Evaluation of XIS-Reverse, a Model-Driven Reverse Engineering Approach for Legacy Information Systems

André Reis and Alberto Rodrigues da Silva

INESC-ID, Instituto Superior Técnico, Universidade de Lisboa, Lisbon, Portugal,
{andre.filipe.reis, alberto.silva}@tecnico.ulisboa.pt

Abstract. Companies have been struggling to manage and maintain their legacy information systems because upgrading those systems has been a complex challenge. Many times, requirements changes are difficult to be properly managed, leading to legacy information system requirements deterioration. To overcome or reduce such problems we propose the XIS-Reverse, a software reverse engineering approach. XIS-Reverse is a model-driven reverse engineering approach that takes database artefacts and user preferences as input, and generates high-level models and specifications of these legacy information systems. This paper presents the evaluation of XIS-Reverse using two real-world information systems, provides an assessment of its interoperability with an existent framework and discusses its main challenges and benefits.

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

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 [1]. 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 [1]. Those kind of specifications are usually hard to keep up to date while the software applications are being developed, leading to deterioration. 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 [2]. 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 [3]. 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 [2].

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 [3]. 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 [4]. 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 [5,6]. 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)) [3]. Moreover, metamodelling 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.

XIS-Reverse [7] has been created precisely to mitigate legacy information system's requirements deterioration and maintenance, reducing human effort and improving productivity. These goals can be achieved by using reverse engineering techniques based on a model-driven approach, able 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.

This paper extends the previous work that introduced the XIS-Reverse approach [7] with the following novel contributions: (i) an extensive discussion of the relevance of this approach based on the evaluation of two real-world cases studies with large databases (e.g., the case study B involves more than 150 data entities and more than 200 associations); (ii) a discussion showing how it is possible to combine the XIS-Reverse approach with other forward engineering approaches, namely the XIS-Web approach [8] in the scope of the XIS* framework and technologies, namely by showing that the extracted XIS* models (with the XIS-Reverse) can then be involved in models validation, model-to-model and model-to-text transformations; (iii) finally, a comparison of the XIS-Reverse with other approaches and a discussion of the related work.

Futhermore, regarding the main contributions of XIS-Reverse, we have to highlight the following aspects: (i) the semi-automatic heuristics that can identify certain relationships between entities, namely implicit generalizations and specialization of associations (aggregations), and also (ii) the possibility to ex-

tract values from the source database to enrich the target models or specifications. All that combined allows to enhance the understanding of each entity role in the produced models.

XIS-Reverse has been developed over the last sixteen months following the well-known Action Research methodology [9]. Over time it was necessary to evaluate the level of detail and correctness of the XIS-Reverse extracted specifications. This evaluation process was done in an iterative way. Initially, this was done by testing only a subset of the approach, namely the domain entities extraction, with simple case studies. However, throughout this period, we got the chance to use real-world applications to evaluate it, which allowed to evolve XIS-Reverse with richer case studies.

The outline of this paper is as follows. Section 2 presents the context. Section 3 gives an overview of the XIS-Reverse. Section 4 presents and analyses the evaluation performed to XIS-Reverse, using two real-world applications. Section 5 presents and analyses the interoperability evaluation of XIS-Reverse with an existent framework. Section 6 analyses and compares this proposal with the related work. Finally, Sect. 7 summarizes the main conclusions of this work along with some future work perspectives.

2 Background

This research has been developed at the Instituto Superior Técnico, Universidade de Lisboa, in the scope of the MDDLingo[1] and the RSLingo[2] initiatives.

MDDLingo is an umbrella researching initiative that aggregates several projects around MDE topics, namely involving the definition of a family of languages, also known as XIS*. This set of modelling languages derives from the XIS-UML profile [10,11], involving namely XIS-Mobile [12,13], XIS-CMS [14] or XIS-Web [8]. 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 (which includes 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 1 illustrates a simple XIS* Domain view which aggregates domain classes (XisEntity), their attributes (XisEntityAttribute) and relationships (XisEntityAssociation and XisEntityInheritance).

Figure 2 shows a 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.

Figure 3 shows the UseCases View. This view details the operations an actor can perform over the business entities when interacting with the system [12].

RSLingo is a general approach defined to rigorously specify and validate software requirements using lightweight Natural Language Processing techniques to

[1] `https://github.com/MDDLingo`

[2] `https://github.com/RSLingo`

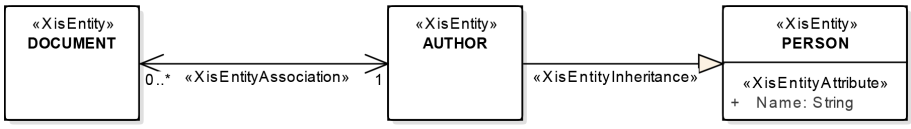

Fig. 1. Example of a XIS* Domain view.

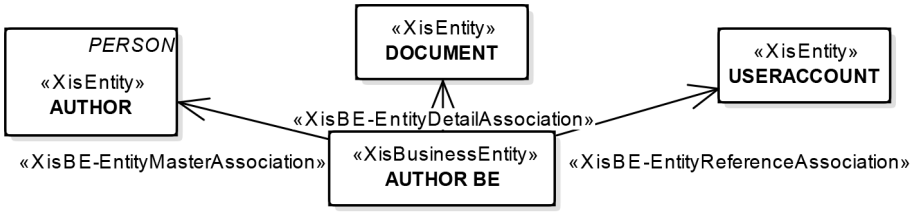

Fig. 2. Example of a XIS* BusinessEntities view.

(partially) translate informal requirements into a rigorous representation provided by a language specially designed for Requirements Engineering. Overtime, following the RSLingo's approach, several projects have been developed, namely RSLingo4Privacy [15,16] and RSLingo's RSL[3] [17]. Moreover, RSLingo's 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 [17]. 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 [17]. The most relevant RSLingo's RSL concepts regarding our research are: Data Entities, Data Entity Views, User Stories, Functional Requirements, Goals, Business Processes and Terms.

3 XIS-Reverse Overview

The XIS-Reverse [7] is a MDRE approach that allows to extract high-level specifications from legacy application artefacts. As illustrated in Fig. 4, the XIS-Reverse approach starts by extracting the application data model from an avail-

[3] `https://github.com/RSLingo/RSL`

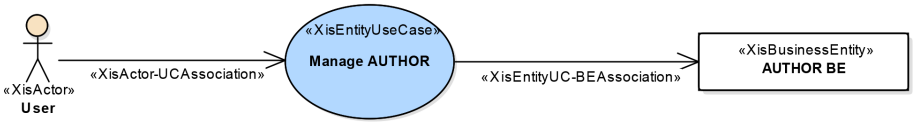

Fig. 3. Example of a XIS* UseCases view.

able database, and from that and from the user configuration (second stage), the reverse engineering execution takes place, by applying several reverse engineering heuristics on those artefacts, and then generating the extracted knowledge in the form of models and specifications.

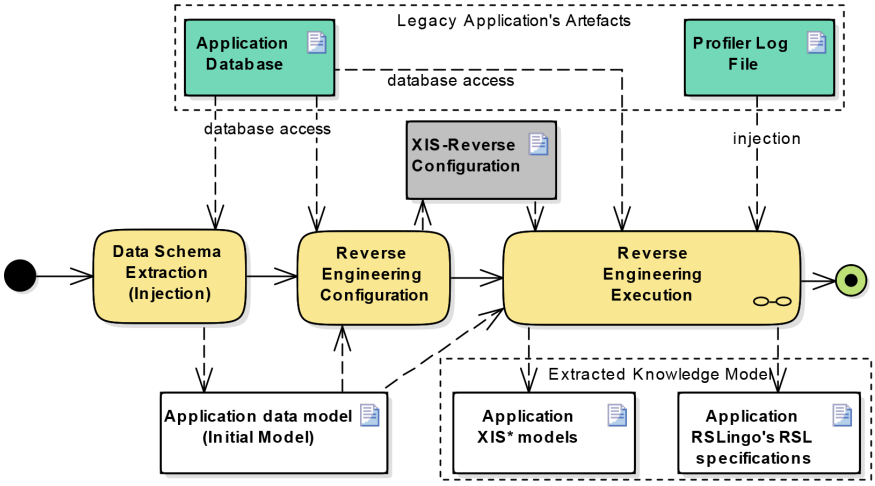

Fig. 4. Overview of the XIS-Reverse approach.

XIS-Reverse was implement on top of the Sparx Systems EA[4], as an EA plug-in. The XIS-Reverse's first stage (application data model extraction) relies on the native capability 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 (available from GitHub[5]). The configuration stage provides an user interface (see Fig. 5) that can be split into 4 different areas:

- **Input** - to specify input artefacts, namely the application data model, database name and additional artefacts, namely a database access or a profiler log file;
- **Output** - to select additional output representations, namely XIS-Web and RSLingo's RSL;
- **Transformation Rules Guidance** - to provide configuration points to the following features: Simple Principal Entities (to identify aggregations); Attribute Values Extraction (to extract attribute values); and Generalization Discovery (to detect implicit generalizations);
- **Appearance** - to improve the readability of the produced specifications.

[4] `http://www.sparxsystems.com/products/ea`
[5] `https://github.com/MDDLingo/xis-reverse`

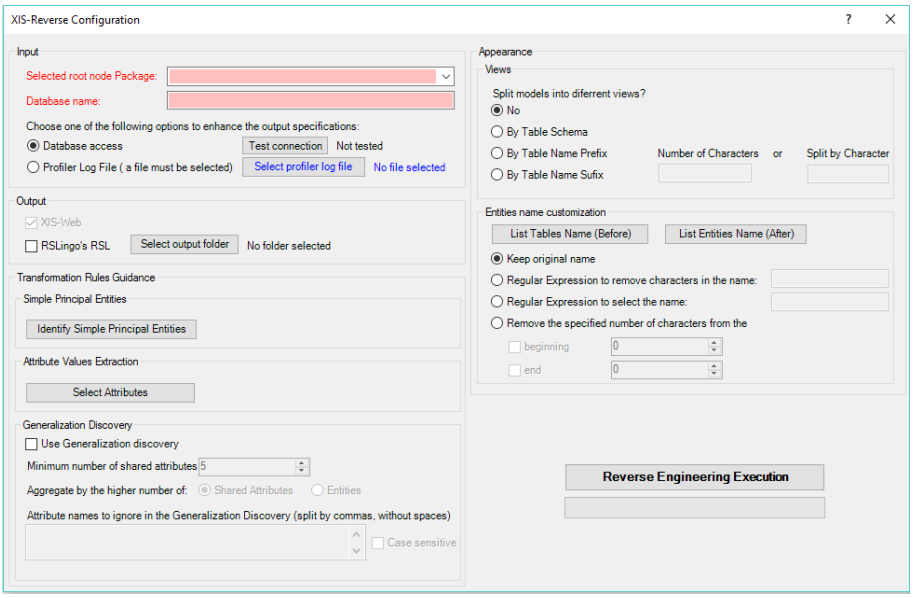

Fig. 5. Main configuration panel of the tool.

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 [8] and RSLingo's RSL specifications [17] from Microsoft SQL Server databases.

Finally, the reverse engineering execution stage is supported by M2M transformations that use the application data model and user configurations to generate XIS* models and RSLingo's RSL specifications. 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.

4 Evaluation

In this section, two case studies are introduced and used to assess the XIS-Reverse 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 be not 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 aplying 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:

$$Reis(x, y) = 0.5x + 0.5y \tag{1}$$

To better explain the Reis function and its variables, a simple domain model illustrated in Fig. 6 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 Fig. 6).

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 Fig. 6, since pairs A-B share at most 2 attributes and C-F share at most 1 attribute, for example).

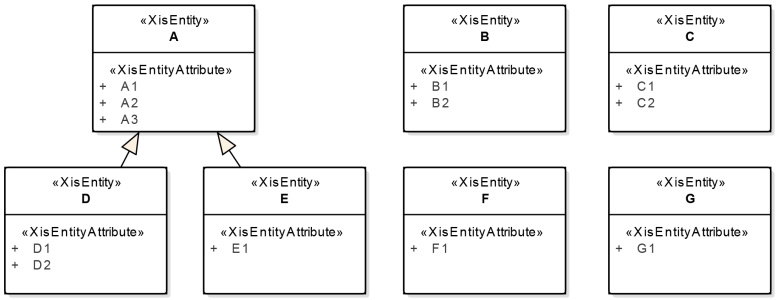

Fig. 6. Support example to explain the Reis function.

4.1 Case Study A: ProjectIT-Enterprise

The ProjectIT[6] [18,19] 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 web application, called ProjectIT-Enterprise [20,21], 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.

Regarding the aggregation rules, since we had access to the domain model specifications of this application (Fig. 7), 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 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 Fig. 7.

[6] `http://isg.inesc-id.pt/alb/ProjectIT`

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 Fig. 7).

Taking into account Fig. 7, 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).

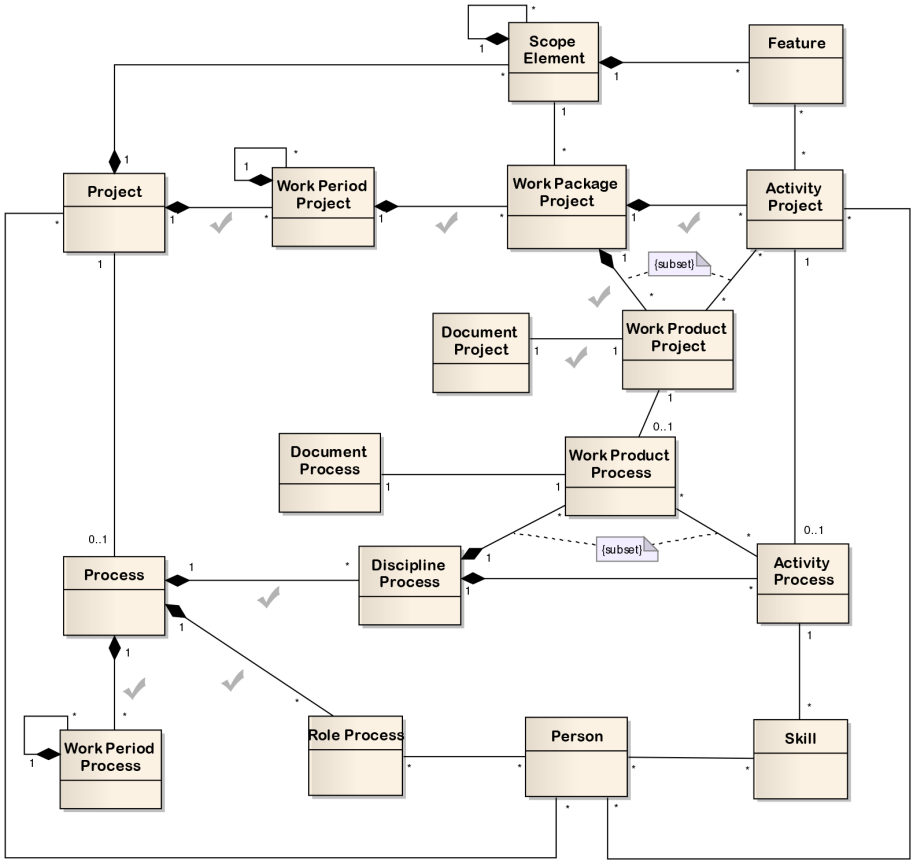

Fig. 7. Case Study A - domain model of the project dimension (adapted from [20]).

Table 2 shows the results of applying the aggregation rules using the XIS-Reverse. These results are analysed bellow, in each of the configuration scenarios.

Furthermore, to support each scenario, Table 3 presents the overall picture of the extracted elements.

Scenario A: Extraction without configuration. In this scenario, the unique configuration performed is the connection setting with the database. With that,

Table 1. Case Study A - equivalence between application database and domain model.

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

Table 2. Case Study A - evaluation of aggregation scenarios.

	Rule-1				Rule-2	
Results / Scenarios	**True Positive**	**False Positive**	**False Negative**	**True Negative**	**Configs. without aggregations**	**Configs. with aggregations**
A: Without Configs.	6	1	1	0	4	1
B: M = 20	0	0	7	1	5	0
B: M = 10	2	1	5	0	5	0
B: M = 5	3	1	4	0	4	1
C: Semi-guided	**6**	**1**	**1**	**0**	**5**	**0**

Table 3. Case Study A - overall results of the reverse engineering.

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

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 (rule E-1 [7]). This problem occurred due to the generic definition of such heuristic that did not take into account composite primary keys. Moreover, that issue 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 that heuristic (updated listing available on GitHub[7]).

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-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 EA-2-b ([7]) 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.

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 ([7]), we started by using the default value (20) to filter entities by the maximum number of rows that a table can have in the database. And then selected all those entities. Moreover,

[7] `https://github.com/MDDLingo/xis-reverse`

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 correctly 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 ([7]). 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, 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:

$$Reis'(a, b) = 0.5(\frac{a}{31}) + 0.5(\frac{b}{51}) \tag{2}$$

Table 4. Case Study A - Scenario B - evaluation of generalization simulations.

Minimum # of shared attributes	**Aggregated by**	**Ignored names**	**Subclasses (a)**	**Inherited attributes (b)**	**Reis'**
5	attributes	-	4	12	0.18
5	entities	-	4	12	0.18
4	attributes	-	4	12	0.18
4	entities	-	5	10	0.18
3	**attributes**	**-**	**8**	**18**	**0.31**
3	entities	-	10	12	0.28
3	attributes	name,description	4	8	0.14
3	entities	name,description	4	8	0.14
2	attributes	name,description	6	10	0.19
2	entities	name,description	8	8	0.21

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.

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 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.

4.2 Case Study B: Social Security application

During this research period, we worked 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 (further XisEntityInheritance transformation rule [7]), 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 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.

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.

Table 5. Case Study B - overall results of the reverse engineering.

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

5 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 8 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.

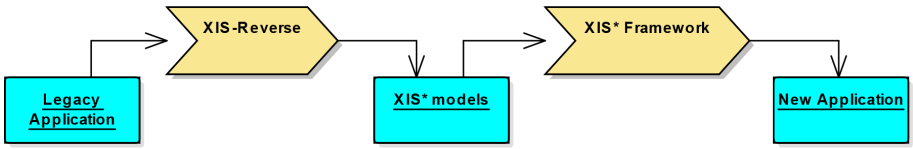

Fig. 8. Interoperability with XIS* frameworks.

Once again, we used the XIS-Web technology [8], 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) Model-to-Text (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 [8]. Then, using the XIS-Web specifications from the ProjectIT-

[8] `https://github.com/MDDLingo/xis-web`

Enterprise (Sect. 4.1), that were extracted with the XIS-Reverse tool, we did our evaluation step by step:

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.

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

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.

6 Related Work Discussion

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 and reverse engineering of databases. Moreover, those contributions are also compared with our approach.

6.1 Data Schema Extraction

The main properties of the research works analysed in this subsection are shown in Table 6. This table specifies for each approach, its input, the existence of data schema extractors, if it extracts all table properties and its output. The last row categorises our approach.

Gra2MoL [22] Text-to-Model (T2M) language and MoDisco [3] framework have been specially tailored for data schema extraction (model injection).

Table 6. Classification of some related works on data schema extraction (adapted from [7]).

Approach	Input	Output	Schema Extractors	Properties
Gra2Mol [22]	any textual artefact	any model	no	n.a.
MoDisco [3]	several sources	any model	no	n.a.
Schemol [23]	data stored into DB	any model	no	n.a.
DB-MAIN [24]	several sources	GER	yes (e.g. ODBC)	yes
SQL2XMI [25]	SQL DDL schema	UML in XMI	yes (only MySQL)	no
EA (XIS-Reverse)	several sources	UML	yes (e.g. ODBC)	yes

Gra2MoL is a Domain Specific Language (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 [23] is another tool for injecting models. However, this tool allows injecting data stored into 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 [24] and SQL2XMI [25] 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 data definition language (DDL).

To sum up, 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 we used EA.

6.2 Reverse Engineering

A reverse engineering approach can be classified in several ways. Table 7 gives a properties summary of the thereafter analysed research works. These properties include: kind of analysis, output, the existence of tools for automating the

approach, automation level of those tools in regards to the reverse engineering stage, if the tool allows extension and main contributions (A - Aggregations Extraction, G - Implicit Generalization Extraction and V - Attribute Values Extraction).

Table 7. Classification of some works on reverse engineering (adapted from [7]).

Approach	Analysis	Output	Contribs.	Tools	Auto.	Extension
[26]	schema	OMT class diagram	G	no	n.a	n.a.
[27]	data	Extended ER	n.a.	no	n.a	n.a.
NoWARs [28]	data	Conceptual schema	n.a.	yes	Semi	n.a.
RAINBOW [29]	screen	ER model.	n.a.	yes	n.a.	n.a
[30]	program (static)	Extended ER	n.a.	no	n.a	n.a.
[31]	program (static)	Object-Oriented class diagram	n.a.	no	n.a	n.a.
[32]	program (dynamic)	n.a.	n.a.	yes	n.a.	n.a.
[33]	program (dynamic)	n.a.	n.a.	yes	n.a.	n.a.
Modisco [3]	schema and program (static)	KDM or UML	n.a.	yes	Auto.	yes
Relational Web [34]	schema	UML	n.a.	yes	Auto.	yes
DB-MAIN [24]	schema and program (static)	GER	G	yes	Semi	yes
XIS-Reverse	schema, data and program	XIS* and RSLingo's RSL	A;G;V	yes	Semi	yes

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 [26] is mainly focused on spotting similarities in names, value domains and representative patterns. This technique may help identify missing constructs (e.g. foreign keys). Moreover, [26] specification of a manual process was adapted in our approach to semi-automatically and automatically identify generalizations and many-to-many associations, respectively.

Data analysis [27,28] uses content mined from a database. Firstly it can be used to analyse the database normalisation 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. In addition, our approach uses this technique in a similar way, however it is applied to classify associations.

Screen analysis [29] 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. Moreover, our approach did not consider this kind of analysis.

Static [30,31] and Dynamic [32,33] 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. Furthermore, our approach uses static program analysis in the profiler log file, aiming to classify associations.

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 MDRE framework [3] 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. Our approach could be implemented extending this framework, however that would require the definition from scratch of all the three stages (discoverers, transformations and generators) needed to produce the desired artefacts.

Polo et al. propose a method and a tool, called Relational Web [34] 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 diagram is manipulated 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. However, 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.

As previously stated, DB-MAIN [24] 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. However, this tool will require the user to apply all the needed transformations, thus the degree of automation achieved in our approach is higher. Furthermore, DB-MAIN supports generalization representation, but once again it must be identified by the user.

Regarding to the main contributions of this paper, we do not find any other approach that specializes associations (e.g. distinguishing between associations and aggregations), nor any approach that allows to extract attribute values and their representation into the target conceptual schema.

7 Conclusion

XIS-Reverse approach allows to automatically extract high-level models and specifications from legacy applications. This approach benefits from a flexible set of configuration points and new features not found in the related work, that allows to produce more detailed models and specifications, that overall will help the user to get a better understanding of the application domain.

In terms of aggregations detection, at least when using a system with a good amount of data (usage), our heuristics can 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. In the extreme scenario, if the user has a good understanding 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 (Sect. 5), 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.

Regarding the future work, and considering the extensibility of the proposed process, we would like to evaluate the similarity between the combined results of pipelining XIS-Reverse with XIS-Web processes to generate a new application, and then compare it with the original legacy application. A divide-and-conquer approach could be used to manage the complexity of identifying implicit generalizations, for example splitting sets of entities by their schema. Additionally, we would like to extend the XIS-Reverse approach to support new input and

output technologies and include more type of analysis in the reverse engineering process (e.g. screen).

References

1. Garcia, J. M. E.: *Requirements Change Management based on Web Usage Mining.* PhD thesis, University of Porto, 2016.
2. Chikofsky, E. J. and Cross, J. H.: "Reverse engineering and design recovery: A taxonomy," *IEEE software*, vol. 7, no. 1, pp. 13–17, 1990.
3. Bruneliere, H., Cabot, J., Dupé, G., and Madiot, F.: "Modisco: A model driven reverse engineering framework," *Information and Software Technology*, vol. 56, no. 8, pp. 1012–1032, 2014.
4. Canfora, G., Di Penta, M., and Cerulo, L.: "Achievements and challenges in software reverse engineering," *Communications of the ACM*, vol. 54, no. 4, pp. 142–151, 2011.
5. Ruiz, F. *et al.*: *An approach for model-driven data reengineering.* PhD dissertation, University of Murcia, 2016.
6. da Silva, A. R.: "Model-driven engineering: A survey supported by the unified conceptual model," *Computer Languages, Systems & Structures*, vol. 43, pp. 139–155, 2015.
7. Reis, A. and da Silva, A. R.: "XIS-Reverse: A model-driven reverse engineering approach for legacy information systems," in *Proceedings of the 5th International Conference on Model-Driven Engineering and Software Development - Volume 1: MODELSWARD*, pp. 196–207, 2017.
8. Seixas, J.: *The XIS-Web Technology: A Model-Driven Development Approach for Responsive Web Applications.* MSc dissertation, Instituto Superior Técnico, University of Lisbon, 2016.
9. Baskerville, R. L.: "Investigating information systems with action research," *Communications of the AIS*, vol. 2, no. 3es, p. 4, 1999.
10. da Silva, A. R.: "The XIS approach and principles," in *29th Euromicro Conference*, pp. 33–40, IEEE, 2003.
11. da Silva, A. R., Saraiva, J., Silva, R., and Martins, C.: XIS-UML profile for extreme modeling interactive systems. In *Model-Based Methodologies for Pervasive and Embedded Software, 2007. MOMPES'07. Fourth International Workshop on*, pages 55–66. IEEE, 2007.
12. Ribeiro, A. and da Silva, A. R.: "XIS-Mobile: a dsl for mobile applications," in *29th Annual ACM Symposium on Applied Computing*, pp. 1316–1323, ACM, 2014.
13. Ribeiro, A. and da Silva, A. R.: "Evaluation of XIS-Mobile, a domain specific language for mobile application development," *Journal of Software Engineering and Applications*, vol. 7, no. 11, p. 906, 2014.
14. Filipe, P., Ribeiro, A., and da Silva, A. R.: "XIS-CMS: Towards a model-driven approach for developing platform-independent cms-specific modules," in *MODELSWARD*, SCITEPRESS, 2016.
15. da Silva, A. R., Caramujo, J., Monfared, S., Calado, P., and Breaux, T.: Improving the specification and analysis of privacy policies. *ICEIS 2016*, page 336, 2016.
16. Caramujo, J. and da Silva, A. R.: Analyzing privacy policies based on a privacy-aware profile: The facebook and linkedin case studies. In *Business Informatics (CBI), 2015 IEEE 17th Conference on*, volume 1, pages 77–84. IEEE, 2015.
17. da Silva, A. R.: "Linguistic Patterns and Styles for Requirements Specification: The RSL/Business-Level Language," in *Proceedings of the European Conference on Pattern Languages of Programs*, ACM, 2017.

18. da Silva, A. R., Saraiva, J., Ferreira, D., Silva, R., and Videira, C.: "Integration of RE and MDE paradigms: the ProjectIT approach and tools," *IET software*, vol. 1, no. 6, pp. 294–314, 2007.
19. da Silva, A. R.: "O programa de investigação projectit," *White Paper, Relatório INESC-ID, Grupo de Sistemas de Informação*, 2004.
20. Pinto, M. A. P.: *Gestão de Projectos com Processos Ágeis.* MSc dissertation, Instituto Superior Técnico, University of Lisbon, 2010.
21. Martins, P. V. and da Silva, A. R.: "ProjectIT-enterprise: a software process improvement framework," in *Industrial Proceedings of the 17th EuroSPI Conference*, pp. 257–266, 2010.
22. Izquierdo, J. L. C., Cuadrado, J. S., and Molina, J. G.: "Gra2mol: A domain specific transformation language for bridging grammarware to modelware in software modernization," in *Workshop on Model-Driven Software Evolution*, 2008.
23. Díaz, O., Puente, G., Izquierdo, J. L. C., and Molina, J. G.: "Harvesting models from web 2.0 databases," *Software & Systems Modeling*, vol. 12, no. 1, pp. 15–34, 2013.
24. Hainaut, J.-L., Englebert, V., Henrard, J., Hick, J.-M., and Roland, D.: "Database evolution: the DB-MAIN approach," in *International Conference on Conceptual Modeling*, pp. 112–131, Springer, 1994.
25. Alalfi, M. H., Cordy, J. R., and Dean, T. R.: "SQL2XMI: Reverse engineering of uml-er diagrams from relational database schemas," in *15th Working Conference on Reverse Engineering*, pp. 187–191, IEEE, 2008.
26. Premerlani, W. J. and Blaha, M. R.: "An approach for reverse engineering of relational databases," in *Reverse Engineering, 1993., Proceedings of Working Conference on*, pp. 151–160, IEEE, 1993.
27. Chiang, R. H., Barron, T. M., and Storey, V. C.: "Reverse engineering of relational databases: Extraction of an eer model from a relational database," *Data & Knowledge Engineering*, vol. 12, no. 2, pp. 107–142, 1994.
28. Pannurat, N., Kerdprasop, N., and Kerdprasop, K.: "Database reverse engineering based on association rule mining," *arXiv preprint arXiv:1004.3272*, 2010.
29. Ramdoyal, R., Cleve, A., and Hainaut, J.-L.: "Reverse engineering user interfaces for interactive database conceptual analysis," in *International Conference on Advanced Information Systems Engineering*, pp. 332–347, Springer, 2010.
30. Petit, J.-M., Kouloumdjian, J., Boulicaut, J.-F., and Toumani, F.: "Using queries to improve database reverse engineering," in *International Conference on Conceptual Modeling*, pp. 369–386, Springer, 1994.
31. Di Lucca, G. A., Fasolino, A. R., and De Carlini, U., "Recovering class diagrams from data-intensive legacy systems," in *Software Maintenance, 2000. Proceedings. International Conference on*, pp. 52–63, IEEE, 2000.
32. Cleve, A., Meurisse, J.-R., and Hainaut, J.-L.: "Database semantics recovery through analysis of dynamic sql statements," in *Journal on data semantics XV*, pp. 130–157, Springer, 2011.
33. Cleve, A., Noughi, N., and Hainaut, J.-L.: "Dynamic program analysis for database reverse engineering," in *International Summer School on Generative and Transformational Techniques in Software Engineering*, pp. 297–321, Springer, 2011.
34. Polo, M., García-Rodríguez, I., and Piattini, M.: "An mda-based approach for database re-engineering," *Journal of Software Maintenance and Evolution: Research and Practice*, vol. 19, no. 6, pp. 383–417, 2007.