An Extension of NDT to Model Entity Reconciliation Problems

J. G. Enríquez, F. J. Domínguez-Mayo, J. A. García-García and M. J. Escalona
Computer Languages and Systems Department, University of Seville, Av. Reina Mercedes s/n, 41012, Seville, Spain

Keywords: Web Engineering, Model-driven Engineering, Entity Reconciliation, NDT.

Abstract: 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 grotesque the volume of information that it is generated daily. In the management of all this information, it appears the entity reconciliation problem, 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. To this end, the NDT methodology has been taken as a reference and has been extended adding new activities, artefacts and documents to cover this problem.

1 INTRODUCTION

At present, the development and creation of web applications is imposed in the world as a technological tool to unite the 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 (Gubbi et Al., 2013), it appears a very big and difficult problem when someone tries to look for an information, the heterogeneity of data.

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 the most of these data sources do not share the same structure and even data that they store, even though they are related to the same subject, may not be the same.

In this context it appears the problem of the entity reconciliation. 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 equals.

This paper aims to give a solution to this kind of problems through a model-driven web perspective. To achieve this goal, it has been taken as reference the Navigational Development Techniques (NDT) methodology and it has been extended with a set of activities and documents to carry out an entity reconciliation problem. It has been selected NDT because of the

The remainder of this paper is organized as follow: section 2 summarizes the background of this paper, presenting the two main pillar of this proposal, MDE and NDT. Section 3 describes the problem of the entity reconciliation. In section 4 it is explained the different activities that this paper proposes in order to extend NDT for covering an entity reconciliation problem into a software development process. Finally, section 5 states a set of conclusions and future works.

2 BACKGROUND

2.1 Model-Driven Engineering (MDE)

In the scientific activity, abstraction has been and is widely used, and often referred to it as the activity of modeling. 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 (García-Borgoñón, 2015).

Enríquez, J., Domínguez-Mayo, F., Escalona, M. and García-García, J.
An Extension of NDT to Model Entity Reconciliation Problems.
DOI: 10.5220/0006393604300436
In *Proceedings of the 13th International Conference on Web Information Systems and Technologies (WEBIST 2017)*, pages 430-436
ISBN: 978-989-758-246-2

The complexity of software development has been growing up drastically. In this sense, developers noted 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 (Schmidt, 2006). 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., 2012) defined 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 (Metzger, 2008). 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 moment, it appears the concept of metamodel. 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 (Mellor et al., 2004).
- **Transformations** are the mechanisms that allow 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 (Thiry & Thirion, 2009). 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, (2005) 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.

2.2 Navigational Development Techniques (NDT)

NDT is a Model-Driven Engineering-based methodology that provides formal and complete set of processes that allow to support for 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 (Escalona & Aragón, 2008).

NDT Framework (Figure 1), stablishes six big groups of processes that allow 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.

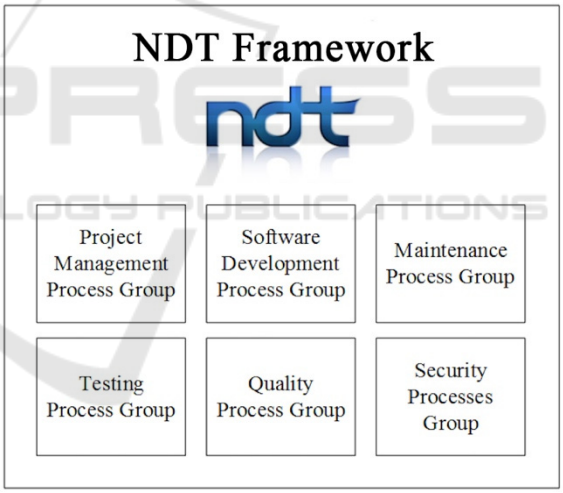

Figure 1: NDT Framework.

Software development process group describes all the processes, activities, artefacts and documents that makes a software engineer be able to satisfy the software development lifecycle. In this sense, NDT methodology, to carry out this objective, stablishes the following set of processes: feasibility process, requirements process, system analysis process, system design process, system building process and system implementation process.

- **Feasibility Process**. In this process the feasibility of a particular software project

should be analysed. It is usually done on demand of the project or when the project is developed in a complex or little known environment that may make the feasibility of the project doubtful.

- **Requirement Process**. Defines a requirements catalogue that must define the system requirements. The catalogued requirements should be established according to their typology and should not go into any detail about how development will be solved.
- **System Analysis Process**. Products resulting from the analysis, definition and structuring of the requirements established in the previous process, independently of the technological platform that is finally used to develop the software.
- **System Design Process**. Specific aspects of how the analysis will be implemented in the machine. It is oriented to the concrete platform with which it is going to work and must correspond with the structure of the future code.
- **System Building Process**. This process refers to the implementation or development of the software system.
- **System Implementation Process**. This process refers to the implementation period of the product developed in the final customer.

In this context, what this paper proposes is 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 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 (Getoor and Machanavajjhala, 2012; Wang et al., 2013).

Figure 2, illustrates a very clear example of entity reconciliation proposed by McCallum et al., (2000). 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.

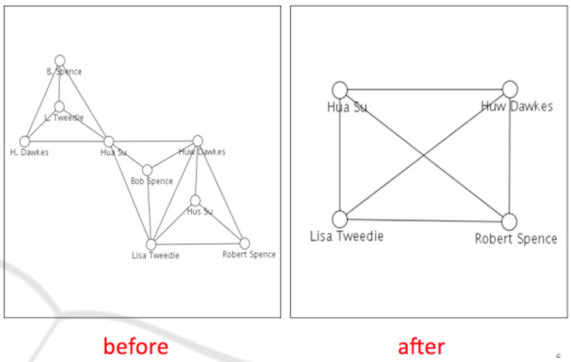

Figure 2: Entity Reconciliation example (McCallum et al., 2000).

Entity reconciliation is a well-known problem and it has been investigated since the birth of relational databases (Whang & Garcia-Molina, 2014) and it can be applied to many different domains. If to everything mentioned, 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 it arises.

4 NDT-RECONCILIATION

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 taking into account the size of software development process group of NDT, this paper will only cover the block of requirement and system analysis processes.

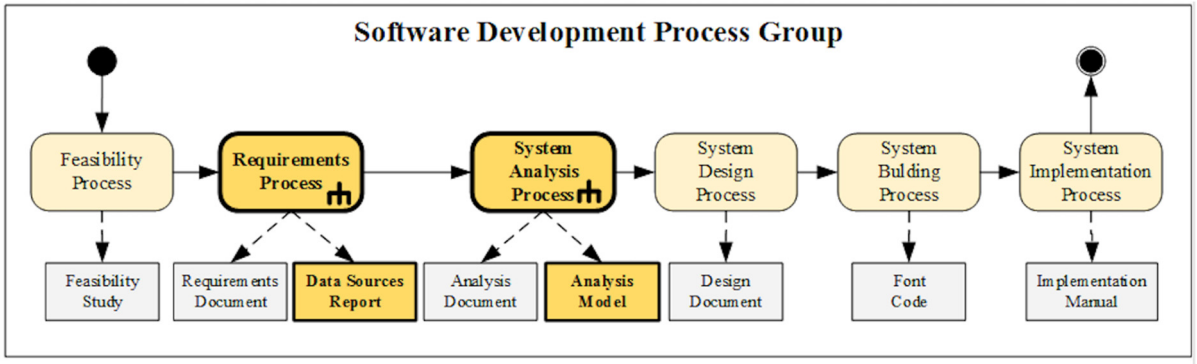

Figure 3: Extended Software Development Process Group of NDT.

Figure 3 shows the extended software development process group of NDT. The strong and bold yellow boxes show the process blocks that have been extended and also, the new documents that must be generated. This extension, proposes the generation of two new documents: the data sources report and the analysis model.

The main goal of the requirement process of NDT is to define a requirements catalogue that define the system requirements. To achieve this goal, the following activities are proposed: model objectives, model services, model storage requirements, model actors, model functional requirements, model interaction requirements, model non-functional requirements and generate requirements document. This proposal presents a new activity called: "Analyse Data Sources" (Figure 4).

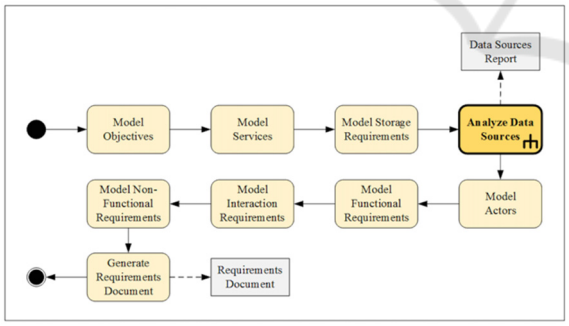

Figure 4: Extended Requirement Process Group of NDT.

"Analyse Data Sources" activity is the activity where the data analysts study and analyse the different data sources that the data consumers have presented for their problem. This activity is divided in four main steps (Figure 5). Three of them can be performed in parallel: analyse data source format, analyse data access and analyse number of records of the data source. Once these activities have been finished, it takes place the generate data sources report activate where the data source report is generated.

- **Analyse Data Source Format** is the activity where the data analysts will have to study and analyse the data structure of the data sources that the data consumers have defined as problematics. In this sense, they have to analyse if each data source refers to a plane text file, a relational database, Oracle, MySQL, a non-relational database or any other type.
- **Analyse Data Access** is the activity where the data analysts will have to study and analyse the data access way of the data sources that the data consumers have defined as problematics. In this sense, they have to analyse if each data source refers web service, ODBC any other type of database connection or access.
- **Analyse Records Number** is the activity where the data analysts will have to study and analyse the number of records of each of the data source that the data consumers have defined as problematics (hundred, thousands, millions, etc.). This activity will provide some knowledge and help to choose the technology in which the system will be implemented in the design phase.
- **Generate Data Sources Report** it the activity where the data analysts will have to create the data sources report with all the information studied and analysed in previous steps.

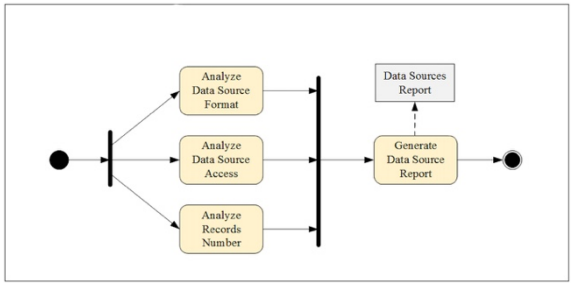

Figure 5: Description of "Analyse Data Sources" activity.

After Requirement Process block, NDT methodology continues with the System Analysis Process. Once this process is completed, it is obtained products resulting from the analysis, definition and structuring of the requirements established in the previous phase. This block of processes is composed of a set of activities, these are: define services, perform the analysis class model, perform navigational model, perform the set of prototypes and generate the DAS document. This proposal presents three new activities called: "Review Data Sources Report", "Define Entity Reconciliation Problem" and "Generate Analysis Model". (Figure 6).

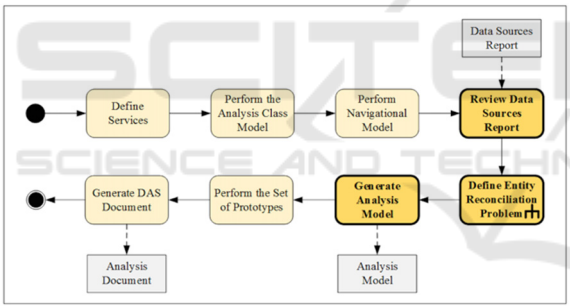

Figure 6: Extended System Analysis Process Group of NDT.

"Review the Strategy Document" activity is based on the on the study, analysis and review by the software engineer of the strategy document generated in the requirement process. This activity receives as input the strategy document generated in the requirement process and generates the analysis document model as result. The software engineers in charge of carrying out this phase will have to, with the support of the strategy document, analyze data stored in each database, in order to define: (i) a clear vision of what represents an entity in the problem where the user is working on (studying their attributes and relationships) and (ii), what will be the data structure in which the result of reconciliation will be stored.

"Define the Entity Reconciliation Problem" activity (Figure 7) is the activity where the software engineer has to model the entity reconciliation problem. This activity will be carried out through the use of the MaRIA Tool (Enríquez et al., 2015). MaRIA (Model-Driven entity ReconcilIAtion) Tool is a domain specific language (DSL) (Cook, Jones, Kent, & Wills, 2007) that allows a software engineer to model entity reconciliation problems. This activity is divided in seven main steps. The four first ones, define wrappers, define data sources, define entities, define attributes, can be performed in parallel. Once defined, the software engineer hast to define the connectors, the data structure and finally, define the transformations.

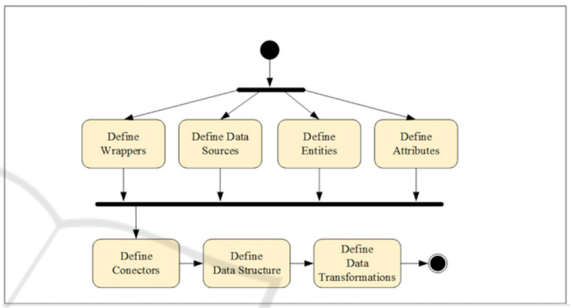

Figure 7: Description of "Define the Entity Reconciliation Problem" activity.

- **Define Wrappers** is the activity where the software engineer will have to model the wrappers that will allow the transfer of information from the data sources that the data consumers have defined as problematics into the entities that will be defined later. In this sense, the software engineer has to use the wrapper element of the MaRIA tool and model in the diagram one element for each data source.
- **Define Data Sources** is the activity where the software engineer will have to model the data sources that the data consumers have defined as problematics. In this sense, the software engineer has to use the data source element of the MaRIA tool and model in the diagram one element for each data source.
- **Define Entities** is the activity where the software engineer will have to model the entities where the information coming from the data sources will be stored in. In this sense, the software engineer has to use the data entity element of the MaRIA tool and model in the diagram one element for each data source.

- **Define Attributes** is the activity where the software engineer will have to model the attributes that compose the entities. In this sense, the software engineer has to use the data source attribute element of the MaRIA tool and model as many attributes as each entity needs.
- **Define Connectors** is the activity where the software engineer will have to model the connectors between the elements that have been defined in the three previous steps. The connectors will relate the wrappers with the data sources, the data sources with the entities and the entities with their attributes. In this sense, the software engineer has to use the different types of connectors element that MaRIA tool offers depending on the elements needed to be related.
- **Define Data Structure** is the activity where the software engineer will have to model the data structure where the data reconciled of the final solution will be stored. For performing this activity, the user will have to use the entity, attribute and connector elements of the MaRIA tool. In this software engineer will have to create the entities, related between them if necessary and the attributes that describe each entity. For each data source must be created a structure and in addition to these, it has to be created another one that will store the final solution.
- **Define Data Transformations** is the activity where the software engineer will have to model the data transformations between the different attributes already created or between the data structures. In this sense, the software engineer has to use the transformation operations that the MaRIA tool offers and use it for relating attributes or data structures depending on the necessities of the problem.

Finally, the "Generate Analysis Document" activity is presented. The main function of this activity is to collect all the information generated in the different previous activities of the analysis phase in a single document, which will serve as a knowledge base and input for the design phase. Also it will be provided the model defined using the MaRIA tool.

5 CONCLUSIONS AND FUTURE WORK

In this paper, 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 taking into account the scope of this paper, it has been extended the requirement and system analysis processes of the NDT Framework 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 is proposed.

Taking into account that NDT Methodology covers the complete process of the software development, the main future work of this proposal is to extend this methodology in order to cover the remaining blocks of processes that have not been covered in this proposal. Also, new methodologies are being considered in order to see how the activities described and added to the NDT should be integrated for checking the scalability of this proposal.

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

Bézivin, J. (2005). On the unification power of models. *Software and Systems Modeling*, *4*(2), 171–188. http://doi.org/10.1007/s10270-005-0079-0

Brambilla, M., Cabot, J., & Wimmer, M. (2012). *Model-Driven Software Engineering in Practice*. *Synthesis Lectures on Software Engineering* (Vol. 1). http://doi.org/10.2200/S00441ED1V01Y201208SWE001

Cook, S., Jones, G., Kent, S., & Wills, A. C. (2007). *Domain Specific Development with Visual Studio DSL Tools*. *Library*.

Enríquez, J. G., Domínguez-Mayo, F. J., Escalona, M. J., García-García, J. A., Lee, V., & Masatomo, G. (2015). Entity Identity Reconciliation based Big Data Federation-A MDE approach. In *International Conference on Information Systems Development (ISD2015)*. Retrieved from http://aisel.aisnet.org/isd2014/proceedings2015/MDD Concepts/7/

Escalona, M. J., & Aragón, G. (2008). NDT. A model-driven approach for web requirements. *IEEE Transactions on Software Engineering*, *34*(3), 377–394. http://doi.org/10.1109/TSE.2008.27

García-Borgoñón, L. (2015). *Un marco de referencia para facilitar la interoperabilidad y mantenibilidad de los modelos de procesos de software*.

Getoor, L., & Machanavajjhala, A. (2012). Entity resolution: Theory, practice & open challenges. *Proceedings of the VLDB Endowment*, *5*(12), 2018–2019. http://doi.org/10.14778/2367502.2367564

Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, *29*(7), 1645–1660. http://doi.org/10.1016/j.future.2013.01.010

McCallum, A., Nigam, K., & Ungar, L. L. H. (2000). Efficient clustering of high-dimensional data sets with application to reference matching. *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 169–178. http://doi.org/10.1145/347090.347123

Mellor, S., Scott, K., Uhl, A., & Weise, D. (2004). MDA Distilled - Principles of Model Driven Architecture. *Addison Wesley*. Retrieved from http://dx.doi.org/10.1007/3-540-46105-1_33

Metzger, A. (2008). A Systematic Look at Model Transformations. *Nature*, *451*(7), 644–647. http://doi.org/10.1038/451644a

Schmidt, D. C. (2006). Guest Editor's Introduction : Model-Driven Engineering. *IEEE Computer*, *39*(2), 25–31. http://doi.org/http://doi.ieeecomputersociety.org/10.1109/MC.2006.58

Thiry, L., & Thirion, B. (2009). Functional metamodels for systems and software. *Journal of Systems and Software*, *82*(7), 1125–1136. http://doi.org/10.1016/j.jss.2009.01.042

Wang, F., Wang, H., Li, J., & Gao, H. (2013). Graph-based reference table construction to facilitate entity matching. *Journal of Systems and Software*, *86*(6), 1679–1688. http://doi.org/10.1016/j.jss.2013.02.026

Whang, S. E., & Garcia-Molina, H. (2014). Incremental entity resolution on rules and data. *VLDB Journal*, *23*(1), 77–102. http://doi.org/10.1007/s00778-013-0315-0