MARIA: A PROCESS TO MODEL ENTITY RECONCILIATION PROBLEMS

J.G. ENRÍQUEZ, M. OLIVERO, A. JIMÉNEZ-RAMÍREZ, M.J. ESCALONA and M. MEJÍAS

Computer Languages and Systems Department. University of Seville. Av. Reina Mercedes s/n, 41012, Seville, Seville.

{jose.gonzalez, miguel.olivero}@iwt2.org, {ajramirez, mjescalona, risoto}@us.es

Within the development of software systems, the development of web applications may be one of the most widespread at present due to the great number of advantages they provide such as: multiplatform, speed of access or the not requiring extremely powerful hardware among others. The fact that so many web applications are being developed, makes enormous the volume of information that it is generated daily. In the management of all this information, the entity reconciliation (ER) problem occurs, which is to identify objects referring to the same real-world entity. This paper proposes to give a solution to this problem through a web perspective based on the Model-Driven Engineering paradigm. To this end, the Navigational Development Techniques (NDT) methodology, that provides a formal and complete set of processes that bring support to the software lifecycle management, has been taken as a reference and it has been extended adding new activities, artefacts and documents to cover the ER. All these elements are defined by a process named Model-Driven Entity ReconcilIAtion (MaRIA), that can be integrated in any software development methodology and allows one to define the ER problem from the early stages of the development. In addition, this proposal has been validated in a real-world case study helping companies to reduce costs when a software product that must give a solution to an ER problem has to be developed.

Key words: Web Engineering, Model-Driven Engineering, Entity Reconciliation, NDT

1 Introduction

At present, the development and creation of web applications is used in the world as a technological tool to unite regions, create business, support companies, appear in the market and plethora of applications according to the perspectives of people and their scope, finding on the internet a vital source of job creation, effective and intelligent business and great help in achieving objectives and approaches.

In this world where there are more than 6.400 million devices connected to the Internet generating information, where more than 1.570 terabytes of information per minute are transferred and where the information about any topic takes a lot of interest by the civilization [1], it appears a very big and difficult problem when someone tries to look for information, due to the heterogeneity of data [2].

The problem of heterogeneity of data resides in the multiple data sources where the information related to the same topic is stored adding to this problem that most of these data sources do not share

the same structure, and data that they store, even though they are related to the same subject, may not be the same.

In this context the problem of the entity reconciliation occurs. This problem lies in the difficulty of identifying entities in different data sources that describe the same real-world entity although the information that describes this entity are not equal.

This paper aims to give a solution to this kind of problem through a model-driven web perspective. During the last years, the Web engineering community has been proposing several different methodologies for modeling Web applications with different concepts and definitions, such as UWE (UML-based Web Engineering) [3], WebML (The Web Modeling Language) [4], OOH4RIA[5], RUX-Method [6] or Navigational Development Techniques (NDT) [7] methodology, among others [8]. Particularly, it was NDT the methodology selected as the base for this proposal to realize how these activities behave in a real context for two reasons: (i) the experienced background in the development of projects our research group has achieved, and (ii) the high performance and acceptance rate that this methodology has demonstrated in several projects.

The remainder of this paper is organized as follows: section 2 describes the related work that has been done about methodologies proposed to solve the entity reconciliation problem. Section 3 summarizes the background of this paper, presenting the two main pillars of this proposal, Model-Driven Engineering (MDE) and NDT and a brief introduction to the Entity Reconciliation (ER) problem. In section 4, the MaRIA Process is described. In section 5 the different activities that this paper proposes in order to extend NDT for covering an entity reconciliation problem into a software development process are explained. Section 6 describes a real-world case study where this proposal has been validated and finally, section 7 states a set of conclusions and future work.

2 Related Work

Similar researches have been carried out to solve this problem, and others papers conduct systematic researches about approaches for solving the entity reconciliation issue [9]. There are some relevant similar works being undertaken recently. Beheshti et al. [10] and Papadakis et al. [11], relates to a specific methodology, but using a reduced number of databases not higher than five, and in a specific entity relationship problem in contrast with our current work, which aims to be for general ER problem purposes.

Another research related with this field is the one presented by Cucerzan [12], in which he reviews this issue historically and proposes entity disambiguation in a large-scale environment. In this paper, authors implemented a Web browser that analyzes a document or webpage and translate from words to concepts.

There are also some approaches including researches with Word Sense Disambiguation (WSD) and Entity Linking. In his study Moro [13] discusses how WSD is closely related with EL since both of them have to solve lexical ambiguity of language, but differs in that EL the text can be linked to an entity which could or could not contain an exact coincidence, there is an exact coincidence to, at least, the word lemma. This merge could help the entity reconciliation accuracy by linking more references to the same entity. The author employs lexicographic and encyclopedic knowledge to unify EL and WSD in any language covered by their semantic network.

Shen et al. [14] discuss this issue and refer to some record linkage and entity linking related works in its survey. In this survey, Shen highlights the fact that when working with different data sources, the information may appear in different formats and with different characteristics. Despite the current stage of the researches, there is still much room for further improvement on recognizing these data. Furthermore, there is some research being carried out to extract information from a tweet and associate each mention to an entity by Liu et al. [15]. Additionally, Shen et al. [14] mentions the lack of analysis of computational complexity while maintaining high accuracy in entity reconciliation as an additional future work in this area.

However, despite the obtained results of these researches, there are still some issues at the time of linking when a proper name is also a word, whether or not it is a polysomic word. When a common word is also used to name a specific concept such as a song, a building or a painting, among other sets of concepts, the accuracy of the framework used may decrease substantially.

3 Background

3.1 Model-Driven Engineering (MDE)

In the scientific activity, abstraction has been and is widely used, and often referred to it as the activity of modelling. If a model is defined as a partial or simplified reality representation, that allows the final user to address a complex task for a specific purpose, it can be considered a model as the result of abstraction [16].

The complexity of software development has been growing drastically. In this sense, developers identified in models an alternative for addressing this complexity. MDE emerged to address the complexity of software systems in order to express the concepts of the problem domain in an effective way [17]. Thus, in the early stages of development, models are more abstract than in the final stages where the models are much closer to implementation. It means, abstract models are transformed into concrete ones the aim of producing software. Studying this process, Brambilla et al. [18] described the two fundamental pillars of the MDE paradigm for creating software automatically: models and transformations.

- **Models** must be defined according to the rules of a concrete Modelling Language (ML). This language defines the syntax and semantic of the model [19]. The ML syntax is composed of a concrete and an abstract syntax. The abstract one defines the language structure and how the different elements can be combined, regardless of its representation. The semantic one, that provides the static and dynamic part, poses restrictions and establishes the meaning of the elements of the language and different ways to combine them. In this context, the concept of a metamodel is required. A metamodel can be defined as a special type of model that specifies a ML. The metamodel defines the structure and constraints for a family of models [20].
- **Transformations** are the mechanisms that allow one to derive models from other existing ones. A transformation between models represent a relation between two abstract syntaxes and it is defined by a set of relations between the elements of the metamodels [21]. There are two types of transformations: horizontal (the derived model and the original one have the

same abstraction level) and verticals (the derived model has a lower abstraction level than the original one).

A very interesting concept found in the MDE literature is proposed by Bézivin et al. [22] where "Everything is a model". In this sense, transformations themselves, are also considered as models. Generally, a transformation models program takes as input a model according to an origin metamodel and produces as output a model according to the target metamodel. The transformation program, should be considered as a model itself.

One of the advantages of MDE is its support for automation, as the models can be automatically transformed from the early stages of development to the final stages. Therefore, MDE allows automating the tasks involved in a software development, such us the testing tasks. The versatility offered by MDE makes it applicable to many different environments such as: quality, programming languages or business process management among others [8], [23]–[25].

3.2 Navigational Development Techniques (NDT)

NDT is a Model-Driven Engineering-based methodology that provides a formal and complete set of processes that allow bring support to the software lifecycle management. Using NDT, it is possible cover the phases of the software engineering life cycle in a structured way, reducing errors and redundancies [7].

NDT Framework (Figure 1), establishes six big groups of processes that allow one to develop a complete software system in all its phases. These groups are: project management process, software development process, maintenance process, testing process, quality process and security process. For the scope of this paper, the focus will be put in the software development process group.

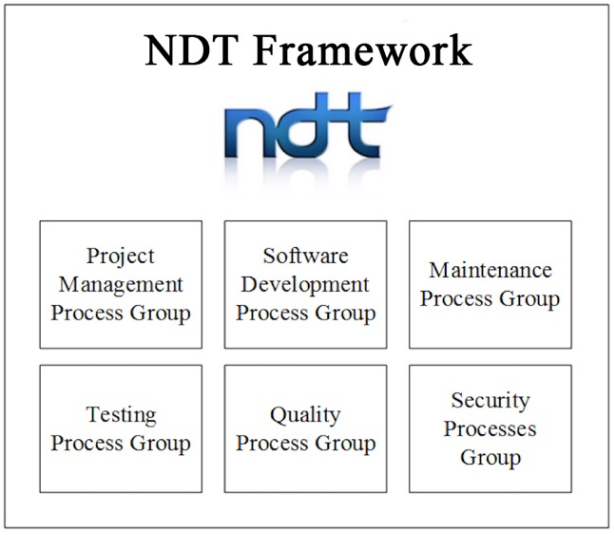

Figure 1 An iterative design model.

Software Development Process Group defines all the processes necessary for the development of a software project. To achieve this goal, NDT methodology proposes six groups of processes: feasibility, requirements, system analysis, system design, system building and system implementation.

- **Feasibility Process**. In this process, the feasibility of a particular software project should be analyzed. It is usually done if stakeholders require it or when the project is developed in a complex or little known environment that may make the feasibility of the project doubtful.
- **Requirements process**. In this process, all the tasks related to complete the requirements catalogue of the system to be developed must be performer. The deliverable of this phase is generated by Enterprise Architect (EA) tool using an NDT-Suite plugin [7].
- **System analysis process**. It determines the technical sheet for system analysis. It is EA tool by means of NDT-Suite plugin that helps to achieve the deliverable of this phase.
- **System design process**. It is the process in which the environment where the system is implemented becomes concrete. In previous phases, the project has been platform independent. In this process, the concrete architecture of the work is defined and the project is prepared so that it can be implemented.
- **System building process**. It defines the technical sheet for system building, defining three environments: development, pre-production and production.
- **System implementation process**. It studies the technical sheet for system implementation, defining three environments: development, pre-production and production.

In this context, what this paper proposes is to extend this software development process group of NDT Framework, adding a set of activities, documents and artefacts in order to make a software engineer able to carry out an entity reconciliation problem.

3.3 Entity Reconciliation

Entity reconciliation (also called entity resolution or ER) is a fundamental problem in data integration. It refers to combining data from different sources for a unified vision or, in other words, identifying entities from the digital world that refers to the same real-world entity. It is an uncertain process because the decision to allocate a set of records with the same entity, cannot be taken with certainty, unless these records are identical in all their attributes or they have a common key [17, 18].

Figure 2, illustrates a very clear example of entity reconciliation proposed by McCallum [28]. This example is based on the bibliographic author names of a database where the entitles to reconcile are the authors. Left side of the Figure 2 shows the “before” part of an entity reconciliation process. It is possible to see that there are different authors related between them, and some of them, seems to be the same author although it is named by a different way. Right side of the Figure 2 shows the “after” part of an entity reconciliation process. It is possible to see that the result structure after applying an entity reconciliation process is much more clear, understandable and clean, eliminating all possible duplications, both in existing relationships and in the entities themselves.

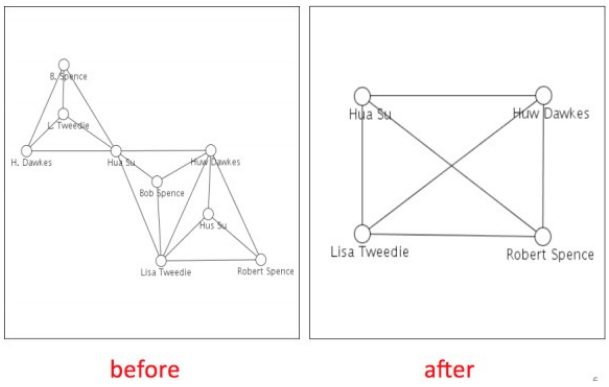

Figure 2 Entity Reconciliation example [28]

Entity reconciliation is a well-known problem and it has been investigated since the birth of relational databases [29] and it can be applied to many different domains. If, to everything mentioned above, a very trending topic nowadays such as the Big Data is added, this problem receives a much more significant attention due to the new challenges that arise.

4 MaRIA Process

Model-driven entity ReconcilIAtion (MaRIA) Process takes its name because of the close relationship with the Model-Driven Engineering paradigm and the Entity Reconciliation process.

MaRIA Process emerges proposing a solution to the gap found by Enriquez et al. [9] where it was noticed that all the solutions proposed to carry out an entity reconciliation problem have been defined in the execution phases, that is in the final stages of a software development product. In this sense, it is proposed to analyze this problem from the early stages of the software development trying to reduce costs to the company.

MaRIA Process is defined by a set of activities to be incorporated into any software development methodology of any organization to allow to prepare the system that is being developed to address an entity reconciliation problem. Concretely, this proposal indicates which are the activities and the artifacts necessary to be able to carry out this type of software development. MaRIA Process is an emerging proposal, in this sense, as it is possible to see in Figure 3, it only covers the requirements, analysis and testing (in its early stages) set of processes of the lifecycle of the software product development.

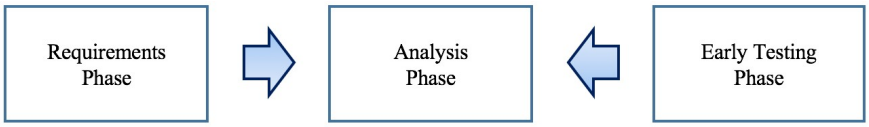

Figure 3 MaRIA Process Phases

It is very important to mention that the definition of the MaRIA Process has been performed according to the recommendations and restrictions of ISO/IEC JTC1 - Big Data [30] preliminary report and in addition, the testing process group has been defined in compliance with the international testing standard ISO/IEC/IEEE 29119 [31].

MaRIA Process (Figure 4) is composed of seven activities and three documents. The requirements phase is composed by two activities: "analyze data sources" and "define data sources report" and one document: the "data sources report". The analysis phase is composed of three activities: "review data sources report", "analyze entity reconciliation problem" and "define analysis model" and one document: the "analysis model". Finally, the early testing phase is composed of two activities: "generate business rules document" and "apply specific criteria" and one document: the "business rule document". Next all these activities and documents will be explained in detail.

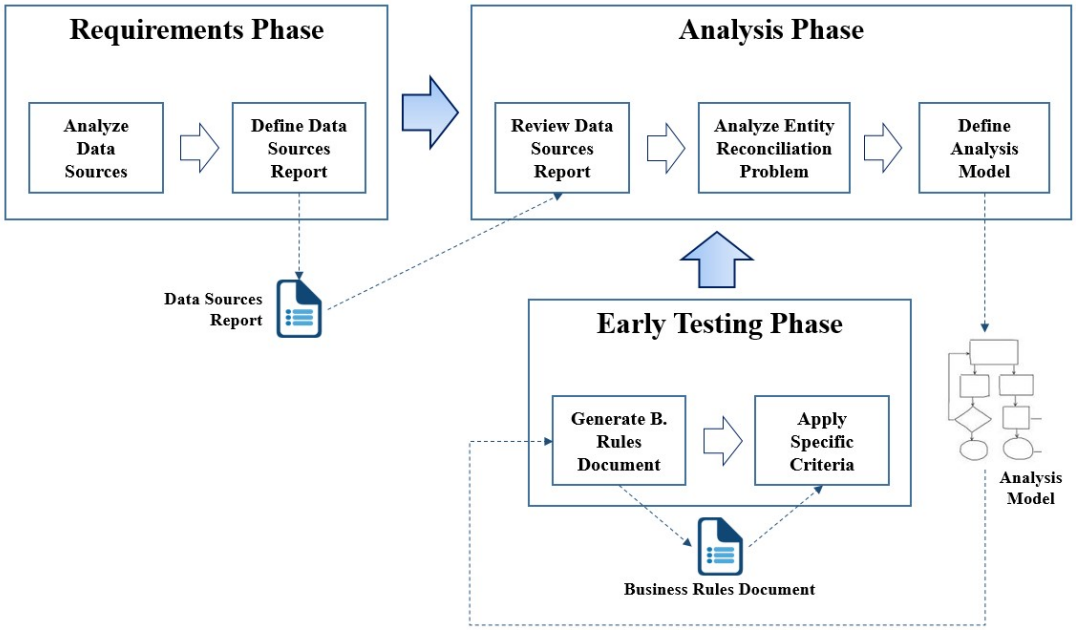

Figure 4 MaRIA Process Description

4.1 Requirements Phase

The main goal of the requirement phase of a software methodology is to create a requirements catalogue that defines the system requirements. In addition to all the activities that this phase proposes in any methodology, MaRIA Process establishes two obligatory activities: analyze data sources and define data sources report.

- **Analyze Data Sources** is the activity where the data analysts study and explore the different data sources that data consumers have presented for their problems. This activity is composed of four main activities (Figure 5). Three of them can be performed in parallel: analyze data

source format, data access and number of data source records. Once these activities have been executed, the data source report is generated. Each of these activities is described in detail as follows.

- **Analyze Data Source Format** is the activity where the data analysts must study and analyze data structure of data sources that data users have defined as relevant. Thus, they must analyze whether each data source refers to a plain text file, a relational database, Oracle, MySQL, a non-relational database or any other type of format.
- **Analyze Data Access** is the activity where the data analysts must study and control the way to access data within data sources that data users have defined as relevant. Thus, they must analyze whether each data source refers to Web service, ODBC or any other type of database connection or access.
- **Analyze Records Number** is the activity where the data analysts must study and supervise the number of records of each data source that data users have defined as relevant (hundreds, thousands, millions or even higher numbers). This activity provides some knowledge and helps to choose the technology that implements the system in the design phase.

- **Generate Data Sources Report** is the activity where the data analysts must create the data sources report from all the information studied and analyzed in previous steps.

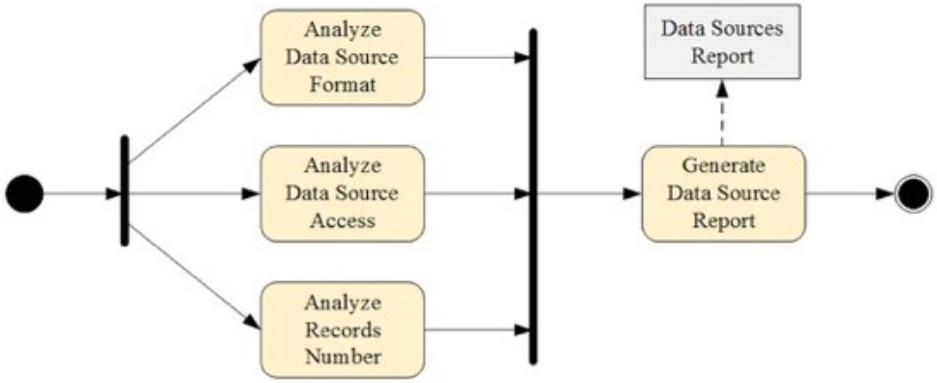

Figure 5 MaRIA Process – Analyze Data Sources

4.2 Analysis Phase

Pressman [32], establishes that the task of the analysis of requirements is a process of discovery, refinement, modelling and specification. It refines in detail the scope of software and creates models of data requirements, information flow and control and operational behavior. In this context, MaRIA Process defines three obligatory activities:

- **Review Data Sources Report** activity is based on the study, analysis and review, carried out by the software engineer, of the strategy document generated in the requirements process

group (this document is received as input for performing this activity). This action provides the software engineer with the knowledge of the data sources needed to model the ER problem in the next activity.

- **Analyze Entity Reconciliation Problem** is the activity where the software engineer must model the ER problem. It involves the supporting tool developed in this research program, MaRIA Tool. This activity can be divided into seven steps (Figure 6). Steps 1 to 4 defining wrappers, data sources, entities and attributes can be performed in parallel. Once done, from steps 5 to 7 the software engineer has to define connectors, data structure and transformations. They are further explained below:

    - **Define Wrappers** is the activity where the software engineer must model the wrappers that allow transferring information from data sources that data users have defined as relevant into the entities that will be defined later. The software engineer must use the wrapper element of MaRIA Tool and model one element for each data source in the diagram.
    - **Define Data Sources** is the activity where the software engineer must model data sources that data users have defined as relevant. The software engineer must use the data source element of MaRIA Tool and model one element for each data source in the diagram.
    - **Define Entities** is the activity where the software engineer must model the entities where the information coming from data sources is stored. The software engineer must use the data entity element of MaRIA Tool and model one element for each data source in the diagram.
    - **Define Attributes** is the activity where the software engineer must model the attributes that compose entities. The software engineer must use the data source attribute element of MaRIA Tool and model as many attributes as each entity needs.
    - **Define Connectors** is the activity where the software engineer must model the connectors among the elements that have been defined in the three previous steps. Such connectors relate wrappers to data sources, data sources to entities and entities to their attributes. The software engineer must use the different types of connector elements that MaRIA Tool offers depending on the elements that need to be associated.
    - **Define Data Structure** is the activity where the software engineer must model the data structure that will store the data reconciled of the final solution. For performing this activity, the user must utilize entity, attribute and connector elements of MaRIA Tool. In this activity, the software engineer must create entities and link them, if necessary, as well as the attributes that describe each entity. For each data source, a structure must be created together with another one that will store the final solution.
    - **Define Data Transformations** is the activity where the software engineer must model data transformations among the different attributes already created or data

structures. The software engineer must use the transformation operations that MaRIA Tool offers and use them for relating attributes or data structures depending on the necessities of the problem.

- **Define Analysis Document** activity is to produce the defined model, once the previous activities have been completed.

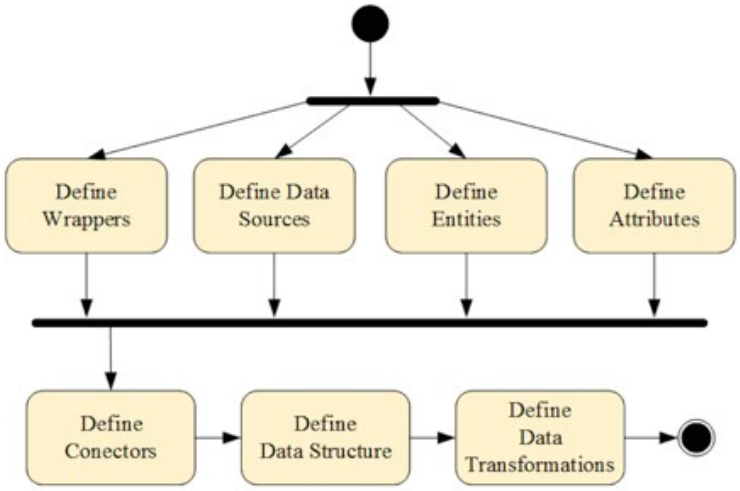

Figure 6 MaRIA Process – Analyze Entity Reconciliation Problem

4.3 Early Testing Phase

One of the most important and critical aspects to improve the quality assurance in software is to improve the testing process by utilizing techniques and tools, which will enhance the software testing process, making it more robust [33]. In this sense, MaRIA Process proposes two activities to carry out: generate business rules document and apply specific criteria and a generation of one new document: the business rules document.

- **Generate Business Rules Document** activity receives the "Analysis Model" generated in the requirements process group as input. This model contains the definition of the ER problem to solve in the software system that it is being developed. It also includes a set of derivations defined for the transformations that the information must undergo during the ER process. The execution of this model generates the "Business Rules Document".
    - **Business Rules Document** is the document generated after performing the execute analysis model activity. This document contains all the business rules that define the test requirements of the software system that it is being developed after the application of concrete criteria.
- **Apply Specific Criteria** activity receives the business rules document as input. To generate test cases of the software system that it is being developed, it is necessary to apply a specific

criterion to the business rules that the business rules document contains. An example of criterion may be Modified Condition/Decision Coverage (MCDC) [34]. This coverage criterion has demonstrated its utility in previous work, such as [35] (for testing SQL queries) and [36] (for testing the user-database interaction).

5 Extension of NDT Methodology

This paper proposes the extension of NDT in order to integrate the entity reconciliation process into a software development process. This integration has been proposed with the aim of formalizing all the activities that must be carried out to perform the entity reconciliation process within a software development. As mentioned before and considering the size of software development process group of NDT, MaRIA Process will cover the requirements, analysis and testing set of processes of this methodology.

Figure 7 shows the extended Software Development Process Group of NDT. The strong and bold boxes show the process blocks that have been extended and also, the new documents and models that must be generated.

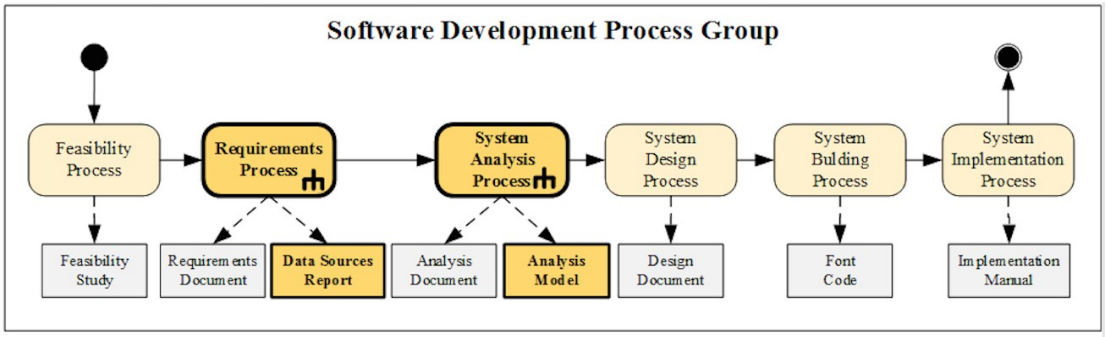

Figure 7 Extended Software Development Process Group of NDT

Figure 8 shows the new Requirements Process of NDT Methodology including what has been defined in MaRIA Process. Figure 9 shows the new Analysis Process of NDT Methodology including what has been defined in MaRIA Process.

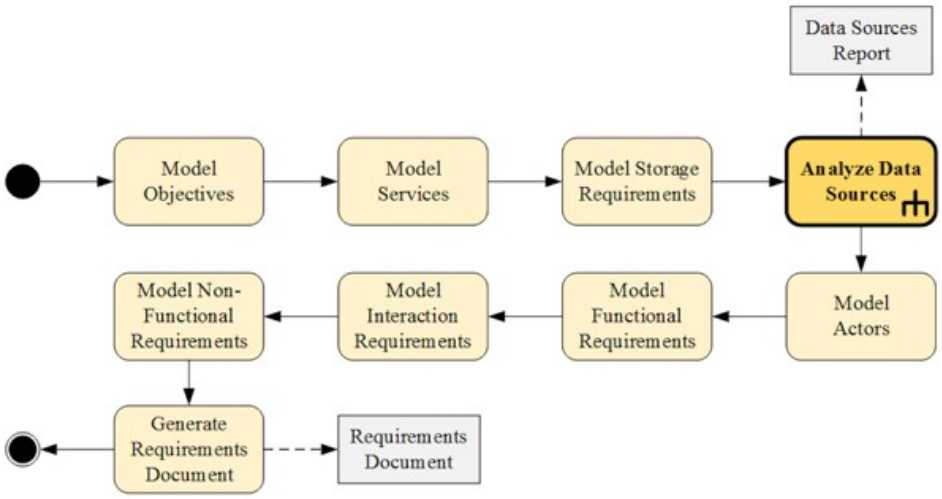

Figure 8 Extended Requirement Process Group of NDT

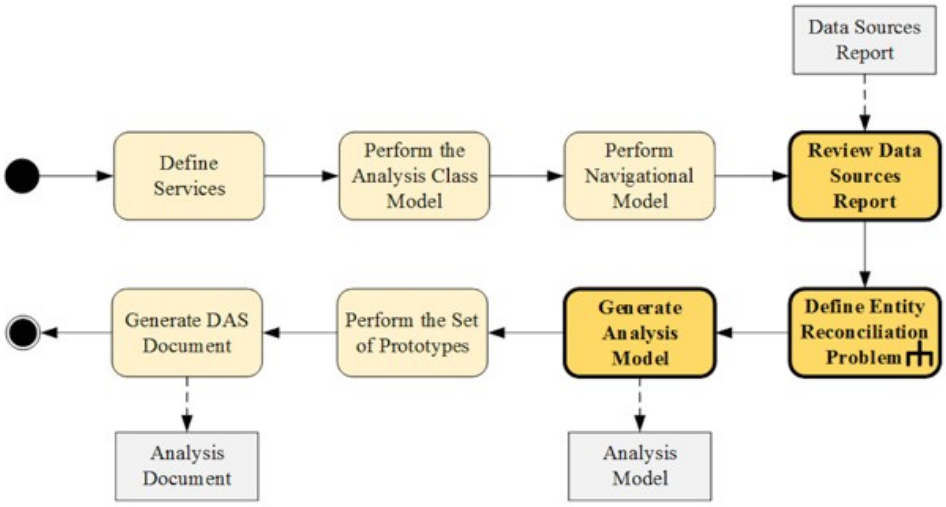

Figure 9 Extended System Analysis Process Group of NDT

Figure 10 shows the extended Testing Process Group of NDT and finally, Figure 11 shows the new Run the Tests Process Group of NDT Methodology including what has been defined in MaRIA Process.

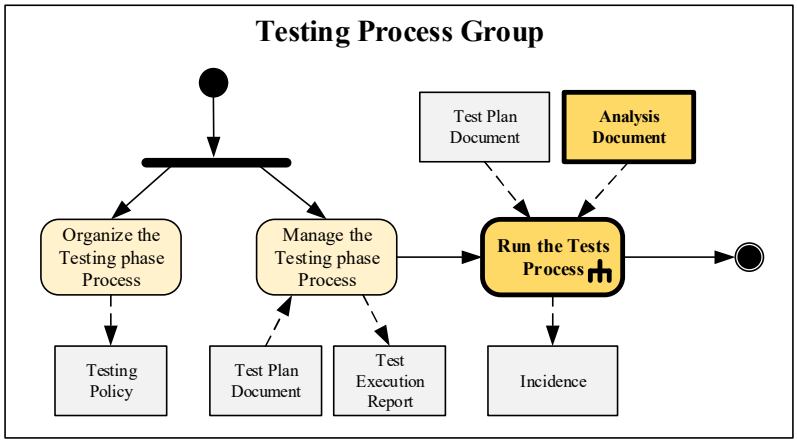

Figure 10 Extended Testing Process Group of NDT

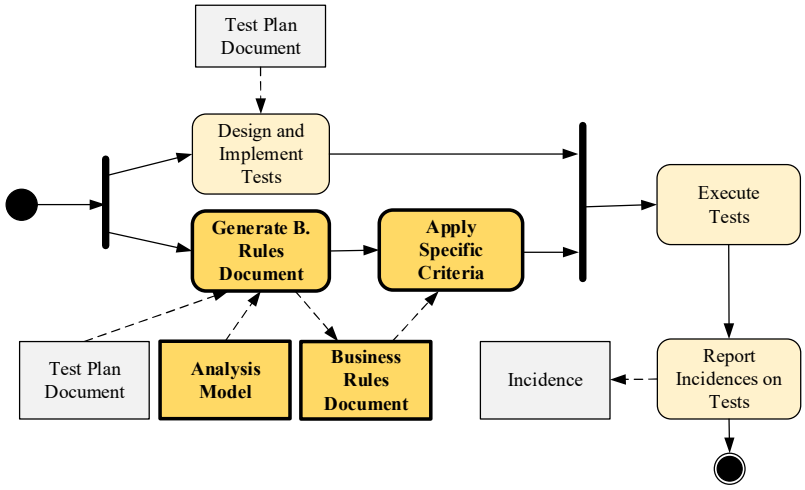

Figure 11 Extended Run the Tests Process Group of NDT

6 Validation

The extension of NDT Methodology with MaRIA Process has been validated in a real-world project named ADAGIO. The main objective of the ADAGIO project is the development of an information system that allows the aggregation, consolidation and normalization of data of various semantic contexts referenced by the geolocation of the objective data. The objective data is obtained from heterogeneous data sources with different levels of aggregation.

The NDT Methodology has been used to model the requirements of the platform to be developed in this project. Considering that ADAGIO project raises an entity reconciliation problem, the MaRIA Process has been used extending NDT Methodology to model this problem. In this context, all the activities proposed by MaRIA Process have been carried out in the different phases of the software development process of ADAGIO project. Then, all these activities and how they have been performed are described. Also it is important to consider that the Early Testing phase has not yet been covered in the project.

As stated before, the activities defined for the requirements phase are: analyze data sources and define data sources report. In this sense, software engineers of ADAGIO Project have defined this document studying all the characteristics that the data source to be reconciled in ADAGIO offers. Three big data repositories have been analyzed in this project: the national catalogue of open data of the Government of Spain (GOV) [37], The World Bank (TWB) [38] and the International Labour Organization (ILO) [39]. For each data source studied from these repositories, the following characteristics have been analyzed: name of data source, format, observations, URL, domain, language, number of records, geographical coverage and access type. An example of the analysis of three very different data sources is given in Table 1.

Name	Format	Obs.	URL	Domain	Lang	Records	Geolocation	Access
Hospital morbidity survey 2015	XLS	N/A	Link	Sanitary	Spanish	199 series of 199 cells	Spain	Plain text
Local Cultural Centers	RDF	N/A	Link	Social	Spanish	2.464 triples	Madrid	API
Caceres Restaurants	RDF	N/A	Link	Entertainment	Spanish	1.571 triples	Caceres	API

Table 1. Extract data sources report

Finally, the MaRIA Tool (the tool proposed to give support to MaRIA Process) has been used to define the ER problem. Figure 12 shows a reduced version of the ER model proposed for ADAGIO Project. In total, 17 data sources have been modelled, however, to simplify Figure 12, only two of them are shown. The two sources of data refer to information related to hospitals that are distributed in a dispersed way.

7 Conclusions and Future Work

This paper, has presented the MaRIA Process, a set of activities, documents and artefacts to be added in any software development methodology to prepare the system to be developed to carry out an entity reconciliation problem.

To illustrate how this process may be added into a software development methodology, it has been proposed an extension of the NDT methodology with the aim of giving support to cover an entity reconciliation problem during a system software development.

Concretely and considering what MaRIA Process defines, it has been extended the requirements, system analysis and testing set of processes of that NDT methodology proposes.

The activities proposed for the requirement process aim to understand the problem as well as the data sources that must be reconciled for giving a solution to the entity reconciliation problem. The activities proposed for the system analysis process aim to model the entity reconciliation problem. To achieve this purpose, a DSL-based tool called MaRIA Tool is proposed. The activities proposed for the Testing process aim to systematically test the future system to develop based on what has been specified in requirements and analysis processes.

Considering that NDT Methodology covers all the processes that allow to develop a complete software system, the main future work of this proposal is based on: (i) to extend MaRIA Process to cover all the set of processes of a software development and (ii) to extend NDT Methodology to cover the remaining blocks of processes that have not been covered yet. In addition, ADAGIO Project is now involved in the Early Testing phase and soon it will be possible to obtain the rules to test the final application once developed.

Also, new methodologies are being considered in order to see how the activities described and added to the NDT Methodology should be integrated for checking the scalability of this proposal.

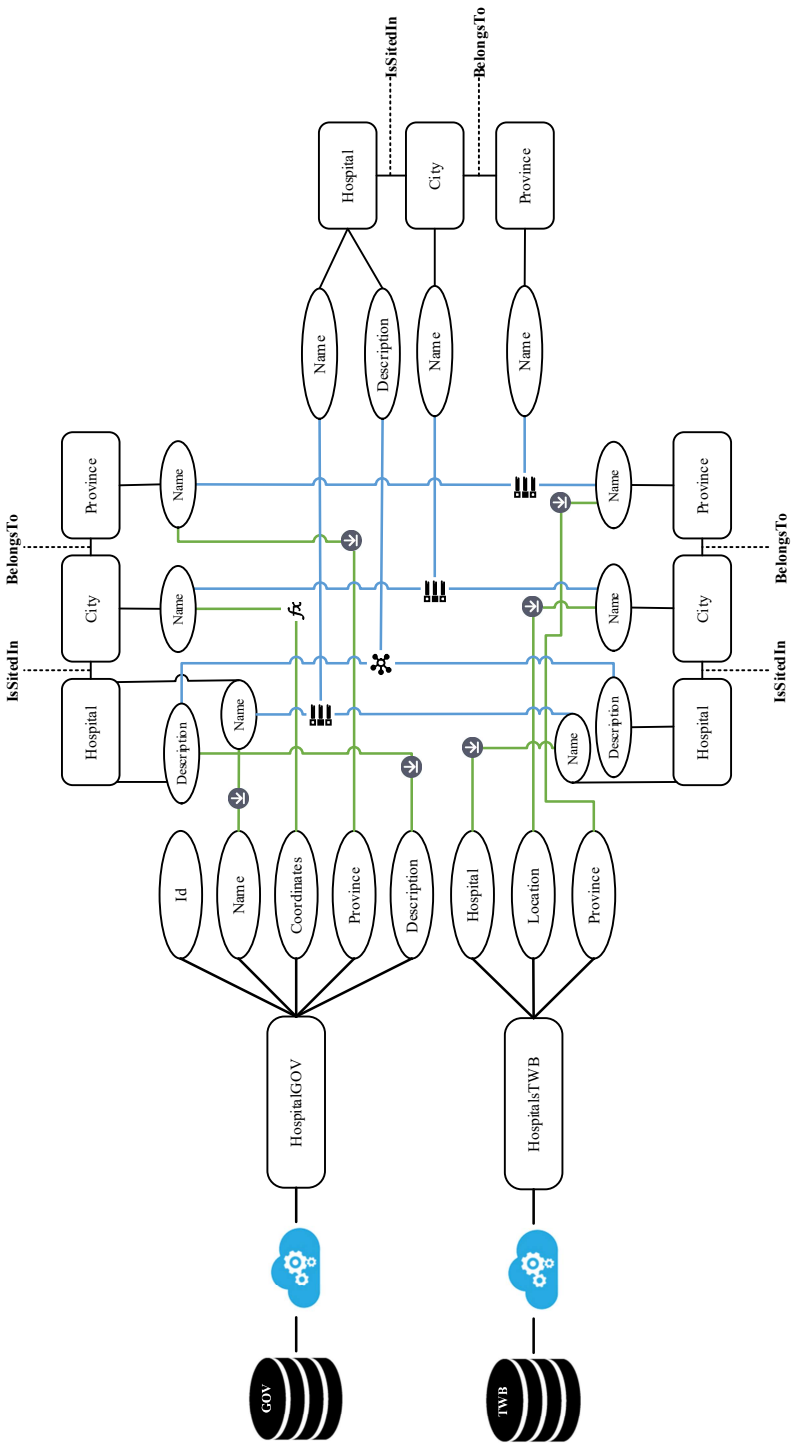

Figure 12 Reduced version of entity reconciliation model for ADAGIO Project

Acknowledgements

This research has been supported by the MeGUS project (TIN2013-46928-C3-3-R), Pololas project (TIN2016-76956-C3-2-R), by the SoftPLM Network (TIN2015-71938-REDT) of the Spanish the Ministry of Economy and Competitiveness and Fujitsu Laboratories of Europe (FLE).

References

[1] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Futur. Gener. Comput. Syst.*, vol. 29, no. 7, pp. 1645–1660, 2013.

[2] J. G. Enríquez, R. Blanco, F. J. Domínguez-Mayo, J. Tuya, and M. J. Escalona, "Towards an MDE-based Approach to Test Entity Reconciliation Applications," in *Proceedings of the 7th International Workshop on Automating Test Case Design, Selection, and Evaluation*, 2016, pp. 74–77.

[3] N. Koch, A. Knapp, G. Zhang, and H. Baumeister, "UML-based web engineering: An Approach Based on Standards," *Web Eng. Model. Implement. Web Appl.*, pp. 157–191, 2008.

[4] S. Ceri, P. Fraternali, and A. Bongio, "Web modeling language (WebML): a modeling language for designing Web sites," *Comput. Networks*, vol. 33, no. 1, pp. 137–157, 2000.

[5] S. Meliá, J. Gómez, S. Pérez, and O. Díaz, "A model-driven development for GWT-based rich internet applications with OOH4RIA," in *Proceedings - 8th International Conference on Web Engineering, ICWE 2008*, 2008, pp. 13–23.

[6] M. Linaje, J. C. Preciado, and F. Sánchez-Figueroa, "Engineering rich internet application user interfaces over legacy web models," *IEEE Internet Comput.*, vol. 11, no. 6, pp. 53–59, 2007.

[7] M. J. Escalona and G. Aragón, "NDT. A model-driven approach for web requirements," *IEEE Trans. Softw. Eng.*, vol. 34, no. 3, pp. 377–394, 2008.

[8] F. J. DOMINGUEZ-MAYO, M. J. ESCALONA, M. MEJIAS, M. ROSS, and G. STAPLES, "Towards a Homogeneous Characterization of The Model-Driven Web Development Methodologies," *J. web Eng.*, vol. 13, no. 1–2, pp. 129–159, 2014.

[9] J.G. Enríquez, F.J. Domínguez-Mayo, M.J. Escalona, M. Ross, and G. Staples, "Entity Reconciliation in Big Data Sources: a Systematic Mapping Study," *Expert Syst. Appl.*, vol. 80, pp. 14–27, 2017.

[10] S.-M.-R. Beheshti, B. Benatallah, S. Venugopal, S. H. Ryu, H. R. Motahari-Nezhad, and W. Wang, "A systematic review and comparative analysis of cross-document coreference resolution methods and tools," *Computing*, pp. 1–37, 2016.

[11] G. Papadakis, J. Svirsky, A. Gal, and T. Palpanas, "Comparative Analysis of Approximate Blocking Techniques for Entity Resolution," *Pvldb*, vol. 9, no. 9, pp. 684–695, 2016.

[12] S. Cucerzan, "Large-Scale Named Entity Disambiguation Based on Wikipedia Data," in *EMNLP-CoNLL 2007*, 2007, pp. 708–716.

[13] A. Moro, A. Raganato, and R. Navigli, "Entity Linking meets Word Sense Disambiguation: a Unified Approach," *Trans. Assoc. Comput. Linguist.*, vol. 2, no. 0, pp. 231–244, 2014.

[14] W. Shen, J. Wang, and J. Han, "Entity linking with a knowledge base: Issues, techniques, and solutions," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 2, pp. 443–460, 2015.

[15] X. Liu, Y. Li, H. Wu, M. Zhou, and Y. L. Furu Wei, "Entity Linking for Tweets," *Acl '13*, pp. 1304–1311, 2013.

[16] L. García-Borgoñón, "Un marco de referencia para facilitar la interoperabilidad y mantenibilidad de los modelos de procesos de software," 2015.

[17] D. C. Schmidt, "Guest Editor's Introduction : Model-Driven Engineering," *IEEE Comput.*, vol. 39, no. 2, pp. 25–31, 2006.

[18] M. Brambilla, J. Cabot, and M. Wimmer, *Model-Driven Software Engineering in Practice*,

vol. 1, no. 1. 2012.

[19] A. Metzger, “A Systematic Look at Model Transformations,” *Nature*, vol. 451, no. 7, pp. 644–647, 2008.
[20] S. Mellor, K. Scott, A. Uhl, and D. Weise, “MDA Distilled - Principles of Model Driven Architecture,” *Addison Wesley*, 2004.
[21] L. Thiry and B. Thirion, “Functional metamodels for systems and software,” *J. Syst. Softw.*, vol. 82, no. 7, pp. 1125–1136, 2009.
[22] J. Bézivin, “On the unification power of models,” *Softw. Syst. Model.*, vol. 4, no. 2, pp. 171–188, 2005.
[23] F. J. Domínguez-Mayo, M. J. Escalona, and M. Mejías, “QuEF (Quality Evaluation Framework) for model-driven web methodologies,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2010, vol. 6385 LNCS, pp. 571–575.
[24] F. J. Domínguez-Mayo, M. J. Escalona, and M. Mej ías, “Quality issues on model-driven web engineering methodologies,” in *Information Systems Development: Asian Experiences*, 2011, pp. 295–306.
[25] F. J. Dominguez-Mayo, M. J. Escalona, M. Mejias, and a. H. Torres, “A Quality Model in a Quality Evaluation Framework for MDWE methodologies,” *Res. Challenges Inf. Sci. (RCIS), 2010 Fourth Int. Conf.*, 2010.
[26] L. Getoor and A. Machanavajjhala, “Entity resolution: Theory, practice & open challenges,” *Proc. VLDB Endow.*, vol. 5, no. 12, pp. 2018–2019, 2012.
[27] F. Wang, H. Wang, J. Li, and H. Gao, “Graph-based reference table construction to facilitate entity matching,” *J. Syst. Softw.*, vol. 86, no. 6, pp. 1679–1688, 2013.
[28] A. McCallum, K. Nigam, and L. H. Ungar, “Efficient clustering of high-dimensional data sets with application to reference matching,” *Proc. sixth ACM SIGKDD Int. Conf. Knowl. Discov. data Min. KDD 00*, pp. 169–178, 2000.
[29] S. E. Whang and H. Garcia-Molina, “Incremental entity resolution on rules and data,” *VLDB J.*, vol. 23, no. 1, pp. 77–102, 2014.
[30] ISO/IEC JTC 1, “ISO/IEC CD 20546 - Big data report,” vol. 31, no. 5, pp. 498–513, 2014.
[31] ISO/IEC/IEEE, “INTERNATIONAL STANDARD ISO/IEC/IEEE 29119,” vol. 2013, 2013.
[32] R. S. Pressman, *Software Engineering A Practitioner’s Approach 7th Ed - Roger S. Pressman*. 2009.
[33] J. G. Enríquez, J. A. García-García, F. J. Domínguez-Mayo, and M. J. Escalona, “ALAMEDA Ecosystem: Centering efforts in Software Testing Development,” *Qual. Control Assur. - An Anc. Greek Term Re-Mastered*, vol. 1, no. 1, pp. 155–172, 2017.
[34] J. J. Chilenski, “An investigation of three forms of the modified condition decision coverage (MCDC) criterion,” *Security*, no. April, 2001.
[35] J. Tuya, M. J. Suárez-Cabal, and C. De La Riva, “Full predicate coverage for testing SQL database queries,” *Softw. Test. Verif. Reliab.*, vol. 20, no. 3, pp. 237–288, 2010.
[36] R. Blanco, J. Tuya, and R. V. Seco, “Test adequacy evaluation for the user-database interaction: A specification-based approach,” in *Proceedings - IEEE 5th International Conference on Software Testing, Verification and Validation, ICST 2012*, 2012, pp. 71–80.
[37] Goverment of Spain. Retrieved January 2018 from: “http://datos.gob.es/.”
[38] The World Bank (TWB). Retrieved January 2018 from: “www.worldbank.org.”
[39] International Labour Organzacion (ILO). Retrieved January 2018 from: “www.ilo.org”