XIS-Reverse: A Model-Driven Reverse Engineering Approach for Legacy Information Systems

André Filipe Arnedo Reis

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisor: Prof. Alberto Manuel Rodrigues da Silva

Examination Committee
Chairperson: Prof. Daniel Jorge Viegas Gonçalves
Supervisor: Prof. Alberto Manuel Rodrigues da Silva
Member of the Committee: Prof. Paulo Jorge Fernandes Carreira

June 2017
Abstract

With the explosion of technologies, companies have been struggling to manage and maintain their legacy information systems, thus upgrading those systems became a complex challenge. Moreover, requirements changes are difficult to be properly managed, leading to legacy information system requirements deterioration. To overcome or reduce such problems, software reverse engineering approaches can be used.

This research work proposes a model-driven reverse engineering approach that aims to support the mentioned challenge. This approach, named XIS-Reverse, has 3 main stages, namely (i) data schema extraction, (ii) reverse engineering configuration and (iii) reverse engineering execution. It uses information system database’s artefacts (stages (i) and (ii)), but also user preferences (stage (ii)) to better guide the reverse engineering process, and from these artefacts it is possible to extract high-level specifications from a legacy information system, through model-to-model transformations (stage (iii)) based on a set of well defined rules.

Moreover, the main contributions of this proposal (compared with the state of the art) are the semi-automatic identification of implicit generalizations and aggregations and the possibility to automatically extract attribute values to enrich the produced model.

This dissertation presents XIS-Reverse (approach and tool), provides an evaluation of its main contributions (based on two real-world information systems), presents an assessment of its interoperability with an existent framework and discusses its main challenges and benefits in the context of the reverse engineering.

Keywords: Model-Driven Engineering, Model-Driven Reverse Engineering, Model-Driven Reengineering, Database, Legacy Information System
Resumo

Com a explosão a nível tecnológico, as empresas têm dificuldade em gerir e manter os seus sistemas de informação legados, consequentemente, melhorar estes sistemas tornou-se um problema complexo. Além disso, é difícil gerir alterações de requisitos, levando à deterioração dos requisitos dos sistemas. De forma a superar ou reduzir tais problemas, podem ser usadas abordagens de engenharia reversa aplicadas a software.

Esta dissertação propõe uma abordagem de engenharia reversa orientada a modelos, que pretende sustentar o desafio mencionado. Esta abordagem, chamada XIS-Reverse, é constituída por 3 fases: (i) extração do esquema de dados; (ii) configuração do processo de engenharia reversa; e (iii) execução do processo de engenharia reversa. O XIS-Reverse utiliza artefatos da base de dados do sistema de informação (fases (i) e (ii)) e preferências especificadas pelo utilizador (fase (ii)) para guiar o processo de engenharia reversa. A partir desses artefatos extraem-se especificações de alto nível do sistema de informação, utilizando transformações modelo-para-modelo (fase (iii)) baseadas num conjunto de regras bem definidas.

Além disso, as principais contribuições da abordagem proposta, comparando com os trabalhos de investigação nesta área, são a identificação semiautomática de generalizações implícitas e agregações e a extração automática de valores de atributos, enriquecendo os modelos produzidos.

De uma forma geral, esta dissertação apresenta a abordagem e a ferramenta XIS-Reverse, fornecendo uma avaliação às contribuições principais (usando dois sistemas de informação reais), é avaliada a sua interoperabilidade com uma framework existente, são ainda discutidos e apresentados os seus principais desafios e benefícios no contexto da engenharia reversa.

Palavras-Chave: Engenharia Orientada por Modelos, Engenharia Reversa Orientada por Modelos, Reengenharia Orientada por Modelos, Base de Dados, Sistema de Informação Legado
# Contents

[**List of Tables**](#) xi

[**List of Figures**](#) xiii

[**Acronyms**](#) 1

## 1 Introduction

1.1 Research Context ................................................. 4

1.2 Problem and Research Questions .................................. 4

1.3 Thesis Statement .................................................. 5

1.4 Research Proposal .................................................. 5

1.5 Methodology ....................................................... 6

1.6 Publications ...................................................... 9

1.7 Outline ............................................................ 9

## 2 Background

2.1 Basis of Model-Driven Reverse Engineering ..................... 11

  2.1.1 Modeling Languages and Metamodeling .......................... 11

  2.1.2 Model-Driven Engineering ..................................... 12

  2.1.3 Model-Driven Reengineering ................................... 12

  2.1.4 Model-Driven Reverse Engineering ............................. 13

2.2 MDDLingo .......................................................... 13

2.3 RSLingo ............................................................. 15

2.4 Related Work ...................................................... 17

  2.4.1 Data Schema Extraction ....................................... 17

  2.4.2 Reverse Engineering ........................................... 18

## 3 XIS-Reverse Approach

3.1 Overview .......................................................... 21

3.2 Running Example ................................................... 21

3.3 Data Schema Extraction ........................................... 22

3.4 Reverse Engineering Configuration ............................... 23

3.5 Reverse Engineering Execution ................................... 24

  3.5.1 Initialization ................................................ 24

  3.5.2 Transformations to XIS models ............................... 25

  3.5.3 Transformations to RSLingo’s RSL models .................... 31

  3.5.4 Reverse Engineering of the Running Example ................. 36

3.6 Tools Support ..................................................... 38
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.6.1 Data Schema Extraction</td>
<td>38</td>
</tr>
<tr>
<td>3.6.2 Reverse Engineering Configuration and Execution</td>
<td>39</td>
</tr>
<tr>
<td>4 Evaluation</td>
<td>41</td>
</tr>
<tr>
<td>4.1 Real-World Case Studies</td>
<td>41</td>
</tr>
<tr>
<td>4.1.1 Case Study A: ProjectIT-Enterprise</td>
<td>42</td>
</tr>
<tr>
<td>4.1.2 Case Study B: Social Security application</td>
<td>48</td>
</tr>
<tr>
<td>4.2 Interoperability with XIS* frameworks</td>
<td>49</td>
</tr>
<tr>
<td>4.3 Related Work Discussion</td>
<td>51</td>
</tr>
<tr>
<td>4.3.1 Data Schema Extraction</td>
<td>51</td>
</tr>
<tr>
<td>4.3.2 Reverse Engineering</td>
<td>51</td>
</tr>
<tr>
<td>4.4 Summary</td>
<td>53</td>
</tr>
<tr>
<td>5 Conclusion</td>
<td>55</td>
</tr>
<tr>
<td>5.1 Main Contributions</td>
<td>56</td>
</tr>
<tr>
<td>5.2 Future work</td>
<td>57</td>
</tr>
<tr>
<td>Bibliography</td>
<td>59</td>
</tr>
</tbody>
</table>
List of Tables

4.1 Case Study A - Equivalence between application database and domain model . . . . . . 44
4.2 Case Study A - Evaluation of aggregation results . . . . . . . . . . . . . . . . . . . . . . 45
4.3 Case Study A - Overall results of the reverse engineering . . . . . . . . . . . . . . . . 45
4.4 Case Study A - Scenario B - Evaluation of implicit generalization results . . . . . . . . 47
4.5 Case Study B - Overall results of the reverse engineering . . . . . . . . . . . . . . . . 49
4.6 Classification of some related works on data schema extraction . . . . . . . . . . . . . . 51
4.7 Classification of some works on reverse engineering . . . . . . . . . . . . . . . . . . . . 52
List of Figures

1.1 XIS-Reverse approach and technologies. ........................................ 5
1.2 Action Research Methodology cycle (extracted from [1]). .............. 6

2.1 Relationships between metamodel, model and modeling language (adapted from [2]). ........................................ 12
2.2 Model-driven reengineering process (extracted from [3]). .......... 13
2.3 The multi-view organization of XIS* (extracted from [4]). ......... 14
2.4 Example of a XIS* Domain view. ............................................. 14
2.5 Example of a XIS* BusinessEntities view. .................................. 14
2.6 Example of a XIS* UseCases view. ........................................... 15

3.1 Overview of the XIS-Reverse approach. ...................................... 22
3.2 Running Example: SS-App data model. ...................................... 22
3.3 Running Example: SS-App database in Microsoft SQL Server Management Studio. ........................................ 23
3.4 Main configuration panel of the tool. ........................................ 23
3.5 Configuration panel for Simple Principal Entities (left) and Attribute Values Extraction (right). ......................... 25
3.6 XE-1 transformation example. ............................................... 26
3.7 XE-2 transformation example. ............................................... 26
3.8 XEA-1-a transformation example. ........................................... 26
3.9 XEA-1-b transformation example. ........................................... 26
3.10 XEA-2-a transformation example. ......................................... 27
3.11 XEA-2-b transformation example. ......................................... 27
3.12 XEA-2-c transformation example. ......................................... 28
3.13 XEA-3 transformation example. ........................................... 28
3.14 XEI transformation example. ............................................... 29
3.15 XBE-1 transformation example. ........................................... 30
3.16 XBE-1-a transformation example. ......................................... 30
3.17 XBE-1-b transformation example. ......................................... 31
3.18 XEU C transformation example. ........................................... 31
3.19 RDE-1 transformation example. ........................................... 32
3.20 RDE-2-b transformation example. ......................................... 32
3.21 RDEV transformation example. ........................................... 33
3.22 RDEV-a transformation example. ......................................... 33
3.23 RUC transformation example. ............................................. 34
3.24 RUS transformation example. ............................................. 34
3.25 RFR transformation example. ............................................. 35
3.26 RG transformation example. ............................................... 35
3.27 RBP transformation example. ............................................. 35
3.28 RT transformation example. ............................................... 36
3.29 SS-App XIS* Domain view ................................................................. 36
3.30 SS-App XIS* BusinessEntities view .................................................. 38
3.31 Technologies used in the XIS-Reverse approach ................................. 39

4.1 Support example to explain the Reis function ....................................... 42
4.2 Case Study A - Domain Model of the Project Dimension (adapted from [5]) . 43
4.3 Interoperability with XIS* frameworks .................................................. 49
Acronyms

DBMS  Database Management System. 22

DDL  Data Definition Language. 12  18

DSL  Domain Specific Language. 12  17

EA  Enterprise Architect. 5  7  38  39  49  51  55–57

M2M  Model-to-Model. 5  8  12  15  24  31  50  55

M2T  Model-to-Text. 12  15  50

MDE  model-driven engineering. 3  4  12  13  17–19  51  56

MDRE  model-driven reverse engineering. 3–6  9  13  17  18  21  55

T2M  Text-to-Model. 12  17  18
Chapter 1

Introduction

One of the main reasons why software projects tend to fail is the difficulty to manage its requirements, mainly due to the fact that requirements changes are difficult to be properly managed [6]. Without a proper way to manage requirements, software projects may have negative impact, namely excessive development and management costs, a system which does not meet stakeholders needs, and so on. However, new methods to collect, analyse, document and maintain requirements have been rising, their software requirements specifications are still mainly written in natural language [6]. Those kind of specifications are usually hard to keep up to date while the software applications are being developed. To overcome or reduce such problems, software reverse engineering approaches can be used.

Reverse engineering was initially used in hardware analysis, but it quickly extended its scope to software systems [7]. Then, following the huge expansion and advent of software from the end of the 80s, the reverse engineering topic has been mainly used in the context of legacy information systems, which are often still running crucial and critical operations for companies [8]. Reverse engineering can be defined as the process of examining an already implemented software system to create a higher abstraction level representation in a different form [7].

The main objective of such representations is to provide a better understanding of the software system's current state. These can be used to correct (e.g. fix bugs), update (e.g. alignment with updated user requirements), upgrade (e.g. add new capabilities), or even completely reengineer the system under study [8]. These operations are happening now more than ever due to new user requirements and expectations, adaptation to emerging business models, updated legislation, new technology innovation and preserving the system structure from deteriorating [9]. Since the reverse engineering of an information system is a time-consuming and error-prone process, any reverse engineering solution that could increase the automation level of the process would benefit the users of such complex task, and thus facilitate its larger adoption.

Model-driven engineering (MDE) approaches are increasingly gaining acceptance in the software engineering field to tackle software complexity and to improve software productivity [3] [2]. These approaches promote the systematic use of models, raising the level of abstraction at which software is specified and increasing the automation level of software development. Although most of the MDE approaches aim to build new information systems through forward engineering, MDE can also be used as a reverse engineering technique (model-driven reverse engineering (MDRE)) [8]. Moreover, meta-modelling and model transformations have proven to be useful in the automation of certain reverse engineering activities, such as representing source code at a higher-level of abstraction.
1.1 Research Context

This research work has been developed at the Information and Decision Support Systems of INESC-ID (Instituto de Engenharia de Sistemas e Computadores - Investigação e Desenvolvimento), under the supervision of Professor Alberto Rodriges da Silva, in the scope of the Master Degree in Information Systems and Computer Engineering at Instituto Superior Técnico, Universidade de Lisboa. This research work results from the common interest in the area of MDE and its application to reverse engineering legacy systems.

The Information and Decision Support Systems is responsible for a variety of projects in the area of MDE and Requirements Engineering, namely the projects related with the research initiatives of MDDLingo and RSLingo, briefly introduced bellow:

- **MDDLingo** initiative aggregates projects that are related to MDE, namely the ones of the XIS family of languages (XIS*) like XIS [10], XIS-Mobile [11, 12], XIS-CMS [13], XIS-Web [4], XIS-Analytics and XIS-Reverse.
- **RSLingo** initiative aggregates projects that are related with Requirements Specification, namely RSL [14], RSLingoPrivacy [15] and ReBox.

The XIS* and the RSLingo’s RSL suit as guidance to represent the produced high-level specifications of legacy information systems. Thus, this work is mainly focused in the extraction of legacy information systems’ specifications based in XIS* and RSL languages (more details later). Since this work focuses on reverse engineering, unlike the aforementioned projects that promoted forward engineering approaches, we named the proposed approach as XIS-Reverse.

1.2 Problem and Research Questions

With the explosion of technologies, it is usually challenging to maintain information systems inside organizations, usually caused by incorrect requirements management, more precisely due to the hard task which is updating information system requirements specification during its lifetime, leading to deterioration.

One of the solutions to this problem is the use of MDRE approaches that can extract updated specifications from a given information system. And then, these specifications can be used to correct, update, upgrade or even completely reengineer the information system.

The problems our research addresses are: legacy information system’s requirements deterioration and maintenance.

The following research questions summarize the situation to address:

- **RQ.1**: How to extract legacy information system’s specifications?
- **RQ.2**: How to apply a MDE approach to reverse engineering a legacy information system?
- **RQ.3**: Which concepts should be extracted?
- **RQ.4**: How to represent the extracted specifications?

The ideal scenario would be to develop a tool capable to extract high-level information system’s specifications from an existent information system, reducing the analysis time.

[https://github.com/RSLingo](https://github.com/RSLingo) (Accessed on June 2017)
1.3 Thesis Statement

The research thesis argues that MDRE can be successfully applied to extract high-level specifications from a legacy information system, reducing human effort.

These goals can be achieved by using reverse engineering techniques based on a model-driven approach, capable to produce high-level specifications of the legacy information system through Model-to-Model (M2M) transformations. This is accomplished using and extending the Sparks Systems Enterprise Architect (EA) tool with those transformations.

1.4 Research Proposal

This research proposes the XIS-Reverse approach and its companion tool, a solution to mitigate the previously mentioned problems. This approach lays down in the extraction of high-level platform-independent specifications from information system database artefacts (e.g. schema), based upon predefined and partially configurable reverse engineering rules. Those specifications are defined in the XIS family of languages and the RSLingo’s RSL.

Figure 1.1 overviews the XIS-Reverse approach and the technologies that support its tool. The XIS-Reverse process starts with the extraction of the data schema from the application database. Second, using the tool, the user is able (not only to introduce his knowledge about the application through some available configuration points to better guide the reverse engineering process, but also) to include other application artefacts in the process, namely a database connection or a profiler log file, to enhance the produced specifications. Third, the tool applies a chain of reverse engineering model transformations to all the aforementioned inputs, and produces high-level specifications (XIS* and RSLingo’s RSL) of the current legacy application.

![Figure 1.1: XIS-Reverse approach and technologies.](http://www.sparxsystems.com/products/ea)

This process is mainly supported by the Sparks Systems EA tool. This tool allows to perform the data schema extraction using its native capabilities. Moreover, the following tasks in the process, namely the reverse engineering tasks, are supported by the XIS-Reverse tool, that is implemented as an EA extension/add-in, and produces the output specifications.
The XIS-Reverse tool aims to help its users, such as system analysts, to produce information system specifications with a lower amount of analysis, or even in the extreme scenario, to produce specifications without any prior user knowledge being required.

Given the proposed solution, it is also important to highlight the goals of this research:

- **G1:** Collect and analyse the related work done in the scope of data schema extraction and **MDRE** of information systems.
- **G2:** Develop an approach capable to extract high-level specifications from application database artefacts.
- **G3:** Implement a tool that supports that approach, including needed user configurations and generation of specifications.
- **G4:** Evaluate the produced specifications through analysis of real-world case studies.

## 1.5 Methodology

Since the research goal is to define and implement an approach in an iterative and gradual way, we follow the Action Research methodology [1]. Figure 1.2 depicts such methodology as a cyclical process that can be applied to a client-system infrastructure environment (specification and agreement that constitutes the research environment). This Action Research methodology is composed of five stages [1]:

![Action Research Methodology cycle](extracted_from_1.png)

- **Diagnosing** - Identify the main problem domain and its relevance;
- **Action Planning** - Plan and define a solution and identify the needed changes to solve the previous step problem;
• **Action Taking** - Implement the previous step solution;

• **Evaluating** - Determine if the previous step implementation actually contributed to solve the identified problem, by analysing the performed actions.

• **Specific Learning** - Identify the lessons learned in the actual cycle, and sometimes, the ones used while preparing the next iteration of the action research cycle.

Based on the Action Research methodology, our research work is subdivided in 3 iterations:

**Iteration 1 (Month 1 to Month 5):** In the first place, the state of the art analysis was performed focused in the reverse engineering topic, namely reverse engineering of relational databases, taking into account schema, data and program analysis. Since the main objective in the beginning was to ultimately represent models using profiles from the XIS family of languages, the tools and technologies that produce XIS* models were also explored. Taking into account that Sparx Systems [EA] was used to design information systems in the other XIS family of languages, and this tool also provides the native capability to reverse engineer a database schema, thus we decided to implement our approach on top of this tool. During this first iteration we focused in the schema extraction from the legacy database.

• **Diagnosing** - Review of the existing literature and technologies regarding the definition of the XIS-Reverse approach and tool. Preliminary definition and implementation of the XIS-Reverse approach.

• **Action Planning** - Definition of the main transformations regarding the Domain view generation. Definition of the main technology to support a preliminary version of the XIS-Reverse.

• **Action Taking** - Implementation of the preliminary XIS-Reverse approach using [EA].

• **Evaluating** - Test with a simple case study and also with a real-world legacy application. Submission of a short paper with the preliminary results.

• **Specific Learning** - Success on generating a Domain view from a database schema. However, the preliminary results were still not relevant compared with the already available approaches, which led to negative response to the paper submission. Need to generate more detailed domain specifications, namely classifying entity relationships. Need to generate Business Entities and Use Cases views. Need to increase readability of the produced models, since when we generated models using a real-world legacy application we realized that large domains would produce complex views due to the number of entities and relationships.

**Iteration 2 (Month 6 to Month 8):** The second iteration was mainly focused in the enhancement of the produced domain views, which led to a redefinition of the preliminary approach. In order to update and define new heuristics, we took into account more details, for example new artefacts, such as log files with the executed Stored Procedures. This allowed to identify implicit generalizations, specializing associations (distinguishing between associations and aggregations) and thus, allowed to generate detailed Business Entities views and therefore Use Case views. Additionally, some configurable settings were implemented in order to increase readability of the produced models, namely arranging models into smaller views and changing entities names to a clearer one.

• **Diagnosing** - Analysis of the problems and needs discovered in the previous iteration. Research about reverse engineering approaches with multiple input artefacts.
• **Action Planning** - Definition of more detailed M2M transformations taking into account more information, in order to generate more detailed domain views, and from those generate business entities and use cases views. Definition of the process in order to accept user configurations, and choose the appropriate configuration points.

• **Action Taking** - Definition and implementation of new heuristics which allowed to identify aggregation associations and implicit generalizations, other ones which allowed to generate business entities and use cases views, and another ones that were responsible to organize smaller sets of entities and to change the name of entities.

• **Evaluating** - Reverse engineering of a real-world legacy application using our tool, also including a profiler log file generated using such application and a user configuration, in order to identify aggregations and implicit generalizations, and to increase readability of the models.

• **Specific Learning** - Sometimes, a profiler log file is not easy to generate and a small profiler log file (few hundreds of megabytes) may not be enough to capture the usual application usage, thus an alternative input should also be included. Need to extract and represent attribute values in the produced specifications, in order to give a better understanding of the role of certain entities in the domain. Additionally, a different output representation was decided to be included.

**Iteration 3 (Month 9 to Month 16):** The third iteration started with the definition of M2M transformations in order to produce RSLingo’s RSL specifications. Moreover, a new input artefact was taken into account, namely database access, which allowed to define an alternative way to identify aggregation associations and in the other hand, it also allowed to extract attribute values from the database. After that, the evaluation of those improvements originated the publication of a paper [16]). And last but not the least, the last month of this iteration was used to finish the writing part of this thesis.

• **Diagnosing** - Analysis of the problems and needs discovered in the previous iteration, namely limitations of using a profiler log file as input, the need to extract attribute values and the need to generate an additional output representation (RSLingo’s RSL).

• **Action Planning** - Definition of M2M transformations to integrate a new type of analysis in the approach, to extract attribute values from a database and to generate RSLingo’s RSL specifications.

• **Action Taking** - Implementation to integrate the new input in the process, whose technology was also used to establish a database connection to retrieve attribute values. Moreover, there was an update in the tool, in order to optionally produce RSLingo’s RSL specifications.

• **Evaluating** - Testing with two real-world legacy applications. Testing interoperability with a XIS* framework. Publication of a full paper with the achieved results using one of those legacy applications.

• **Specific Learning** - Analysis of the results obtained from the evaluation performed with the two real-world applications, which revealed that the identification of aggregations works properly in systems with a good amount of data, and the identification of implicit generalizations also works well. Regarding the interoperability with a XIS* framework, the produced models using XIS-Reverse are suitable to be used with the XIS-Web framework. As future research directions, there are ideas to extend the approach to support different technologies and reverse engineering approaches.
1.6 Publications

The following paper was published in an international conference:


(This paper describes the last version of XIS-Reverse (approach and tool), and presents the results of its evaluation using a real-world application.)

1.7 Outline

The remainder of this dissertation is organized as follows:

**Chapter 2** - This chapter provides an overview of the main concepts and technologies that underlie this research.

**Chapter 3** - This chapter describes the proposed approach, known as XIS-Reverse. **A MDRE** approach for legacy information systems.

**Chapter 4** - This chapter presents and analyses the evaluation performed to XIS-Reverse. **Using two real-world applications and a XIS* framework.**

**Chapter 5** - This chapter summarizes the main conclusions of this work along with some future work perspectives.
Chapter 2

Background

This chapter introduces the concepts and the main technologies supporting this research. Firstly, Section 2.1 introduces the concepts of modeling languages and metamodeling, and covers the core concepts of model-driven reverse engineering, such as model-driven engineering and model-driven reengineering. Then, regarding the produced specifications with the XIS-Reverse approach, Section 2.2 and 2.3 present the research initiatives of MDDLingo and RSLingo. Finally, Section 2.4 introduces related work particularly focused in data schema extraction and reverse engineering.

2.1 Basis of Model-Driven Reverse Engineering

This section presents the core concepts that support this research. Namely, Section 2.1.1 defines the concepts of Modeling Language, Metamodel and Model. And then, Sections 2.1.2, 2.1.3 and 2.1.4 present the overall picture of model-driven engineering, model-driven reengineering and model-driven reverse engineering, the relationship between them and the model transformations used in each process.

2.1.1 Modeling Languages and Metamodeling

Although there is not a consensual definition of metamodels, some authors suggest the following definitions [2]: (1) a metamodel is a model that defines the language for expressing a model; (2) a metamodel is a model of a language of models; (3) a metamodel is a specification model for which the systems under study being specified are models in a certain modelling language.

From the aforementioned mentioned definitions, as illustrated in Figure 2.1 we can infer that a metamodel is an instance of a model and that a model is an element of a modeling language, which in turn is defined by a metamodel. Thus, a model should satisfy the rules defined in the metamodel, in other words, o model conforms with the metamodel. Moreover, metamodels define the abstract syntax of metamodeling languages, such as Ecore [17] and MOF, providing the core constructs of the object-oriented conceptual modelling.

Modeling languages can be split in two categories: (1) General-Purpose Modeling Languages or Domain Specific Modeling Languages. The first one usually presents a larger number of generic constructs, promoting a widespread use in different fields of application. An example of such type of modeling languages is the Unified Modeling Language specified by the OMG, which is primarily used for specifying and documenting software systems in an object-oriented way.

On the other hand, **Domain Specific Languages (DSLs)** enable to describe the domain in which they are applied, in a more accurate way. This can be achieved by using language elements that are closely connected to the constructs of the application domain. Moreover, while UML is used to model any system in a generic way, a DSL by being closely related to the problem domain, is able not only to do it in a more expressive way but also to capture more details. However, UML allows to extend its concepts, by specifying a profile. Thus, with such specification it is possible to specialize the UML, creating a DSL merging the best features of both approaches: wide usage of UML, and the ability to collect more details specification using DSL.

Regarding to model-driven reverse engineering, models can be used to represent the extracted knowledge from the legacy information system. Furthermore, in our approach we use a UML profile to represent the extracted knowledge.

**2.1.2 Model-Driven Engineering**

MDE is a software engineering approach that promotes the use of models throughout all engineering disciplines and in any application domain, by considering models not just as documentation artefacts but also as first-class citizens [2]. Moreover, a MDE approach can be seen as a chain of model transformations that produce the target software artefacts from more abstract models [3]. Using this kind of approach, the most common model transformations are: M2M and Model-to-text (M2T) [2]. M2M transformations generate a target model from a source model, which is typically closer to the solution domain or that satisfies specific stakeholders’ needs. M2T transformations produce a textual representation from a source model, such as source code, XML and so on.

**2.1.3 Model-Driven Reengineering**

Model-driven reengineering is the application of MDE principles and techniques to reengineer an application. As illustrated in Figure 2.2, a complete model-driven reengineering process is composed of three main stages [7]: (i) reverse engineering, (ii) restructuring and (iii) forward engineering. First, the reverse engineering stage aims to extract knowledge defined in low abstraction representations through introspection and analysis of the initial source artefacts (initial models); this process collects valuable information that makes it easier to define current legacy application requirements. Second, the system restructuring stage involves establishing a mapping between the source and the target systems. Third, the forward engineering stage produces the artefacts to the new target system. Each of these stages is a transformational process that can be implemented by means of a model transformation chain whose initial set of models is injected. This model injection is performed with a Text-to-Model (T2M) transformation applied to textual software artefacts [3], such as SQL Data Definition Language (DDL) scripts.
2.1.4 Model-Driven Reverse Engineering

MDRE uses MDE principles and core techniques to build effective reverse engineering solutions. Such solutions are able to produce relevant (model-based) views from legacy systems, thus facilitating their understanding and subsequent manipulation [8]. These models are then used as the starting point of reverse engineering activities. Therefore, MDRE directly benefits from the genericity, extensibility, coverage, reusability, integration and automation capabilities that MDE technologies usually provide [8].

In addition, a MDRE process can be seen as a part of the model-driven reengineering process, merely focused in the reverse engineering stage.

Moreover, this research proposes a MDRE approach to produce high-level specifications of legacy information systems in a human-controlled way. These specifications are defined according to domain specific languages and a language for Requirements Engineering, produced from the injection and the reverse engineering stages.

2.2 MDDLingo

MDDLingo is an umbrella researching initiative that aggregates several projects around MDE topics, namely involving the definition of a family of languages known as XIS*. This set of modelling languages derives from the XIS-UML profile [10, 18] and involve the XIS-Mobile [11, 12], XIS-CMS [13] and XIS-Web [4]. XIS-UML is a set of coherent constructs defined as an UML profile that allows a high-level and visual modeling way to design business information systems. In general these languages include the following views: Entities (aggregating Domain and Business Entities views), UseCases (containing Actors and Use Cases views), Architectural and User-Interfaces (composed by Interaction Space and Navigation Space views). Figure 2.3 illustrates the aforementioned set of views.

Figure 2.4 illustrates a simple XIS* Domain view which aggregates domain classes (XisEntity), their attributes (XisEntityAttribute) and relationships (XisEntityAssociation and XisEntityInheritance).

Figure 2.5 shows the BusinessEntities view which allows to define higher-level entities (XisBusinessEntity), that aggregate XisEntities and that in the context of a given use case can be easily manipulated. There are 3 relationship types between XisBusinessEntity and XisEntities: MasterAssociation, a unique and mandatory relationship with the main XisEntity that will be managed by XisUseCases using...
this XisBusinessEntity; DetailAssociation, a relationship with a XisEntity that intrinsically belongs to the master entity (e.g. aggregation); ReferenceAssociation, similar to the second one, however in this case, the XisEntity does not have to intrinsically belong to the master entity. Moreover, a XisBusinessEntity can have zero or more detail and reference associations.

Figure 2.3: The multi-view organization of XIS* (extracted from [4]).

Figure 2.4: Example of a XIS* Domain view.

Figure 2.5: Example of a XIS* BusinessEntities view.

Figure 2.6 shows the UseCases View. This view details the operations an actor can perform over the business entities when interacting with the system [11].

XIS* languages are very alike in terms of the Entities view, however, there are some discrepancies on their Use Cases, Architectural and User-Interfaces views due to the specific needs of each platform target application. For example, XIS-CMS is oriented to CMS-based applications and does not have for instance an Architectural view [13].
It is to point out that all of these languages are capable to generate the User-Interface view, using a smart approach based on M2M transformations given the set of Domain, Business Entities, (Architectural) and Use Cases views. Once all those views are specified, the corresponding XIS\* framework is able to generate the specific target platform source code for a new application, through M2T transformations.

2.3 RSLingo

RSLingo is a general approach defined to rigorously specify and validate software requirements using lightweight Natural Language Processing techniques to (partially) translate informal requirements into a rigorous representation provided by a language specially designed for Requirements Engineering. Over time, following the RSLingo's approach, several projects have been developed, namely RSLingo4Privacy \cite{19, 20} and RSLingo's RSL\cite{14}.

RSLingo’s RSL (or just RSL for brevity) is a System Requirements Specification language (structured way to collect the requirements of a system), which follows the multi-view architecture and it is inspired on the design of several former languages such as RSL-IL \cite{21, 22}, XIS\* \cite{18, 23} and SilabREQ \cite{24}, and that is aware of the problems and practical recommendations discussed in \cite{18, 25}. Overall, RSL is a control natural language (restricted use of a natural language grammar and a set of terms to be used in a restricted grammar) to help the production of software requirements specifications in a more systematic, rigorous and consistent way \cite{14}. Such specifications are usually specified as a set of .rsl files, and later they can be validated and used by different types of users such as requirement engineers, business analysts, or domain experts \cite{14}. The most relevant RSL concepts regarding our research are explained below.

Listing \ref{lst:rslingo} shows a simple RSL description of dataEntities. These entities are somehow similar to XisEntities, and their specifications are very alike SQL. Moreover, each dataEntity has a type (Principal or Secondary), a subtype (Simple or Complex), a name, and may have an dataEntity superclass (through an isA relationship), a set of attributes, primary key and foreign key constraints (referencing another dataEntity), and a description. Each attribute has an identifier, a type, and may also have a name, NotNull and Unique properties, and extracted values.

\begin{quote}
\begin{lstlisting}[language=RSL]
XisEntity {dataEntity}:
  type: [Principal | Secondary]
  subtype: [Simple | Complex]
  name: string
  [isA: XisEntity {dataEntity}]
  [attributes:
    [name: string, 
     [NotNull | Unique]
    ]
  ]
\end{lstlisting}
\end{quote}

\cite{https://github.com/RSLingo/RSL} (Accessed on June 2017)
Listing 2.1: RSLingo’s RSL example of DataEntities.

dataEntity e_AUTHOR : Principal : Simple [ name "Author" attribute IdNumber : Integer [name "Idnumber" NotNull] primaryKey (IdNumber) description "An author is a main entity"
]
dataEntity e_DOCUMENT : Secondary : Complex [ name "Document" isA e_Superclass_DOCUMENT_SYSTEMDOCUMENT attribute Id: Integer [name "Id" NotNull] attribute AuthorIdNumber : Integer [name "Authoridnumber" NotNull] ... (more attributes) primaryKey (Id) foreignKey e_AUTHOR(AuthorIdNumber) foreignKey e_DOCUMENTCATEGORY(CategoryId)
]
dataEntity e_SYSTEMDOCUMENT : Principal : Simple [ ...
  attribute Status : String [ name "Status" values "Accepted; Declined; Expired"
  ] ...
]

Listing 2.2 shows a simple RSL description of dataEntityViews. These entities follow most of the XisBusinessEntities specifications. Moreover, each dataEntityView has a type (Complex, Simple, or VerySimple), a name, a master relationship (with the corresponding dataEntity), and may have detail and reference relationships with dataEntities.

dataEntityView ev_AUTHOR : Simple [ name "Author" master e_AUTHOR detail e_DOCUMENT description "Author"
]
dataEntityView ev_SYSTEMDOCUMENT : VerySimple [ name "Systemdocument" master e_SYSTEMDOCUMENT description "Systemdocument"
]

Listing 2.3 shows a simple example of RSL Use Cases. These specifications are similar to XisUse-Cases views, allowing to define system Use Cases that are related with a specific dataEntityView, and the actions an actor can perform in such use case.

useCase uc_1: EntitiesManage [ name "Manage Author" actorInitiates AU_User dataEntityView ev_AUTHOR actions Create, Read, Update, Delete, Search, Filter ]

Listing 2.4 shows a simple example of RSL User Stories. These specifications allow to define the desired system user stories, in which the required privileges and features for a certain user are specified.
Listing 2.4: RSLingo’s RSL example of User Stories.

userStory us_1: UserStory [name "Manage Author" asA AU_User iWant "Manage Author with Create, Read, Update, Delete, Search, and Filter features" soThat "some reason"
]

Listing 2.5 shows a simple example of RSL Functional Requirements. These specifications allow to define the desired system functional requirements, in which the participant stakeholder and the priority level are specified.

Listing 2.5: RSLingo’s RSL example of Functional Requirements.

FR fr_1: Functional [name "Manage Author" stakeholder stk_user priority Should]

Listing 2.6 shows a simple example of RSL Goals. These specifications allow to define the desired system goals, in which the participant stakeholder and the priority level are specified.

Listing 2.6: RSLingo’s RSL example of Goals.

goal g_1: Concrete [name "Manage Author" stakeholder stk_user priority Should]

Listing 2.7 shows a simple example of RSL Business Processes. These specifications allow to define the desired system business processes, in which the participant stakeholder is specified.

Listing 2.7: RSLingo’s RSL example of Business Processes.

businessProcess bp_ManageAUTHOR: User [name "Manage Author" participant stk_user]

Finally, Listing 2.8 shows a simple example of RSL Terms. These specifications allow to define the desired system terms, in which the type of entity that the term is applicable to is specified.

Listing 2.8: RSLingo’s RSL example of Terms.

term t_AUTHOR : Noun [name "Author" applicableTo DataEntity description "].."

2.4 Related Work

Reverse engineering of software applications have been extensively studied in the last decades, and their main contributions have been crucial to produce better results in this complex activity, and furthermore to stimulate the development of reverse engineering tools. More recently, MDE started to be applied to reverse engineering (MDRE), promoting a more systematic and flexible process. Moreover, MDRE approaches have been extended in order to completely reengineer a source application into a new target application through model-driven reengineering techniques.

This section overviews the most relevant research studies, covering data schema extraction (section 2.4.1) and reverse engineering of information systems (section 2.4.2).

2.4.1 Data Schema Extraction

Gra2MoL [26] T2M language and MoDisco [8] framework have been specially tailored for data schema extraction (extraction of a database schema). Gra2MoL is a DSL to write transformations between any textual artefact which conforms to a grammar (e.g. source code) and a model which conforms to a target metamodel. On the other hand, MoDisco is a Java framework intended to facilitate the implementation of MDRE approaches. Regarding to data schema extraction, MoDisco facilitates the implementation of discoverers (model injectors), and it currently provides discoverers for Java, JSP and XML. A drawback for both approaches is that they would require the definition of such transformations and discoverers, respectively, to extract the database schema.
Schemol \[27\] is another tool for injecting models. However, this tool only allows injecting data stored in a database by specifying transformations that express the correspondence between the source database schema and the target metamodel.

Furthermore, in terms of database model injection (to ease this T2M transformation) it is possible to transform a database schema into a graphical representation using a variety of commercial and academic tools. DB-MAIN \[28\] and SQL2XMI \[29\] are two examples of such academic tools. Firstly, DB-MAIN is a toolbox that supports database reverse engineering, and includes legacy database schemas extractors, through several sources such as ODBC drivers or SQL files. Secondly, SQL2XMI is entitled as a lightweight transformation of data models from SQL Schemas to UML-ER expressed in XMI. To our knowledge, this tool is still limited compared with DB-MAIN since it does not infer entity types or cardinalities, and for now it is only compatible with the MySQL implementation of the SQL DDL.

2.4.2 Reverse Engineering

Regarding reverse engineering techniques, several approaches have been proposed, which are usually distinguished by the particular application artefact used as main information source. The most relevant research works, following such distinction, are described below:

- **Schema analysis** \[30\] is mainly focused on spotting similarities in names, value domains and representative patterns. This technique may help identify missing constructs (e.g. foreign keys).

- **Data analysis** \[31, 32\] uses content mined from a database. Firstly it can be used to analyse the database normalization and secondly to verify hypothetical constructs suggested by other techniques. Given the combinatorial explosion that can affect the first approach, data analysis technique is mainly used with the purpose of the second approach.

- **Screen analysis** \[33\] state that user interfaces can also be sources of useful information. These user-oriented views over a database may display spatial structures, meaningful names and, at run time data population and errors combined with data-flow analysis may provide information about data structures and schema properties.

- **Static** \[34, 35\] and **Dynamic** \[36, 37\] **program analysis** can easily give valuable information about field structure and meaningful names, or identifying complex constraint checking and foreign keys after a complex analysis. A main challenge of dynamic program analysis is the extraction of highly dynamic interactions between a program and a database. Moreover, the analysis of SQL statements is one of the most powerful variant of source code analysis.

Additionally, a set of approaches, concerning the application of MDE are also taken into account. Our analysis focused on their injection and reverse engineering stages.

As previously introduced, MoDisco MDE framework \[8\] has a huge potential in order to support reverse engineering activities, due to its generic and extensible properties. Besides its legacy application discoverers (model injectors), MoDisco also allows the definition of transformations and generators, responsible for restructuring and forward engineering tasks over the system models, respectively.

Polo et al. propose a method and a tool, called Relational Web \[38\] specially designed for database reengineering. The starting point is a relational database, whose physical schema is reverse engineered into a class diagram representing its conceptual schema. In the restructuring stage, the class descriptions are completed with state machines by the user and then passed as input to the forward engineering step. Moreover, this tool supports the definition of new database managers to be used as input and the implementation of new code generators. Furthermore, on the one hand this approach only uses as input
the physical schema, and user knowledge, and its tool does not take advantage of the existing MDE
techniques nor technologies.

As previously stated, DB-MAIN [28] is a toolbox that offers a complete functionality with which to
apply database reverse engineering. Regarding to reverse engineering DB-MAIN includes features
such as extractors of legacy database schemas, transformations between schemas, data and code
analysis tools, among others. This tool is one of the most mature ones, in regard to database reverse
engineering, meaning that it includes several features that have been the result of a great number of
research contributions from Namur University. DB-MAIN supports a lot of common transformations and
extraction tools thus, a user with such tools can handle almost any needed transformation to create a
good conceptual schema.
Chapter 3

XIS-Reverse Approach

This chapter presents the proposed approach. The approach is composed by three main stages which are briefly introduced in section 3.1. Section 3.2 introduces the running example that is used to illustrate the transformations used in this approach. Sections 3.3, 3.4 and 3.5 describe in detail the stages of the approach. Finally, Section 3.6 still describes the technology behind the XIS-Reverse software tool.

3.1 Overview

XIS-Reverse is the MDRE approach proposed in this research. This approach focuses in the first two stages of the broader reengineering process as discussed in Section 2.1.3 (see Figure 2.2), namely: Injection and Reverse Engineering stages. XIS-Reverse is represented in Figure 3.1 and starts with the extraction of the application database schema from an existent data source (through injection). Thereafter, the Application data model is generated. Secondly, the user is able to define his own preferences by tweaking some heuristic's parameters, producing the XIS-Reverse configuration. The reverse engineering stage takes as input these two artefacts and generates XIS* models and RSLingo's RSL models. Then, the user can analyse the produced artefacts and introduce some refinements, such as changing automatically identified relationships into different ones in the Entities view, enhancing the Use-Cases views, etc.

3.2 Running Example

For better understanding and simplicity of the explanation, we introduce a simple but practical example: the “Tiny Social Security” (SS-App) legacy application.

The SS-App is a simple application that manages beneficiaries, dependents and tutors, associated documents, system's documents and user accounts.

This domain is modelled as 8 tables: Beneficiary and Dependent which primary keys are foreign keys to a Person table. Dependents table represents the relationship between a Beneficiary and a Dependent (through its foreign keys linked to each of those tables). A Dependent may have a Tutor, which is linked by a foreign key. Each Beneficiary has a set of documents (since a Document has a foreign key to a Beneficiary table) and a unique UserAccount (due to the unique constraint that UserAccount table has over its BenIdNumber foreign key). Furthermore, several documents not linked to any Beneficiary may exist as a SystemDocument. Each of those tables, columns, constraints and number of rows in the database are represented in Figure 3.2.
3.3 Data Schema Extraction

Following the aforementioned process of Figure 3.1 during the Data Schema Extraction stage, in order to extract the application data model using our approach, a connection with a Database Management System (DBMS) is established and the database schema is extracted. For example, Figure 3.3 illustrates the SS-App database in Microsoft SQL Sever Studio which is extracted and represented as an application data model (illustrated in Figure 3.2).
3.4 Reverse Engineering Configuration

A snapshot of the tool is illustrated in Figure 3.4, which is used to explain the main configuration features.

The configuration panel of the tool has four main areas:

- **Input** - This area allows to specify the root node Package where the application data model was imported, to introduce the database name and to choose how to enhance the output specifications. Moreover, if database access is chosen to enhance specifications, it is also possible to test if it is
possible to establish connection with the database identified by the specified database name. On the other hand, if the Profiler Log File option is selected, a file must be selected in order to start the reverse engineering execution.

- **Output** - This area allows the user to select additional output representations, namely XIS-Web or RSLingo’s RSL representations, and their output location (if applicable).

- **Transformation Rules Guidance** - This area allows the user to contribute with his knowledge to enhance the Reverse Engineering process. This configuration can be split into three areas: (i) Simple Principal Entities (ii) Attribute Values Extraction and (iii) Generalization discovery. The user can select which tables are Simple Principal Entities (which we consider as entities without many relationships, such as configuration entities or “kind of” entities). Moreover, the user can enhance the produced specifications by extracting values from a selected column of a certain table. Furthermore, in both cases the user can also filter the list of tables by specifying the maximum number of rows per table, which allows the user to find “kind of” tables more easily, since usually they have a small number of instances (rows). The selection window for both (i) and (ii) areas can be seen in Figure 3.5. Finally, the user can also activate generalization discovery and add some knowledge, such as a list of XisEntityAttribute names to ignore, the minimum number of shared XisEntityAttributes (e.g. 5 or 10), and to choose how to aggregate those entities, namely by the higher number of shared attributes or by the higher number of aggregated entities.

- **Appearance** - This area allows to improve readability of the produced specifications, by arranging models into smaller views (Views area) and changing the name of the entities to a clearer one (Entities name customization area). In regards to arranging models, it is possible to do it by table schema and also by the table name prefix/suffix, since it is also a recurrent pattern used to organize database tables. Using the last two options, the user must introduce the number of characters to take into account or specify a character which will split every table name (e.g. underscore). On the other hand, it is possible to clean the name of each entity by removing some characters from the corresponding table name, applying a regular expression or specifying the number of characters that will be removed from the beginning and/or the end of each name.

Moreover, during the Reverse Engineering Execution, additional information about the progress and the current state of the execution can be found in the title of the window allied with the progress bar below the Reverse Engineering Execution button.

### 3.5 Reverse Engineering Execution

Taking as input (i) the application data model, (ii) the XIS-Reverse configuration and (iii) the profiler log file or the database connection, the reverse engineering process is composed by a set of transformation tasks, that will be applied. Those include a chain of M2M transformations which are used to specify the application as a set of XIS* models and RSLingo’s RSL specifications.

#### 3.5.1 Initialization

Before M2M transformations take place, the profiler log file or the database connection, if provided, are used to extract more detailed information about the legacy application, that may enhance the produced specifications.
From the profiler log file it is possible to extract SQL statements (INSERT, SELECT, UPDATE and DELETE), and from those statements the set of used tables. Moreover, from that we generate 4 different lists, one for each type of statement, where each one of them holds for each table the number of occurrences for the corresponding kind of statement.

A database connection can be used to collect for each table in the application data model, the number of rows in the specified application database.

### 3.5.2 Transformations to XIS models

The following transformations are broken down into the different set of input elements.

#### 3.5.2.1 XIS Entities transformation rules

For each table of the extracted application data model, the following XIS Entities (XE-i) rules are applied:

(XE 1) If a table has exactly 2 foreign keys, each one to a distinct table, does not contain more attributes (excluding its foreign keys and primary keys), and one of the two things (1) the primary key is composed of exactly two primary keys which are also the foreign keys (two pfK) or (2) both foreign keys are not primary keys. Following that, this table will be translated into a XisEntityAssociation between the two referenced tables, using a many-to-many cardinality, ie. *:* cardinality (e.g. Dependents in the running example - illustrated in Figure 3.6).

(XE 2) Otherwise the table will map a XisEntity (e.g. Beneficiary in the running example - illustrated in Figure 3.7).

#### 3.5.2.2 XIS Entity Attributes transformation rules

Regarding table columns, the following XIS Entity Attributes (XEA-j) rules are applied:

Let A, B be two tables related by a foreign key constraint from B (referencing table) to A (referenced table):
(XEA 1) For the complete Primary Key:

a) If all the foreign key columns in B are also its complete primary key, in other words B and A have a 1:1 relationship since B primary key references A primary key. Then B foreign keys will be translated into a XisEntityInheritance relationship, where the class equivalent to A will be the superclass and the equivalent class to B will be the subclass (e.g. Person (A) and Beneficiary (B) in the running example - illustrated in Figure 3.8).

b) Otherwise, the complete Primary Key will not be represented (e.g. User Account in the running example - illustrated in Figure 3.9). Due to XIS* Domain View similarity to a Class Diagram, each XisEntity can be seen as a Class, which can have objects, and each object is different by definition, so there is no need for a unique identifier for each XisEntity.
(XEA 2) For the remaining columns with Foreign Key constraints:

a) If the foreign key in B constitute a Unique Index, the XisEntityAssociation will have a 1:1 cardinality, preserving the relationship direction (e.g. BenIdNumber of UserAccount in the running example (the underlined column name is used to illustrate the unique index property) - illustrated in Figure 3.10).

![Figure 3.10: XEA-2-a transformation example.](image)

b) If A was not selected as Simple Principal Entity by the user (configuration stage), and one of the following options is also true:

- The profiler log file was provided, both A and B tables were referenced in that file, and table B had the same or higher amount of INSERT operations as A did.
- The database connection was provided and table B had the same or higher amount of rows as A did (if B foreign key column allows NULL values, only rows with NOT NULL values on that column will be taken into account).

Then we assume that there is an aggregation relationship between entity A and B (A aggregates B), that is translated into a XisEntityAssociation, preserving the foreign key cardinality and represented with a bidirectional arrow (e.g. Beneficiary with 500 rows (A) and Document with 1000 rows (B) in the running example - illustrated in Figure 3.11).

This reasoning made sense for us since that, assuming that A and B are respectively strong and weak entities, A is the one which aggregates B, it is fair to assume that if that foreign key was made for that purpose, then in a system with good amount of data there must be at least the same amount of B instances created compared with A.

![Figure 3.11: XEA-2-b transformation example.](image)
c) Otherwise, the XisEntityAssociation will have a 1:* cardinality between table B and A, preserving the relationship direction (e.g. Dependent (A) and Tutor (B) in the running example - illustrated in Figure 3.12).

![Figure 3.12: XEA-2-c transformation example.](image)

(XEA 3) Otherwise the column will be translated into a XisEntityAttribute with a corresponding XIS* attribute type following a predefined mapping (e.g. Status from table SystemDocument in the running example (without annotation) - illustrated in Figure 3.13). Moreover, if previously configured, the user can explicitly select if values from a certain XisEntityAttribute from a particular XisEntity should be extracted and written as an annotation, which will be linked to the XisEntity (e.g. the same example as before, but now including the annotation with the values from that column - illustrated in Figure 3.13). This feature is a significant contribution since the produced specifications are enhanced with those values, allowing a better understanding of the XisEntities captured from the legacy domain.

![Figure 3.13: XEA-3 transformation example.](image)

**Note:** in both b) and c) cases, from XEA-2, when the Not Null property is not set for that column (in other words, that column can be NULL), the XisEntityAssociation target’s cardinality is the concatenation of 0.. with the original target cardinality. (e.g. 1 was the original target cardinality and the column allowed NULL values, then it becomes 0..1)
3.5.2.3 Implicit XIS Entity Inheritance transformation rule

For each XisEntity, after the previously mentioned XisEntities and XisEntityAttributes rules are applied and if the Generalization Discovery feature was configured, the following XIS Entity Inheritance (XEI) rule is applied:

(XEI) A XisEntityInheritance identification is performed comparing every XisEntity and their XisEntityAttributes with each other. We assume that two XisEntityAttributes are the same if they share the same name and type. During this comparison, if previously defined, some XisEntityAttributes can be excluded by their name. Moreover, we cluster 2 or more XisEntities if they share at least the same or higher configured number of shared XisEntityAttributes. Furthermore during this comparison only XisEntities without inheritance relationship are taken into account.

After this clustering procedure, there are two ways of finding inheritance, depending on the configuration, the list can be ordered by the descending number of shared XisEntityAttributes or ordered by the descending number of entities in each cluster. While this list has clusters of XisEntities, a Superclass is created for each cluster. This Superclass will have the identified XisEntityAttributes, each of the XisEntities in that cluster will not own those XisEntityAttributes and each XisEntity will be linked to the superclass using a XisEntityInheritance (e.g. SystemDocument and Document share 3 identical columns (Path, Description and Notes) in the running example - illustrated in Figure 3.14).

Although, the concept of generalization is based on shared characteristics between entities, that can be attributes, associations, or methods. From the available characteristics (attributes and associations) we did not take into account associations, since those may require deep knowledge about the system. For example, beneficiary and dependent entities could have an association with a country entity, however the relationship in the first case could be used to specify the residence country of the beneficiary, and in the second case it could be used to specify the country where the dependent was born. Moreover, the same entity can have multiple associations, with different meanings, to the same entity.

Furthermore, in order to implement the first phase of this algorithm, given the complexity that is comparing every XisEntity with each other taking into account all their XisEntityAttributes, in domains that can easily be compound by hundreds or thousands of elements, we based our approach in the MapReduce [39] programming model which allows to handle large data sets.

Figure 3.14: XEI transformation example.
3.5.2.4 XIS Business Entities transformation rules

For each XisEntity, the following XIS Business Entities (XBE-k) rules are applied:

(XBE 1) If the current XisEntity aggregates other XisEntities or is not aggregated by another XisEntity (i.e. we do not consider weak entities as Business Entities), then a corresponding XisBusinessEntity will be created, followed by the creation of a XisBE-EntityMasterAssociation that will link that XisBusinessEntity with its XisEntity from the Domain view (e.g. Beneficiary in the running example - illustrated in Figure 3.15).

And, for each XisEntityAssociation:

(a) If the XisEntityAssociation is an aggregation (the current XisEntity aggregates another XisEntity) then it will be translated into a XisBE-EntityDetailAssociation between that XisBusinessEntity and the target XisEntity of the XisEntityAssociation (e.g. Beneficiary and Document in the running example - illustrated in Figure 3.16). Moreover, if the corresponding target XisBusinessEntity exists, there will be a XisBE-EntityReferenceAssociation between the target XisBusinessEntity and the source XisEntity.

(b) Otherwise, if the current XisEntity has a 0..* cardinality in that XisEntityAssociation, it will be translated into a XisBE-EntityReferenceAssociation between that XisBusinessEntity and the other XisEntity of the XisEntityAssociation (e.g. Tutor and Dependent in the running example - illustrated in Figure 3.17). Therefore, if the XisEntityAssociation is a many-to-many association, both XisBusinessEntities will have a XisBE-EntityReferenceAssociation to the other XisEntity (e.g. Beneficiary and Dependent in the running example).

(XBE 2) Otherwise there is no mapping.

Figure 3.15: XBE-1 transformation example.

Figure 3.16: XBE-1-a transformation example.

Figure 3.17: XBE-1-b transformation example.
3.5.2.5 XIS Entity Use Cases transformation rules

For each XisBusinessEntity, the following XIS Entity Use Case (XEUC) rule is applied:

(XEUC) There will be a corresponding XisEntityUseCase that is named “Manage” followed by the XisBusinessEntity master XisEntity name, associated to the XisBusinessEntity by a XisEntityUC-BEAssociation. And a default XisActor will be linked to the XisEntityUseCase using a XisActor-UCAssociation (e.g. Beneficiary in the running example - illustrated in Figure 3.18). Additionally, each XisEntityUseCase has boolean tagged values representing the CRUD operations for the master, detail and reference entities (e.g. CreateMaster, ReadDetail). And in our approach, all of them will be set to true.

Figure 3.18: XEUC transformation example.

3.5.3 Transformations to RSLingo’s RSL models

First of all, although RSLingo’s RSL specifications are textual artefacts, we consider the transformation chain which produces such artefacts a M2M transformation chain, since RSLingo’s RSL specifications conform to a well defined grammar, allowing to perform validations and then those specifications can also be forward engineered to produce several specifications conforming other formats, such as EXCEL files.

RSLingo’s RSL M2M transformations start together with the XIS M2M transformations, in order to collect all the needed information as RSLingo’s RSL specifications.
3.5.3.1 RSLingo’s RSL Data Entities transformation rules

Since RSLingo’s RSL uses a specification for Entities very similar to SQL, the following RSLingo’s RSL Data Entities (RDE-i) rules are applied:

(RDE 1) Every table is mapped into a RSLingo’s RSL dataEntity (e.g. Beneficiary in the running example - illustrated in Figure 3.19). Moreover, if the number of attributes of this dataEntity is higher than 5, its subtype will be Complex, otherwise will be Simple.

(RDE 2) Every column is mapped into a RSLingo’s RSL attribute. Also taking into account their NOT NULL properties, and converting their type into an equivalent RSLingo’s RSL attribute type (e.g. IdNumber column from Beneficiary in the running example - illustrated in Figure 3.19 line 4).

a) If a given column has a Primary Key or a Foreign Key constraint, the RSLingo’s RSL dataEntity will represent that as well, including the referenced table for the second case (e.g. IdNumber column from Beneficiary in the running example - illustrated in Figure 3.19 line 5 and 6 respectively).

b) Likewise XIS Entity Attributes rule XEA-3, values from a certain attribute can be extracted (e.g. Status from table SystemDocument in the running example - illustrated in Figure 3.20 line 7).

(RDE 3) For each identified XisBusinessEntity, the corresponding RSLingo’s RSL dataEntity will have Principal as dataEntity type (e.g. Beneficiary in the running example - illustrated in Figure 3.19 line 1). The left over RSLingo’s RSL dataEntities are classified as Secondary (e.g. Document in the running example).

(RDE 4) Every RSLingo’s RSL dataEntity that its corresponding XisEntity has a Superclass, will have an isA reference to the Superclass entity (e.g. Person is the Superclass of Beneficiary in the running example - illustrated in Figure 3.19 line 3). In the same way as aforementioned in 3.5.2.3.
implicit generalizations can be detected comparing every XisEntity generated. From that, found generalizations will also be added as RSLingo’s RSL dataEntities and their subclasses will have an isA reference to the entity they inherit from (e.g. SystemDocument Superclass in the running example - illustrated in Figure 3.20 line 3).

3.5.3.2 RSLingo’s RSL Data Entity Views transformation rules

For each XisBusinessEntity created, the following RSLingo’s RSL Data Entity Views (RDEV) rules are applied:

(RDEV) There will be an equivalent RSLingo’s RSL dataEntityView, with the same name, and same master, detail and reference associations to RSLingo’s RSL dataEntities. (e.g. Beneficiary in the running example - illustrated in Figure 3.21).

Figure 3.21: RDEV transformation example.

a) If the produced RSLingo’s RSL dataEntityView has detail and reference associations or the sum of detail and reference associations is higher than 5 it is classified as Complex, otherwise, if it has detail or reference associations it is classified as Simple, otherwise it is classified as VerySimple (e.g. Beneficiary (left), Dependent (middle) and Person (right) respectively in the running example - illustrated in Figure 3.22 line 1).

Figure 3.22: RDEV-a transformation example.

3.5.3.3 RSLingo’s RSL Use Cases transformation rule

For each XisEntityUseCase created, the following RSLingo’s RSL Use Cases (RUC) rule is applied:

(RUC) There will be an equivalent RSLingo’s RSL useCase with, the same name, EntitiesManage as type, a default actor which initiates the use case, an equivalent RSLingo’s RSL dataEntityView, and the following actions: Create, Read, Update, Delete, Search and Filter (e.g. Beneficiary in the running example - illustrated in Figure 3.23).

33
3.5.3.4 RSLingo’s RSL User Stories transformation rule

For each XisEntityUseCase created, the following RSLingo’s RSL User Stories (RUS) rule is applied:

(RUS) There will be an equivalent RSLingo’s RSL userStory with, UserStory as type, the same name, a default actor, and the following iWant property: “Manage «user story name» with Create, Read, Update, Delete, Search, and Filter features” (e.g. Beneficiary in the running example - illustrated in Figure 3.24).

3.5.3.5 RSLingo’s RSL Functional Requirements transformation rule

For each XisEntityUseCase created, the following RSLingo’s RSL Functional Requirements (RFR) rule is applied:

(RFR) There will be an equivalent RSLingo’s RSL FR with, Functional as type, the same name, a default stakeholder, and Should as priority (e.g. Beneficiary in the running example - illustrated in Figure 3.25).
3.5.3.6 RSLingo’s RSL Goals transformation rule

For each XisEntityUseCase created, the following RSLingo’s RSL Goals (RG) rule is applied:

(RG) There will be an equivalent RSLingo’s RSL goal with, Concrete as type, the same name, a default stakeholder, and Should as priority (e.g. Beneficiary in the running example - illustrated in Figure 3.26).

3.5.3.7 RSLingo’s RSL Business Processes transformation rule

For each XisEntityUseCase created, the following RSLingo’s RSL Business Processes (RBP) rule is applied:

(RBP) There will be an equivalent RSLingo’s RSL businessProcess with, User as type, the same name, and a default participant (e.g. Beneficiary in the running example - illustrated in Figure 3.27).

3.5.3.8 RSLingo’s RSL Terms transformation rule

For each XisEntityUseCase created, the following RSLingo’s RSL Terms (RT) rule is applied:
There will be an equivalent RSLingo's RSL term with, Noun as type, the same name, applicable to a DataEntity, and a default description (e.g. Beneficiary in the running example - illustrated in Figure 3.28).

Figure 3.28: RT transformation example.

3.5.4 Reverse Engineering of the Running Example

To give an overall view of the produced XIS* models and RSLingo's RSL specifications with the XIS-Reverse approach, this section presents the generated models and specifications in a wider way.

To do so, the small but also detailed SS-App will be used (see Section 3.2 for further information). In order to generate the specifications we used the following set of configurations:

- **Input**: selected database access to enhance the output specifications.
- **Output**: selected both XIS-Web and RSLingo's RSL.
- **Transformation Rules Guidance**: selected Status column from SystemDocument to extract its values (Attribute Values Extraction); and activated generalization discovery by the higher number of shared attributes with the minimum of 3 attributes (Generalization discovery).

Figure 3.29 illustrates the produced XIS* Domain view and Listing 3.1 shows an excerpt of the produced RSLingo's RSL dataEntities. The main difference between the produced specifications is that only the XIS* models are capable to represent the aggregation between Beneficiary and Document through XEA2-b.

Figure 3.29: SS-App XIS* Domain view.
Listing 3.1: Excerpt of SS-App RSLingo’s RSL dataEntities.

dataEntity e_BENEFICIARY : Principal : Simple [
    name "Beneficiary"
    isA e_PERSON
    attribute IdNumber : Integer [name "Idnumber" NotNull]
    primaryKey (IdNumber)
    foreignKey e_PERSON(IdNumber)
    description "has 500 records (by xis-reverse in 2017/04/29)"
]
dataEntity e_Superclass_DOCUMENT_SYSTEMDOCUMENT : Principal : Simple [
    name "Superclass Document Systemdocument"
    attribute Path : String [name "Path"]
    attribute Description : String [name "Description"]
    attribute Notes : String [name "Notes"]
    description "has 0 records (by xis-reverse in 2017/04/29)"
]
dataEntity e_DOCUMENT : Secondary : Simple [
    name "Document"
    isA e_Superclass_DOCUMENT_SYSTEMDOCUMENT
    attribute Id: Integer [name "Id" NotNull]
    attribute BenIdNumber : Integer [name "Benidnumber" NotNull]
    primaryKey (Id)
    foreignKey e_BENEFICIARY(BenIdNumber)
    description "has 750 records (by xis-reverse in 2017/04/29)"
]
dataEntity e_SYSTEMDOCUMENT : Principal : Simple [
    name "Systemdocument"
    isA e_Superclass_DOCUMENT_SYSTEMDOCUMENT
    attribute Id: Integer [name "Id" NotNull]
    attribute Status : String [name "Status" values "Accepted; Declined; Expired"]
    primaryKey (Id)
    description "has 200 records (by xis-reverse in 2017/04/29)"
]

Figure 3.30 illustrates the extracted XisBusinessEntities and their relationships with the XisEntities from the Domain View, namely Master, Reference and Detail, using the XBE-k rules. Listing 3.2 shows an excerpt of the equivalent entities, as dataEntityViews, through the RBEV rule.

Listing 3.2: Excerpt of SS-App RSLingo’s RSL dataEntityViews.

dataEntityView ev_BENEFICIARY : Complex [
    name "Beneficiary (eview)"
    master e_BENEFICIARY
    detail e_DOCUMENT
    reference e_DEPENDENT
    reference e_USERACCOUNT
    description "Beneficiary (eview)"
]
dataEntityView ev_SYSTEMDOCUMENT : VerySimple [
    name "Systemdocument (eview)"
    master e_SYSTEMDOCUMENT
    description "Systemdocument (eview)"
]

Regarding Use Cases, User Stories, Functional Requirements, Goals, Business Processes and Terms, those won’t be illustrated since they are not as distinct and detailed as the domain and business entities.
3.6 Tools Support

Since Sparx Systems [EA] is the tool used to design information systems in the other XIS* projects, in order to reduce dependencies for future work and learning curve for the user, we also decided to implement the XIS-Reverse tool, as an add-in implemented in C#, on top of [EA].

A view of the technologies behind the XIS-Reverse approach implementation is illustrated in Figure 3.31. This view aggregates the main technologies used in the main stages, which will be further explained in the following sub sections. As illustrated, both reverse engineering configuration and execution stages (Figure 3.1) are supported by the XIS-Reverse tool.

The XIS-Reverse tool and an User guide are available on GitHub [2].

3.6.1 Data Schema Extraction

Following the aforementioned process of Figure 3.31 during the Data Schema Extraction stage, in order to extract the application data model our approach depends on the EA native capability [3] to reverse engineer a database model through an ODBC connection. Moreover, EA supports several ODBC data sources [4] thus the only pre-requisite is to have the needed ODBC installed in the user machine.

Furthermore, in the end of this stage, the application data model will be aggregated to the repository of the [EA] project being used.

3.6.2 Reverse Engineering Configuration and Execution

In order to develop the XIS-Reverse tool we used a set of provided technologies by the EA, namely the Automation Interface and the Model Driven Generation. The Automation Interface allows to access and apply operations over the EA’s repository, which contains all its packages (including the extracted Application data model) and its elements. On the other hand, the Model Driven Generation technology was also required to use the already defined XIS* languages (e.g. XIS-Web) as UML profiles, including their compliant toolboxes, diagram types, model patterns and model templates. Thus, with this two technologies it is possible not only to access the application data model, but also to create/edit/delete packages, diagrams and elements in several XIS* languages, enabling to add the Application XIS-Web views to the EA’s repository.

As an EA plug-in, the XIS-Reverse tool is accessible through the XIS-Reverse sub-menu in the EA Extensions menu, which leads the user to the main configuration panel of the tool (Figure 3.4). After the user configuration (explained in Section 3.4), the Reverse Engineering Execution takes place. This last stage takes as input the Application data model (Initial Model), the user configuration and it also uses a database access or a profiler log file.

Currently, this tool only supports SQL Server connections to perform database access.

Regarding the profiler log file, it must have Stored Procedures events recorded while the legacy application was used. During initialization, those events will be parsed, extracting every table name that was used. In order to do it, the set of available tools to parse SQL statements was analysed, and we choose the one witch was more complete (in terms of recognized statements) and free to use. This tool is named JSqlParser and it is a Java SQL statement parser and is capable to extract names of tables from SQL statements. Since JSqlParser was currently distributed as a JAR file, we had to converted it into a C# assembly (DLL) using IKVM, enabling that all the development was done in C#.

---

7 https://github.com/JSQLParser/JSqlParser - version 0.9.6 (Accessed on May 2017)
To use this SQL parser some preprocessing was applied to the SQL statements in order to increase its results due to some syntax limitations. Those limitations were mainly in terms of SQL Server operations that were not covered by the latest version of this parser, at the time it was used, which would lead to errors while running the parser.

Moreover, currently the XIS-Reverse tool only supports profiler log files generated from the Microsoft SQL Server Management Studio Profiler Tool. A user guide with the instructions to produce the aforementioned file, using the Microsoft SQL Server 2014 Management Studio, is also available on GitHub9.

Regarding the extracted knowledge model, the produced RSLingo’s RSL specifications are generated using the Visual Studio runtime text templates (T4 Text Templates 10), a template based text generator, which allows to generate text given a template. These templates are composed by a mixture of static text, and fragments of program code.

Additionally, although this tool was mainly directed to produce XIS-Web and RSLingo’s RSL specifications, it can be extended to support different representations.

Chapter 4

Evaluation

This chapter presents and discusses the evaluation of the XIS-Reverse approach. First, Section 4.1 presents and analyses the main results of using our tool with two real-world legacy applications. Second, Section 4.2 shows an assessment of the interoperability between the XIS-Reverse and a XIS* framework. Third, Section 4.3 presents a discussion about the related work. Finally, Section 4.4 summarizes the key aspects of the evaluation and its results.

4.1 Real-World Case Studies

In this section, two case studies are introduced and used to assess our approach.

Both applications were supported by SQL Server databases and our experiments only considered database access in order to enhance the output specifications since it is harder to generate a profiler log file that represents well the normal usage of such applications.

To assess the overall results in each Case Study we divided this evaluation into three levels of configuration scenarios: Without configuration, Blind configuration and Semi-guided configuration. Within each scenario we extracted: number of XisEntities (including explicit and implicit superclasses); number of XisAssociations (also including Aggregations and Many-to-many associations); number of Aggregations; number of Many-to-many associations; number of explicit and implicit subclasses and superclasses; number of XisBusinessEntities and each of their types of associations and number of XisEntityUseCases.

Moreover, we defined some heuristics to evaluate the obtained results in a deeper way, namely in terms of aggregation associations and implicit generalizations. However, the following heuristics will not be applied to the Case Study B due to privacy constraints.

Regarding aggregations, we defined two rules. The first one requires to have an updated domain model in the available application requirements, in which entity associations are classified (e.g. one-to-one or aggregation associations). The second one, requires to have every entity manually classified as main entity (e.g. relevant entity in the domain), configuration entity (e.g. “kind of” entity) or association entity (e.g. entity which main purpose is to link two or more entities).

Rule-1: Number of associations well classified in terms of aggregations. We assess this rule by applying the concepts of a confusion matrix to the results (after the experiment), thus we count the number of: (1) actual aggregations that were correctly classified as aggregations (true positive); (2) non-aggregations that were incorrectly classified as aggregations (false positive); (3) aggregations that were incorrectly marked as non-aggregations (false negative); (4) all the remaining associations correctly classified as non-aggregations (true negative).
Rule-2: Number of configuration entities that do not aggregate main nor association entities. We assess this rule by counting how many of those did and did not aggregate main or association entities (after the experiment).

Regarding generalizations, we want to extract implicit generalizations which maximize both the number of subclasses found (variable $x$) and the number of inherited attributes (variable $y$), based on the following function:

$$Reis(x, y) = 0.5x + 0.5y$$  \hfill (4.1)

To better explain the Reis function and its variables, a simple domain model illustrated in Figure 4.1 will be used.

Variable $x$ can be determined by the number of subclasses found (after the experiment), divided by the maximum number of subclasses that could be found (number of entities without generalization and with at least 1 attribute (before the experiment), such as A, B, C, F and G (5) in Figure 4.1).

Taking into account that generalizations with the exact number of two subclasses will maximize the number of superclasses that can be found, and thus, maximize also the number of inherited attributes:

Variable $y$ can be determined by the sum of all the superclass attributes found (after the experiment), divided by the sum of the maximum number of attributes that could be inherited (the sum of the maximum number of attributes every pair of entities can share (before the experiment), taking into account all pairs of entities that can be grouped, by the descending order of attributes number, such as 3 in Figure 4.1 since pairs A-B share at most 2 attributes and C-F share at most 1 attribute, for example).

![Figure 4.1: Support example to explain the Reis function.](image)

4.1.1 Case Study A: ProjectIT-Enterprise

The ProjectIT\textsuperscript{1}\cite{40,41} initiative aggregates several research topics, such as software engineering and software development. The main goal behind this initiative is to provide a complete software development workbench, with support for project management, requirements engineering, analysis, design and code generation features. Moreover, within this initiative a collaborative tool with Web interface was developed. This tool, called ProjectIT-Enterprise\textsuperscript{5,42}, provides a mechanism to process definition, collaborative support for team work, emphasizing project management, project-process alignment, workflows and documents management.

Although, the ProjectIT-Enterprise was mainly used and tested in an academic and research scope, it is a mature one, with well defined concepts and requirements. Since we had the chance to use it, we decided to perform an exhaustive experiment to assess the XIS-Reverse.

\textsuperscript{1}http://isg.inesc-id.pt/alb/ProjectIT (Accessed on June 2017)
Regarding the aggregation rules, since we had access to the domain model specifications of this application (Figure 4.2), and it was granted that there were no significant updates in the application database since this specification was defined, we used that specification to evaluate against our experiment (required for Rule-1). With that, we established a mapping between every database table and the corresponding entity in the domain model (Table 4.1), and then with some domain knowledge, we classified those entities/tables as main entities, configuration entities and association entities (required for Rule-2). This mapping and classification will be used during the evaluation to compare the extracted specification (using the XIS-Reverse) with the aforementioned domain model, showed in Figure 4.2.

Moreover, we identified the direct relationships between the main entities in the domain, defined as foreign key constraints in the application database (checked symbols in Figure 4.2).

Taking into account Figure 4.2, from the total of 8 direct relationships identified, 7 were aggregations (relaxing the composition definition) and 1 was an one-to-one relationship (not an aggregation).

Table 4.2 shows the results of applying the aggregation rules using the XIS-Reverse. These results are analysed below, in each of the configuration scenarios.

Furthermore, to support each scenario, Table 4.3 presents the overall picture of the extracted elements.
Table 4.1: Case Study A - Equivalence between application database and domain model.

<table>
<thead>
<tr>
<th>Application Database Table</th>
<th>Domain Model Entity</th>
<th>Manual Classification</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActivityEffort</td>
<td>-</td>
<td>Main</td>
</tr>
<tr>
<td>ActivityMembers</td>
<td>-</td>
<td>Association</td>
</tr>
<tr>
<td>ActivityProcess</td>
<td>Activity Process</td>
<td>Main</td>
</tr>
<tr>
<td>ActivityProcessSkills</td>
<td>-</td>
<td>Association</td>
</tr>
<tr>
<td>ActivityProject</td>
<td>Activity Project</td>
<td>Main</td>
</tr>
<tr>
<td>ActivityProjectTemplate</td>
<td>-</td>
<td>Main</td>
</tr>
<tr>
<td>ActivityProjectTemplateSkills</td>
<td>-</td>
<td>Association</td>
</tr>
<tr>
<td>Country</td>
<td>-</td>
<td>Configuration</td>
</tr>
<tr>
<td>DisciplineProcess</td>
<td>Discipline Process</td>
<td>Main</td>
</tr>
<tr>
<td>DisciplineProjectTemplate</td>
<td>-</td>
<td>Main</td>
</tr>
<tr>
<td>DocumentProcess</td>
<td>Document Process</td>
<td>Main</td>
</tr>
<tr>
<td>DocumentProject</td>
<td>Document Project</td>
<td>Main</td>
</tr>
<tr>
<td>DocumentProjectTemplate</td>
<td>-</td>
<td>Main</td>
</tr>
<tr>
<td>PrivacyLevel</td>
<td>-</td>
<td>Configuration</td>
</tr>
<tr>
<td>Process</td>
<td>Process</td>
<td>Main</td>
</tr>
<tr>
<td>ProcessDefinition</td>
<td>-</td>
<td>Main</td>
</tr>
<tr>
<td>Project</td>
<td>Project</td>
<td>Main</td>
</tr>
<tr>
<td>ProjectMembers</td>
<td>-</td>
<td>Association</td>
</tr>
<tr>
<td>RoleActivities</td>
<td>-</td>
<td>Association</td>
</tr>
<tr>
<td>RoleProcess</td>
<td>Role Process</td>
<td>Main</td>
</tr>
<tr>
<td>RoleSkills</td>
<td>-</td>
<td>Association</td>
</tr>
<tr>
<td>Skill</td>
<td>Skill</td>
<td>Configuration</td>
</tr>
<tr>
<td>State</td>
<td>-</td>
<td>Configuration</td>
</tr>
<tr>
<td>TimePeriod</td>
<td>-</td>
<td>Configuration</td>
</tr>
<tr>
<td>UserProfile</td>
<td>Person</td>
<td>Main</td>
</tr>
<tr>
<td>UserSkills</td>
<td>-</td>
<td>Association</td>
</tr>
<tr>
<td>WorkPackage</td>
<td>Work Package Project</td>
<td>Main</td>
</tr>
<tr>
<td>WorkPackageMembers</td>
<td>-</td>
<td>Association</td>
</tr>
<tr>
<td>WorkPeriodProcess</td>
<td>Work Period Process</td>
<td>Main</td>
</tr>
<tr>
<td>WorkPeriodProject</td>
<td>Work Period Project</td>
<td>Main</td>
</tr>
<tr>
<td>WorkPeriodProjectTemplate</td>
<td>-</td>
<td>Main</td>
</tr>
<tr>
<td>WorkProductProcess</td>
<td>Work Product Process</td>
<td>Main</td>
</tr>
<tr>
<td>WorkProductProject</td>
<td>Work Product Project</td>
<td>Main</td>
</tr>
<tr>
<td>WorkProductProjectTemplate</td>
<td>-</td>
<td>Main</td>
</tr>
</tbody>
</table>

### 4.1.1.1 Scenario A: Extraction Without configuration

In this scenario, the unique configuration performed is the connection setting with the database. With that, we aim to extract and analyse the simplest scenario used with the XIS-Reverse. Then, from these results, compare with the scenarios that use complex configurations (Scenario-B and Scenario-C).

The first execution of this configuration scenario allowed to identify a problem in our approach, namely the identification of many-to-many associations. This problem occurred due to the generic definition of such heuristic that did not take into account composite primary keys. That problem occurred in cases that, an entity had at least 2 primary keys (which only one of them was a foreign key), there was only one attribute and that attribute had a foreign key constraint. Taking that into account, we redefined this heuristic as it is explained in the Section 3.5.2.1, rule XE-1.

After updating that heuristic, a new execution was performed in which 34 XisEntities were found with 45 XisEntityAssociations established, from which 42 were classified as aggregations. Moreover, regarding our aggregation evaluation Rule-1, from the 8 direct relationships, 6 aggregations were well identified. However, 1 aggregation was misinterpreted as a simple XisEntityAssociation and the one-
<table>
<thead>
<tr>
<th>Results</th>
<th>Rule-1</th>
<th>Rule-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scenarios</td>
<td>True Positive</td>
<td>False Positive</td>
</tr>
<tr>
<td><strong>A:</strong> Without Configs.</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td><strong>B:</strong> M = 20</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>B:</strong> M = 10</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td><strong>B:</strong> M = 5</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td><strong>C:</strong> Semi-guided</td>
<td>6</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 4.3: Case Study A - Overall results of the reverse engineering.

<table>
<thead>
<tr>
<th>Element / Scenario</th>
<th>A</th>
<th>B M=20</th>
<th>B M=10</th>
<th>B M=5</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>XisEntities</td>
<td>34</td>
<td>34</td>
<td>34</td>
<td>34</td>
<td>38</td>
</tr>
<tr>
<td>XisEntity Associations</td>
<td>45</td>
<td>45</td>
<td>45</td>
<td>45</td>
<td>45</td>
</tr>
<tr>
<td>XisEntity Associations (Aggregations)</td>
<td>42</td>
<td>6</td>
<td>13</td>
<td>32</td>
<td>38</td>
</tr>
<tr>
<td>XisEntity Associations (Many-to-many)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Explicit subclasses</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Explicit superclasses</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Implicit subclasses</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>Implicit superclasses</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>XisBusinessEntities</td>
<td>14</td>
<td>31</td>
<td>28</td>
<td>23</td>
<td>15</td>
</tr>
<tr>
<td>XisBusinessEntities Master Associations</td>
<td>14</td>
<td>31</td>
<td>28</td>
<td>23</td>
<td>15</td>
</tr>
<tr>
<td>XisBusinessEntities Detail Associations</td>
<td>42</td>
<td>6</td>
<td>13</td>
<td>32</td>
<td>38</td>
</tr>
<tr>
<td>XisBusinessEntities Reference Associations</td>
<td>12</td>
<td>36</td>
<td>27</td>
<td>21</td>
<td>13</td>
</tr>
<tr>
<td>XisEntityUseCases</td>
<td>14</td>
<td>31</td>
<td>28</td>
<td>23</td>
<td>15</td>
</tr>
</tbody>
</table>

to-one relationship was wrongly classified as an aggregation. The first problem occurred due to the difference of rows’ number of each entity in the database, and since that difference goes against the rule XEA-2-b and there is no available configuration able to correct this problem, this type of issue had to be solved manually. On the other hand, the second problem was due to the absent of a Unique Index property in that foreign key, which was probably forgotten or relaxed.

Moreover, following the Rule-2, from all the configuration entities, only the Skill entity had aggregation associations with main or association entities. This can be solved by classifying this entity as Simple Principal Entity during the configuration stage.

Furthermore, there were no many-to-many associations identified, neither explicit generalizations. Thus, from this configuration level results, to improve the quality of the obtained specifications, the main configurations that make sense to explore, in the following configuration scenarios, are the identification of Simple Principal Entities and Generalization discovery.

4.1.1.2 Scenario B: Extraction with Blind configuration

After the previous configuration results, the goal of the blind configurations is to improve the results of the defined evaluation heuristics executed in a blind way. Thus, in this section we cover two distinct situations, namely aggregations and then generalizations.
Aggregation

This situation is focused in the assessment of the obtained results using different Simple Principal Entity configurations in a blind way, following the defined evaluation heuristics.

Regarding the Simple Principal Entities selection menu, we started by using the default value (20) for the maximum number of rows that a table can have in the database. And then selected all those entities. Moreover, a similar process is used in the following situations but with different numbers. To simplify we define this number as M.

- **M = 20** - With this configuration, 6 aggregations were identified. However, regarding the Rule-1, none of the 6 aggregations that were well classified in the Scenario-A was now correctly identified, thus the number of False Negatives increased to 7. Moreover, the one-to-one association, wrongly identified as aggregation in the Scenario-A, was not classified as an aggregation this time. Regarding the Rule-2, none of the configuration entities aggregated a main entity or an association entity. Due to the low number of identified aggregations, it only makes sense to test again with a lower M number.

- **M = 10** - With this configuration, 13 aggregations were found. In terms of the Rule-1, 2 aggregations were correctly identified, thus the number of False Negatives decreased to 5. This time, the one-to-one association was wrongly classified as an aggregation, since this time DocumentProject entity was not selected as Simple Principal Entity (number of False Positives is 1). Moreover, following the Rule-2, the result was the same of the previous test. Since with this new value for M, the number of identified aggregations is still less than half of the number of entities, we will decrease M once again.

- **M = 5** - With this configuration, the number of aggregations increased to 32. Following the Rule-1, the number of correctly identified aggregations increased by 1, thus there were still 4 aggregations wrongly identified as simple XisEntityAssociations (False Negatives). The number of False Positives remained the same. Regarding the Rule-2, this time 1 entity (Skill) had aggregation associations with main or association entities, likewise in the scenario without configurations. Moreover, since the number of entities with 5 or less attributes (M = 5) is only one (Process), it does not make sense try lower M values, because the results that we would get would be the same as we got in the scenario without configurations.

With these results, we can say that without domain knowledge about the ProjectIT-Enterprise, in terms of aggregations, we would get the best result without using the Simple Principal Entities configuration in a blind way. However, we think that the tests with this kind of configuration did not show interesting results, mainly due to the reduced application usage, which was reflected in the low amount of main entities rows, such as project and process. And thus, since our heuristic assumes that the number of rows of aggregated entities is greater or equal to the number of rows of the entities that aggregate them, and that the number of rows of Simple Principal Entities is usually a lot smaller, compared with the others, we conclude that this configuration did not benefit the obtained results in this case study.
Generalization

This situation is focused in the identification of implicit generalizations and the assessment of the obtained results. In order to perform this evaluation we will activate Generalization discovery and use its configuration points. Moreover, since our generalization evaluation heuristic tries to maximize both the number of subclasses and the number of inherited attributes, we will use our two options to aggregate entities every iteration, i.e. every time we change other configuration points.

Table 4.4, summarises the main results during this evaluation, taking into account, for each configuration, the number of generated subclasses (a), the sum of inherited attributes (b) and the application of such values in our evaluation function. Moreover, from our Scenario-A we could extract that the maximum number of subclasses that can be found is 31, and the sum of the maximum number of attributes that can be inherited is 51, thus we can rewrite our function as:

\[
Reis'(a,b) = 0.5\left(\frac{a}{31}\right) + 0.5\left(\frac{b}{51}\right)
\]

Table 4.4: Case Study A - Scenario B - Evaluation of implicit generalization results.

<table>
<thead>
<tr>
<th>Minimum # of shared attributes</th>
<th>Aggregated by</th>
<th>Ignored names</th>
<th>Subclasses (a)</th>
<th>Inherited attributes (b)</th>
<th>Reis'</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 attributes</td>
<td>-</td>
<td>4</td>
<td>12</td>
<td>0.18</td>
<td></td>
</tr>
<tr>
<td>5 entities</td>
<td>-</td>
<td>4</td>
<td>12</td>
<td>0.18</td>
<td></td>
</tr>
<tr>
<td>4 attributes</td>
<td>-</td>
<td>4</td>
<td>12</td>
<td>0.18</td>
<td></td>
</tr>
<tr>
<td>4 entities</td>
<td>-</td>
<td>5</td>
<td>10</td>
<td>0.18</td>
<td></td>
</tr>
<tr>
<td>3 attributes</td>
<td>name,description</td>
<td>8</td>
<td>18</td>
<td>0.31</td>
<td></td>
</tr>
<tr>
<td>3 entities</td>
<td>name,description</td>
<td>10</td>
<td>12</td>
<td>0.28</td>
<td></td>
</tr>
<tr>
<td>3 attributes</td>
<td>name,description</td>
<td>4</td>
<td>8</td>
<td>0.14</td>
<td></td>
</tr>
<tr>
<td>2 entities</td>
<td>name,description</td>
<td>4</td>
<td>8</td>
<td>0.14</td>
<td></td>
</tr>
<tr>
<td>2 attributes</td>
<td>name,description</td>
<td>6</td>
<td>10</td>
<td>0.19</td>
<td></td>
</tr>
<tr>
<td></td>
<td>name,description</td>
<td>8</td>
<td>8</td>
<td>0.21</td>
<td></td>
</tr>
</tbody>
</table>

We started our evaluation by using the default value for the minimum number of shared attributes (5). And then, from the obtained results we decided which configuration should be used in the next iteration.

Overall, from the first iteration we got reasonable results (Reis’ = 0.18), namely 4 subclasses and 12 inherited attributes were found. Then, due to lower number of subclasses found, it only made sense to iterate with lower numbers for the minimum number of shared attributes. We, reduced to 4, and we got slightly similar results, so we decided to reduce again to 3, from which we got better results (Reis’ = 0.31), namely 8 subclasses found and 18 inherited attributes, by aggregating our entities by the higher number of shared attributes. Then, we reduced to 2 and a combinational explosion happened, thus no results were generated. However, we analysed the inherited attributes from the last configuration successfully used, and we noticed that two attributes (“name” and “description”) were inherited by almost every superclass, thus we decided to do more iterations but this time ignoring such attributes. During these iteration, the best result we got (Reis’ = 0.21) was slightly higher than the firsts we got, but anyway closer to our best one.

4.1.1.3 Scenario C: Extraction with Semi-guided configuration

The main goal of this scenario is to try to improve the results obtained from the two previous configuration scenarios, by introducing some domain knowledge in the configuration parameters.

In terms of aggregations, we just identified the configuration entities as Simple Principal Entities, from the previously generated Table 4.1. And as expected, with that configuration, we got the best
results of all the configuration scenarios used, namely by improving results of the Rule-2 . As stated before, no matter how much domain knowledge the user has, the False Negative and the False Positive problems identified can only be solved manually, since none of the available configurations can correct such issues.

Regarding generalization discovery, even with a good domain knowledge (e.g. average number of entity attributes), the user would always need to perform a similar approach as the blind one, to get good results in terms of implicit generalizations.

Thus, to get the best results overall, we had to select the known configuration entities as Simple Principal Entities and activate Generalization Discovery with at least 3 shared attributes, ordered by the higher number of shared attributes.

### 4.1.2 Case Study B: Social Security application

During this research period, we got the chance to work in the TT-MDD-Mindbury/2015 project, which main goal was to develop a real-world Social Security application on top of an existent legacy one. This was a good opportunity to apply our approach to a real-world application, already with some usage by its users. Due to privacy concerns for the client and the development company, details about this application must be kept in secret. Due to these constraints, the analysis and evaluation of this case study is not so much detailed as the Case Study A, only highlighting the most relevant results.

Moreover, since this project lasted for 12 months, we got a deep domain knowledge of this application. And such knowledge is used to guide the reverse engineering process for this application, in order to reduce the number of iterations in this process. We evaluate this application based on this knowledge, since the available documentation of this application was outdated.

In terms of its database, this application was designed with 187 tables, and most of its tables shared a common set of attributes, namely timestamps for those entities, etc. Thus, using the XIS-Reverse approach to find implicit generalizations (XEI rule), can easily lead to a combinational explosion.

This evaluation only stresses 3 different configuration scenarios, namely a scenario without configuration; one blind configuration scenario, selecting every entity with 20 or less rows as Simple Principal Entity; and a semi-guided configuration scenario, based on the selection of Simple Principal Entities selection and on Generalization discovery.

Table 4.5 shows the overall results obtained from these three configuration scenarios based on previously defined metrics.

- **Scenario A - Extraction Without configuration:** We found that many entities, that we could classify as Simple Principal Entities, were wrongly aggregating main entities and no explicit generalizations were found. Thus, the obtained results were not accurate, and in general were wrong in terms of aggregations.

- **Scenario B - Extraction with Blind configuration:** This scenario was executed, in order to evaluate if a user could get better results by selecting Simple Principal Entities in a blind way. In this experiment, every entity with 20 or less rows in the database was selected as Simple Principal Entity (132 entities). With our domain knowledge we were able to identify that from those 132 entities, 14 were wrongly selected and 12 Simple Principal Entities were not selected since they had more than 20 rows (e.g. country). However, the quality of the results increased drastically, since overall most of the Simple Principal Entities (around 91%) were well identified using this approach.
- **Scenario C - Extraction with Semi-guided configuration**: With the domain knowledge, we can improve this results even more, by identifying every Simple Principal Entity (with the help of the available filters, due to the large number of entities to select). Moreover, with our knowledge we know the average number of attributes per entity and that some entities shared some properties, thus we can reduce the number of iterations to obtain results in terms of implicit generalizations. With that, following a semi-guided configuration we got better results, not only in terms of aggregations that made more sense to exist, but also in terms of implicit generalizations found.

<table>
<thead>
<tr>
<th>Element / Scenario</th>
<th>A</th>
<th>B</th>
<th>M=20</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>XisEntities</td>
<td>168</td>
<td>168</td>
<td>176</td>
<td></td>
</tr>
<tr>
<td>XisEntityAssociations</td>
<td>224</td>
<td>224</td>
<td>224</td>
<td></td>
</tr>
<tr>
<td>XisEntityAssociations (Aggregations)</td>
<td>126</td>
<td>38</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td>XisEntityAssociations (Many-to-many)</td>
<td>19</td>
<td>19</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>Explicit subclasses</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Explicit superclasses</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Implicit subclasses</td>
<td>0</td>
<td>0</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>Implicit superclasses</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>XisBusinessEntities</td>
<td>140</td>
<td>156</td>
<td>161</td>
<td></td>
</tr>
<tr>
<td>XisBusinessEntities Master Associations</td>
<td>140</td>
<td>156</td>
<td>161</td>
<td></td>
</tr>
<tr>
<td>XisBusinessEntities Detail Associations</td>
<td>82</td>
<td>23</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>XisBusinessEntities Reference Associations</td>
<td>124</td>
<td>190</td>
<td>215</td>
<td></td>
</tr>
<tr>
<td>XisEntityUseCases</td>
<td>140</td>
<td>156</td>
<td>161</td>
<td></td>
</tr>
</tbody>
</table>

Furthermore, during this research, this case study was used several times to support the evaluation of the development iterations. One of the issues that we discovered by using this complex system, besides the time to reverse engineering, was the combinational explosion that a Generalization discovery could easily trigger while comparing every entity and their attributes.

This problem can happen when there is a large amount of entities which share identical attributes, leading to a large set of entities to be compared, which can exponentially increase the amount of time and memory required to find generalizations. During our experiments, the aforementioned combinational explosion was usually stopped due to memory constraints of the EA application (usual Windows application constraints), which led to a crash of the application. Thus, the only solution to execute this feature in large domains (like this one), is to ignore some of the most used attribute names.

### 4.2 Interoperability with XIS* frameworks

In this section, an analysis of the XIS-Reverse interoperability with a XIS* framework is performed, in order to access how well both tools can work together. Figure 4.3 illustrates the main goal of this evaluation, which is to successfully use XIS-Reverse to generate XIS* models given a Legacy Application, and use those models to generate a New Application using a XIS* framework.

![Figure 4.3: Interoperability with XIS* frameworks.](image)
Once again, we used the XIS-Web technology [4], since it is the most recent XIS* technology. In a nutshell, this framework supports the XIS-Web forward engineering process, which is applied to XIS-Web models. This is accomplished by following three steps: (1) Models validation; (2) M2M transformation; and (3) M2T transformation. Step-1 uses a set of rules (built-in in the framework) to validate the XIS-Web models. Then, step-2 generates the user-interface views, namely the Interaction Space and Navigation Space views. Finally, step-3 generates the target application’s source code.

To perform this evaluation we used the most recent version of the XIS-Web framework [2].

Firstly, in order to validate the XIS-Web models, the Transformations to XIS models (Section 3.5.2) had to be extended to take into account the specific properties of the XIS-Web language. Namely, in the XEUC rule, each XisEntityUseCase will also have the following XIS-Web tagged values:

- If the corresponding XisEntity was selected as Simple Principal Entity, that XisEntityUseCase’s type will be EntityConfiguration, otherwise will be EntityManagement;
- If the corresponding XisBusinessEntity is the Business Entity with the most relationships, that XisEntityUseCase will have its isStartingUseCase tagged value set to true, or false otherwise.

Then, using the XIS-Web specifications from the ProjectIT-Enterprise (Section 4.1.1), that were extracted with the XIS-Reverse tool, we did our evaluation step by step:

1. **Models validation** - In this step, we got one relevant error in the first validation, namely that XisEntities had to have at least 1 XisEntityAttribute. Since this error can easily occur, for example in subclasses, we think that the only viable solution for this mismatch is to update the XIS-Web framework in order to omit this rule. Furthermore, to bypass this problem, we added a fictitious XisEntityAttribute to those XisEntities and the second validation was successful.

2. **M2M transformation** - Then, we executed the model generation, which successfully produced the contents of the Interaction Space and the Navigation Space views.

3. **M2T transformation** - In this step, we also got an error during the code generation. The error stated that the same file was generated several times, i.e. some files were overwritten. This problem occurred due to the limitation of the XIS-Web framework to generate Interaction Space and the Navigation Space views for XisEntities with many-to-many relationships. Then, once again, the most logic solution to this problem is to extend the XIS-Web framework in order to support that scenario. However, to bypass this problem, we removed each many-to-many relationship and, for each one of them, we added a XisEntity linked to each of those entities through an one-to-many relationship and after rerunning the M2M transformation the M2T transformation was successful.

4.3 Related Work Discussion

In this section, a further analysis considering the introduced approaches in Section 2.4 (namely in the area of Data Schema Extraction and Reverse Engineering), is used to explain and discuss the main differences and influences between those proposals and the XIS-Reverse approach.

4.3.1 Data Schema Extraction

To sum up the related work introduced in Section 2.4.1, given that a set of existing tools already support data schema extraction from several sources, without additional specification of transformations, we took advantage of such tools, more precisely of EA. The main aspects of the related work previously analysed are shown in Table 4.6, namely: input format, output format, the existence of data schema extractors (i.e. mechanisms to extract the database schema from the source application) and if it extracts all table properties (namely constraints (NOT NULL, UNIQUE, PRIMARY, FOREIGN, CHECK, DEFAULT and INDEX) and column properties (name and type)). The last row categorises our approach.

<table>
<thead>
<tr>
<th>Approach</th>
<th>Input Format</th>
<th>Output Format</th>
<th>Schema Extractors</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gra2Mol [26]</td>
<td>any textual artefact</td>
<td>any model</td>
<td>no</td>
<td>n.a.</td>
</tr>
<tr>
<td>MoDisco [8]</td>
<td>several sources</td>
<td>any model</td>
<td>no</td>
<td>n.a.</td>
</tr>
<tr>
<td>Schemol [27]</td>
<td>data stored into DB</td>
<td>any model</td>
<td>no</td>
<td>n.a.</td>
</tr>
<tr>
<td>DB-MAIN [28]</td>
<td>several sources</td>
<td>GER</td>
<td>yes (e.g. ODBC)</td>
<td>yes</td>
</tr>
<tr>
<td>SQL2XMI [29]</td>
<td>SQL DDL schema</td>
<td>UML in XMI</td>
<td>yes (only MySQL)</td>
<td>no</td>
</tr>
<tr>
<td><strong>EA</strong> (our approach)</td>
<td>several sources</td>
<td>UML</td>
<td>yes (e.g. ODBC)</td>
<td>yes</td>
</tr>
</tbody>
</table>

Table 4.6: Classification of some related works on data schema extraction.

4.3.2 Reverse Engineering

As previously presented in Section 2.4.2, Reverse Engineering approaches started by focusing in a single input artefact. However, recent works have proven that is possible to enhance the obtain results by using more than one artefact (through schema, screen, data and program analysis, introduced in Section 2.4.2). From those, our approach uses schema, data and program (Stored Procedures) analysis. Regarding schema analysis, the specification of a manual process in [30] was adapted in our approach to semi-automatically and automatically identify generalizations and many-to-many associations, respectively. In terms of data analysis, our approach uses this technique in a similar way as presented in Section 2.4.2 however it is applied to classify associations. And finally, our approach uses static program analysis in the profiler log file, also aiming to classify associations.

Moreover, taking into account the set of approaches concerning the application of MDE, our approach could be implemented extending MoDisco framework [8]. However, that would require the definition from scratch of all the three stages (discoverers, transformations and generators) needed to produce the desired artefacts. Regarding Relational Web [39], this approach defined foreign key’s semantic extraction techniques, to identify inheritance relationships and associations, that were adapted and extended by our approach, in order to identify aggregations. Furthermore, DB-MAIN [28] tool requires the user to apply all the needed transformations, thus the degree of automation achieved in our approach is higher. And although, DB-MAIN supports generalization representation, once again it must be identified by the user.

Regarding to the main contributions of this research, we do not find any other approach that specializes associations (distinguishing between associations and aggregations), nor any approach that allows to extract values and their representation into the target conceptual schema.
Finally, since a reverse engineering approach can be classified in several ways, taking into account the most relevant aspects for our research, Table 4.7 gives a summary of the previously analysed research works and compares them with the XIS-Reverse approach.

These aspects include for each approach: analysis type (data, schema, screen and program); output format(s); which XIS-Reverse main contributions they support (A - Aggregations Extraction, G - Implicit Generalization Extraction and V - Values Extraction); existence of tools automating the approach; automation level of those tools (regarding the reverse engineering stage); and if those tools can be extended.

<table>
<thead>
<tr>
<th>Approach</th>
<th>Analysis</th>
<th>Output</th>
<th>Contributions</th>
<th>Tools</th>
<th>Auto.</th>
<th>Extension</th>
</tr>
</thead>
<tbody>
<tr>
<td>[30]</td>
<td>schema</td>
<td>OMT class diagram</td>
<td>G</td>
<td>no</td>
<td>n.a</td>
<td>n.a.</td>
</tr>
<tr>
<td>[31]</td>
<td>data</td>
<td>Extended ER</td>
<td>n.a.</td>
<td>no</td>
<td>n.a</td>
<td>n.a.</td>
</tr>
<tr>
<td>RAINBOW [33]</td>
<td>screen</td>
<td>ER model</td>
<td>n.a.</td>
<td>yes</td>
<td>n.a.</td>
<td>n.a.</td>
</tr>
<tr>
<td>[34]</td>
<td>program (static)</td>
<td>Extended ER</td>
<td>n.a.</td>
<td>no</td>
<td>n.a.</td>
<td>n.a.</td>
</tr>
<tr>
<td>[35]</td>
<td>program (static)</td>
<td>Object-Oriented class diagram</td>
<td>n.a.</td>
<td>no</td>
<td>n.a.</td>
<td>n.a.</td>
</tr>
<tr>
<td>[36]</td>
<td>program (dynamic)</td>
<td>n.a.</td>
<td>n.a.</td>
<td>yes</td>
<td>n.a.</td>
<td>n.a.</td>
</tr>
<tr>
<td>[37]</td>
<td>program (dynamic)</td>
<td>n.a.</td>
<td>n.a.</td>
<td>yes</td>
<td>n.a.</td>
<td>n.a.</td>
</tr>
<tr>
<td>Modisco [8]</td>
<td>schema and program (static)</td>
<td>KDM or UML</td>
<td>n.a.</td>
<td>yes</td>
<td>Auto.</td>
<td>yes</td>
</tr>
<tr>
<td>Relational Web</td>
<td>schema</td>
<td>UML</td>
<td>n.a.</td>
<td>yes</td>
<td>Auto.</td>
<td>yes</td>
</tr>
<tr>
<td>DB-MAIN [28]</td>
<td>schema and program (static)</td>
<td>GER</td>
<td>G</td>
<td>yes</td>
<td>Semi</td>
<td>yes</td>
</tr>
<tr>
<td>XIS-Reverse (our approach)</td>
<td>schema, data and program</td>
<td>XIS* and RSLingo’s RSL</td>
<td>A;G;V</td>
<td>yes</td>
<td>Semi</td>
<td>yes</td>
</tr>
</tbody>
</table>
4.4 Summary

The XIS-Reverse approach to reverse legacy information systems, is able to extract high-level specifications from a legacy application. This approach benefits from a flexible set of configuration points and new features not found in the related work, producing more detailed results, that overall will help the user to get a better understanding of each entity in the domain.

In terms of aggregations detection, at least when using a system with a good amount of data (usage), our heuristics are able to correctly identify those relationships and help the user to get better results by specifying Simple Principal Entities (with and without domain knowledge).

Regarding implicit generalizations discovery, our approach proved that can extract accurate results. However this feature does not benefit much from user domain knowledge, since several experiments with different configurations scenarios must be executed to find the best results. Yet, in the extreme scenario, if the user has a good knowledge of each attribute of each entity, this feature should be disabled, since the identification of generalizations can be easily done manually.

Although not evaluated, we assume that extraction of attribute values, independently of the user domain knowledge level, can benefit the user if values from certain entity attributes can be extracted, giving him a better understanding of the entity role in the domain.

Additionally, in terms of interoperability with the XIS-Web framework, despite the aforementioned errors, which were probably found due to the size and complexity of the case study, overall, the produced XIS-Web specifications with the XIS-Reverse tool, are suitable to be used with the XIS-Web framework.
Chapter 5

Conclusion

Companies have been struggling to maintain their legacy systems due to large costs associated with management, development, legacy infrastructure, legacy licensing, new user requirements and expectations, and so on. Those are usually caused due to increased complexity of applications overtime, which also leads to reduced flexibility of their technologies and outdated or poor documentation. To overcome these problems, software reengineering approaches can be used. In a nutshell, a reengineering approach tries to extract specifications from a legacy system and generate a new application conforming the target system. Regarding, the first stage of this approach, also known as reverse engineering, several approaches have contributed to the development of this stage. However, some of them do not make use of user domain knowledge or provide significant configuration points to improve the specification level of elements, in order to improve understandability of the extracted knowledge models.

In this research work we proposed the XIS-Reverse, a MDRE approach to extract high-level specifications from legacy application artefacts. Moreover, the presented approach is focused on the extraction through schema, data and program analysis techniques. Overall, this approach starts by extracting the application data model from an available database, and from that and a user configuration, the reverse engineering execution takes place, by applying several reverse engineering heuristics to those artefacts, and then generating the extracted knowledge models and specifications.

XIS-Reverse was implement on top of Sparx Systems EA as an EA plug-in. The XIS-Reverse tool first stage (application data model extraction) relies on the native feature of EA to reverse engineer a database schema through an ODBC connection. Then, the following stages (reverse engineering configuration and execution) are supported by the XIS-Reverse tool. The configuration stage is an interface that can be split into 4 different areas:

- **Input** - allowing to specify input artefacts, namely initial model, database name and additional artefacts, such as database access or a profiler log file;
- **Output** - allowing to select additional output representations;
- **Transformation Rules Guidance** - providing configuration points to our 3 main contributions;
- **Appearance** - allowing to improve readability of the produced specifications.

Moreover, although the number of available input technologies and output specifications can be extended, for now our approach is able to produce XIS-Web language models and RSLingo’s RSL specifications from Microsoft SQL Server databases. Finally, the reverse engineering execution stage is composed by M2M transformations that use the application data model and the user configurations to generate such output specifications.
XIS-Reverse has been developed following the Action Research methodology and has been evaluated in order to assess its usefulness, feasibility and adequacy to the purpose of reverse engineering legacy applications. The evaluation process consisted of three aspects:

- **Case study assessment of two real-world applications, focused on the main contributions of the XIS-Reverse.** In terms of aggregations, we used different configuration scenarios, and, as expected, configurations supported by some user knowledge are capable to extract specifications almost identical to the real ones, as presented in Table 4.2. In addition, although several iterations may have to be performed in order to extract good implicit generalizations, this feature is able to identify similarities between entities, saving time for the user in such a complex and error-prone task. It is also to point out that, from using this feature with a real-world case study, we got good results, which are summarized in Table 4.4.

- **Interoperability with XIS* frameworks.** Despite the two errors that can be solved by updating the XIS-Web framework, overall the interoperability between the XIS-Reverse and that framework is good, allowing to automatically produce the source code for a new application, merely from the extracted specifications.

- **Discussion and comparison with the related work.** In terms of data schema extraction, we took advantage of the already available tools, namely the [EA](#) tool, which allows to extract the complete application data model. Regarding the reverse engineering stage, since some of the existent approaches did not make use of user domain knowledge or provide significant configuration points, we decided to develop a new approach.

Furthermore, every goal established in Section 1.4 was achieved during the development of this research work.

### 5.1 Main Contributions

XIS-Reverse provides a suitable solution to reverse engineer legacy information systems, in order to obtain updated and detailed specifications described using a UML-profile representation and/or a Requirements Specification Language. Regarding the main contribution of the reverse engineering approach described in this thesis, we highlight the semi-automatic heuristics that can identify certain relationships between entities, namely implicit generalizations and specialization of associations (aggregations), and also the possibility to extract values from the source database to enrich the target models or specifications, all that combined in order to enhance the understanding of each entity role in the produced models.

This research also provided the following contributions aligned with the research questions originally defined (Section 1.2):

- **RQ.1: How to extract legacy information system’s specifications?**
  Through the use of reverse engineering approaches applied to legacy information system’s artefacts. Namely, by analysing those artefacts in order to extract a good knowledge about the information system.

- **RQ.2: How to apply a MDE approach to reverse engineering a legacy information system?**
  Through the definition of an approach that uses model transformations applied to the input artefacts.
• **RQ.3: Which concepts should be extracted?**

The input artefacts, such as database schemas, user configurations, profiler log files and database access, allow to extract not only explicit properties (e.g. constraints) and relationships (e.g. associations and generalizations), but also to infer implicit characteristics that are capable to enrich the explicit extraction, by applying complex approaches of analysis.

• **RQ.4: How to represent the extracted specifications?**

Through the definition of reverse engineering approaches that use complex techniques (e.g. model transformations, clustering) in order to generate models and specifications, namely by producing detailed Domain entities (relevant relationships and entities properties), and from those generate higher-level views, such as Business Entities and Use Cases views. Nevertheless, the extracted models and specifications are always a tentative approximation.

### 5.2 Future work

During the XIS-Reverse development, some ideas for future work and research directions came up, namely from feedback in a conference presentation. However, they were not considered either due to time constraints or due to their complexity, they are briefly introduced below:

- **Analyse similarity between the combined results of XIS-Reverse and XIS-Web and the original Legacy Application:** Once the XIS-Reverse and the XIS-Web approaches are aligned, then a new application, using XIS-Web technology, can be generated. Thus, it would be interesting to analyse how different that new application is from the legacy application.

- **Use a Divide-and-conquer approach to solve combinational explosion:** In large domains, Implicit Generalization Discovery can easily lead to a combinational explosion. That problem was partially solved in our approach by using a specified list of names to be ignored, introduced by the user. However, we think that, a Divide-and-conquer approach could help to solve this problem, by splitting entities into groups of related entities and try to work from that.

- **Incorporate support for different Relational Databases technologies:** Although EA supports several ODBC data sources, the XIS-Reverse tool only supports additional database artefacts (profiler log files and database connections), that use the Microsoft SQL Server technology. Thus, we think that adding support for more technologies would be a good contribution, allowing to expand the scope of the tool to more information systems. Furthermore, that extension is performed by adding a mechanism that allows to chose the wanted technology in the XIS-Reverse configuration panel. Additionally, an interface to support the database connection and a method to parse the profiler log file, with the new technology, should also be added. However, from our experience, the most complex task here is related with the parsing, since each technology has different language nuances (e.g. functions) and each database management system has a distinct way to represent profiler log files.

- **Extend this approach to easily support different output representation:** The main idea behind this is to create an extension (e.g. plug-in), merely focused in the representation of different output specifications. Allowing anyone to establish a mapping between our produced specifications and any other language.
• **Extend this approach, incorporating screen and additional program analysis**: Although very complex, due to its dependency to the used technology, screen and program analysis can be very useful to enrich the produced use case models, or ultimately to produce the interaction and navigation space views, by analysing the application interfaces content, links between each screen or the application’s source code.

• **Support user restructuring changes**: Although our approach is focused in the reverse engineering stage, the user is free to make any changes after this process is complete, for example updating some relationships that might be wrongly interpreted. Thus, we propose a new improvement to this afterwords task, that could be a tool that from an (updated) domain model would regenerate the equivalent business entities and the use cases models.
Bibliography


