given. As it can be seen in the Figure 1, ETL workflow generator has four template objects which contain information what is needed for certain source types (databases, files, ...), sources (MySQL, PostgreSQL, XML, TXT, etc.), transformations (split, merge, duplicate, date format, ...) and destination (star schema, snowflake).

The internal architecture of ETL workflow generator (further in text just referred as ETL generator) is based on messages and its internal structure is presented as message flows (Figure 2). Flow goes from the moment the generator gets the first message, that is the mappings and transformation (M&T) specification to the final specification for some existing ETL tool. The figure uses the symbols for message flows which represent standard message patterns.

Used messaging patterns are one type of enterprise integration patterns and are used to successfully manipulate with messages. These patterns and their symbols were taken from the book Enterprise Integration Patterns from Hohpe and Woolf [31] and are shown in Table 1. To build the diagram MS Visio was used with using templates available at Apache Camel web site [16].

The benefit of using messaging is that the initial message of M&T can be split into smaller messages. These messages represent one unique transformation that needs to be performed on one, two or more attributes and sources. Now each of this single messages can be processed separately and in parallel. With system like this instead of just generating the specification for existing ETL systems one can use it to perform the ETL process. In this work the focus is on generating the specification. So while each message is processed, the necessary description is added and sent further to other processing parts.



**Figure 2:** ETL workflow generator - message flow

### 3.2 Description of message flow

To better understand what is going on the whole message flow will be described. First, initial message with M&T is created based on input data. Input data is got from existing system that recognizes the needed mappings and transformations. This initial message is transformed (by message transformer) into XML format which will be used for message inside of ETL generator. XML is chosen because it can be easily manipulated (like adding a new child or adding properties as attributes). At the same moment as the initial message is sent, one command message is sent to first router to configure the router.

System consists of three main modules: GEN1, GEN2 and GEN3. GEN1 module is responsible for adding information to messages which contain information about M&T that require only one source and transformations that are made on one attribute (like date convert). One Content Enricher exists for every source which adds data to messages that it gets based on templates.

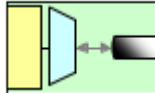
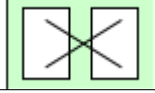
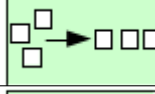
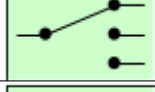
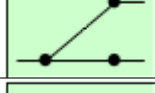
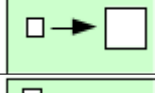

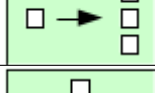
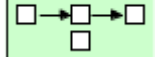
GEN2 module is responsible for messages that require one source and transformations are made on two or more attributes (like merge transformation). One GEN2 module consist of multiple Content Enrichers for every source and it adds data to incoming messages based on templates. Each of Content Enrichers knows what it has to fully describe the source that it is responsible for. GEN3 module works similar to one GEN2 module, but differs in that it is responsible for messages that require multiple sources.

When the initial message is in the correct format it is split into multiple messages where each represents one transformation with info on which source attribute is performed and to what destination should be loaded.

This message could look as flows:

```
<message id="1">
  <source>
    <user_source>Source_1</user_source>
    <name>employees</name>
    <attribute>date of employment
  </attribute>
```

**Table 1:** Used enterprise integration patterns

Pattern name	Symbol [31, 16]	Short description
Chanel Adapter		Enables that one system connects to a message channel.
Message translator		Translates one data format into another, it has the same purpose as Adapter from GOF [32].
Resequencer		Transforms the stream of incoming messages into correct order.
Content Based Router		Routes messages based on their content.
Detour		Some messages go directly to destination while some go over context based router to process.
Content Enricher		Extends the message with missing information.
Aggregator		Aggregate separate related messages into one single message.
Splitter		Splits message into separate messages.
Composed message processor		Is a combination of splitter and router it splits the incoming message flow into separate messages which need to be processed differently and at one point combines them back together.

```

</source>
<destination>
  <name> employees </name>
  <attribute>date</attribute>
</destination>
<transformation>
  <type>convert</type>
  <format>European date</format>
</transformation>
</message>

```

The message has three sub nodes: a) source — that has basic information about the source (source name, table name, and attribute name) from which data should be extracted; b) destination — that has basic information about the destination (table and attribute name) to which data should be loaded. As ETL generator can only be used at the same time to work with workflows for one data warehouse there is no special information about the destination name, because all messages have the same destination name; c) transformation — basic information about transformation (like type and format) that needs to be performed.

Two types of message exits: normal and command message. Command messages are used to configure the system. Normal messages are messages containing

information about mappings and transformation. The messages are resequenced so that command messages are send first. The messages are next routed based on their type. Normal messages go to the first router and are then sent to one of the Content Enrichers inside GEN1 module to which the message belongs. Each message is processed and enriched.

For example, our previous message can be enriched as follows (new parts are written in bold):

```

<message id="1">
  <source>
    <name>employees </name>
    <attribute>date of employment
    </attribute>
    <user_source>Source_1
    <user_source>
      <type>MySQL</type>
      <ip>192.168.5.1</ip>
      <username>pero</username>
      <password>123456</password>
    </source>
    <destination>
      <name> employees </name>
      <attribute>date</attribute>
      <type>DIM</type>
      <ip>192.168.5.2</ip>
      <username>dw</username>
      <password>654321</password>

```

```

</destination >
<transformation >
  <type>convert </type>
  <format>dd.mm.yyyy.</format >
</transformation >
</message >
    
```

Content enricher has added in source and destination part of enriched message following elements: type, ip, username and password. They are information that must be provided for final workflows to be able to connect to the source and destination. Also, transformation part has changed. The format of the transformation is changed from European Date to specific type of the date to which the date attribute should be converted.

Every message might not contain transformation, therefore it could just contain source and destination. Templates of transformation contain for every transformation info what is necessary to do. For example, message would require split transformation on “Name Surname” into “Name” and “Surname”. Then two new messages would be generated. Command messages that involve two attributes from same source or multiple sources are send to modules GEN2 and GEN3.

In GEN2 command messages tell that something must be done on two different attributes of the same source. For example merge of name and surname. In this case the GEN2 instantiates Aggregator and Content Enricher and adds new rule to dynamic router that new channel exists. Aggregator has the info how many messages should it get (in the name and surname case only two messages are expected) and when these messages are received they are copied into one new message and then this message is sent to Content Enricher. The Content Enricher in this case adds info like Content Enricher in GEN1. The original messages that the aggregator got are further sent without entering Content Enricher. If the transformation is name and surname merging of name and surname then these two messages are wrapped around with info needed for merge transformation. The same is with GEN3 command messages. Only difference is that GEN3 module is used, as already described, for messages that work with multiple sources. Normal incoming messages that are not needed in GEN2 or GEN3 skip them and are send directly to the next dynamic router.

One interesting situation is, for example, when Name and Surname would not be stored into destination. In this case only “Name Surname” is stored. The messages Name and Surname would never exist and in GEN2 nothing would happen. So to eliminate this possibility for every such case at the beginning are generated messages for name and surname which don’t have destination info. When such messages get to GEN2 or GEN3 they are copied into new merged message and then destroyed.

After GEN3 there is an aggregator which merges all messages into one specification. Based on the command message that it gets at the beginning it knows how many messages it should receive. When all mes-

sages are merged final specification is send to router which then routes the specification message to one adapter which will convert that message to specification expected by existing ETL tools.

The benefits of this system are that messages are processed independent of one another which enables parallel execution. Also, the system can be distributed on several servers for example each module from GEN1 to GEN3 could be on separate server and elements inside one GEN module could also be on separated servers. If the system would be distributed over several servers then there would be a bigger benefit to perform the ETL process itself then just generate the specification.

### 3.3 Implementation

Java programming language was used for a system implementation. Figure 3 presents used patterns and important classes that are important. Template based meta programming was used for the generation of features in GEN2 and GEN3 modules. The used meta model is described in [33]. Templates were created for ETLGen2AgregatorAndCE, ETLGEN2Router, ETLGEN3Router, ETLSourceRouter, and so on. Every template contains keywords that can be configured during execution like “//ADD RULE” or “//ADD FROM”.

In the next example the source-code of ETLSourceRouter template is presented.

```

package templates;
import org.apache.camel.builder.RouteBuilder;
public class ETLSourceRouter
    extends RouteBuilder{
    @Override
    public void configure() throws Exception {
        from("file:target/gen1_router?noop=false").
            choice().
            when(xpath("/message/command = 'yes2'")).
                to("file:target/cmd_router_gen2").
            otherwise().
            when(xpath("/message/command = 'yes3'")).
                to("file:target/cmd_router_gen3");
        //ADD RULE
    }
}
    
```

The keyword //ADD RULE? can be seen at the end of the router. We will explain how the system is built based on this simple example. When a command message is received that has a new source, the code for

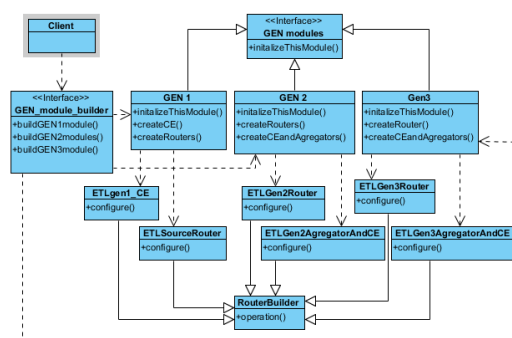


Figure 3: Class diagram - generation of GEN modules

this source is added to the ETLSourceRouter template. More precisely the template is copied and the rule is added to the copy of the template which is used for execution.

When the new copy of the ETLSourceRouter is ready the class is compiled and imported into the program. At this point the new class is ready to be used. This process is shown in the following code example.

```
static String compileTemplateClass(File f)
    throws CompilationException {

JavaCompiler jc =
    ToolProvider.getSystemJavaCompiler();
StandardJavaFileManager fileManager =
    jc.getStandardFileManager(null, null, null);
DiagnosticCollector<JavaFileObject> diag =
    new DiagnosticCollector<JavaFileObject>();

List<File> fileList = Arrays.asList(f);
Iterable<? extends JavaFileObject>
    compilationUnits1 = fileManager.
        getJavaFileObjectsFromFiles(fileList);
boolean ok = jc.getTask(null, fileManager,
    diag, null, null, compilationUnits1).call();

try { fileManager.close(); }
catch (IOException ex) { ... }

printDiagnostics(diag, !ok);

//MOVE THE CLASS TO BUILD FOLDER
String f = f.getName().
    substring(0, f.getName().lastIndexOf("."));
String file_class_name = f_name + ".class";
String build_dir =
    System.getProperty("user.dir")
    + File.separator + "build"
    + File.separator + "classes";
String file_parent_dir =
    f.getParent().substring(f.getParent().
        indexOf("src") + 4);
build_dir += File.separator + file_parent_dir;

moveCompiledClass(f.getParent(),
    file_class_name, build_dir);

String class_name = file_parent_dir
    + File.separator + f_name;

return class_name.replace(File.separator, ".");
}
```

One problem with this implementation is that a compiled code is created in the same directory as the source-code file so it needs to be moved to the build folder. To import the newly compiled class Java reflection is used which is shown in the next example.

```
Class c = Class.forName(fullClassName);
etlsourcerouter =
    (RoutesBuilder) c.newInstance();
...
//add etlsourcerouter to Apache Camel context
context.addRoutes(etlsourcerouter);
```

Finally, since the system is built as a messaging system every template is basically a class that extends RouteBuilder class from Apache Camel, so the last thing that needs to be done is to add this class to

the running context as shown in the previous example. Once this is done the RouterBuilder can start doing his job and that is read incoming messages, process it and generate output messages. The source-code is available at [https://github.com/matnovak-foi/ETL\\_WorkflowGenerator](https://github.com/matnovak-foi/ETL_WorkflowGenerator) under the GPLv3 license.

## 4 Conclusion and future work

In this work it was presented the architecture to automatically integrate the system which generates mappings and transformations based on ontologies using traditional ETL tools. The architecture of presented system is based on messages and it enables processing multiple different messages at the same time.

The system built in this way is flexible and can be implemented as distributed system. Also it is very fast because many messages can be processed in parallel. Presented system generates specification that can be used in existing ETL tools which is the main difference to existing semantic ETL tools which are implemented from scratch. Also, the system can be easily modified to generate source code or to perform the ETL process right away. Other benefit of this architecture is possibility to change destination type. This is possible because a destination is described by templates. Different destination could be described by other template and load data into any kind of destination. In future we plan to use this ETL system for other purposes than DW.

Prototype implementation of the presented ETL generator was made in Camel because it supports all the necessary messaging patterns that are needed to implement the intended ETL generator.

The current work is focused on usage of generative programming techniques to make the system more flexible. Right now for every new case the system needs to be manually configured. For every new source type manual creation of new Content Enricher in GEN1 module is needed. The future plan is to automate this process based on message content. Other possible extension to this ETL generator would be to include subscriber pattern so that the tool could change existing specification based on changes that occur in the sources.

## References

- [1] R. Kimball and J. Caserta, *The Data Warehouse-ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data*. Wiley, 2004.
- [2] M. Novak and K. Rabuzin, "Prototype of a Web ETL Tool," *International Journal of Advanced Computer Science and Applications*, vol. 5, no. 6, pp. 97–103, 2014.

- [3] K. Sun and Y. Lan, "SETL: A scalable and high performance ETL system," in *System Science, Engineering Design and Manufacturing Informatization (ICSEM), 2012 3rd International Conference on*, vol. 1, (Software Engineering Institute, Beihang University, Beijing, China), pp. 6–9, 2012.
- [4] K. Rabuzin and M. Novak, "WebETL Tool - A Prototype in Action," in *ICCGI 2014, The Ninth International Multi-...*, (Sevilja, Spain), pp. 67–71, 2014.
- [5] A. Simitsis, P. Vassiliadis, and T. Sellis, "Optimizing ETL processes in data warehouses," in *Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference on*, pp. 564–575, 2005.
- [6] A. Behrend and T. Jörg, "Optimized incremental ETL jobs for maintaining data warehouses," in *14th International Database Engineering and Applications Symposium, IDEAS '10*, (University of Bonn, Germany), pp. 216–224, 2010.
- [7] A. Simitsis, D. Skoutas, and M. Castellanos, "Representation of conceptual ETL designs in natural language using Semantic Web technology," *Data & Knowledge Engineering*, vol. 69, no. 1, pp. 96–115, 2010.
- [8] D. Skoutas, A. Simitsis, and T. Sellis, *Ontology-Driven Conceptual Design of ETL Processes Using Graph Transformations*, vol. 5530 of *Lecture Notes in Computer Science*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009.
- [9] X. Zhang, W. Sun, W. Wang, Y. Feng, and B. Shi, "Research on generating incremental ETL processes automatically," *Jisuanji Yanjiu yu Fazhan/Computer Research and Development*, vol. 43, no. 6, pp. 1097–1103, 2006.
- [10] A. Berro, I. Megdiche, and O. Teste, "A Content-Driven ETL Processes for Open Data," *18th East European Conference on Advances in Databases and Information Systems and Associated Satellite Events, ADBIS 2014*, vol. 312, pp. 29–40, 2015.
- [11] N. Du, X. Ye, and J. Wang, "A schema aware ETL workflow generator," *Information Systems Frontiers*, vol. 16, no. 3, pp. 453–471, 2012.
- [12] M. Poess, T. Rabl, H.-A. Jacobsen, and B. Caulfield, "TPCDI: The first industry benchmark for data integration," *Proceedings of the VLDB Endowment*, vol. 7, no. 13, pp. 1367–1378, 2014.
- [13] D. Skoutas and A. Simitsis, "Ontology-Based Conceptual Design of ETL Processes for Both Structured and Semi-Structured Data," *International Journal on Semantic Web and Information Systems*, vol. 3, no. 4, pp. 1–24, 2007.
- [14] S. K. Bansal, "Towards a Semantic Extract-Transform-Load (ETL) Framework for Big Data Integration," in *2014 IEEE International Congress on Big Data*, pp. 522–529, IEEE, 2014.
- [15] S. Bergamaschi, F. Guerra, M. Orsini, C. Sartori, and M. Vincini, "A semantic approach to ETL technologies," *Data & Knowledge Engineering*, vol. 70, no. 8, pp. 717–731, 2011.
- [16] The Apache Software Foundation, "Apache Chamel: Enterprise Integration Patterns."
- [17] P. Vassiliadis, "A Survey of Extract-Transform-Load Technology," *International Journal of Data Warehousing and Mining*, vol. 5, no. 3, pp. 1–27, 2009.
- [18] S. Dupor and V. Jovanovic, "An approach to conceptual modelling of ETL processes," in *2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pp. 1485–1490, IEEE, 2014.
- [19] R. Kimball, *The Data Warehouse Lifecycle Toolkit*. The Data Warehouse Lifecycle Toolkit, John Wiley & Sons, 2008.
- [20] L. Muñoz, J.-N. Mazón, and J. Trujillo, "A family of experiments to validate measures for UML activity diagrams of ETL processes in data warehouses," *Information and Software Technology*, vol. 52, no. 11, pp. 1188–1203, 2010.
- [21] J. Trujillo and S. Lujan-Mora, "A UML based approach for modeling ETL processes in data warehouses," in *Conceptual modeling - ER 2003 Proceedings*, vol. 2813 of *Lecture notes in computer science*, (Berlin, Germany), pp. 307–320, Springer-Verlag Berlin, 2003.
- [22] S. Luján-Mora, P. Vassiliadis, and J. Trujillo, "Data mapping diagrams for data warehouse design with UML," in *Lecture Notes in Computer Science*, vol. 3288 of *Lecture notes in computer science*, (University of Alicante, Spain), pp. 191–204, Springer-Verlag Berlin, 2004.
- [23] D. Skoutas and A. Simitsis, "Designing ETL processes using semantic web technologies," in *Proceedings of the 9th ACM international workshop on Data warehousing and OLAP - DOLAP '06*, (New York, New York, USA), p. 67, ACM Press, 2006.
- [24] N. Du, X. Ye, and J. Wang, "A semantic-aware data generator for ETL workflows," *Concurrency and Computation: Practice and Experience*, vol. 28, no. 4, pp. 1016–1040, 2016.

- [25] A. Abello, O. Romero, T. Pedersen, R. Berlanga Llavori, V. Nebot, M. Aramburu, and A. Simitis, "Using Semantic Web Technologies for Exploratory OLAP: A Survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. PP, no. 99, pp. 1–1, 2014.
- [26] J. Chakraborty, A. Padki, and S. K. Bansal, "Semantic etl - state-of-the-art and open research challenges," in *2017 IEEE 11th International Conference on Semantic Computing (ICSC)*, pp. 413–418, 2017.
- [27] R. P. Deb Nath, K. Hose, and T. B. Pedersen, "Towards a programmable semantic extract-transform-load framework for semantic data warehouses," in *Proceedings of the ACM Eighteenth International Workshop on Data Warehousing and OLAP, DOLAP '15*, (New York, NY, USA), pp. 15–24, ACM, 2015.
- [28] J. P. McCusker, K. Chastain, S. Rashid, S. Norris, and D. L. McGuinness, "Setlr: the semantic extract, transform, and load-r," *PeerJ Preprints*, vol. 6, p. e26476v1, 2018.
- [29] Talend, "Talend Open Studio Integration Software Platform."
- [30] Pentaho Corporation, "Data Integration - Pentaho Business Analytics Platform."
- [31] G. Hohpe and B. Woolf, *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Addison-Wesley, 2012.
- [32] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Pearson Education, Limited, 2005.
- [33] M. Novak, I. Magdalenić, and D. Radošević, "Common Metamodel of Component Diagram and Feature Diagram in Generative Programming," *Journal of Computer Science*, vol. 12, no. 10, pp. 517–526, 2016.