Universidade do Minho

Escola de Engenharia

Fernando Tiago Fernandes da Silva Pereira

Methodological Approaches and Techniques for Designing Ontologies in Information Systems Requirements Engineering

December 2022

Methodological Approaches and Techniques for Designing Ontologies in Information Sysytems Requirements Engineering

Fernando Tiago Fernandes da Silva Pereira

UMinho | 2022

Funding:

Centro de Computação Gráfica
Investigação & Desenvolvimento Tecnológico
Research & Technological Development

CENTROALGORITMI

SEMAG
software engineering and management group

Universidade do Minho
Escola de Engenharia

This work was carried out within the project "PHC Voice Framework" reference LISBOA-01-0247-FEDER-045341, co-funded by Fundo Europeu de Desenvolvimento Regional (FEDER), through Portugal 2020 (P2020).

This work was carried out within the project "IDT4CTI - Interoperability and Digital Thread for a Competitive Textile Industry" reference POCI-01-0247-FEDER-045343, co-funded by Fundo Europeu de Desenvolvimento Regional (FEDER), through Portugal 2020

This work was carried out within the project "iITEC Smart Automation I4.0" reference NORTE-01-0247-FEDER-045073, co-funded by Fundo Europeu de Desenvolvimento Regional (FEDER), through Portugal 2020 (P2020).

This work was carried out within the project "CityCatalyst" reference POCI/LISBOA-01-0247-FEDER-046119, co-funded by Fundo Europeu de Desenvolvimento Re-gional (FEDER), through Portugal 2020 (P2020).

This work was carried out within the project "STVgoDigital: Digitalização da Cadeia de Valor do STV" reference POCI-01-0247-FEDER-046086, co-funded by Fundo Europeu de Desenvolvimento Regional (FEDER), through Portugal 2020 (P2020).

COMPETE 2020
PORTUGAL 2020
UNIÃO EUROPEIA
Fundo Europeu de Desenvolvimento Regional
FCT Fundação para a Ciência e a Tecnologia



Universidade do Minho
Escola de Engenharia

Fernando Tiago Fernandes da Silva Pereira

Methodological Approaches and Techniques for Designing Ontologies in Information Systems Requirements Engineering

PhD Thesis
Doctoral Program in Information Systems and Technology

Work done under the guidance of
Professor Doutor Ricardo J. Machado
Doutor Carlos E. Salgado

December 2022

Copyright

Este é um trabalho académico que pode ser utilizado por terceiros desde que respeitadas as regras e boas práticas internacionalmente aceites, no que concerne aos direitos de autor e direitos conexos.

Assim, o presente trabalho pode ser utilizado nos termos previstos na licença abaixo indicada.

Caso o utilizador necessite de permissão para poder fazer um uso do trabalho em condições não previstas no licenciamento indicado, deverá contactar o autor, através do RepositóriUM da Universidade do Minho.

Licença concedida aos utilizadores deste trabalho



Atribuição-NãoComercial

CC BY-NC

https://creativecommons.org/licenses/by-nc/4.0/

Acknowledgments

I dedicate this space to all those who are present along my path, and who directly or indirectly helped me to achieve my goals. Being aware that it is not possible to thank everyone as I should, there are, however, some to whom I cannot fail to express my appreciation and deep and sincere gratitude.

To the director of the doctoral program in Information Systems, Professor João Varajão, I thank the opportunity I had to attend this program that allowed me to become richer in knowledge.

To Professor Ricardo J. Machado, I express my most sincere and profound gratitude for his guidance and for the unconditional support he always had for me, for having fed me with his wisdom and knowledge, for his opinions, for his criticism, for his words of incentive, for his excellent professionalism, for his availability, and also for his friendship. It is a privilege to have him as a mentor.

I would like to thank Professor Carlos E. Salgado for his co-guidance in this project, for the availability that he always gave me, for the great help in this work, for the teachings that he transmitted to me that allowed me to evolve, and also for his friendship. It is definitely an excellent help.

I would like to give special thanks to the CCG/ZGDV institution and all the colleagues that share the day to day work with me.

To my friends and family for their affection, constant support and strength, my enormous gratitude.

To my wife, for always having been with me, for the trust she always put in me, for her patience, for her affection, for the words of encouragement that made me never give up, for the strength she always transmitted to me, for laughing with me in the best moments and for never letting me get discouraged in the most difficult moments of this journey "Thank you Vânia de Sousa".

To my mother Teresa Fernandes, my father Fernando Pereira, João Bravo, and my brother João Pedro, thank you very, very much, for always believing in me, for supporting me, for helping me get around obstacles, for not letting me down, for laughing with me, for conversations, for concern, for teaching me the most important values, and for the pride they have in me.

Finally, to my grandparents Herminio and Cândida, although they are no longer present, my gratitude for the importance they had and continue to have in my life.
To them I dedicate this PhD.

—— This page is intentionally left blank ——

Declaration of integrity

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the University of Minho.

—— This page is intentionally left blank ——

Abstract

The way we interact with the world around us is changing as new challenges arise, embracing innovative business models, rethinking the organization and processes to maximize results, and evolving change management. Currently, and considering the projects executed, the methodologies used do not fully respond to the companies' needs. On the one hand, organizations are not familiar with the languages used in Information Systems, and on the other hand, they are often unable to validate requirements or business models. These are some of the difficulties encountered that lead us to think about formulating a new approach. Thus, the state of the art presented in this paper includes a study of the models involved in the software development process, where traditional methods and the rivalry of agile methods are present. In addition, a survey is made about Ontologies and what methods exist to conceive, transform, and represent them.

Thus, after analyzing some of the various possibilities currently available, we began the process of evolving a method and developing an approach that would allow us to design ontologies. The method we evolved and adapted will allow us to derive terminologies from a specific domain, aggregating them in order to facilitate the construction of a catalog of terminologies. Next, the definition of an approach to designing ontologies will allow the construction of a domain-specific ontology. This approach allows in the first instance to integrate and store the data from different information systems of a given organization. In a second instance, the rules for mapping and building the ontology database are defined. Finally, a technological architecture is also proposed that will allow the mapping of an ontology through the construction of complex networks, allowing mapping and relating terminologies.

This doctoral work encompasses numerous Research & Development (R&D) projects belonging to different domains such as Software Industry, Textile Industry, Robotic Industry and Smart Cities. Finally, a critical and descriptive analysis of the work done is performed, and we also point out perspectives for possible future work.

Keywords: Ontologies, Graph Database, Methods for Designing Ontologies, Interoperability, Semantic, Complex Networks

Resumo

A forma como interagimos com o mundo à nossa volta está a mudar à medida que novos desafios surgem, abraçando modelos empresariais inovadores, repensando a organização e os processos para maximizar os resultados, e evoluindo a gestão da mudança. Atualmente, e considerando os projetos executados, as metodologias utilizadas não respondem na totalidade às necessidades das empresas. Por um lado, as organizações não estão familiarizadas com as linguagens utilizadas nos Sistemas de Informação, por outro lado, são muitas vezes incapazes de validar requisitos ou modelos de negócio. Estas são algumas das dificuldades encontradas que nos levam a pensar na formulação de uma nova abordagem. Assim, o estado da arte apresentado neste documento inclui um estudo dos modelos envolvidos no processo de desenvolvimento de software, onde os métodos tradicionais e a rivalidade de métodos ágeis estão presentes. Além disso, é efetuado um levantamento sobre Ontologias e quais os métodos existentes para as conceber, transformar e representar.

Assim, e após analisarmos algumas das várias possibilidades atualmente disponíveis, iniciou-se o processo de evolução de um método e desenvolvimento de uma abordagem que nos permitisse conceber ontologias. O método que evoluímos e adaptamos permitirá derivar terminologias de um domínio específico, agregando-as de forma a facilitar a construção de um catálogo de terminologias. Em seguida, a definição de uma abordagem para conceber ontologias permitirá a construção de uma ontologia de um domínio específico. Esta abordagem permite em primeira instância, integrar e armazenar os dados de diferentes sistemas de informação de uma determinada organização. Num segundo momento, são definidas as regras para o mapeamento e construção da base de dados ontológica. Finalmente, é também proposta uma arquitetura tecnológica que permitirá efetuar o mapeamento de uma ontologia através da construção de redes complexas, permitindo mapear e relacionar terminologias.

Este trabalho de doutoramento engloba inúmeros projetos de Investigação & Desenvolvimento (I&D) pertencentes a diferentes domínios como por exemplo Indústria de Software, Indústria Têxtil, Indústria Robótica e *Smart Cities*. Finalmente, é realizada uma análise critica e descritiva do trabalho realizado, sendo que apontamos ainda perspetivas de possíveis trabalhos futuros.

Palavras-chave: *Ontologies, Graph Database, Methods for Designing Ontologies, Interoperability, Semantic, Complex Networks*

TABLE OF CONTENTS

Copyright iv
Acknowledgments v
Declaration of integrity vii
Abstract ix
Resumo x
Table of Contents xi
List of Figures xv
List of Tables xix
Acronyms / Abbreviations xxi

PART I - INTRODUCTION

Chapter 1: Introduction 1
1.1 Context and Motivation 3
1.2 Core Concepts 4
1.3 Research Question and Objectives 9
1.4 Research Method 12
1.5 Demonstration Cases 15
1.6 Document Structure 20
1.7 References 21

PART II – STATE OF THE ART

Chapter 2: Information Systems Development 25
2.1 Introduction 27
2.2 Process Models for Systems Development 29
2.3 Systems Interoperability 43
2.4 Ontologies in Requirements Engineering 50
2.5 Conclusions 59
2.6 References 59

Chapter 3: Ontologies Design and Complex Networks....65
3.1 Introduction....67
3.2 Ontological Development....69
3.3 Complex Networks and Information Visualization....79
3.4 Technologies for Complex Networks Constructing....85
3.5 Conclusions....92
3.6 References....92

PART III – CONTRIBUTIONS

Chapter 4: On the Construction of Terminologies....101
4.1 Introduction....103
4.2 The 4-Step Rule Set Method for Terminologies Derivation....105
4.3 Demonstration Case....117
4.4 Conclusions....124
4.5 References....124

Chapter 5: On the Design of Ontologies....127
5.1 Introduction....129
5.2 A Processual View of Ontologies Design....130
5.3 Demonstration Cases....140
5.4 Conclusions....163
5.5 References....164

Chapter 6: On the Construction of Ontologies Representation....167
6.1 Introduction....169
6.2 Complex Networks for Ontologies Representation....170
6.3 Demonstration Cases....182
6.4 Conclusions....204
6.5 References....205

PART IV – CONCLUSIONS

Chapter 7: Conclusions 209
7.1 Focus of the Work 211
7.2 Scientific Outputs 213
7.3 Critical Analysis and Future Work 218
7.4 References 221

APPENDIXES

Appendix A – Information Visualization: Talend Pipeline 226
Appendix B – General Terminologies Catalog STVgoDigital 232

—— This page is intentionally left blank ——

List of Figures

Figure 1 - Research question and associated goals 10

Figure 2 - Design Science Research Methodology 13

Figure 3 - Alignment Plan to the Design Science Research 15

Figure 4 - Data, Information Knowledge and Wisdom Pyramid 29

Figure 5 - Transition phases of the DIKW pyramid 32

Figure 6 - Systems development life cycle 34

Figure 7 - Waterfall Model 36

Figure 8 - Incremental Model 37

Figure 9 - Spiral Model 38

Figure 10 - V-Model 39

Figure 11 - V-Model Adaption for Business and IT-Alignment 41

Figure 12 - Overview of the LISI Interoperability Maturity Model 49

Figure 13 - Automatic ontology generation process 78

Figure 14 - Information Visualization Process 84

Figure 15 - DB-Engines Ranking database management systems 90

Figure 16 - Service specification with recursive 4SRS execution 106

Figure 17 - Service specification with recursive 4SRS execution 108

Figure 18 - 4SRS Method for Ontological Design 110

Figure 19 - Identification and analysis of Terms in a domain 112

Figure 20 - Representation of Terms 114

Figure 21 - Term Packaging and Aggregation 115

Figure 22 - Direct and inferred association of Terms 116

Figure 23 - Unification of terminologies process at STVgoDigital 118
Figure 24 - Identification and analysis of Terms in the domain 119
Figure 25 - Representation of Terms 120
Figure 26 - Term Packaging 121
Figure 27 - Term Aggregation 121
Figure 28 - Direct and inferred association of terms 122
Figure 29 – Ontological Approach for Information Systematization version 1 133
Figure 30 - Ontological Approach for Information Systematization version 2 134
Figure 31 - Ontological Approach for Information Systematization (version 3) 135
Figure 32 – Ontological Approach for Information Systematization (version 3) included into V-Model 136
Figure 33 - Ontological Mapping Schema. - Bloom Interface 140
Figure 34 - PHC Voice Assistant 142
Figure 35 - Voice Interaction Framework architecture proposed 143
Figure 36 - Process Used in Ontology Design. 144
Figure 37 - Excerpt of the PHC demonstration case code 146
Figure 38 - Excerpt from the ontological mapping performed in Neo4J. 147
Figure 39 - Representation of the context of the demonstration case scenario 149
Figure 40 - Characterization of the ascending phases of the right edge of the V-Model for Ontological Design 150
Figure 41 - Excerpt of the Class Diagram 153
Figure 42 - Excerpt of the F3M demonstration case code 153
Figure 43 - Ontological Database Schema mapped in Neo4J 155
Figure 44 - Code to import purchase orders for fabric manufacturer 155
Figure 45 - Code for eliminating null properties 155
Figure 46 - Mapping of purchase order information 156
Figure 47 - Representation of the context of the demonstration case scenario 158

Figure 48 - Mapping of the class diagram 161
Figure 49 - Schema for the semantic database mapped in Neo4J 163
Figure 50 - Technological architecture for visualizing complex networks 171
Figure 51 – “Generation Method” - Extractions and mapping 171
Figure 52 - Improved architecture 173
Figure 53 – Technological Architecture for Information Visualization 174
Figure 54 - yFiles visualization of schema 176
Figure 55 –yfiles type of filters 177
Figure 56 - yFiles visualization of Neo4J mapping 177
Figure 57 - Graph XR visualization of Neo4J mapping 178
Figure 58 - Bloom Visualization 179
Figure 59 – Bloom filters 180
Figure 60 - Graph Algorithms Playground - centralities visualization 180
Figure 61 - Graph Algorithms Playground - "samples are sampled from" 181
Figure 62 - Building and storing the terminology catalog 186
Figure 63 - Talend Pipeline 187
Figure 64 - Visualization of the architecture of all Talend jobs - relations 188
Figure 65 - Complex Network STVgoDigital Project 189
Figure 66 - Applying filters to the ontological model 190
Figure 67 - Demonstratio case ITEC 191
Figure 68 - Example of a Practice Area (CMMI) 195
Figure 69 - OpenUp Tasks by Category 196
Figure 70 - Technological Architecture for mapping information in graph databases 199
Figure 71 - CMMI Maturity Levels Mapping 200
Figure 72 - Mapping between the different CMMI categories 200
Figure 73 - Mapping the Practice Summary of CMMI's Doing category 201

Figure 74 – OpenUP tasks and disciplines mapping 202
Figure 75 – Mannaging Category Mapping 203
Figure 76 – Full alignment mapping 204
Figure 77 - Terminologies Output Pipeline 226
Figure 78 - Insert Neo4J configurations into Talend 226
Figure 79 - Delete nodes from Neo4J 226
Figure 80 - HDFS connection into Talend 227
Figure 81 - Select file from HDFS to connect 227
Figure 82 - tSample Row 228
Figure 83 - Attributes mapping 228
Figure 84 - tUniqRow component 228
Figure 85 - tNeo4jOutputs connection definitions 229
Figure 86 - Neo4J mapping 229
Figure 87 - Visualization of the architecture of all Talend jobs - relations 230
Figure 88 - Mapping relations 230
Figure 89 - Relationships configuration 231

List of Tables

Table 1 - Ranking of graph database technologies 89

Table 2 - Characteristics of graph database tools (top-4 solutions) 90

Table 3 - Characteristics of graph database tools (cont.) 91

Table 4 - General Terminology Identification (excerpt). 151

Table 5 - Identification of terminologies in the CityCatalyst Project 159

Table 6 - Excerpt from the General Catalogue of Terminologies 185

Table 7 - Coverage Levels Categorization 194

Table 8 - Relationship between CMMI and OpenUp 198

Table 9 - Demonstration case's inputs towards the research contributions 217

—— This page is intentionally left blank ——

Acronyms / Abbreviations

4SRS – 4-Step Rule Set

AMOD - Agile Methodology for Ontology Development

CCG – Center for Computer Graphics

CNA - Complex Networks Analysis

CPS – Cyber Phisical System

CQs - Competency Questions

CRISP-DM - Cross-Industry Standard Process for Data Mining

DAML - DARPA Agent Markup Language

DDD - Domain-driven Design

DSRM- IS - Design Science Research methodology for IS

EIF - European Interoperability Framework

ERP - Enterprise Resource Planning

ESB - Middleware

GSO - Graduation Screen Ontology

HCI – Human-Computer Interaction

IDEs - Integrated Development Environments

IDT4CTI – Interoperability and digital thread for a more competitive textile industry

IoT - Internet of Things

IP - Internet Protocol

IS - Information System

ISO - International Organization for Standardization

KAOS - Knowledge Acquisition in Automated Specification

LIMS - Laboratory Information Management System

LISI - Levels of Information Systems Interoperability

M2M - Machine-Machine

MVC - Model-View-Controller

NLP - Natural Language Process

NS - Network Science

OBDA - Ontology-Based Data Access

OC - Ontological Component

OIL - Ontology Inference Layer

OS - Operating Systems

OWL - Web Ontology Language

RDF - Resource Description Framework

RDMS - Relational Database Management Systems

RE - Requirements Engineering

RFID - Radio-Frequency Identification tags

S2T - Speech-to-Text

SaaS - Software-as-a-Service

SAMOD - Simplified Agile Methodology for Ontology Development

SDKs - Software Development kits

SDLC – Systems Development Lifecycle

SOA - Service Oriented Architecture

SQL - Structured Query Language

SR - Speech Recognition

TCP - Transmission Control Protocol

TOVE - Toronto Virtual Enterprise

UML – Unified modelling language

UPON - Unified Process for ONtology Building

VAD - Voice Activity Detection

XML - Extensible Markup Language

XTM - XML Topic Map

PART I - INTRODUCTION

—— This page is intentionally left blank ——

CHAPTER 1

INTRODUCTION

Summary: This first chapter initiates with the context and motivation that led to establishing the research theme of this thesis. Afterwards, the research design is structured in three parts, starting by the proposed research question and associated goals, then the description of the chosen research method (based on design science research) and, lastly, the activities planned for the time period prescribed. This chapter ends with an outline of the structure of this document and a synthesis of its contents.

CHAPTER TABLE OF CONTENTS

1.1 Context and Motivation 3

1.2 Core Concepts 4

1.3 Research Question and Objectives 9

1.4 Research Method 12

1.5 Demonstration Cases 15

1.6 Document Structure 20

1.7 References 21

—— This page is intentionally left blank ——

CHAPTER 1

INTRODUCTION

"We are what we repeatedly do. Excellence then is not a way of acting, but a habit."

– Aristoteles

1.1 Context and Motivation

An Information System (IS) within an organization comprehends all of the information flows between people, processes, and machines, even including paper and oral communications, constituting an organized system for the collection, organization, storage, and communication of information. An Information System can also be described as a combination of hardware, software, data, business processes, and functions that can be used to increase the efficiency and management of an organization.

Accordingly, the Industry 4.0 future trend of automation and data exchange in manufacturing technologies fosters the use of intelligent software and machines to interact with one another and with people, autonomously, both in the factory plant and through the cloud. This concept assumes the usage of state-of-the-art information technology and automation to collect and monitor all relevant information flows to increase efficiency, optimize management, and automate tasks according to the principles of Requirements Engineering, interoperability, virtualization, decentralization, real-time capability, service orientation, and modularity.

Industry 4.0 is a popular term used to describe the impending changes in the industry landscape, particularly in the manufacturing industries of the developed world (Perspective *et al.*, 2014). However, there are some problems that have been detected in some of the target industries. Here, the fact is that in many of these organizations there is excessive dependence on documentation in physical format

(paper, notebooks, excel sheets), mostly verbal communication, and independent systems that later require human interaction to unite information (Liao *et al.*, 2017).

On a more positive note, there is the opportunity to participate in projects in the most diverse areas of activity and where the work to be carried out allows workers to have better working methods, new technologies, and an optimization of their work so that the organization is also able to be driven and differentiated from its competitors.

In this context, the analysis of the evolution of the different scenarios under study from the stated point of view, for this declared vision, raises a series of problems and needs. In many of the analyzed contexts, there is a lack of a common language and a clear definition of terms and a large number of heterogeneous applications to support the daily tasks within the different departments of the organizations. Also, there is the leap to an Internet of Things vision based on the support of hardware (mainly sensors), global infrastructure connectivity, and the use of intelligent services and software, where interoperability between all these elements plays a crucial role.

As the focus is to increase the value of the cases under study, it is necessary first to characterize the organization or the context of action and also to define the problem, where the current organizational environment is analyzed in order to perceive and detect how the work is accomplished. With this analysis, it is easier during the requirements gathering process to understand the real needs of workers and to make more accurate and adaptable process models.

1.2 Core Concepts

This section introduces the four basic concepts mentioned throughout this work in order to provide the reader with a better understanding of the context of the subjects discussed. Therefore, this brief contextualization focuses on Information Systems, Interoperability, Ontologies, and Complex Networks.

Information Systems are treated as a field of specialization throughout this PhD work and are a highly important theme in terms of the contributions resulting from this work. Regarding Interoperability, a contextualization is provided in order to enlighten the reader on this topic and differentiate the many types of existent interoperability. Regarding Ontologies, these are introduced with an explanation of what they are and the problems they aim to solve. Finally, the notion of Complex Networks is introduced, and in

relation to the contributions demonstrated throughout this text, they are used to visually express the ontologies. Thus, these principles are described in further depth in the following section.

Information Systems

Information systems are an emerging field of human knowledge that faces challenges due to a lack of consensus over its most fundamental ideas and terminology. The terms "information" and "Information Systems" are troublesome. Attendees of information systems-related conferences and readers of relevant books and articles will quickly learn the following: It is obvious that the phrase "information system" has several different meanings.

That is, individuals who discuss or write about information systems do not always refer to the same entity. The distinctions between the various objects that can be considered information systems are not always clear because they share many characteristics: all deal with information; all are somewhat related to organizations or the work performed by organizations; and all are related to information technology, either because they can benefit from its use or because they are created with computers or computer-based devices.

According to the definitions of information and system supplied by different writers, an information system is either: (i) an active object that interacts with (processes) information; or (ii) an active entity whose mission is to inform (Carvalho, 2000).

Interoperability

Interoperability is the capacity of two or more software components to work together regardless of variations in language, interface, and execution platform (Wegner, 1996).

When referring to specific conditions, the interoperability level must be specified. a) "Interoperability is the capacity of information systems to share and communicate information." b) Conditions, attained to varying degrees, in which information systems and/or its components can share information directly and satisfactorily. c) The ability to execute software and exchange data across a diverse network (i.e., one large network comprised of several different local area networks). d) Systems or programs capable of

data exchange and efficient collaboration. "Interoperability necessitates resolution at various levels. Interoperability consists of four layers: technical, syntactic, semantic, and organizational.

1. Technical Interoperability is accomplished between communications electronics systems or communications-electronics equipment when services or information may be shared directly and satisfactorily between them and their users. Regarding specific situations, the interoperability level must be indicated. Technical interoperability is typically associated with hardware/software components, systems, and platforms that enable machine-to-machine communication. This type of interoperability typically involves communication protocols and the infrastructure required for such protocols to operate (Lampathaki *et al.*, 2012).
2. The ability to interchange data is the definition of syntactic interoperability. Data formats are often related to syntactic interoperability. Even if just in the form of bit tables, the grammar and encoding of the messages sent through communication protocols should be well-defined (Veer & Wiles, 2008).
3. The definition of semantic interoperability is the capacity to operate on the data in accordance with agreed-upon semantics. Semantic interoperability is typically associated with the definition of content and focuses on the human interpretation of this content as opposed to machine interpretation. Therefore, interoperability at this level signifies that participants share a shared understanding of the concept of the content (information) being exchanged (Veer & Wiles, 2008).
4. Organizational interoperability refers to the ability of organizations to effectively communicate and transfer meaningful data (information) despite the use of a variety of information systems over significantly different types of infrastructure, potentially across a variety of geographic regions and cultures (Veer & Wiles, 2008). Interoperability of the technological, syntactic, and semantic aspects is essential for organizational interoperability.

Ontologies

Lately, there has been a lot of talk about ontologies and how they are beneficial when it comes to handling and organizing data. But what are ontologies anyway? Ontologies are data models that represent a set of concepts within a domain and the relationships between them. Basically, ontologies can be characterized as an information organization technique, mainly regarding the formal representation of knowledge (Guizzardi, 2007). They are usually created by experts and, as their structure is based on the description

of concepts and the semantic relationships between them, they allow the generation of a formal and explicit specification of a shared conceptualization. And why have they become so important? As we well know, the volume of data we deal with today - on the Internet and especially at the organizational level - is very vast and diverse, and the trend is for it to become larger and larger.

Due to these high growth rates, it has become necessary to use techniques that allow a better and more effective treatment and organization of data, acting mainly in its selection, processing, retrieval, and dissemination - the ontologies. Ontologies can be defined by creating catalogs of terminologies, glossaries, or dictionaries, by classification or categorization through taxonomies or from concepts and their relationships, with great focus on the use of semantic networks (Calero *et al.*, 2006). When ontologies are used, the different terms or ways of talking about something are brought together. This makes it easier to understand and model the ecosystem in question.

The use of ontologies has as main advantages the possibility of communication between different types of machines, creating an interoperable network about a particular knowledge; and the formalization, which is related to the specification of the ontology, allowing the elimination of contradictions and inconsistencies in the representation of knowledge. In terms of knowledge representation, ontologies form a consensual vocabulary that allows the representation of knowledge of a specific domain at a high level of abstraction, thus ensuring a potential for reuse.

On the other hand, the most common problem that usually arises is: "What is the most correct methodology to use?" Just as in software development, in ontology design there are also several associated methodologies, some more traditional and more exhaustive, and others more recent that have an agile basis. At Center for Computer Graphics (CCG), we have been developing our own methodology, which we have been applying to the various projects where we are inserted.

Complex Networks

A complex network is sometimes shown as a graph, G = (V, E), consisting of two conjuncts, V and E, where V represents the group of G's vertices (nodes or points) and E represents the group of E's arestas (ligatures or lines), which represent pairs of V's linked elements. Each vertex is identified by an independent integer p = 1,.... N, and each area is identified by a pair (p, q) connecting the two vertex pairs p and q. If an area of the p to q vertex as denoted by (p, q) is equal to an area of the q and p vertex,

the graph G is said to be non-directed, i.e. (p, q) E (q, p) E. On the other side, a directed graphic is often (p, q) E (q, p) E.

A graph may have loops, or arestas of a single vertex by themselves, or many arestas, or vertex pairs connected by more than one aresta. More generally, arestas (p, q) supplemental weights Wpq may be assigned. By convention, Wpq is often defined as 0 of (p, q) / E. In this instance, a directed ponderation graph may be completely described by its weight matrix W, where each entry Wpq represents the weight of the connection between the vertices p and q (A *et al.*, 2019).

Considering the preceding contextualization of the core concepts mentioned throughout this dissertation, some brief definitions of data, information, knowledge, graphs, and semantics are then provided. These definitions are emphasized because there is a direct relationship between each of them.

Data

The data is unprocessed. It simply exists and is only relevant in reference to itself (in and of itself). It can exist in any form, regardless of its usefulness. It does not by itself suggest anything. In computer terminology, a spreadsheet frequently begins with data (Russell Ackoff, 1989).

Information

Information is herein understood as symbolic objects (as opposed to material and energetic objects) deliberately built in order to enable communication and the formation of knowledge (Carvalho, 2000).

Knowledge

Knowledge is a highly valued state in which a person has cognitive contact with reality. Therefore, there is a linkage. There exists a conscious subject and a reality component with which the knower is either directly or indirectly related. Although directness is a matter of degree, it is useful to consider knowledge of things to be more direct than indirect knowledge of things. The former is generally referred to as acquaintance knowledge because the subject has direct experiential touch with the component of reality

known, whereas the latter is propositional knowledge since what the subject knows is a statement about the world that is based on facts (Zagzebski, 2017).

Graphs

A graph intended to accumulate and convey knowledge of the real world, whose nodes represent entities of interest and whose edges represent potentially different relations between these entities. The graph of data (a.k.a. data graph) conforms to a graph-based data model, which may be a directed edge labelled graph, a heterogeneous graph, a property graph, and so on (HOGAN *et al.*, 2021).

Semantic

Semantic, specifically semantic parsing, is a shallow semantic parsing task that seeks to identify the predicate-argument structure of each predicate in a phrase, such as who did what to whom, where and when, etc. In particular, we aim to recognize arguments and characterize their semantic functions given a predicate (He *et al.*, 2018).

1.3 Research Question and Objectives

Figure 1 presents a schematic of the research problem. Thus, it begins by defining the research question that guided a review of the literature on the current state of the art for this work. Related to the primary research question (RQ), three research objectives (RG) are established and discussed throughout this document, respectively, in each contribution chapter. Individual research objectives are then pursued through the execution of tasks that generate artifacts and result in publications associated with their respective research and development initiatives.

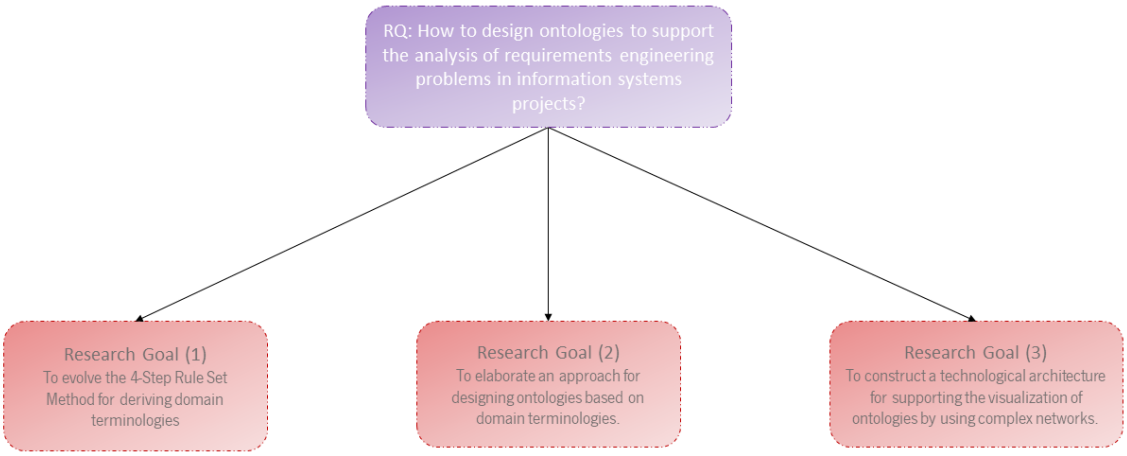


Figure 1 - Research question and associated goals

"How to design ontologies to support the analysis of requirements engineering problems in information systems projects?"

Developing ontologies associated with successful and evolving information systems is a challenging task that necessitates the application of methods that can adapt to rapidly changing conditions and achieve superior performance. In addition, the correct use of terms and expressions is crucial when seeking strategic and operational alignment between various contexts and information systems.

As detailed in this document chapters on the state-of-the-art, these issues are major obstacles to the development of any project, and the currently available methods are incapable of addressing them in their entirety. Consequently, our research question seeks methods and approaches that permit the development of requirement-based solutions for designing ontologies from various domains.

Thus, it is intended that this research will generate a method for building ontologies. To ensure proper alignment between all the phases required to build an ontology, it is necessary that requirements gathering and identification be addressed from the initial phase of any project. Thus, the research objectives are based on the evolution of a method and the design of a targeted approach for the construction of ontologies based on complex networks. In order to answer this research question, we identified three objectives:

01: To evolve the 4-Step Rule Set Method for deriving domain terminologies

In this first objective we intend to adapt and evolve a currently existing method in order to work on and derive terminologies for a given domain. Thus, we have as a starting point the 4-Step Rule Set method, developed within the research team of the Computer Graphics Center. This method, 4SRS, is applied to transform the requirements resulting from the modeling of the traceability process into a logical architecture.

Thus, this goal is to adapt it by developing a new variant, the 4SRS-Onto, which allows deriving terminologies from requirements, allowing the user to optimize the construction of a terminology catalog in highly complex contexts.

02: To elaborate an approach for designing ontologies based on domain terminologies.

In general, an ontology should provide adequate workflows and the ability to design process sequences for the development, maintenance, and standardization of the information under study. Ontology creation is largely a process of knowledge integration. Thus, we aim to define, develop, and align an approach that draws on the concepts of data, information, knowledge, and wisdom (DIKW pyramid).

Therefore, it is defined as an objective the construction of an ontological approach to systematize information based on three main phases: Data characterization, Information specification and Ontology mapping.

03: To construct a technological architecture for supporting the visualization of ontologies by using complex networks.

After collecting requirements and modeling business processes in the companies and projects under investigation, it is verified that many of the processes previously described are not performed in their entirety and often are not performed correctly. Thus, and considering all the interactions with the various stakeholders of the companies, it is identified as a goal the definition and implementation of a technological architecture for building complex networks to represent ontologies.

This architecture specification will allow employees to have access to an intuitive visualization tool that can be customized through user interface interactions. These visualization components also provide the visualization of dynamic data catalogs in which all terms are standardized.

1.4 Research Method

The choice of the research method to follow depends on the research topic and on the research questions addressed in the PhD research work. In the information system discipline, information system professionals are involved in the design and implementation of IT artifacts, and research on Information System (IS), which is characterized by two paradigms: Behavioral Science and Design Science. In this context, the paradigm of Design Science is more appropriate since it allows extending the boundaries of human and organizational capabilities by creating new and innovative IT artifacts.

In recent years, several researchers have succeeded in bringing Design Science into the IS research community. Typically, research projects that use Design Science in IS produce four types of IT artefacts: constructs (vocabulary and symbols), models (abstractions and representations), methods (algorithms and practices), and instantiations (operationalization of the constructs, models, and methods).The authors (Alan R. Hevner *et al.*, 2004) provide practice rules for conducting research in the IS discipline that describe the characteristics of well carried out research.

The most important of these is that the research must produce an “artifact created to address a problem". The research should represent a verifiable contribution, and rigor must be applied in both the development of the artifact and its evaluation. The development of the artifact should be a search process that draws from existing theories and knowledge to come up with a solution to a defined problem. Also, the research must be effectively communicated to appropriate audiences. In this context, as the work in this PhD is based on the development of an artifact, the adopted approach is Design Research, while the Design Science Research methodology for IS (DSRM-IS) from (Peffers *et al.*, 2007) will be the adopted research methodology.

Considering the nature of this PhD research work, which involves the development of an artifact to address a specific problem, the DSRM-IS is the methodology chosen for this work. Therefore, the DSRM-IS incorporates principles, practices, and procedures required to carry out such research while meeting three objectives: it is consistent with prior literature; it provides a nominal process model for doing DS research; and it provides a mental model for presenting and evaluating DS research in IS. The DS process includes six steps: problem identification and motivation; definition of the objectives for a solution; design and development; demonstration; evaluation; and communication (Peffers *et al.*, 2007).

Figure 2 presents the DSRM-IS with four different entry points: problem-centered initiation; objective-centered solution; design and development centered solution and client/context initiated. Next we describe the projected work involved in the six tasks which comprise the associated process. Then, in Figure 3, we present the predictable plan aligned to the Design Science research methodology.

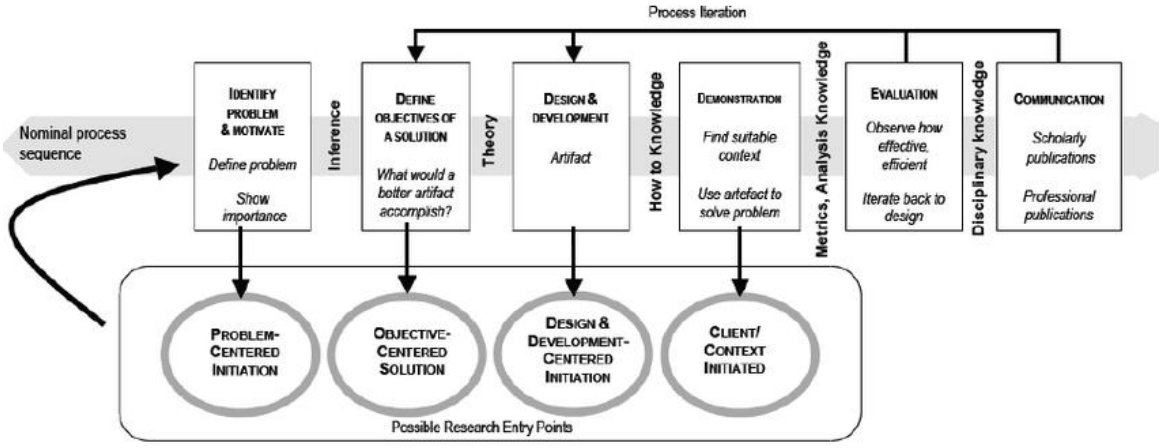


Figure 2 - Design Science Research Methodology

Identify problem and motivate – In this activity, the specific research problem is defined and the problem definition will be used to develop the artefact that can effectively provide a solution. The justification of the value to the solution is defined. In terms of contributions, the main input in this activity are the state-of-the-art in Software development lifecycle, Ontologies and Interoperability. The activity focus on defining the research problem showing the importance and motivation.

The problem definition is related to the lack of a common language and clear definition of terms and a large number of heterogeneous applications to support the daily tasks within the different departments of companies, as well as the leap towards an Internet of Things vision based on hardware support (mostly sensors), a global infrastructure connectivity, and the use of intelligent services and software, where interoperability between all these elements plays a crucial role. Since the problem definition, will be used to develop the artifact that will effectively provide a solution.

Define the objectives of a solution – This activity contemplates the objectives definition of a solution from the problem definition. The objectives can be quantitative or qualitative. If quantitative, the proposed solution should be better than the existing ones. If they are qualitative, the solution should describe how the new artifact supports the solution of the problem. At this stage, in addition to knowing state of the art, it is necessary to understand other solutions and their effectiveness to serve as a term of comparison. In

this work plan, these objectives correspond to the expected results, which are mainly qualitative. However, the expected results from the evaluation of the method are quantitative and is used to assess the performance of this Framework.

The expected results from the evaluation will be established according to the state-of-the-art and will be related to others Methods with the same purpose, which makes possible to evaluate if the proposed method provides adequate instantiations. This activity aims to suggest a solution for the identified problem, which should be implemented, evaluated and aiming to increase the existing body of knowledge. Regarding this doctoral thesis, the suggested solution consists on create or evolve a method for designing Ontologies to support It interoperability architectures.

Design and development – Typically, research projects that use DS in IS produce four types of IT artefacts: constructs, models, methods and instantiations. In this activity the artifact is created and in this doctoral project the artifact is a method and will be conceived at this stage. The input used in this activity will be the state-of-the-art related with methods for software development, Ontologies and Interoperability. The design and the development will incorporate the Framework for designing Ontologies.

Demonstration – In this activity is used the artifact to solve one or more instances of the problem. The method will be instantiated to demonstrate its application for solving a real problem in research projects in industry.

Evaluation – This activity involves comparing the expected objectives of a solution to the actual observed results from the artifact in the demonstration. The evaluation could be done in various forms and depends on the nature of the problem and the artifact. IT could include: objective quantitative performance measures as budgets, items produced, client feedback the satisfaction of results or simulations. In this doctoral thesis, includes the evaluation of the Framework accomplishes the results that are considered satisfactory. The alignment should be validated and compared to the expected results.

Communication – Communicating the proposed solutions to approach the problem is important to demonstrate the utility and novelty of the artifact, the rigor of its design and its effectiveness to researchers and other relevant audiences. This activity involves writing and publishing the scientific publications in conferences and journals, as well as the doctoral thesis itself (Figure 3).

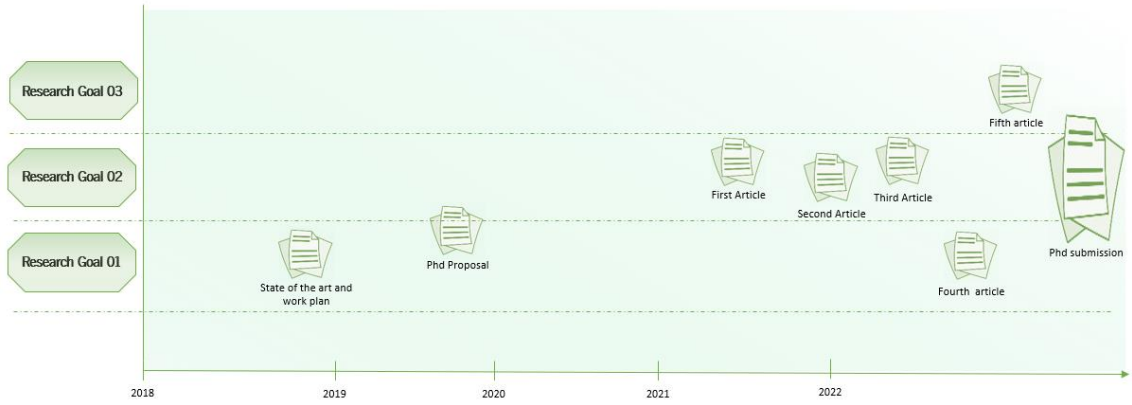


Figure 3 - Alignment Plan to the Design Science Research

1.5 Demonstration Cases

In this thesis, the research projects are used as demonstration cases, separately, within the scope of DSR cycles. Each project had a clearly defined input for the research and span a variety of areas such as the textile industry, software industry, robotics industry, and smart cities.

Besides the projects presented below, other projects could have been mentioned throughout this document. However, we will privilege those that effectively translated into an ontology design component.

The F3M Case: IDT4CTI (Interoperability and digital thread for a more competitive textile industry) project

This project proposes to investigate and develop a solution, following the paradigm of a highly distributed and globalized "ecosystem", implemented in an infrastructure that provides relevant and current information of the production status, which speeds up the decision-making process. Specifically, the project will contribute to the automation of the integration of productive systems that are geographically distributed and heterogeneous that should be able to interoperate both technically (syntactic interoperability) and semantically (semantic interoperability). In this way, it will allow to "inject" into the market the horizontal interoperability pillar that is evangelized by Industry 4.0.

By allowing a faster development of new technology-based solutions for optimization of the production process, this project will contribute to innovation in industries, specifically textile and clothing, which will

enhance greater flexibility and productive adaptability and, consequently, the strengthening of the positioning of these industries in global value chains. In this context, the project and its underlying objectives enhance the competitive advantages not only of the promoting company but also at the level of the Northern region, where F3M is based, and the country itself.

The CityCatalyst Case: (A Catalyst for Sustainable Cities) project

Cities are currently faced with a very diverse set of challenges, the result of the complexity of the urban experience of citizens, active elements in different segments and in cross-cutting axes of the city such as, for example, in the use of energy or mobility. The ambition of the project "City Catalyst - Catalyst for Sustainable Cities" is thus to address these challenges through research, development and validation, in a real context, of innovative technological solutions and services that enhance integrated urban management, more efficient and effective, and a catalyst for innovation and sustainable development through specific contributions to the implementation and interoperability of urban platforms.

The project is thus structured around different lines of research and development / pioneering PPS in the perspective of Smart and Sustainable Cities related to sensing infrastructures, communication and computational processing (PPS1); security and privacy of information (PPS2); data models and interfaces (PPS3), as well as processes of open innovation and co-creation for the competitiveness of smart cities and sustainable (PPS5). The project also includes the implementation of a set of demonstrators with the aim of validating the operation of the solutions to be developed in 5 Portuguese cities (Porto, Aveiro, Guimarães, Famalicão and Cascais).

To ensure the pursuit of the R&D and market objectives, a consortium of 22 entities (companies and national ENESIIs) was formed, which are committed to promoting integrated urban management that is more efficient and effective and a catalyst for innovation and competitiveness. The City Catalyst project thus aims to contribute to the affirmation of Portugal as a reference market in the export of innovative solutions for Smart and Sustainable Cities.

PHC Case: The Voice Interaction Framework

Throughout its almost 30 years of existence, PHC has been a pioneer in Portugal with regard not only to the products it places on the market, but also in relation to the positioning it assumes. As a software house that develops management software, it is currently positioned not only as an IT expert, but above all as a Management Specialist. In this context, it intends to:

- Prepare its software for an expansion in the market, responding to the needs, new expectations and ways of working of the clients;
- Be a pioneer in Portugal, with the introduction of a new technological piece of voice recognition in ERP, distinguishing it from the competition;
- Increase customer productivity with a change in the way they interact with the software, with new actions that have voice as the main input;
- Develop a solution that is transversal to all devices and multi-language;
- Increase investment in research and development at PHC.

The CCG will allocate competencies from three of its applied research domains to the challenges listed above, namely: CVIG (Computer Vision, Interaction & Graphics), the EPMQ (Engineering Process, Maturity & Quality) and the HTIR (Human-Technology interaction and Usability.

The CCG proposes to support PHC in the pursuit of technical and scientific strategic objectives, presented in the application for an individual project supported by the incentives system for research and technological development, namely:

- Study of the state of the art of natural language processing tools (Speech to Text and Text to Speech), specifically those associated with / developed for ERP systems.
- Improving the performance of algorithms and natural language processing methods, specifically to ensure a more intuitive interface with ERP systems, in Portuguese, Spanish and English languages.
- Controlled survey and comprehensive systematization of interaction strategies with the PHC CS system. Use of this knowledge to define the linguistic and semantic content of the Natural Language Processing (NLP) dictionaries.

- Evaluation of the suitability of acoustic hardware (e.g. microphones) for use in noisier contexts of use (e.g. office environment). Use of this data to develop digital acoustic signal processing strategies for the different contexts of use of the PHC CS system.
- To specify and implement an ontological model or ontology (semantic tree of concepts), which allows unifying and identifying work instructions (e.g. queries to PHC CS) and common terminologies or terminologies with the same meaning.
- Define and develop a semantic interoperability layer in order to automate the ontology defined in the context of the project. This layer will allow the information collected by the speech recognition component to be automatically recognized, converted and processed by the PHC CS.

The STVgoDigital Case: (Digitalization of the T&C sector) project

The STVgoDIGITAL: Digitalization of the T&C sector project is a structural project of the Portuguese Textile Cluster: Technology and Fashion, which aims to comprise a set of research initiatives with a strong collective impact and high inductor and demonstrator effect. Clearly aligned with the Portuguese Textile Cluster strategy, namely with the strategic pillar Industry 4.0 which attempts to promote the digitalization and the adoption of this technologies by the Textile and Clothing sector.

The project will bring together the Textile and Clothing sector and other complementary sectors that will enhance the transition to this new paradigm of Industry 4.0 by building new and complementary value chains. The project leader is *TMG - Tecidos para Vestuário e Decoração, S.A*. The technical and administrative coordination will be assured by CITVE. The project is organized in 5 technical PPS (Products, Processes and Services) that aim the development of innovative solutions and business models in different areas:

PPS 1 - Sustainable and Circular Textile ID 4.0: Innovative solution that allows the Portuguese Textile Cluster to deliver to the customer an extended product, with the additional information of the product (Product ID).

PPS 2 - Supply Chain 4.0: an interoperable system that intermediates the different actors along the value chain, defining an interaction/ communication/ coordination protocol that allows each agent to interact, question and answer to the different requests from other agents.

PPS 3 - Fashion Ecosystem 4.0: a system that enables to successfully respond to small or even 1 size orders, reaching unit production, driven by a greater demand of customized and taking advantage of local or regional identity, both in terms of creativity and production capacity.

PPS 4 - Worker 4.0: innovative solution focused on assisting the movements performed by the T&C workers, which will allow an increase in their comfort and safety, performance and the ability to operate with the expected flexibility of production.

PPS 5 – Artificial Intelligence for ITV 4.0: a platform for quality analysis of textile materials and products that makes intensive use of image processing techniques, algorithms and artificial intelligence.

Additionally, a transversal PPS (PPS6 – Project management, dissemination and exploitation) will be dedicated to project management, dissemination and exploitation of the results. The STVgoDIGITAL project gathers 23 entities, 16 companies and 7 non-business R&D organizations (ENESII).

The ITEC Case: Smart Automation i4.0

The ITEC Smart Automation I4.0 project aims to develop technological solutions that will translate into the creation of value in ITEC products in response to the challenges of Industry 4.0 and the growing demand of the automotive sector, with knowledge and skills generated in Portugal.

Concretely, the ITEC Smart Automation I4.0 project has as main objectives to develop I&DT solutions for transversal incorporation in ITEC products available to the market in the domains of Robotics, Industrial Automation and Quality Control Systems, namely:

- Automatic visual inspection: A new system for vision, optics and automatic inspection, equipped with computer vision and Artificial Intelligence (AI) algorithms, which solves the limitations found in the current systems related to visual inspection carried out by humans. The system should characterize and solve several tasks, critical for ITEC, and contain functionalities such as the traceability of the process inspected by the vision system, the collection of statistics on the

inspection system, and also the ability to debug the operation of the system on the assembly line in order to determine which method failed to detect the defect in a product.

- Smart machines: Smart Machine systems that enable ITEC equipment with monitoring, self-diagnostic and self-adjustment mechanisms, namely for defects related to dispensing systems (fluids) and screw machine failures (wiring).
- Systematized software development processes: New life cycle management process and quality for software products - to endow the company with a systematized software development process with standardization of the base software development activities (system specification, implementation, testing, version control, etc.) and to guarantee the robustness of the developed software through functionalities for software quality verification and validation.

These developments will provide ITEC with technological solutions, competencies and internal specialized knowledge in the areas of intervention of this project that will be reflected in highly innovative and competitive products for the global market in their business areas.

1.6 Document Structure

Following the described context and towards our aim to develop a method for designing ontologies to support IT interoperability in software architectures, this document presents a state-of-the-art literature review in the associated topics of System Analysis, Requirements Engineering Semantics, and System Integration. In this first chapter an introduction is made, including the motivation, the literature review process and the document overview as a way to synthesize and make known everything it contains.

The following chapters, chapter 2 and chapter 3, present all the state-of-the-art surveys to date, framed with the components, information system development, system integration, requirements and ontologies, complex networks, and technologies used for visualization of complex networks. Then, chapters 4, 5, and 6 elaborate on the produced work, specifically the 4SRS-Onto method, the approach to construct an ontology, and the technological architecture developed for mapping the ontologies over complex networks. In these chapters, the demonstrations of the applicability of the method and the associated approach are based on real cases in live R&D projects.

Finally, Chapter 7 presents the conclusions and future work to be done to continue answering the research questions and this process of developing an ontology design method using complex network mapping technologies.

1.7 References

A, Y. Z., Donner, R. V., Marwan, N., Donges, J. F., & Kurths, J. (2019). Complex network approaches to nonlinear time series analysis. *Physics Reports*, 1–97.

Alan R. Hevner, March, S. T., Park, J., & Ram, S. (2004). DESIGN SCIENCE IN INFORMATION SYSTEMS RESEARCH. *MIS Quarterly*, *28*, 75–105.

Calero, C., Ruiz, F., & Piattini, M. (2006). Ontologies for Software Engineering and Software Technology. In *Springer*.

Carvalho, J. A. (2000). Information System? Which One Do You Mean? In E. D. Falkenberg, K. Lyytinen, & A. A. Verrijn-Stuart (Eds.), *IFIP — The International Federation for Information Processing book series* (pp. 259–277). Springer US.

Guizzardi, G. (2007). On ontology, ontologies, conceptualizations, modeling languages, and metamodels. *Frontiers in Artificial Intelligence and Applications*, *155*, 18–39.

He, S., Li, Z., Zhao, H., Bai, H., & Liu, G. (2018). Syntax for semantic role labeling, to be, or not to be. *ACL 2018 - 56th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)*, *1*(61672343), 2061–2071.

HOGAN, A., BLOMQVIST, E., COCHEZ, M., D'AMATO, C., MELO, G. DE, GUTIERREZ, C., KIRRANE, S., GAYO, J. E. L., NAVIGLI, R., NEUMAIER, S., NGOMO, A.-C. N., POLLERES, A., RASHID, S. M., RULA, A., & SCHMELZEISEN, L. (2021). Knowledge graphs. *Communications of the ACM*, *64*(3), 96–104.

Lampathaki, F., Koussouris, S., Agostinho, C., Jardim-Goncalves, R., Charalabidis, Y., & Psarras, J. (2012). Infusing Scientific Foundations into Enterprise Interoperability. *Comput. Ind.*, *63*(8), 858–866.

Liao, Y., Deschamps, F., Loures, E. de F. R., & Ramos, L. F. P. (2017). Past, present and future of Industry 4.0 - a systematic literature review and research agenda proposal. *International Journal of Production Research*, *55*(12), 3609–3629.

Peffers, K., Tuunanen, T., Rothenberger, M., & Chatterjee, S. (2007). A design science research methodology for information systems research. In *Journal of Management Information Systems* (Vol. 24).

Perspective, A. I., Brettel, M., Friederichsen, N., Keller, M., & Rosenberg, M. (2014). *How-Virtualization-Decentralization-and-Network-Building-Change-the-Manufacturing-Landscape–An-Industry-40-Perspective*. *8*(1), 37–44.

Russell Ackoff. (1989). From Data to Wisdom. *Journal of Applied Systems Analysis*, *15*, 3–9.

Veer, H. Van Der, & Wiles, A. (2008). Achieving Technical Interoperability - the ETSI Approach. *ETSI White Paper*, *3*(3), 1–30.

Wegner, P. (1996). *Interoperability*. *28*(1).

Zagzebski, L. (2017). What is Knowledge? In *The Blackwell Guide to Epistemology* (pp. 92–116).

PART II – STATE OF THE ART

—— This page is intentionally left blank ——

CHAPTER 2

INFORMATION SYSTEMS DEVELOPMENT

Summary: This chapter presents existing research in information systems development, with a focus on interoperability and ontologies. First, it introduces information systems development by identifying some models for information systems development. Subsequently, the chapter discusses the impact of the evolution of information systems and the triggering of interoperability needs. At this stage, the existing types of interoperability are also identified. Afterwards, a section concerning ontologies is included, where existing approaches are discussed and how they can be used to solve interoperability problems. This present chapter ends with the conclusions on the work previously presented.

CHAPTER TABLE OF CONTENTS

2.1 Introduction....27
2.2 Process Models for Systems Development....29
2.3 Systems Interoperability....43
2.4 Ontologies in Requirements Engineering....50
2.5 Conclusions....59
2.6 References....59

—— This page is intentionally left blank ——

CHAPTER 2

INFORMATION SYSTEMS DEVELOPMENT

"I wish to be a creator of myths, which is the highest task which can be accomplished by anyone in humanity"

– Fernando Pessoa (1813-1934)

2.1 Introduction

In Information Systems development, the analysis of requirements encompasses components which deal with the research, definition, and scope of new systems. Therefore, requirements analysis is an integral part of the development process of information systems in which the analysis and gathering of needs or requirements must be performed with the customer. Also, requirements analysis is one of the first activities of software development (Fernandes & Machado, 2015).

The expected result of the work in requirements specification, is to define the scope of a project in two dimensions: functional requirements and non-functional requirements. It is in this phase that the analyst holds the first meetings with the client or software users to learn about the functionalities of the system that will be developed. It is also in this phase that the first errors and inconsistencies are detected because customers do not always know or have well defined what they want for their new or renewed system (Hofmann & Lehner, 2001).

However, as applications vary from organization to organization and as the realities encountered are also distinct, it is necessary at an early stage to understand the business involved. This is an extremely important step in Information Systems (IS) development. Thus, it is necessary to understand in detail the organizational context in which we are inserted and where we will operate. Nevertheless, from a technical point of view, in some cases, legacy information systems are rigid with pre-defined functionalities created at design time as an interpretation of the then valid rules and regulations (Benbya *et al.*, 2020).

The rigidity of such legacy systems makes them less sustainable to cope with changes in the company's organization in order to be adapted to the rapidly changing environment (Agostinho *et al.*, 2017). Thus, as described throughout this chapter, it is possible to resort to ontologies as a way to address the lack of interoperability that many companies still face. Within organizations, information systems comprise all information flows between people, processes, and machines (including paper and oral communications), forming an organized system for collecting, organizing, storing, and conveying information. Due to all of these information flows, it is essential that there be a way to standardize the way we handle a domain within the organization. And this is where some barriers to the process start to arise (Reynolds & Stair, 2020).

For all of this to work correctly, we need a coherent and homogeneous business ecosystem at the level of data and its applications. However, we often have to deal with unstructured data, applications that work on their own, and too much paper. So, if we want a company to have an interoperable system, we must first give it the tools and processes it needs and then do a survey of existing software applications and the terms they use so that we can standardize them through a catalog of synonyms. It is precisely here that ontologies act as problem solvers (Teichert, 2019).

Ontologies refer to knowledge sources, as dictionaries are related to literary works. As with the dictionary, ontologies collect and organize terms of reference. Similarly, the definitions in a dictionary indicate the relationships between words, whereas ontologies give us the relationships between terms (Mitra *et al.*, 2000). So, since ontologies let us show how terms relate to each other, and if the terms are defined in a database, we can use graphs to show how an ontology works.

2.2 Process Models for Systems Development

Information Systems (IS) are sociotechnical communication systems that use artifacts to represent and transport data (Beynon-Davies, 2009). They have existed for thousands of years, long before the development of contemporary information and communication technologies. For instance, the Inca civilization operated an effective information system without the use of written language; they recorded intricate messages on colored cotton cords and with us (Beynon-Davies, 2007). Data, Information, Knowledge, and Wisdom (DIKW) (see Figure 4), also known as the Knowledge Pyramid and the Knowledge Hierarchy, serves as the foundation for the study of information system architecture (Saša Baškarada & Iskandar, 2018).

Although the actual hierarchy is frequently cited and explicitly or implicitly applied in literature about information systems (information is typically defined in terms of data, knowledge in terms of knowledge, and wisdom in terms of knowledge), there has only been limited direct discussion of the content and structure of the hierarchy (Zins, 2007). In particular, the idea of wisdom has only been briefly discussed in the literature (Saša Baškarada & Iskandar, 2018).

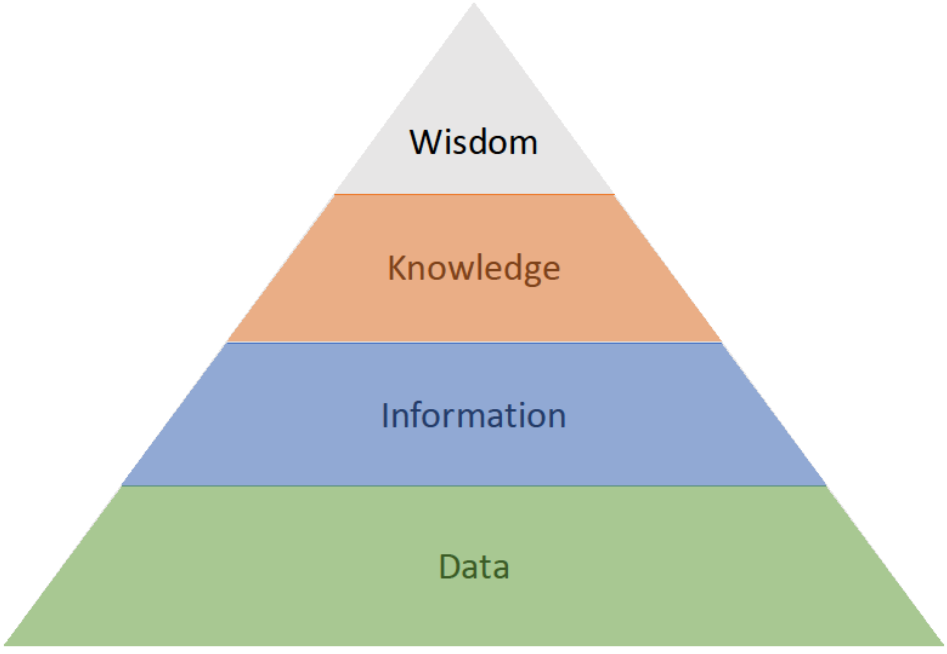


Figure 4 - Data, Information Knowledge and Wisdom Pyramid adapted from (Russell Ackoff, 1989)

In textbooks on information management, information systems, and knowledge management, it is frequently quoted or implicitly used in definitions of data, information, and knowledge. The hierarchy is used to identify and describe the processes involved in transforming an entity at a

lower level in the hierarchy (for example, data) to an entity at a higher level in the hierarchy. It is also used to contextualize data, information, knowledge, and occasionally wisdom in relation to one another (e.g., information). The underlying premise is that information can be produced using data, knowledge can be produced using information, and wisdom can be produced using knowledge (Rowley, 2007).

Each of the higher kinds in the hierarchy "includes the categories that lie below it," according to Ackoff (Russell Ackoff, 1989), whose contribution is frequently recognized when the DIKW hierarchy is mentioned. The DIKW hierarchy's defining function places it at the center of the information management, information systems, and knowledge management models (Rowley, 2007).

Paraphrasing Ackoff as follows: Knowledge is the ability to increase effectiveness. The ability to increase efficiency is a function of intelligence. Knowledge, or expertise, is what makes the transformation of information into instructions possible. The information provides answers to who, what, where, and when questions are asked. Data is defined as symbols that reflect the characteristics of objects, events, and their surroundings. These are the observational products (Rowley, 2007).

According to Ackoff, the first four categories are connected to the past; they focus on what was or is well-known. Only the fifth category, knowledge, deals with the future since it incorporates vision and design. Knowledge allows people to create the future rather than just cling to the past and present. But achieving knowledge is not simple; individuals must progress steadily through other categories.

Another development of Ackoff's definitions follows:

- Data: The data is raw. It just exists and is significant only in relation to itself (in and of itself). Whether or not it is usable, it can exist in any form. It doesn't imply anything by itself. A spreadsheet, as used in computer terminology, often begins with data.
- Information: Data that has been given meaning through a relational link is referred to as information. This "meaning" is not required to be beneficial, although it may be. In computer parlance, a relational database makes information from the data stored within it.
- Knowledge: Knowledge is the proper accumulation of information with the purpose of being of use. The process of learning is deterministic. Someone has acquired knowledge when

they "memorize" facts (as less motivated students who are obligated to take tests frequently do). They can make use of this knowledge, but it does not by itself allow for the kind of integration that would imply further information. Children in primary school, for instance, learn or acquire knowledge of the "times table." They have acquired the information to be able to tell you that "4 x 4 equals 16" (it being included in the times table). However, they are unable to respond appropriately when asked what "2512 x 135" is since that answer is not included in their times table.

Such questions need actual cognitive and analytical skills, which are only present at the next level—understanding—and which are necessary to provide an accurate response. The majority of the computer programs we use (such as modeling, simulation, etc.) make use of some form of stored information.

- **Wisdom**: Extrapolation is a non-deterministic, non-probabilistic process that involves knowledge. It explicitly invokes certain sorts of human programming and all prior levels of awareness (morals, ethical codes, etc.). It calls for knowledge about things about which we previously had no knowledge, and in doing so, it goes far beyond knowledge itself. It embodies all that philosophical inquiry is about. It poses questions that cannot be easily answered and, in certain situations, cannot have a humanly known response, unlike the preceding four levels. Therefore, discernment between good and evil, as well as between right and wrong, is a process we call wisdom. I firmly believe that computers do not currently possess wisdom and never will. A soul is necessary for knowledge since it exists equally in the heart and the mind, making wisdom a distinctively human state. Additionally, robots will never have a soul (or perhaps I should reword that to say, a soul is something that, in general, will never possess a machine)(Russell Ackoff, 1989).

According to G. Bellinger *et al.*, the sequence is a little less complex than what Ackoff stated. The changes from data to information to knowledge and then to wisdom are depicted in the following diagram (Figure 5), and understanding is what facilitates each stage's passage to the next. It is not possible to have understanding on a separate level (Bellinger *et al.*, 2003).

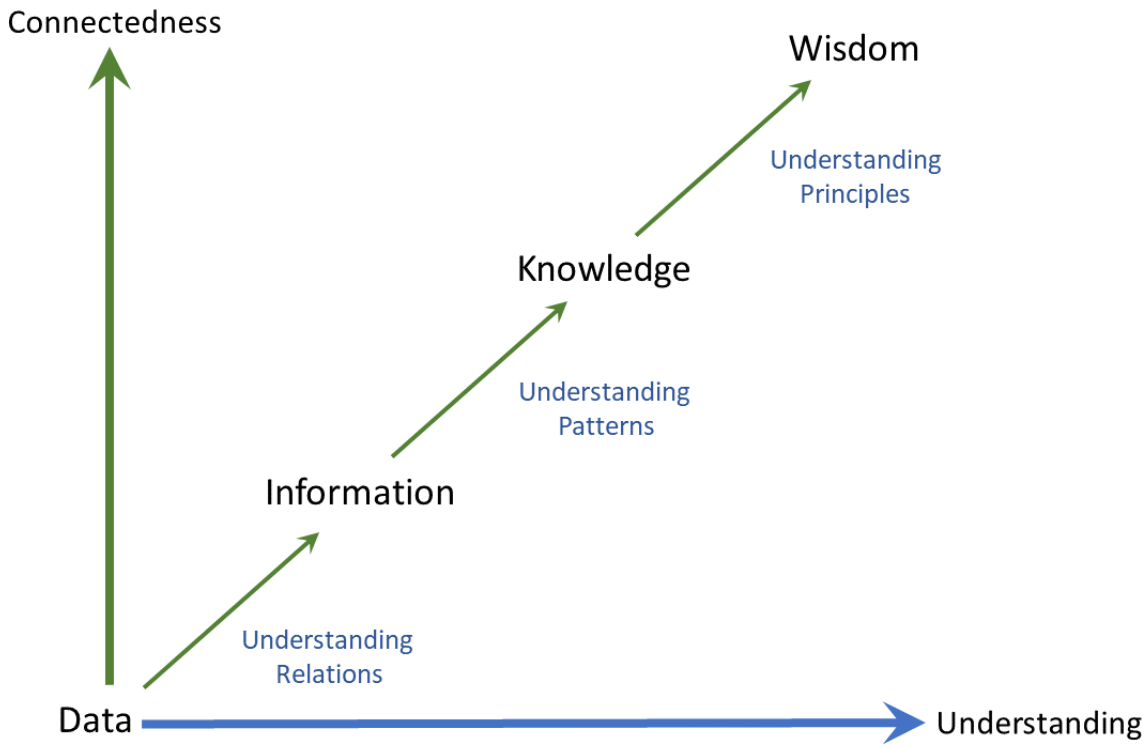


Figure 5 - Transition phases of the DIKW pyramid adapted from (Bellinger *et al.*, 2003)

As organizations use IS for achieving strategic and operational purposes, it is development is regarded as an essential organizational activity. ISs development calls for examining, designing, and executing information technology (IT) systems for boosting business operations (Xia & Lee, 2005).

As organizations have to develop IS for handling their business requirements and challenges, the role and contribution of knowledge sharing in IS projects are unavoidable and essential (Tiwana & Mclean, 2005). ERP is characterized as an IS that can combine information and information-based processes across different areas of a given organization (Kumar & van Hillegersberg, 2000). As it is considered a type of process-based IS, ERP can maintain different business operations such as accounting, finance, human resources, production and logistics (Mirzaee & Ghaffari, 2018).

Information systems are considered one of the most notable elements in the current business context. Besides supporting decision-making and control, ISs can help managers and employees in analyzing problems, visualizing complicated issues, and producing new products. An IS includes a set of interdependent variables that support decision-making and control in organizations via collecting, processing, storing, and distributing information. This system not only helps with

coordinating how an organization works, but it also helps with analyzing and simulating problems and challenges that an organization faces (Wood-Harper & Baskerville, 2016).

Organizations employ numerous information systems. When analyzing the importance of information systems in people's lives, it's helpful to categorize them into four categories based on their sphere of influence: personal information systems, workgroup information systems, enterprise information systems, and interorganizational information systems (Reynolds & Stair, 2020).

A personal information system includes information systems that improve the productivity of individual users in performing stand-alone tasks. A workgroup information system supports teamwork and enables people to work together effectively, whether team members are in the same location or dispersed around the world. An enterprise information system (EIS) is used to meet organization-wide business needs and typically shares data with other enterprise applications used within the organization. Finally, an interorganizational IS enables the sharing of information across organizational boundaries.

Development models are various processes or methodologies, selected to develop the project according to its purpose and objectives. Software development templates help improve their quality as well as the overall development process. There are several models for the Systems Development Lifecycle (SDLC), each developed for certain purposes. The SDLC is an environment that describes the activities performed at each stage of the software development process, consisting of a detailed plan where it is described how the development, maintenance, and replacement of specific software should be conducted. It is also known as the software development process.

The international standard for SDLC is ISO/IEC 12207, where it aims to define all the activities necessary to develop and maintain the software (STOICA *et al.*, 2013). The SDLC is composed of important phases that are essential for who will perform the analysis and development. This includes the phases of analysis and planning, requirements definition, design and architecture, development and implementation, product testing, and further operations and maintenance (Balaji, 2012), as illustrated in Figure 6.

First, there is the planning phase. The purpose of this first phase is to discover the scope of the problem and determine solutions, resources, costs, time, and benefits. In a second phase, the analysis of the system and requirements, is where the requirements of the project and/or solution are considered and defined. This analysis of customer needs aims to ensure that the new system

to be developed meets your expectations. Systems analysis is critical to determining the needs of a business and how these needs can be met.

The third phase, system design, describes in detail the specifications, features, and operations required to meet the functional requirements of the proposed system, which will be in place. The development phase marks the end of the initial section of the process. This phase, as the name implies, means the beginning of the process of developing the solution, taking into account all the requirements raised in the previous phases.

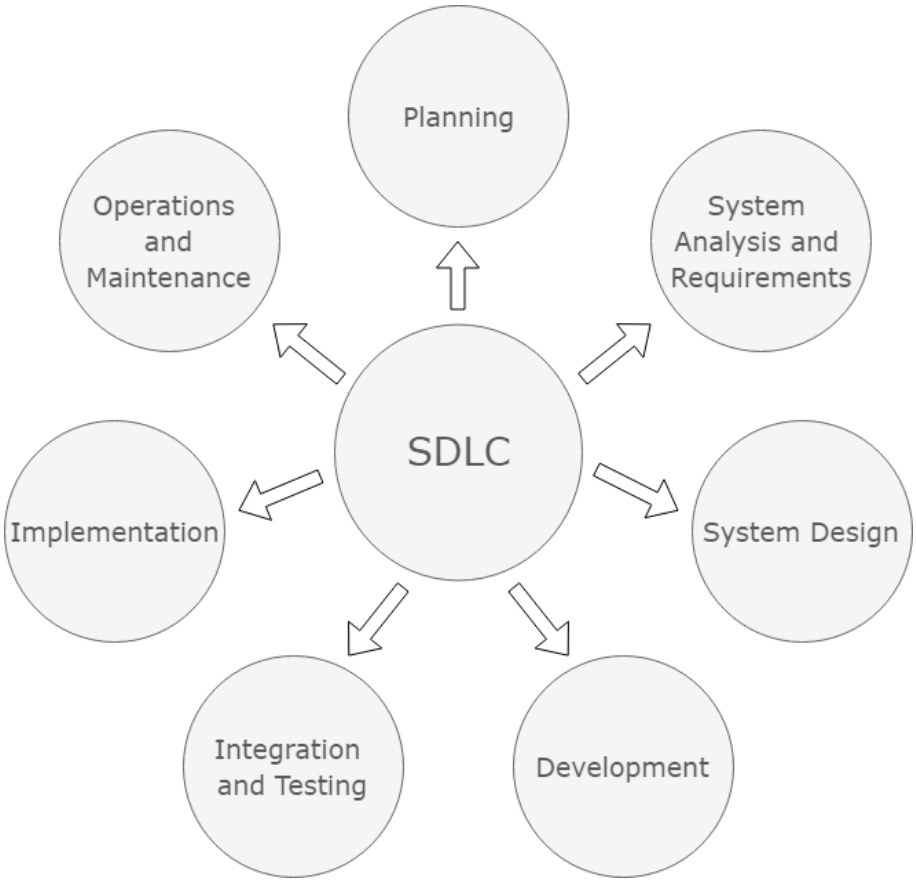


Figure 6 - Systems development life cycle [1]

The fifth phase includes integration and system testing (of programs and procedures) - usually performed by a Quality assurance professional (QA) - to determine if the proposed project meets the initial set of objectives. On the other hand, even at this stage, verification and validation are performed to help ensure that the conclusion is successful. The sixth phase, implementation,

[1] https://www.innovativearchitects.com/KnowledgeCenter/basic-IT-systems/system-development-life-cycle.aspx

comprises the installation of the system to be developed. This step, aims to put the project into production, transferring the data and components from the old system to the new system.

The seventh and final phase of the software development lifecycle involves regular maintenance and upgrades. At this stage, end users can tweak the system if they see fit to increase their performance: adding new features or meeting additional requirements[1].

Thus, with the passing of time several models for software development have emerged: cascade, spiral, V-Model, rapid prototyping, incremental and synchronization and stabilization. Of all the models mentioned here, the oldest and best known is the cascade model, it is composed of a sequence of stages in which each output from one stage is the input of the next. V-Model demonstrates the relationships between each phase of the development lifecycle and its associated test phase.

There is also agile software development (AGILE), which includes software development methodologies based on iterative and incremental development (Cohen *et al.*, 2003). Here, the requirements and solutions evolve through collaboration between self-organized and cross-functional teams, usually run by sprints (Balaji, 2012).

Waterfall

The cascade SDLC model is a sequential software development process in which progress is viewed as fluid, starting at the top and descending (cascade-like) by completing a list of phases that must be performed to successfully build a certain software. Originally, the Waterfall model was proposed by Winston W. Royce in 1970 to describe a possible software engineering practice. The Cascade model defines several consecutive phases that must be completed one after the other and move to the next phase only when the previous phase is completely completed. For this reason, the Waterfall model is recursive, because each phase can be repeated indefinitely until it is perfected. Figure 7 shows the different phases of the SDLC cascade model (Bassil, 2012).

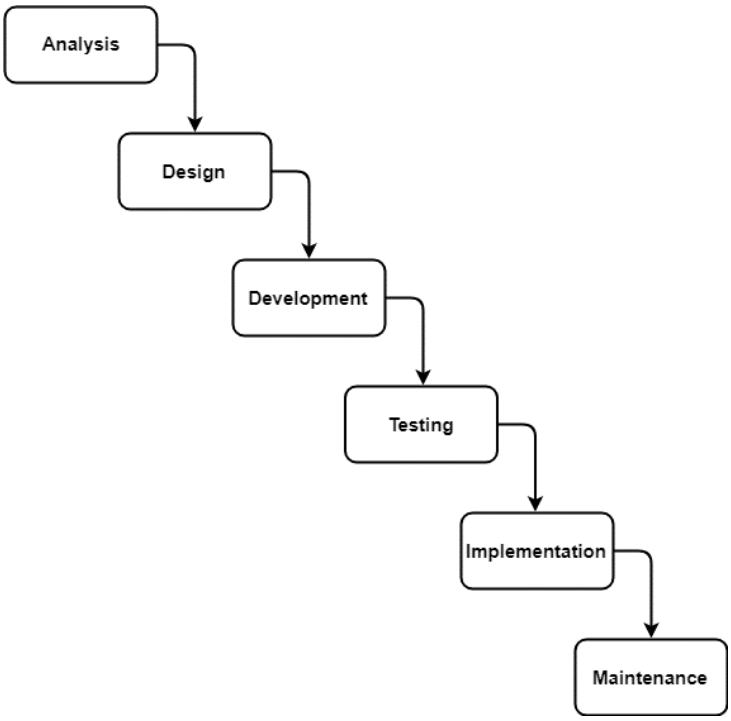


Figure 7 - Waterfall Model

Incremental Model

The incremental model, also known as the iterative cascade model, can be seen as a three-dimensional representation of the cascade model (Ruparelia, 2010). In the incremental model, development begins with a limited number of prioritized requirements, where delivery is a labor increment of the product. A set of activities ranging from requirements to code development is called iteration and, based on incremental functionality, the next batch of requirements is provided to the subsequent iteration (Figure 8).

Then the result of the next iteration is an improved work increment of the product. This process is repeated iteratively until the product achieves the necessary functionality. The number of iterations that follow varies from project to project, depending on its specificity and complexity.

The basic idea behind this method is to develop a system through iterative cycles and in smaller incremental parts, where the development and use of the system contribute with lessons to improve the increments of the next iterations. Iterative and incremental developments are essential parts of the various agile software development methodologies (Salgado, 2016).

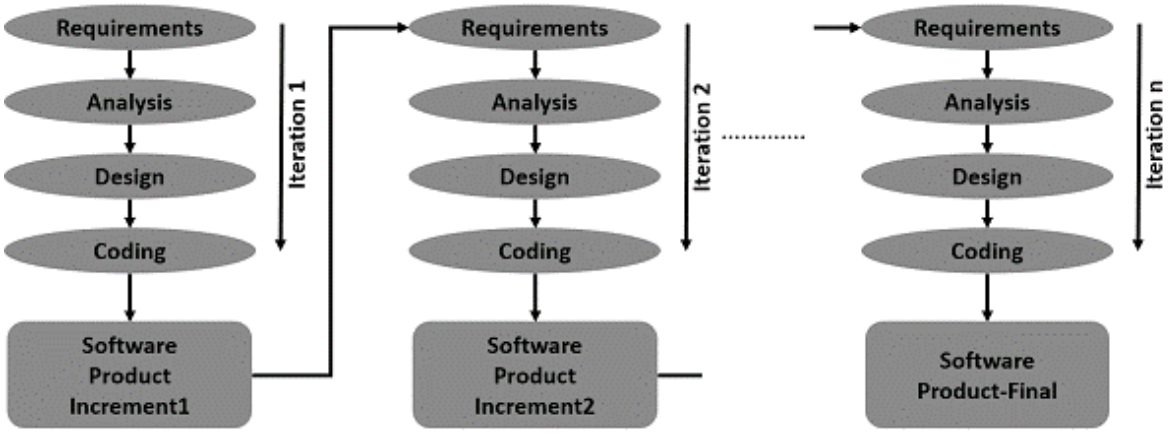


Figure 8 - Incremental Model

Spiral Model

The spiral model, Figure 9, was defined by Barry Boehm in his 1988 paper "A spiral model of software development and improvement". This model was not the first model to address iterative development, but was the first model to explain the importance of iterations. As originally anticipated, the iterations typically had an estimated duration of 6 months to 2 years. Each phase starts with a design goal and ends with the client (which can be internal) by analyzing the progress up to each moment. Analysis and engineering efforts are applied at each phase of the project, aiming at the final goal of the project.

The different phases of the spiral model are widely used for complex industrial software. This helps reduce the complexity of the software being developed. The spiral model is similar to the incremental model, with more emphasis on risk analysis. The spiral model has four phases: Planning, Risk Analysis, Engineering and Evaluation. A software project repeatedly passes through these phases in iterations (called spirals in that model). Each subsequent spiral is based on the baseline spiral. The requirements are raised during the planning phase. In the risk analysis phase, a process is carried out to identify risk solutions and alternatives (Massey & K.J.Satao, 2012).

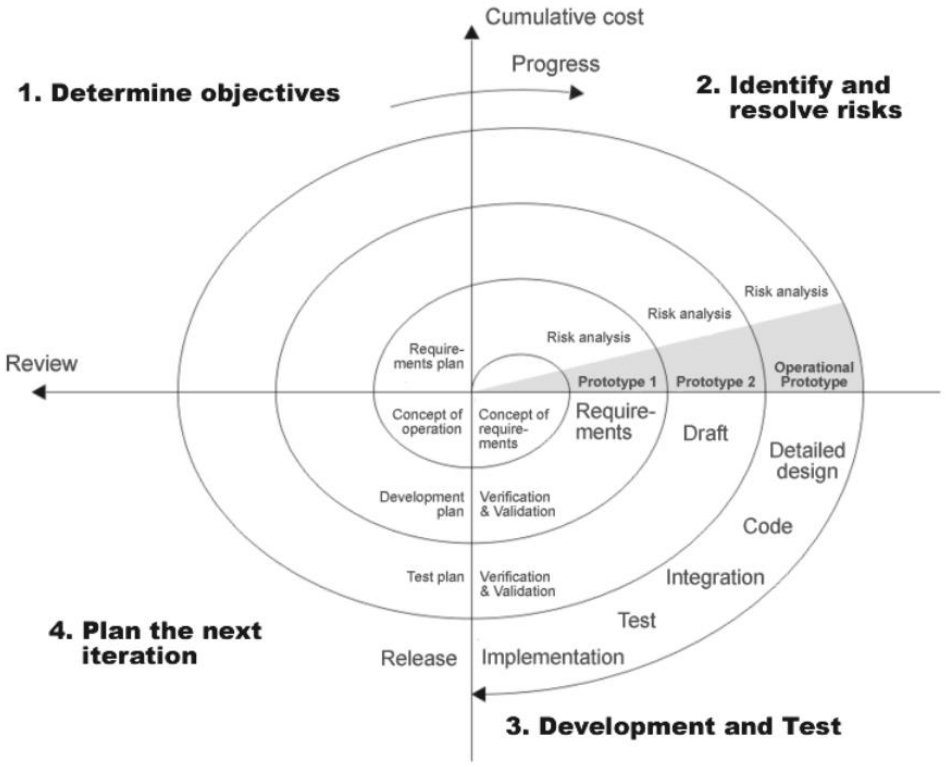


Figure 9 - Spiral Model

V-Model

The V model is a software development process that can be considered the extension of the cascade model. It was first proposed by Paul Rook (Rook, 1986) in the late 1980s and is still used today. Model V demonstrates the relationships between each phase of the development lifecycle and its associated test phase. Instead of descending linearly, the process steps are folded upward after the coding step.

The V model represents a well-structured development process in which each phase can be implemented by the detailed documentation from the previous phase. Testing activities, such as the test project, start early in the project well before coding, and therefore save a lot of project time. The purpose of Model V is to improve the efficiency and effectiveness of software development and to reflect the relationship between test activities and development activities, as shown in

Figure 10. The V model is perhaps the most traditional model followed for managing software testing (Mathur & Malik, 2010).

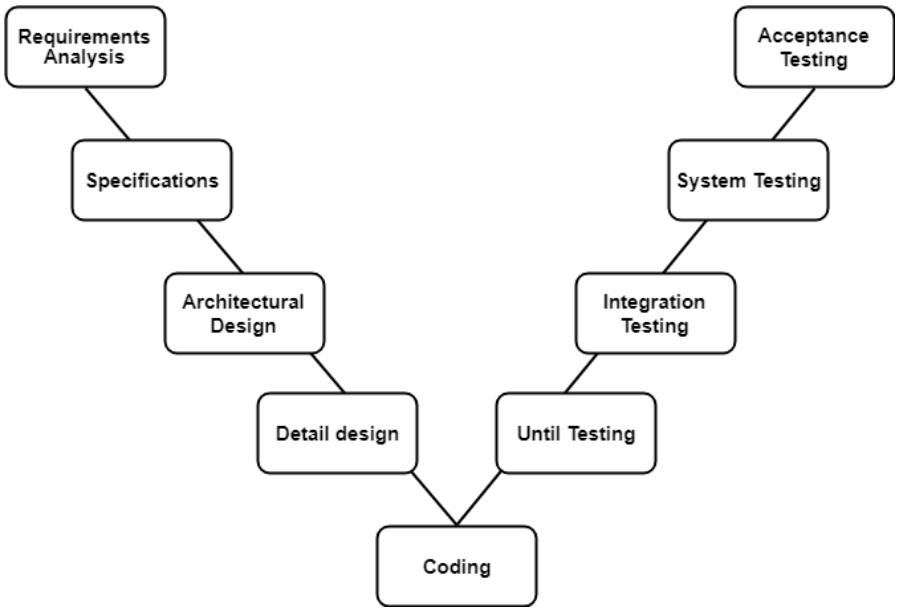


Figure 10 - V-Model

Agile software development

Agile development is based on incremental iterative development, in which requirements and solutions evolve through team collaboration, where an iterative approach over time is recommended, thus encouraging a rapid and flexible response to change. It is a theoretical framework and does not specify any specific practice that a development team should follow.

Among the agile methodologies for software development, SCRUM is the most widely used. Scrum is an agile process structure that specifies the practices needed to be followed. It is the most popular agile framework, which focuses particularly on how to manage tasks within a team development environment. Scrum also uses iterative and incremental development model, but with shorter duration of iterations. It is relatively simple to implement and focuses on fast and frequent deliveries so that the project can be validated and adjusted throughout each sprint (Salgado, 2016).

V Model + 4SRS

The original V-Model is a variation of the V-shape Royce Waterfall model, folded in half, and where the vertex contains the lowest level of decomposition. Regarding the left side of the model, it concerns the decreasing abstraction of user requirements in components by a process of decomposition and definition. On the right side of model V are represented the integration and verification of the components at higher levels of implementation, reducing the level of abstraction. This model allows a balanced representation of the process, ensuring that each step is validated before starting the next step (Santos *et al.*, 2015).

This V-Model method has been developed along the years inside our research group, balancing a traditional with an agile approach. This method is the one chosen to develop the research work, as it has already been adapted from the traditional V-Model and is currently being used by the development teams in the CCG. In addition, it is a model that uses a hybrid approach, that is, it is neither a traditional method nor Agile in its entirety, trying to gather the best of both worlds.

Ferreira *et al*, proposed a new approach to the V-Model. According to the authors, in this model, artifacts are generated based on information in previously defined artifacts, i.e., scenario-based type A sequence diagrams, use case models are based on type A, logical architecture is based on use-case models, and the type B sequence diagrams conform to the logical architecture. Organizational settings are a high-level representation of interactions between business-level entities in a given domain. The set of interactions is based on the business requirements and together with the entities and the stakeholders, is represented with the intention of describing a viable scenario that fulfils a business vision.

This approach uses a representation of Unified Modeling Language (UML) sequence diagrams to describe the interactions in the initial phase of the developmental analysis of the system. These diagrams are represented as sequence diagrams of type A. The sequence diagrams of type B, in this model, allow to derive sequences between architectural elements (AEs) present in the logical architecture.

An architectural element is a representation of the parts from which the logical architecture can be constructed. It is essential to ensure that the modeling of the type B sequence diagrams represents the same information flows that are contained in the modeling of type A diagrams, in addition to

being in accordance with the interactions between the architectural elements in the logical architecture associations (Ferreira *et al.*, 2013).

The V-Model suggested by Ferreira *et al*, Figure 11, the artifacts present on the left side of the "V" are properly aligned with the artifacts present on the right side. Thus, the type B sequence diagrams are aligned with the type A sequence diagrams and the logical architecture is also aligned with the use case models. This alignment between use-case models and the logical architecture is ensured by the application of 4SRS.

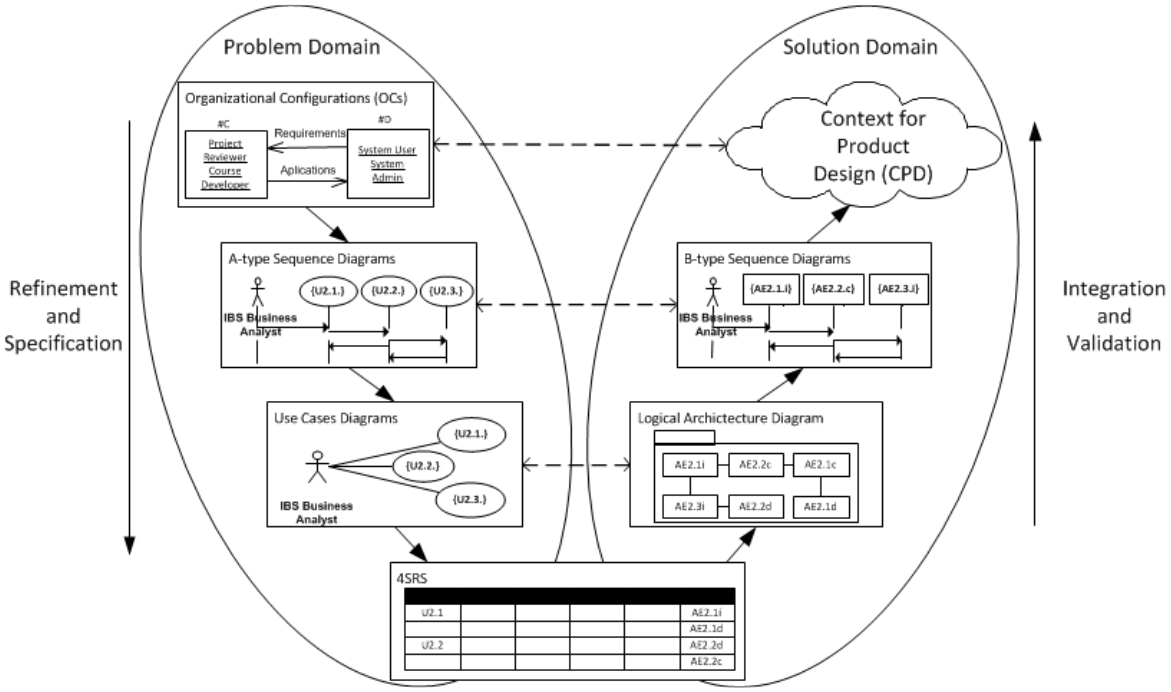


Figure 11 - V-Model Adaption for Business and IT-Alignment

At the Center for Computer Graphics (CCG), we have been using the requirements engineering approach, 4SRS, in numerous projects. One such project using this approach is UH4SP. The UH4SP project emerged as a solution for the cement production industry. In this domain, an industrial unit plant is typically composed by a set of: (1) fabric silos, responsible for storing bulk and bagged materials, which may contain grain, coal, cement, carbon black, woodchips, food products and sawdust; (2) logistics circuits, where trucks follow a path for loading or unloading material; (3) other points for transformation activities, where some industrial activities typically end in storing a good in a silo or warehouse.

The on-premises deployment is a difficulty for promoting a corporate-level management, since in order to the industrial group manager to have an integrated analysis of the group's plants, he was only able to access each plant's ERP one at a time using a remote virtualized environment. The remote business analysis was also impossible to perform in some con-texts, namely within plant's located in low connectivity spaces. The current solution did not enable the incorporation of remote technical interventions. Finally, the cur-rent solution was not able to respond to a previous need to enable third-party access (e.g., forwarders, customers, suppliers) to allow the inclusion of collaborative tools in process execution and analysis (Santos *et al.*, 2018).

To overcome these issues, the settled UH4SP project objectives were: (1) to de-fine an approach for a unified view at the corporate (group of units) level; (2) to develop tools for third-party entities; (3) in-plant optimization; and (4) system reliability. The team that developed this project proposed propose a model-based approach for a fog-based architecture design, using requirements engineering approaches to design service-oriented architectures that respond to elicited requirements. This approach firstly uses typical gathered user requirements, namely functionally decomposed UML Use Cases and a Domain-driven Design (DDD) (Evans, 2004) approach, which are input for the 4SRS method that al-lows modeling of the entire fog computing architecture in a logical architecture diagram (using UML Component notation).

The 4SRS method takes as input a set of UML Use Cases describing the user requirements and derives a software logical architecture using UML Components. The logical architecture is then refined trough successive 4SRS iterations (by recurring to tabular transformations), producing progressively more detailed requirements and design specifications (Santos *et al.*, 2018).

2.3 Systems Interoperability

Enterprise Information Systems users have begun to use a large number of heterogeneous applications to support their business rules. Currently, it is common to find a single company using hundreds of applications designed from different technologies, and running them on different operating systems and databases (Pokraev, 2009). Thus, we propose a layer that can enhance interoperability through service discovery, selection and composition process by considering syntactic, semantic and pragmatic services aspects.

The Internet of Things (IoT) is based on three main pillars, namely, (i) hardware, which includes objects with a unique ID through Radio-Frequency Identification tags (RFID) associated with sensors, (ii) connectivity, which is characterized by the infrastructure that is established between objects and sensors, (iii) services and software that support the intelligence issues so that IoT can operate. The semantic web is one of the technologies that can contribute to this support. Its goal is to process collected data and give meaning to them in specific contexts. In the IoT context, objects can communicate with each other establishing Machine-Machine (M2M) communication.

The adoption of the aforementioned technologies will be evaluated through the progress of researches, such as, (i) advancing the state of the art of semantic interoperability (ii) advancing in relation to supporting pragmatic interoperability in the IS context (iii) advancing the state of the art in organizational interoperability and other levels (iv) defining and coining the term full interoperability, or synonyms, especially in the IS context (v) suggesting ways to achieve full interoperability (vi) promote the adoption of approaches by industry to adopt the various levels of interoperability.

In this context, paired with networking and enterprise integration, enterprise information systems (EIS) interoperability gained utmost importance, ensuring an increasing productivity and efficiency thanks to a promise of more automated information exchange in networked enterprises scenarios. Enterprise integration is an essential component of enterprise engineering, concerning the usage of specific methods, models and tools, to design and to continually maintain an enterprise in an integrated state so that it can fulfil domain objectives. However, from a technical point of view, traditional legacy information systems are rigid with predefined functionality created at design time as an interpretation of the then valid rules and regulations. The rigidity of such legacy systems

makes them less sustainable to face the changes in the enterprise organization in order to be adapted to the rapid changes of its environment (Agostinho *et al.*, 2017).

According to the European Interoperability Framework (EIF, 2010) three layers of interoperability can be defined: (1) Technical interoperability, or syntactic interoperability, at the lowest level, encompassing technical problems of connection of systems, devices, applications and services, including aspects such as communication protocols (TCP / IP), interconnection services, specific data formats (XML or SQL), data integration and middleware (ESB), presentation and exchange of data, accessibility and security services; (2) Semantic interoperability, defining the "meaning" and use of data / messages, allowing data / messages to be received, combined and processed in a given system, sent to another system by which they can be automatically recognized and processed.

The state of the art in the industry is to enrich data and messaging with metadata, resorting to thesauri or management of metadata arming (such as Microsoft SharePoint) or ontological definitions of objects and concepts to be exchanged using representations or languages of common logic (CL) such as the Web Ontology Language (OWL), designed to represent rich and complex knowledge (OWL working group, 2012) in association with the Resource Description Framework (RDF) for Web data exchange (RDF working group, 2014); and (3) Organizational Interoperability, focusing on business process alignment and organization, automated processing of sub-functions in a single automated inter-organizational workflow through the use of a common service... architecture (e.g. SOA) and the lower layers mentioned above.

So there are three levels of integration that can be identified in a company: (1) Physical integration, or level of communication, that deals with system interconnections and data exchange (interconnection of physical devices, computers and database systems through computer networks), (2) Application Integration, or level of cooperation, which deals with the interoperability of software applications and database systems in heterogeneous computing environments and (3) Business Integration or Collaboration Level, which deals with the coordination of roles, processes, and people who manage, control, and monitor company operations (Romero & Vernadat, 2016).

Supply chain is the dominant organizational structure in the industry. This structure can be seen as a global network of suppliers, producers, transporters, and customers who need to share technical and business information in an integrated way. This information, previously shared in a

variety of ways, including telephone conversations, should now be passed electronically and correctly among all partners in the Supply chain. The term "interoperability" is commonly used for this feature (Ray & Jones, 2006). However, as noted earlier, interoperability is much more than an exchange of information between Supply Chain agents. Interoperability is also related to the fact that it is possible to integrate applications in order to reduce and/or optimize the work process performed manually.

In the past, most companies created their own applications and designed their own set of services, but today the situation is changing. In today's globalized and networked society, companies need to collaborate with other companies to satisfy their own aggregate values and exploit market opportunities. An important issue in global collaboration and cooperation is the development of interoperability. Interoperability is also the ability of two or more systems or components to exchange information and use the information that has been exchanged. Developing interoperability can induce many problems that need to be solved to achieve specific goals. The solution to these problems may be a long iterative procedure that may fail due to a lack of consensus among partners or the high cost of solution applicability (Guédria *et a*l., 2011).

Nowadays, it is still possible to observe and detect some of these gaps in countless companies from different activity sectors. Here we refer mainly to the excessive use of verbal communication to effect the passage of information between departments, excessive use of excel sheets as a way of storing information regarding the management and planning of the production process, manual exports of application logs that are later used to enter data in other applications, among other aspects. This lack of interoperability results in this, but in many other cases a delay of the work process and a failure of communication / knowledge in relation to what the other organizational areas are doing.

Industry 4.0 is a term introduced by Siemens and refers to the integration of interconnected systems in industry and is known as the fourth industrial revolution. After the first industrial revolution "Mechanization", as a result of the invention of the steam engine, the second "mass production" with the help of electricity and the third "Scanning" through the use of electronic devices and IT, this marked the emergence of the fourth Industrial Revolution through the use of Cyber Physical Systems (CPS), Internet of Things (IoT). and Services (Bagheri *et al.*, 2015).

The CPS refers to a new generation of systems with integrated computational and physical capabilities that can and do allow interaction with humans through various modalities (Baheti & Gill, 2011). In other words, they are systems that are in close connection with the physical world and its processes, providing and using data access and processing services (Monostori, 2014). CPS connected to the Internet is often called "*Internet of Things*" (Jazdi, 2014).

Nowadays, IoT is defined as a dynamic global network infrastructure with auto configuration capabilities based on standardized and interoperable communication protocols, where physical and virtual things have identities, use intelligent interfaces and are integrated into the network. More specifically, we can provide an example of the integration of sensors, RFID tags and communication technologies, which serve as the basis of IoT, and explains how a variety of physical objects and devices can be associated with the Internet and communicate with each other (Xu *et al.,* 2014).

Service Oriented Architecture (SOA) can be used to assist the Internet of Things (IoT) due to its importance in the integration of heterogeneous systems and devices. IoT is designed to link "things" to the network. Interoperability, adaptability, and reconfiguration needs for sophisticated applications in the automation sector have increased dramatically in recent years. According to Wenbin Dai *et al*, the adoption of service-oriented architectures may be a viable solution to address these challenges (Dai *et al.*, 2017). SOA can be described[2] as a service-oriented architecture, that is, it is a software architecture whose fundamental principle is the implementation of the application functionalities in the form of services. These services are connected and provide accessible interfaces, usually through web-services.

On another view the evolution of a large volumes of data present in organizations, including major industries, is called Big Data. Big data can then be classified as large volumes of datasets with a high degree of complexity. In organizations, the workflow includes data storage, data management, data maintenance, data integration, and data interoperability. Among these levels, data integration and interoperability can be the two main areas of focus in organizations in order to optimize workflows. Data integration and interoperability are complex challenges for companies deploying Big Data architectures because of the heterogeneous nature of the data they handle (Kadadi *et al.*, 2014).

[2] https://pt.wikipedia.org/wiki/Service-oriented_architecture

The size of data stored is growing nowadays, thus it is becoming important to understand large, complex, information-enriched data in several fields (e.g., technology, business, industry, science). It is now relevant in the competitive world to have the ability to extract useful knowledge hidden in the large amount of data and to act on the knowledge (Jothi *et al.*, 2015).

The process of analyzing large information repositories and discovering implicit and potentially useful information is usually designed as Data Mining, or Knowledge Discovery on Databases (Han *et al.*, 2011). Data Mining is able to discover hidden relationships and to reveal unknown patterns and trends by digging into large amounts of data, which can be used to support managerial decisions (Sumathi & Sivanandam, 2006). Data Mining includes several data analytics processing steps, such as business and data understanding, data processing, data modeling (where Machine Learning algorithms are applied), evaluation and implementation.

Interoperability can be defined as the capacity of two or more software components to interact despite differences at the language, interface, and execution platform levels. (Wegner, 1996). Digital interoperability concerning data or information exchange becomes a key enabler for the next evolutions that will massively rely upon digitalization, artificial intelligence, and autonomous systems (Pan *et al.*, 2021).

Interoperability is vital for businesses to efficiently face the challenges of modern competition, discover new business opportunities, and improve customer service. The sharing of semantically coherent business documents is one technique to accomplish interoperability. According to (Rezaei *et al.*, 2014) there are four types of interoperability. The interoperability types are technical, syntactic, semantic, and organizational interoperability.

Technical interoperability is achieved when services or information may be shared directly and satisfactorily between communications-electronics systems or items of communications-electronic equipment. The degree of interoperability must be defined when referring to specific cases. The capacity to share data is defined as syntactic interoperability. Data formats are frequently related with syntactic compatibility. Even if merely in the form of bit-tables, the messages sent by communication protocols should have a well-defined grammar and encoding.

The capacity to operate on data according to agreed-upon semantics is referred to as semantic interoperability. Semantic interoperability is usually associated with the definition of content, and it deals with human interpretation of that content rather than machine interpretation. As a result,

interoperability at this level signifies that participants have a common understanding of the concept of the content (information) being transferred.

Organizational interoperability is concerned with the delineation of authority and responsibility in order for interoperability to occur under favorable circumstances. The ontology-based semantic interoperability framework has proven to be an effective solution for business interoperability (Gyrard *et al.*, 2018).

From the analysis of existing interoperability projects, two main research gaps have been identified. The first is the lack of standard taxonomy on relevant domain concepts. The second is the lack of empirical research and evidence of the quantifiable benefits of ontology-based interoperability in enterprises. This is one of the main reasons why business owners are reluctant to switch to the ontology-based semantic interoperability approach over the conventional approach (Komninos *et al.*, 2016).

The Levels of Information Systems Interoperability (LISI) project was initiated in 1993 and is a reference model and process for assessing information systems' interoperability required. It is a discipline and a process for defining, measuring, assessing, and certifying the degree of interoperability required or achieved between organizations or systems (Kasunic, 2001). This model highlights the stages through which a system should logically progress, or "mature," to improve its interoperability capabilities.

In terms of system interaction and the system's ability to communicate and share information and services, LISI considers five levels of sophistication. Each higher level of system-to-system iteration provides a verifiable improvement in capabilities over the preceding one. A clear prescription of the common suite of necessary capabilities that must be embedded in all information systems that wish to interoperate at a certain level of sophistication is a fundamental component of interoperability assurance. The 5 levels listed from 0 to 4 are shown in the Figure 12.

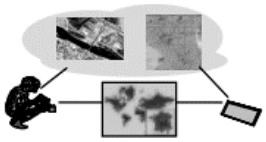
Information Exchange	Level	Computing Environment
• Distributed global information and applications • Simultaneous interactions with complex data • Advanced collaboration, e.g. interactive COP update • Event-triggered global database update	**4** **Enterprise** Interactive manipulation; shared data and applications	
• Shared databases • Sophisticated collaboration, e.g., Common Operating Picture	**3** **Domain** Shared data; "separate" applications	
• Heterogeneous product exchange • Basic collaboration • Group collaboration, e.g., exchange of annotated imagery, maps with overlays	**2** **Functional** Minimal common functions; separate data and applications	
• Homogeneous product exchange, e.g., FM voice, tactical data links, text files, transfers, message, e-mail	**1** **Connected** Electronic connection; separate data & applications	
• Manual gateway, e.g., diskette, tape, hard copy exchange	**0** **Isolated** Non-connected	

Figure 12 - Overview of the LISI Interoperability Maturity Model adapted from (Kasunic, 2001)

Interoperability is shown in a manual context at Level 0, Isolated. Systems are autonomous and do not permit interconnection. This level involves the manual integration and extraction of data across many platforms. Interoperability at level 1, Connected, is peer-to-peer and relies on electronic connections between systems for information sharing. At level 2, Functional interoperability exists in a distributed environment where systems are located on local networks, allowing for the exchange of data across systems. At this level, diverse data represented in a basic information format are merged and shared between functions and systems. Level 3, Domain, is concerned with interoperability in an integrated environment in which various users are permitted data access via system connections with long-distance networks. At this level, apps exchange data with one another based on the domain data model. At the last level, which corresponds to the company, we

speak to interoperability in a global environment where it is possible to used global information accessible across several domains.

At this level we can have access to complex data at the same time by several users, in different applications, being able to benefit from collaboration between applications and having common data throughout the company eliminating ambiguity. In addition, each level's prescription of capabilities must cover all four enabling attributes of interoperability, namely: procedures, applications, infrastructure and data (Kasunic, 2001).

- Procedures: These establish standards and indicate procedures and processes which influence the functional operational requirements and the integration of the system.
- Applications: These refer to the functions that the system must perform. These functions are presented in the form of application programs based on the user who executes them or in the support of a specific set of processes or procedures.
- Infrastructure: This attribute is related to the support to the systems' operations and has 4 sub-components.
- Data: The data structures are used to support both the applications and the system infrastructure.

2.4 Ontologies in Requirements Engineering

Information systems have become increasingly complex due to the constant growth of data, different structures, technologies and constant evolution of requirements by users. Information systems have become increasingly so, and given all these characteristics, the applications that are currently developed, denote a great effort at the level of design and development. In order to give an adequate answer to the needs identified, a detailed and detailed study of the functionalities and fields of the applications is necessary, thus also defining and relating concepts.

However, as the applications vary from organization to organization and as the resulting realities are also distinct, it is necessary at an initial moment to understand the business. This is an extremely important step in the development of Information Systems (IS). Thus, it is necessary to understand in detail the organizational context in which we are inserted and where we will operate. Conceptual models help us explain and make sense of the situations in which we act (Fernandes & Machado, 2015).

The Ontologies come from the philosophy, with the intention of specifying formally and explicitly a conceptualization. A conceptualization consists of an abstract model of a domain where the relevant models and the relations that exist between them are identified. Ontology defines the common terms (meanings) used to describe and represent an area of knowledge. It may vary according to the taxonomy (knowledge of the hierarchical structure), to a thesaurus (words and synonyms), to a conceptual model (with more complex knowledge) or to a logical theory (with more complex, consistent and meaningful knowledge) (Trinkunas & Vasilecas, 2007).

Ontologies refer to sources of knowledge, just as dictionaries are related to literary works. As in the dictionary, ontologies collect and organize terms of reference. Similarly, the definitions of a dictionary indicate the relations between words, whereas ontologies give us the relations between the terms (Mitra *et al.*, 2000). Thus, since ontologies allow us to represent the relations between terms, and if the terms are defined in a database, we can then represent an ontology through graphs. The main measure of success for an information system is how well it does what it was made to do.

Requirements Engineering (RE) is the process of determining that purpose by identifying stakeholders and their requirements and then recording them for future study, communication, and implementation (Nuseibeh & Easterbrook, 2000). Software Requirements Engineering is a subtask of Software Engineering that presents methods and tools to facilitate the defining of all desired software objectives and capabilities (Fernandes & Machado, 2015).

Due to a lack of understanding and divergent interpretations, software developers are frequently need to rebuild and iterate. (Noppen *et al.*, 2007). Various other issues must be resolved during RE activities in order to generate consistent and complete requirements during the early phases of software development and feed succeeding phases effectively. Given the widespread absence of technical knowledge, one of these difficulties is the management of participating organizations (via their stakeholders) in requirements collection.

Effective tools must thus be made available to do a complete analysis, taking into consideration both unique and general needs, and to manage requirements as a collaborative process. According to (Castañeda *et al.*, 2010) the research of an information system's needs should lead to the construction of well-defined functionality and features that all stakeholders agree on. The software may not meet the expectations of users if the functionalities are stated as insufficient or wrong.

There are some factors that could lead to an inadequate process of requirements such as (Castañeda *et al.*, 2010):

- Ambiguous Requirements: which result in wasted time and work and leads to erroneous product testing. Their origins can be traced back to several stakeholders who produce various interpretations of the same demand. Furthermore, the same demand can be interpreted differently by different stakeholders.
- Insufficient Specifications: they result in the absence of critical needs. This causes developers to be frustrated because their work is based on faulty assumptions, and as a result, the desired product is not generated, causing the clients to be dissatisfied.
- Requirements not completely defined: they make the project's secure planning and monitoring impossible. Because of a lack of understanding of the needs, optimistic estimates are made, which backfire when the agreed-upon boundaries are exceeded.
- Dynamic and changing requirements: which necessitates ongoing requirements revision to assist in understanding new customer needs and determining how they might be met.

Ontologies can be employed to mitigate the detrimental effects of the preceding elements on RE processes. (Castañeda *et al.*, 2010). According to (Decker *et al.*, 2005) have taken the use case technique and turned it into a clear example. They supplement the usual use cases documentation with additional papers and structures. These new papers, known as templates, allow knowledge to be captured. Aside from the ontology of the documents, each has metadata. The authors also allow the ontology to be extended to link multiple use cases, making it easier to find documents of the same type in other projects. As previously stated, ontologies have been shown to assist stakeholders in clarifying their information needs and producing semantic representations of materials.

All of these approaches can be used in some way to build an ontology for capturing the structures of RE documents, so encouraging the adaption of the same information in various formats to be understood by all stakeholders. Furthermore, an ontology with this purpose can be reused in a variety of projects to arrange knowledge that is specific to each one (Castañeda *et al.*, 2010). The use of ontologies to describe needs knowledge has been under investigation for quite some time. (Lin *et al.*, 1996) proposed one of the first attempts in this field. They present a generic solution with dependencies and linkages between recorded and stored needs that is unambiguous, precise, reusable, and easy to extend.

Knowledge Acquisition in Automated Specification (KAOS), a goal-oriented requirements engineering approach with a broad set of formal analysis techniques (Lapouchnian, 2005), is one of the most widely used methodologies. KAOS is a multiparadigm framework that allows several levels of expression and reasoning to be combined: semi-formal for modelling and structuring goals, qualitative for selecting among alternatives, and formal when more exact reasoning is required (Schmidt *et al.*, 2004).

Given the importance of knowledge reuse and its use in Requirements Engineering, (Deridder & Wouters, 1999) state that one of the most difficult aspects of reusing use cases is finding ones that are related. As a result, and in order to achieve reuse, they offer a semiformal description that, when combined with a "human" format, can allow use cases to be reused. An ontology can be used to promote information integration or permit communication amongst software agents at run time. In both cases, the RE process is where the ontology is created.

Labels, concepts, and relations are the three types of information in the stated ontology. These notions may be used to develop a variety of rules and queries that, when combined with algorithms and a logic inference machine, allow for the discovery of related use cases. An ontology can be used to promote information integration or permit communication amongst software agents at run time. In both cases, the RE process is where the ontology is created (Castañeda *et al.*, 2010).

The term ontology can be used to refer to a set of knowledge that describes some domain, typically a domain of common knowledge, through a representation vocabulary. Ontologies are defined by (Gruber, 1993) as explicit specifications of a conceptualization. In addressing this definition, (Vickery, 1997) describes that conceptualization means an abstract view of the domain that one wishes to organize for some purpose. As a result, the ontology may be compared to a system's conceptual schema, which will provide a logical explanation for data sharing. An ontology represents a domain in knowledge organization, allowing the establishment of a knowledge base that includes the knowledge symbolically expressed to solve specific issues or answer queries about the domain (Netto & Lima, 2017).

Although ontologies do not always have the same structure, most of them share common characteristics and basic components, and well-defined terms can be identified (Almeida & Bax, 2003). The basic components of an ontology are classes, instances, properties, rules, axioms, and values. According to (Ramalho & Fujita, 2011) the classes and subclasses of an ontology group a

set of elements that are represented and categorized according to their similarities, taking into account a specific domain.

The elements can symbolize anything from inanimate objects to scientific hypotheses or theoretical streams, and they might be physical or intellectual. According to the properties of the classes, relations, and constraints described, instances represent the values of classes and subclasses, forming a representation of things or individuals belonging to the modelled domain. The properties, on the other hand, are traits ascribed to classes or instances to enrich the semantics of the ontology. They can be classified as either descriptive or relational. The former describes the classes' attributes, descriptors, and/or quality, whereas the latter describes the relationships between classes in the same hierarchy or not, as well as the sorts of relationships that exist in the represented domain.

The rules and axioms are logical assertions that allow imposing conditions, officially expressing the ontology's rules, and allowing automatic inferences to be drawn from information that may not be apparent in the domain but is implicit in the ontological framework. Finally, the values in each class describe the formats and types of data that are acceptable.

There are several approaches associated with the type of ontologies that are proposed by several authors. According to (Almeida & Bax, 2003) there are 5 different types of approaches which may be regarding the function, the degree of formalism, the application to the structure and the content. According to Mizoguchi and Ikeda (Mizoguchi *et al.*, 1995), regarding function, ontologies can be domain, task or general. The former provide vocabulary about concepts, relationships, activities and rules. The task ontologies provide systematized vocabulary of terms, specifying the tasks that may or may not be in the domain. Finally, general ontologies include vocabulary related to things, events, time, space, chance, behavior and functions.

The second approach, concerning the degree of formalism, created in 1996 by Uschold and Gruninger (Uschold & Grüninger, 1996) addresses the highly informal ontologies, which is expressed in natural language, the semi-informal ones, also in natural language, but in a restricted and structured way, and the semi-formal ones, which has an artificial language formally defined. In addition, he presents the strictly formal ontologies, whose terms are defined with formal semantics, theorems and proofs.

In relation to its application, Jasper and Uschold (Uschold & Jasper, 1999), define the author-neutral ontologies as specification and common access to information. The neutral authoring ontologies consist of an application that is written in a single language and then converted, through information reuse, so that it may be used in other systems. In turn, in ontologies as specification, an ontology is created for a certain domain that is used for documentation and maintenance in software development. Common access ontologies occur when the vocabulary is inaccessible and the ontology makes the information intelligible, providing shared knowledge of the terms.

Other approach presented by Haav and Lubi (Haav & Lubi, 2001), lies in the structure of the ontologies. These can be high-level, domain and task. The former describes general concepts related to all concepts of the ontology (space, time, among others) that are dependent on the problem and/or the domain. Domain ontologies, as the name implies, describe the vocabulary related to the domain, such as health or computer science. Task ontologies describe a task or activity by inserting the specialized terms in the ontology.

The last approach, related to content, is presented by Var-Heijist Schreiber and Wieling (van Heijst *et al.*, 1997) and presents 7 types of ontologies: terminological, information, knowledge modelling, application, domain, generic and representation. The terminological ontologies specify terms that will be used to represent knowledge in a given domain. In turn, information ontologies specify the record structure of databases and knowledge modelling ontologies specify conceptualizations of knowledge and have a semantically rich internal structure that are refined for the domain in which they are used.

The fourth type of ontology, application ontologies, contain the definitions needed to model the knowledge of an application. Domain ontologies, on the other hand, express conceptualizations specific to a particular domain of knowledge. Finally, generic ontologies are similar to domain ontologies, but the concepts are more general and common to several fields, and representation ontologies explain the conceptualizations that are behind the knowledge representation formalisms.

The use of ontologies, as previously mentioned, has grown exponentially. Ontologies enable a better understanding of a given knowledge area, integrating global and local views of the system, as such, there are numerous benefits associated with their use. Next, two different views are presented regarding the advantages of using ontologies.

According to (Duarte & Falbo, 2000), ontologies are useful to support the specification and implementation of a complex computational system. The development of an ontology allows for a better understanding about a particular area of knowledge that is being modelled. Each individual expresses the individual knowledge they have possessed and, subsequently, a generic model is created that will allow the knowledge about a certain domain to increase. Another advantage mentioned by the authors is the possibility of reaching a consensus about an area of knowledge, eliminating communication problems arising from differences in the definition of the concepts used.

In addition, ontologies help individuals, who are not experts in an area, to acquire knowledge. In this way, individuals will be faced with aggregate knowledge, which gathers the consensus of the community, and there will be fewer divergences and doubts in the areas studied by them. According to (Durán-Muñoz & Bautista-Zambrana, 2017), one of the main advantages of using an ontology is the clarity that it brings to the organization as well as the modeling of expert knowledge through a macro-structure. León, in 2009, explains that a macrostructure is a way to represent the underlying conceptual structure, in general terms, of a given domain, in order to model its basic categories. Ontologies are considered to be one of the most relevant ways of expressing this macro-structure and, as such, they become valuable resources.

Furthermore, ontologies offer the possibility of choosing the level of specificity that should be represented, i.e., they may focus on any of the more specialized levels or more general content, according to the objectives of the project. Another advantage is systematization in information retrieval, i.e. the clear and gradual organization obtained through ontologies results in greater control, harmonized and systematic terminological resources, which, moreover, constructed according to user needs.

Ontologies make it possible to create systematic and coherent definitions. The use of ontologies in terminology meets both requirements since they provide the possibility of organizing and clarifying conceptual information related to concepts and their differences with other similar concepts and, consequently, of producing systematic and coherent definitions. Ontologies, whether in the form of graphs, tables, or other type of representation, have proven to be a useful tool to deal with the phenomenon of multidimensionality found in specialized domains, since they facilitate the organization and modelling of complex knowledge representations, and eliminate (or reduce) the negative effects regarding ambiguity, replication, among others.

Changes in concepts and terms are caused by several reasons: new realities to be named, new uses of terms, obsolescence of concepts/terms, among others. In this context, terminology needs to be prepared to deal with these kinds of changes and with the new communicative situations that may occur. Thus, it is essential to create more flexible and dynamic models for representing expert knowledge, capable of managing and integrating information from different sources, and of adapting information to users' needs.

Finally, multilingualism, although often an essential aspect of terminology projects, can be considered a problem in structuring conceptual information, since concepts are independent of language but not of culture. However, ontologies provide different solutions for adding multilingual information, depending on the objectives of the resource being built, and help terminologists and other users to find accurate data.

Despite the numerous advantages presented when it comes to the use of ontologies, there are some barriers that may make them inconvenient or difficult to use (Durán-Muñoz & Bautista-Zambrana, 2017).

One of the barriers is the existence of a large number of ontology languages to encode or edit ontologies, such as Resource Description Framework (RDF), RDF Schema, Ontology Inference Layer (OIL), DARPA Agent Markup Language (DAML) +OIL and Ontology Web Language (OWL). The existence of these languages facilitates data exchange between different ontology-related applications, but, on the other hand, makes it impossible to exchange or reuse data between systems that do not share the same languages. Another important barrier to mention is the difficulty of transferring expert knowledge from texts or domain experts to abstract and effective conceptual representations.

There are currently two ways of representing synonyms, which causes disorganization in knowledge representation and hinders the reuse and interchangeability of ontologies. The first is through a logical relation that represents an exact correspondence and the second through denominative variants. In the first case, both synonyms are considered different concepts and, therefore, both are included in the ontology, but in the second case, the units are considered one concept with two or more related terms, that is, two graphical representations at the terminological level but only one at the conceptual level.

Finally, the last barrier concerns the lack of adequate tools, since, there are two types of tools that can be used: standard ontology editing tools or an ontology based on a terminology resource editor. Regarding the first option, it is not always easy to adapt standard ontology editors to terminology purposes, and the work involved can be time-consuming. Furthermore, standard ontology editors include many technical features (e.g. logical inferences) that are not generally needed for terminology projects and that may slow down the working process. As for the second option, there are some specific tools for creating ontology-based terminology resources, mainly developed within research groups or projects.

2.5 Conclusions

Information Systems Development, Interoperability, and Ontologies are the themes of the literature evaluated throughout this chapter. Relatively to the Development of Information Systems, it is made in a first instance a framework where the DIKW pyramid is approached with the intuit of contextualizing and highlighting the importance of each of the phases and concepts in it (data, information, knowledge and wisdom). After this framework is presented an examination of the current SDLC models used in software development, namely traditional methodologies like the waterfall model, the spiral model and others considered agile like scrum. Thus, section 2.2 culminates with an explanation of the V + 4SRS Model that will serve as the basis for work to be presented in the input sections.

Next, in section 2.3, a study and survey of the state of the art about the different types of interoperability that exist is conducted. This study is primarily aimed at elucidating semantic concerns related to interoperability.

Finally, in section 2.4, the study focuses on an analysis and explanation of what ontologies are and how the correct elicitation of needs influences the ontology design and its representation.

The connection between the topics of Information Systems Development, Interoperability and Ontologies can be understood with the help of this analysis and systematization of the current state of the art.

2.6 References

Agostinho, C., Ferreira, J., Pereira, J., Lucena, C., & Fischer, K. (2017). Process Development for the Liquid-sensing Enterprise. *Proceedings of the 5th International Conference on Model-Driven Engineering and Software Development*, *Modelsward*, 239–249.

Almeida, M. B., & Bax, M. P. (2003). Uma visão geral sobre ontologias: pesquisa sobre definições, tipos, aplicações, métodos de avaliação e de construção. *Ciência Da Informação*, *32*(3), 7–20.

Bagheri, B., Yang, S., Kao, H.-A., & Lee, J. (2015). Cyber-physical Systems Architecture for Self-Aware Machines in Industry 4.0 Environment. *IFAC-PapersOnLine*, *48*(3), 1622–1627.

Baheti, R., & Gill, H. (2011). *Cyber Physical Systems*. *1*.

Balaji, S. (2012). Waterfall vs v-model vs agile : A comparative study on SDLC. *WATEERFALL Vs V-MODEL Vs AGILE : A COMPARATIVE STUDY ON SDLC*, *2*(1), 26–30.

Bassil, Y. (2012). A Simulation Model for the Waterfall Software Development Life Cycle. *International Journal of Engineering & Technology*, *2*(5), 2049–3444.

Bellinger, G., Castro, D., & Mills, A. (2003). *Data , Information , Knowledge , and Wisdom*. 5–7.

Benbya, H., Nan, N., Tanriverdi, H., & Yoo, Y. (2020). Complexity and information systems research in the emerging digital world. *Mis Quarterly*, *44*(1), 1–17.

Beynon-Davies, P. (2007). Informatics and the Inca. *International Journal of Information Management*, *27*(5), 306–318.

Beynon-Davies, P. (2009). The 'language' of informatics: The nature of information systems. *International Journal of Information Management - INT J INFORM MANAGE*, *29*, 92–103.

Castañeda, V., Ballejos, L., Caliusco, M. L., & Galli, M. R. (2010). The Use of Ontologies in Requirements Engineering. *Global Journal of Researches in Engineering*, *10*(6), 2–8.

Cohen, D., Lindvall, M., & Costa, P. (2003). A State of the Art Report: Agile Software Development. *DACS SOAR Report*.

Dai, W., Vyatkin, V., Christensen, J. H., & Dubinin, V. N. (2017). Comments on bridging service-oriented architecture and IEC 61499 for flexibility and interoperability. *IEEE Transactions on Industrial Informatics*, *13*(4), 1494–1496.

Decker, B., Ras, E., Rech, J., Klein, B., & Hoecht, C. (2005). Self-Organized Reuse of Software Engineering Knowledge Supported by Semantic Wikis. *Workshop on Semantic Web Enabled Software Engineering (SWESE)*, 1–12.

Deridder, D., & Wouters, B. (1999). The Use of Ontologies as a Backbone for Software Engineering Tools. *In: Fourth Australian Knowledge Acquisition Workshop (AKAW'99)*.

Duarte, K. C., & Falbo, R. de A. (2000). Uma ontologia de qualidade de software. *Workshop de Qualidade de Software*.

Durán-Muñoz, I., & Bautista-Zambrana, M. R. (2017). Applying Ontologies to Terminology: Advantages and Disadvantages. *HERMES - Journal of Language and Communication in Business*, *51*, 65–77.

EIF. (2010). *European Interoperability Framework (EIF) for European public services* (p. 40).

Evans, E. (2004). *Domain-Driven Design*. *7873*(415).

Fernandes, J. M., & Machado, R. J. (2015). *Requirements in Engineering Projects*. Springer.

Ferreira, N., Santos, N., Machado, R. J., Fernandes, J. E., & Gasević, D. (2013). A V-model approach for business process requirements elicitation in cloud design. *Advanced Web Services*, *9781461475*, 551–578.

Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition*, *5*(2), 199–220.

Guédria, W., Naudet, Y., & Chen, D. (2011). *Maturity Model for Enterprise Content Management*. *9*(June), 9.

Gyrard, A., Zimmermann, A., & Sheth, A. (2018). Building IoT-Based Applications for Smart Cities: How Can Ontology Catalogs Help? *IEEE Internet of Things Journal*, *5*(5), 3978–3990.

Haav, H.-M., & Lubi, T.-L. (2001). *A Survey of Concept-based Information Retrieval Tools on the Web*.

Han, J., Kamber, M., & Pei, J. (2011). Data Transformation by Normalization. In *Data Mining: Concepts and Techniques*.

Hofmann, H. F., & Lehner, F. (2001). Requirements engineering as a success factor in software projects. *IEEE Software*, *18*(4), 58–66.

Jazdi, N. (2014). *Cyber Physical Systems in the Context of Industry 4.0*. 3.

Jothi, N., Rashid, N. A., & Husain, W. (2015). Data Mining in Healthcare - A Review. *Procedia Computer Science*, *72*(February 2016), 306–313.

Kadadi, A., Agrawal, R., Nyamful, C., & Atiq, R. (2014). Challenges of data integration and interoperability in big data. *Proceedings - 2014 IEEE International Conference on Big Data, IEEE Big Data 2014*, 38–40.

Kasunic, M. (2001). *Measuring systems interoperability Version 1.0*.

Komninos, N., Bratsas, C., Kakderi, C., & Tsarchopoulos, P. (2016). Smart City Ontologies: Improving the effectiveness of smart city applications. *Journal of Smart Cities*, *1*(1), 31–46.

Kumar, K., & van Hillegersberg, J. (2000). ERP experiences and evolution. *Communications of the ACM*, *43*(4), 22–26.

Lapouchnian, A. (2005). *Goal-Oriented Requirements Engineering : An Overview of the Current Research Goal-Oriented Requirements Engineering : An Overview of the Current Research by Alexei Lapouchnian Department of Computer Science*. *January 2005*, 30.

Lin, J., Fox, M. S., & Bilgic, T. (1996). A requirement ontology for engineering design. *Concurrent Engineering Research and Applications*, *4*(3), 279–291.

Massey, V., & K.J.Satao. (2012). Comparing Various SDLC Models And The New Proposed Model On The Basis Of Available Methodology. *International Journal of Advanced Research in Computer Science and Software Engineering*, *2*(4), 2277–128.

Mathur, S., & Malik, S. (2010). Advancements in the V-Model. *International Journal of Computer Applications*, *1*(12), 30–35.

Mirzaee, S., & Ghaffari, A. (2018). Investigating the impact of information systems on knowledge sharing. *Journal of Knowledge Management*, *22*(3), 501–520.

Mitra, P., Wiederhold, G., & Kersten, M. (2000). A Graph-Oriented Model for Articulation of Ontology Interdependencies. *Proceedings of 7th International Conference on Extending Database Technology*, *1777*(2000–20), 86–100.

Mizoguchi, R., Ikeda, M., & Seta, K. (1995). *Ontology for modeling the world from problem solving perspectives*.

Monostori, L. (2014). Cyber-physical Production Systems: Roots, Expectations and R&D Challenges. *Procedia CIRP*, *17*, 9–13.

Netto, C. M., & Lima, G. Â. (2017). Visualização de ontologias: Estudos e perspectivas. *Informacao e Sociedade*, *27*(3), 59–72.

Noppen, J., Van Den Broek, P., & Aksit, M. (2007). Imperfect requirements in software development. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *4542 LNCS*(June), 247–261.

Nuseibeh, B., & Easterbrook, S. (2000). *Requirements Engineering : A Roadmap*. *1*, 35–46.

OWL working group. (2012). *OWL*. https://www.w3.org/2001/sw/wiki/OWL

Pan, S., Trentesaux, D., McFarlane, D., Montreuil, B., Ballot, E., & Huang, G. Q. (2021). Digital interoperability in logistics and supply chain management: state-of-the-art and research avenues towards Physical Internet. *Computers in Industry*, *128*, 103435.

Pokraev, S. V. (2009). *Model-Driven Semantic Integration of Service-Oriented Applications* (Vol. 53).

Ramalho, R. A. S., & Fujita, M. S. L. (2011). Aplicabilidad De Ontologías En Bibliotecas Digitales. *Anales de Documentación*, *14*, 1–19.

Ray, S. R., & Jones, A. T. (2006). Manufacturing interoperability. *Journal of Intelligent Manufacturing*, *17*(6), 681–688.

RDF working group. (2014). *RDF*.

Reynolds, G., & Stair, R. (2020). *Principles of Information Systems* (14th ed.). Cengage.

Rezaei, R., Chiew, T. K., & Lee, S. P. (2014). An interoperability model for ultra large scale systems. *Advances in Engineering Software*, *67*, 22–46.

Romero, D., & Vernadat, F. (2016). Enterprise information systems state of the art: Past, present and future trends. *Computers in Industry*, *79*, 3–13.

Rook, P. (1986). Controlling software projects. *Software Engineering Journal*, *1*(1), 7.

Rowley, J. (2007). The wisdom hierarchy: Representations of the DIKW hierarchy. *Journal of Information Science*, *33*(2), 163–180.

Ruparelia, N. B. (2010). Software development lifecycle models. *ACM SIGSOFT Software Engineering Notes*, *35*(3), 8.

Russell Ackoff. (1989). From Data to Wisdom. *Journal of Applied Systems Analysis*, *15*, 3–9.

Salgado, C. (2016). *An OMG Model-based Approach for Aligning Information Systems Requirements and Architectures with Business*. University of Minho.

Santos, N., Rodrigues, H., Pereira, J., Morais, F., Martins, R., Ferreira, N., Abreu, R., & Machado,

R. J. (2018). *Specifying Software Services for Fog Computing Architectures using Recursive Model Transformations*.

Santos, N., Teixeira, J., Pereira, A., Ferreira, N., Lima, A., Simoes, R., & Machado, R. J. (2015). A demonstration case on the derivation of process-level logical architectures for ambient assisted living ecosystems. *Ambient Assisted Living*, *June*, 103–139.

Saša Baškarada, & Iskandar, A. (2018). Data, Information, Knowledge, Wisdom (DIKW): A Semiotic Theoretical and Empirical Exploration of the Hierarchy and its Quality Dimension. *Australasian Journal of Information Systems*, *23529*(2), 1–45.

Schmidt, H. W., Krämer, B. J., Poernomo, I., & Reussner, R. (2004). *Radical Innovations of Software and Systems Engineering in the Future*. *2941*(I), 310–324.

STOICA, M., MIRCEA, M., & GHILIC-MICU, B. (2013). Software Development: Agile vs. Traditional. *Informatica Economica*, *17*(4/2013), 64–76.

Sumathi, S., & Sivanandam, S. N. (2006). *Introduction to Data Mining and its Applications* (Vol. 29).

Teichert, R. (2019). Digital transformation maturity: A systematic review of literature. *Acta Universitatis Agriculturae et Silviculturae Mendelianae Brunensis*.

Tiwana, A., & Mclean, E. R. (2005). *Systems Development Expertise Integration and Creativity in*. *May 2015*, 37–41.

Trinkunas, J., & Vasilecas, O. (2007). Building ontologies from relational databases using reverse engineering methods. *Proceedings of the 2007 International Conference on Computer Systems and Technologies*, 13:1–13:6.

Uschold, M., & Grüninger, M. (1996). Ontologies: Principles, methods and applications. *The Knowledge Engineering Review*, *11*.

Uschold, M., & Jasper, R. (1999). A Framework for Understanding and Classifying Ontology Applications. *Methods*, 1–12.

van Heijst, G., Schreiber, A. T., & Wielinga, B. J. (1997). Using explicit ontologies in KBS development. *International Journal of Human-Computer Studies*, *46*(2), 183–292.

Vickery, B. C. (1997). Ontologies. *Journal of Information Science*, *23*, 277–286.

Wegner, P. (1996). Interoperability. *ACM Computing Surveys*, *28*(1), 285–287.

Wood-Harper, A. T., & Baskerville, R. L. (2016). *A critical perspective on action research as a method for information systems research* (Volume 2). Springer.

Xia, W., & Lee, G. (2005). Complexity of information systems development projects: Conceptualization and measurement development. *Journal of Management Information Systems*, *22*(1), 45–83.

Xu, L. Da, He, W., & Li, S. (2014). Internet of things in industries: A survey. *IEEE Transactions on Industrial Informatics*, *10*(4), 2233–2243.

Zins, C. (2007). Conceptual approaches for defining data, information, and knowledge. *JASIST*, *58*, 479–493.

CHAPTER 3

ONTOLOGIES DESIGN AND COMPLEX NETWORKS

Summary: This chapter reflects the existing research on Ontology development, more specifically identifying some of the methodologies currently in use to design and visualize ontologies. First, this chapter introduces some methodologies for developing Ontologies, identifying a few and then the topic of visualization and presentation of complex networks is addressed. Then, as a complement to the previous topic (Visualization and Presentation of Complex Networks), we identify some technologies through which it is possible to implement complex networks using a benchmark of those technologies. This chapter ends with the conclusions of the work previously presented.

CHAPTER TABLE OF CONTENTS

3.1 Introduction....67
3.2 Ontological Development....69
3.3 Complex Networks and Information Visualization....79
3.4 Technologies for Complex Networks Constructing....85
3.5 Conclusions....92
3.6 References....92

—— This page is intentionally left blank ——

CHAPTER 3

ONTOLOGIES DESIGN AND COMPLEX NETWORKS

"The only revolution really worthy of such a name would be the revolution of peace, the one that would transform man trained for war into a man educated for peace, because he would have been educated for peace. That, yes, would be the great mental, and therefore cultural, revolution of Humanity. That would be, finally, the much talked about new man."

– José Saramago

3.1 Introduction

An Ontology is a conceptualization of a domain in a machine-readable format (Guarino, 1995) where graphical visualization helps to search and understand the structure of ontologies. While ontologies are becoming increasingly popular modeling schemes for knowledge management services and applications, the focus on developing tools to graphically visualize ontologies is increasing in order to facilitate their evaluation and analysis. As one can see in the next chapter, there are numerous technologies for developing ontologies, which provide an intuitive ontology editor and have extensions for ontology visualization, project management, software engineering and other modeling tasks.

An ontology, according to the definition in (Sivakumar & Arivoli, 2011), is an explicit formal description of a domain, consisting of classes, which are the concepts found in the domain. These classes are organized in a specialization/generalization hierarchy through is-a (or inheritance) links, where each class can have zero, one, or multiple parent classes. Each class has properties that describe various features of the modeled class. These properties are typed, and the types allowed

are either simple types (strings, numbers, booleans, or enumerations) or instances of other classes (references); constraints can also be set on the value ranges of slots (e.g., integers from 1 to 10). Finally, instantiation can be applied to classes to produce items corresponding to individual objects in the domain of discourse (instances), where each instance has a concrete value for each property of the class to which it belongs. Furthermore, classes, together with instances, are said to constitute the knowledge base.

From the above definition, it is clear that the task of visualizing the complete set of features of the ontology is not easy. The properties of ontology are summarized as follows (Sivakumar & Arivoli, 2011):

- Hierarchy. A type of organization that, like a tree, branches into more specific units, each of which is "owned" by the higher-level unit immediately above.
- Properties representation. More than a hierarchy, as it concepts are described by using restrictions on properties.
- Level of detail. Possibility to choose till which level an ontology to be provided.
- History. The concepts that are chosen in the previous steps.
- Filtering. Ontologies could contain hundreds of properties. The user can be interested in only the subset of the ontology, based on the central concept and the properties of the user's choice.
- Multiple geometrical views. The representation of the graph in different geometrical models to better understand the structure of ontology.
- Zoom semantic/geometric. To see more or less details during ontology exploration. With the geometric zoom the visualized object is scaled when the user zooms in/out. The semantic zoom provides the possibility to see more/less details of the object by zooming in/out.

There are several ontology visualizations that have been incorporated into ontology management tools and are used as information retrieval aids in applications that use ontologies (Katifori *et al.*, 2007). Various visualization techniques have been described over the years, such as covering tree layouts, tree maps (Johnson & Shneiderman, 1991), fisheye views, hyperbolic, and 3D hyperbolic layouts, aiming to help understand and analyze complex information structures. The preferences of visualization models vary according to the users' needs and the context of the query (Graham *et al.*, 2000). It also depends on the type and extent of the visualized network. Using a combination

of integrated visualizations of various types has sometimes been shown to be beneficial (RISDEN *et al.*, 2000).

On the other hand, complex networks of multi-dimensional hierarchies and arbitrary relationships are becoming common features of current ontologies. Tools that discriminate some of these features, for example by supporting only spanning trees or hierarchical relationships, may not be appropriate for comprehensive ontology visualization. Ontologies, together with their Knowledge Bases (KBs), could grow to very large information networks, especially if intended to provide scalable services for the Semantic Web. Visualizing large networks has always been a challenge. Studies (Herman *et al.*, 2000) surveyed a wide range of visualization techniques and concluded that all existing algorithms have a size limit beyond which they cannot cope.

3.2 Ontological Development

Semantic web, as a machine-readable web, needs ontologies as its primary and most important component, where they describe conceptions and their associations in the domain of discourse (Gruber, 1993). Ontologies are formal models and machine intelligible descriptions of a domain that are required for knowledge-based applications, which aid in the transfer of domain knowledge to other domains, whether relevant or irrelevant. Because organizational knowledge is scattered, knowledge-based applications must be able to combine knowledge from disparate sources and offer an overview of the knowledge available in the organization using ontologies (Ahmad *et al.*, 2011). In this context, finding a suitable ontology for a domain is one of the bigger research challenges (Tudorache, 2020).

Organizations have used ontologies as a conceptual tool and fundamental component of knowledge-based systems for effective knowledge management in the area of discourse. The industry has showed an interest in developing fresh applications in semantic technology, resulting in widespread acceptance of ontology-based solutions by government, academia, and commercial industry during the last decade. In this context, ontology-based solutions with improved knowledge management help in better decision making. Furthermore, ontology approach makes it easy to share conceptualization of a domain (Gruber, 1993), and this sharing offers more opportunities for stakeholders to solve their real-time problems.

Due to the distributed nature of organizational knowledge, the knowledge-based applications with the help of ontologies, must be able to integrate knowledge of heterogeneous sources and present an overview of the knowledge available in the organization (Ahmad *et al.*, 2011). In this context, finding a suitable ontology for a domain is one of the bigger research challenges (Tudorache, 2020). Ontologies, as a conceptual tool and key component of knowledge-based systems, have been used by organizations for effective knowledge management of the domain of discourse.

The study of principles, methods and tools for designing upper or domain ontologies, is a primary focus of the ontology engineering discipline. In this context, a methodology provides guidelines for the development of ontologies. In order to help and support ontology development, several methodologies have been proposed by researchers (Sattar *et al.*, 2020). Ontologies help in communication for better decision making, promote sharing of knowledge, facilitate storage of information, and support the reuse of knowledge (Ahmad *et al.*, 2011). A methodology is a set of well-designed techniques and methods that assure the quality of the results of an ontology design process. Studies (Silva-López *et al.*, 2014) describe a series of related concepts associated to methodologies for designing ontology.

- Method: The order or a series of steps to develop a product.
- Technique: A procedure for achieving a goal. Therefore, the methodology provides a framework for building ontology for the domain of knowledge.
- Methodology: A set of methods and techniques that assure the quality of the results of an ontology design process.

Ontology is exploited by researchers to describe common vocabulary in any domain for the exchange and reuse of knowledge (Gokhale *et al.*, 2011). In addition, a thorough data model called an ontology for standardizing terminologies is needed to enable an inference-based design for real-life scenarios (Agyapong-Kodua *et al.*, 2013). Understanding the justification for ontology development is also crucial. They are as follows: to enable reuse of domain knowledge, to make domain assumptions clear, to distinguish domain knowledge from operational knowledge, to analyze domain knowledge, and to communicate shared understanding of the structure of information across humans or software agents (Noy, 2001). Ontology creation, which is an iterative engineering process, is really time-consuming and labor-intensive and, as is to be expected, calls for a methodology similar to that of software development.

Ontology development, however, lacks a standardized process (Kapoor & Savita, 2010). Similarly, (Noy, 2001) made it obvious that there is no one ideal process for ontology construction and that the ontology developer must make this decision. Nevertheless, pre-development, development, and post-development processes must be properly taken into account in order to create ontologies that adhere to the idea of reusability and semantic stability (Rajpathak & Chougule, 2011). Based on the huge numbers of literature suitably analyzed, the prominent approaches addressed by developer include: Noy and McGuiness methodology, Gruninger and Fox's methodology, Methontology among others, AMOD and UPON inclusive.

Thus, based on a literature review, methodologies for ontology development are presented. These methodologies indicate a set of guiding principles about the activities identified in the ontology design process and how they should be carried out. Therefore, it is essential to construct ontologies based on both novel and old approaches (Nanda *et al.*, 2006). However, due to the shortcomings in accordance with the standard requirements of software development guilding principles, ontology developers are still faced with the difficulties of generic standard ways of developing the knowledge representation (Vigo *et al.*, 2014).

Some of the approaches have a lengthy history of use in the creation of ontologies. To this purpose, (Jones *et al.*, 1998) studied several ontology development approaches in their survey study in the late 20th century. They include the TOVE (Toronto Virtual Enterprise) ontology engineering process, which comprises of six techniques; the Enterprise Model Approach, which has four stages; the Methontology engineering process, which has seven phases; and the KBSI IDEF5 engineering process, which has five approaches. SENSUS, MENELAS, ONIONS, Ontolingua, and others are among the others. The most popular ontology building approaches are also given in (Lopez *et al.*, 1999). The methodologies discussed include Methontology, Gruninger and Fox, Uschold and King, SENSUS, AMOD (Agile Methodology for Ontology Development), and UPON Lite.

The majority of ontology development approaches may be characterized as an iterative ontology engineering process in that changes can be made at any point in the process, regardless of where the ontology is in its development. There are three alternative methods for locating ontology concepts in every given domain throughout the ontology building process. Top-down, middle-out, and bottom-up tactics are these. Using a top-down method, the most abstract notions are initially determined before being specialized into more precise concepts. Beginning with the most particular notions, a bottom-up strategy then expanded into more abstract ones. The middle-out strategy, on

the other hand, initially identified the most crucial concepts before generalizing and focusing on additional concepts (Corcho *et al.*, 2003). Taking these aspects into account, we now present the characteristics of some methodologies, mentioned above, for the development of ontologies.

Methontology Methodology

The Methontology was created in the Artificial Intelligence Laboratory in Madrid to design new ontologies or reuse ones that already exist by accessing ontology servers. It was created as an ontology management tool using webODE and ontoEdit (Agyapong-Kodua *et al.*, 2013). The strategy was initially a result of the understanding gained by creating a chemical ontology. The approach's step-by-step activities are as follows: specification, where the goal of the ontology must be determined; knowledge acquisition, where knowledge must be extracted from various sources; conceptualization, where domain terms are identified as concepts or relations; integration, where super concepts are related to one another either through classes or properties; implementation, where an ontology representation language is necessary; evaluation; and documentation (Delir Haghighi *et al.*, 2013).

Park *et al* (Park *et al.*, 2008) establishes the process for creating a Graduation Screen Ontology (GSO), and OWL-DL and Protégé put it into practice. Islam *et al* (Islam *et al.*, 2010) validated the methodology and stated that its activities complement and adhere to the software development life cycle and provide an easy-to-understand, methodical approach to ontology building. However, the study found that Methontology was a standout methodology after analyzing and comparing a significant number of techniques based on factors including kind of development, reusability support, and ways for discovering ideas. It was recognized as both a popular and classic strategy by (Gavrilova & Leshcheva, 2015).

The study of Iqbal *et al* (Iqbal et al., 2013), which characterized the activities of the technique as being in complete conformance with IEEE standard for software development process, further supports Methontology's remarkable reputation. Uschold and King, Gruninger and Fox (who suggested a different technique termed TOVE), Methontology, and UPON by De Nicola *et al* (De Nicola *et al.*, 2009) are a few of the methodologies that have been examined and contrasted. The researchers' suggested semantic conflict detection ontology, which exists between messages from web services, was finally built using Methontology. Because of the way it was developed, which the

authors believed to be in accordance with IEEE standard 1074-1999, Methontology was chosen. Similar to this, Águila *et al* (Águila *et al.*, 2014) asserted that Methontology is the finest method for describing the development process among all others. However, there are several significant flaws in the technique. For instance, Methontology lacks the activity of pre-developmental appraisal, but Gruninger and Fox do.

Gruninger & Fox's Methodology

It is a first-order logic-based approach to knowledge-based systems presented by (Gruninger & Fox, 1995). It is a formal approach that taps the power of conventional logic. It acts as a stage in converting informal settings into formal languages in that regard. Its tasks include identifying motivating scenarios and formalizing informal ones, defining the ontology's terminology in a formal language, laying out the competency questions that will help determine the ontology's scope, defining the axioms and definitions for the terms in the ontology, and establishing the criteria for judging the ontology's degree of completion.

The approach was first intended to generate ontologies within the realm of corporate enterprises. However, the approach is now being used in a wide range of academic disciplines. This methodology's assessment process, or collection of Competency Questions (CQs), which helps define the scope of the ontology and validate and ensure that the design is accurate, is a critical component. At this point, formal modeling approaches are important. Due to this strength (that is, CQs), the technique very frequently forms part of combined methods approach (Iqbal *et al.*, 2013). More significantly, reusability is one of the fundamental tenets of ontology. It was noted that none of its operations adhere to the guiding philosophy. The FAO-based technique, on the other hand, took the reusability principle—also known as ontological evolution—into account when considering post-development activities.

Noy and McGuiness Methodology

Noy and McGuiness' iterative methodology is built on a few basic principles. The approach described seven phases for creating ontology. Enumerate key terms in the ontology, define the classes and the class hierarchy, describe the characteristics of the classes, define the value of the slots, and construct instances, to name a few (Noy, 2001). For the purpose of identifying ideas, a top-down method was adopted. Using a series of competence questions, the ontology was created and tested using the protégé tool query and Export Tab plug-in. In order to create their suggested ontology for software maintenance, (Serna M. & Serna A., 2014) similarly followed the technique. In other words, the domain of antimicrobial-microorganisms is where the ontology is formed. Similar to this, the first four steps of the Noy and McGuiness technique are included into the suggested ISI method in the work of Chen *et al* (Chen-Huei Chou, F. Zahedi, 2011) to create the ontology framework for the natural disaster management domain.

Agile Methodology for Ontology Development (AMOD)

The proposed methodology aims to include software engineering's agile ideas and practices into the creation of ontologies. Pre-game, development, and post-game are the three phases that AMOD divides the ontology development process. Additionally, it lists a few auxiliary tasks that take place concurrently with other activities. Ontology owner, ontology engineer, and ontology user are the main responsibilities taken into account by AMOD. Customer demands must be communicated to the ontology engineers via the ontology owner. The ontology will be put into use by an ontology engineer. The ontology is being used with a specific goal in mind by the ontology user.

The pre-game stage is the beginning of the ontology development process. The objective and scope of the ontology, tools and methodologies, competence questions, and available sources are all identified during this phase. The details of each of these initiatives are provided below.

- Ontology Goal and Scope: The first stage in creating ontologies is defining the aim and scope of the ontology. This activity explains the purpose for developing the ontology, its planned applications, and the ontology's users (Abdelaziz *et al.*, 2017). What belongs in the ontology and what doesn't depend on the scope. It restricts the variety of notions that may be examined (Brusa *et al.*, 2006).

- Tools and Techniques: It is important to identify knowledge capture strategies. It is important to choose the language and resources that will be adopted to create the ontology. Users may visually edit, explore, inspect, and code ontologies using ontology construction tools. The ontology construction tools Apollo, Protégé 3.4, IsaViz, and SWOOP are a few examples. Knowledge Interchange Format (KIF), Web Ontology Language (OWL), Resource Description Framework (RDF), DARPA Agent Markup Language and Ontology Inference Layer (DAML+OIL), and others are the most widely used languages for encoding ontologies.
- Ontology Requirements: The needs for the ontology should be acquired after determining the objective and scope of the ontology. A collection of competence questions can be used to express the needs acquired (CQs). The questions to which the ontology must be able to provide solutions are known as CQs. The collection of CQs is kept in a product backlog that is prioritized based on risk and business value. The CQs also offer a means of assessing the ontology.
- Source Selection: The objective of this exercise is to choose sources that may be used to extract domain knowledge. Domain specialists are the primary source for knowledge acquisition. International standards, monographs, technical studies, glossaries, categorization schemes, and reference models are some more sources.

The development phase incorporates multiple and iterative cycles that are called sprints. Sprints are typically 1-4 weeks in length. Each sprint includes the following activities:

- Sprint Planning: The ontology owner and ontology engineers decide which high-priority product backlog items will be implemented during the sprint during sprint planning. The ontology engineers then choose how to put these objects into practice (Schwaber & Sutherland, 2020).
- Knowledge Acquisition: In this stage, knowledge acquisition techniques are used to collect all the pertinent terminology connected to the area of interest (concepts, attributes, relations, etc.). Interviewing, brainstorming, protocol analysis, and the Delphi method are a few of these methods.
- Conceptualization: This activity's objective is to arrange the acquired information into a semi-formal specification based on a collection of intermediary representations (IRs). A glossary of terminology, a concept dictionary, concept classification trees, binary

connection diagrams, and other tools are included in IRs. The ontology conceptual model is the primary product of this effort (Lopez *et al.*, 1999).

- Formalization: The formalization activity codifies the conceptual model using the selected language and technology, converting it into a formal model.
- Integration: The ontologies created over the previous sprints must be merged with the one that was implemented during the sprint. Integration operations and integration-oriented design standards are required for this (Sofia Pinto & Martins, 2001).
- Sprint Review: The sprint's conclusion is when this meeting is held. The ontology engineer and ontology owner assess the sprint's work during this meeting].

The purpose of the final phase is to prepare for a final ontology. It includes the following activities:

- Evaluation: Ontological evaluation is divided into two perspectives, namely verification and validation.
    - While ontology validation assure that the right ontology is being generated, ontology verification verifies that the ontology is being built correctly (Hlomani Hlomani & Deborah Stacey, 2014).
- Maintenance. To reflect the changes in the area of interest that it describes, the resultant ontology must be revised and updated. The resultant ontology may be extended with new concepts or relations to increase its dependability (Looser *et al.*, 2013).

UPON Lite

The suggested Unified Process for ONtology Building (UPON) was proposed by Nicola *et al.* (De Nicola *et al.*, 2009). The widely used software engineering approach called Unified Process is the foundation of UPON (UP). Additionally, it incorporates the benefit of the Unified Modelling Language (UML). The five major processes in UPON are requirements, analysis, design, implementation, and test. The fact that UPON does not take the creation of generic ontologies into account is one of its shortcomings. Additionally, UPON ignores the component of collaborative ontology creation.

Nicola and Missikoff presented the Lightweight Methodology for Rapid Ontology Engineering (UPON Lite) (De Nicola & Missikoff, 2016). It is a derivation of the Unified Process for ONtology Building

(UPON). The UPON Lite methodology (De Nicola & Missikoff, 2016) was created as a simplified way for ontology engineering, which aids in the comprehension and communication of business environment among domain specialists. The series of tasks that must be carried out throughout the creation of an ontology are described by ontology engineering (Gomez-Perez *et al.*, 2004). An explicit statement of a shared conception is how an ontology is described in this context (Guarino *et al.*, 2009). Top-level (or foundational), domain, task, and application (i.e. relying on a task in a specific domain) ontologies can be distinguished based on the degree of generality. UPON Lite specifically focuses at simplicity of usage and a decreased reliance on ontology engineers, in contrast to more rigorous and methodical techniques (De Nicola & Missikoff, 2016).

It is especially helpful for the building of domain, task, and application ontologies since Upon Lite focuses on the pooled input of domain experts. Six interconnected phases make up the UPON Lite development process: I domain terminology; (ii) domain glossary; (iii) taxonomy; (iv) predication; (v) parthood; and (vi) ontology (De Nicola & Missikoff, 2016). De Nicola and Missikoff explain each step's intended goal as well as how it may be accomplished and what obstacles need to be overcome.

Taking into consideration the above and presented methodologies, Bedini adds an automatic ontology generation process in case concrete data sources exist. Thus, according to Bedini, Figure 13, there are 5 stages that represent the main tasks of the ontology design process based on existing data sources. Initially, it is necessary to extract the data, which is then the first stage of extraction. Here, is where the necessary information is gathered to generate the ontology (concepts, attributes, relations, and axioms) from an existing source. The input resources can be structured, semi-structured or unstructured, and the techniques for extracting information can be of different types: Natural Language Process (NLP) techniques, clustering, machine learning, semantics, morphology or lexicon, the most common being the use of a combination of these.

In the second step, Analysis, the matching of information from two or more existing ontologies is performed. This step requires techniques already used in extraction, such as morphological and lexical analysis of labels, a semantic analysis to detect synonyms, homonyms and other such relationships, an analysis of the structure of concepts in order to obtain hierarchical relationships and identify common attributes. The Generation stage, is related to the merging of the ontology, if applicable, and the formalization of the metamodel used by the tool and that will be interpretable by other applications, such as OWL and RDF/S.

In the fourth step, Validation, the concepts and relationships introduced in the previous steps are verified. This step is usually done manually, however, it can be automated. It should also be noted that at the end of each of the previous steps, a validation can be performed. Finally, as an ontology is not a static description of a domain, it can undergo changes or evolution of applications (in quantity and number), the number of concepts and the relationships between the different properties can be added and/or modified. Thus, the Evolution stage, is considered as an addition of new requirements and can therefore be followed by a new extraction of information. However, at this stage, the ability of the tools to solve this problem is also evaluated (Bedini & Nguyen, 2007).

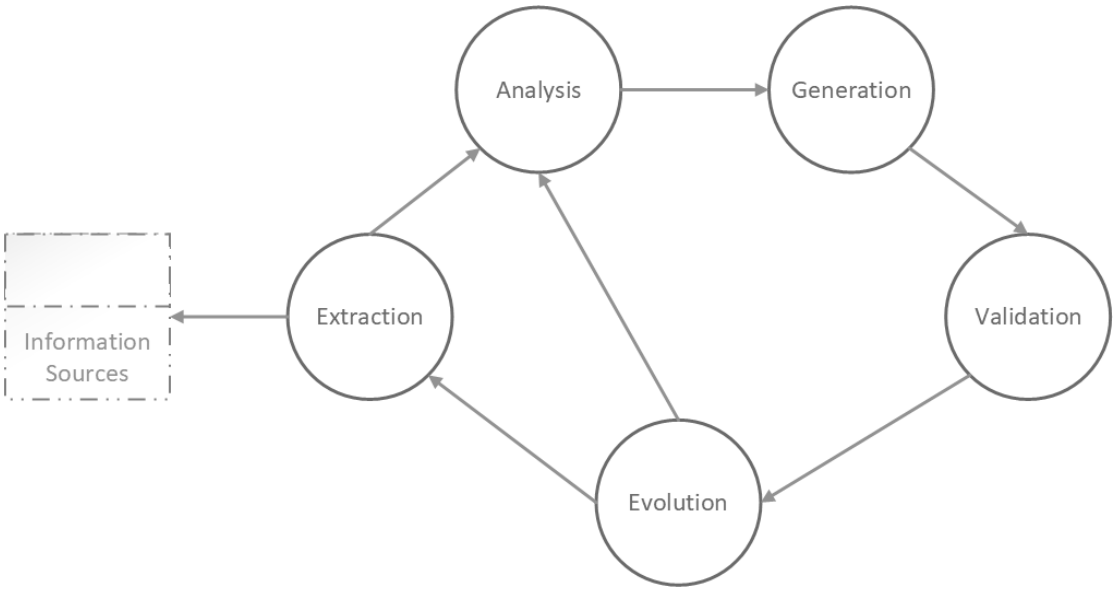


Figure 13 - Automatic ontology generation process (Bedini & Nguyen, 2007)

Ontology representation languages and editors (ontology management tools), in addition to ontology development methodologies, are also a component of the process (Munir & Sheraz Anjum, 2018). Ontology development approaches, however, are the focus of this research since they form the basis of the ontology creation process. The amount of expressiveness for the proposed ontology determines the choice of ontology management tools to apply the selected technique. OWL (both version 1 and version 2), RDF, RDF Schema, DARPA Agent Markup Language and Ontology Interchange Language (DAML+OIL), and the others are ontology representation languages. These languages are classified as World Wide Web Consortium (W3C) standards in this research, while XML Topic Map (XTM) (Femi Aminu & Sunday Adewale, 2015), another ontology language, was classified as an International Organization for Standardization (ISO) standard. OWL is apparently the most widely used W3C standard language due to its expressiveness. Protégé, FAO AGROVOC

Concept Server Workbench Tool, OBO-Edit, SWOOP, Apollo, IsaViz, TopBraidComposer, and citrus ontology created from scratch using Graph Databases technologies (Liao *et al.*, 2013) are also included in the list of ontology editing tools.

3.3 Complex Networks and Information Visualization

Ontologies typically comprise hundreds, if not thousands, of classes and instances, depending on the complexity of the domains to be represented. It's possible that this structure will become difficult to picture and comprehend. Ontology visualisation has been addressed in the scientific literature in the domains of Computing and Information Science with the goal of facilitating the usage of ontologies and the human comprehension of the domain represented by them. (Netto & Lima, 2017).The transformation of abstract data and the complex systems into visuals or images, referred known as visualisations, is part of the information visualisation process. As a result, the goal of visualisation design is to test the human perception capacity so that the user can analyse and grasp the information given while also deducing new knowledge from the spatial relationships represented.

Our present knowledge of the environment, whether geographical or biological, demonstrates that nature is made up of intricate webs of interconnected systems. These systems' networks enable behaviours that, when studied using conventional techniques, are far from being predictable. The network is influenced by each element, and each element is influenced by the network. A new discipline of study called Network Science (NS), or Complex Networks Analysis (CNA), has evolved to better comprehend complex linked systems (Becheru & Badica, 2014). The core of this new branch of study draws from computer science and graph theory. NS looks into non-trivial aspects of graph issues that are typically not covered by random graphs or lattice theory. Since these non-trivial properties regularly appear in issues in the real world, understanding them is of great relevance. The modeling and assessment of overlapping and interrelated events that are neither entirely regular nor purely random is what gives real-world networks their complexity. Additionally, complexity may result from the network's overall size (Becheru & Badica, 2014).

Information systems (ISs) have been progressively more complex in recent years due to the fast growth of manufacturing, information, and management technology. This complexity is preventing many important ISs theories and technologies from being used more widely. Thankfully, significant

progress has been achieved in statistical physics theories during the past 20 years, along with several complicated network applications. It has offered a different method for analyzing ISs. The idea of complex networks has received a lot of attention recently as a means of describing and resolving complex issues (Yongfeng Li, Fei Tao, Ying Cheng, Xianzhi Zhang, 2017).

There is no one definition that applies to all complex systems since there are so many different types of them. It is described as "any system composed of several heterogeneous components, among which local interactions form various layers of collective structure and organization" by the national network of complex systems. As a result, we may define a complex system as one that has some of the qualities listed below (Bihanic, 2015):

- Heterogeneity; a complex system is made up of several agents or entities. These entities often fall under several kinds and possess a certain internal structure;
- Flow processing, where many system components interact or are related to one another;
- The scale of a system and the accompanying processing may be an element of complexity, however this is not a basic property;
- Hierarchical organization, complex systems exhibit hierarchies that create networks made up of connected entities or agents that communicate with one another.

The characteristics and behavior of a complex system's separate components are insufficient to anticipate the system's overall behavior. Any complex system is characterized by its features and environmental qualities, whether it is an ecosystem of living things, the administration of an urban community, or a network of people (in other words, through its relationships between data and its interactions between elements). Data is important nowadays, and "connecting the dots" is key. Big data has virtually transformed from zero to hero in the corporate computing sector over the last several years. With one exception: in actuality, it hasn't. Many people appear to have forgotten that big data exists and is being used effectively. Enterprise data volumes have undoubtedly increased significantly, and businesses have started to realize the potential of these bigger repositories.

An Aberdeen Group study from 2011 found that organizations that successfully integrate complex data can use up to 50% larger data sets for business intelligence and analytics, integrate external unstructured data into business processes twice as successfully, and almost halve the amount of errors that occur. It is obvious that a company's performance is directly related to how well it can use big data (Bihanic & Polacsek, 2012a).

There is still plenty to do. Information Systems (IS) and its underlying concepts are now too complicated to be comprehensible to humans (Bodik *et al.*, 2010). The concepts of perspective and viewpoint have been extensively studied in the fields of software engineering, knowledge representation, and information systems development. A view is defined as "a representation of a whole system from the perspective of a related set of concerns" in accordance with IEEE Standard 1471, and a viewpoint is defined as "a pattern or template from which to develop individual views by establishing the purposes and audience for a view and the techniques for its creation and analysis."

The concepts of perspective and viewpoint are not new, as one will note, and they refer to how an item is regarded from the standpoint of the observer. The idea of perspective may aid in managing and using the complex systems as the IS becomes more complicated (Ahmad Ghazal *et al.*, 2013). In order to meet the demands of graph data management and online graph analysis, a number of graph database systems have recently emerged (L. Y. Ho *et al.*, 2013). Some well-known graph database systems are Neo4j, DEX, OrientDB, Trinity, Titan, etc.

Over the course of more than three decades, several information science laboratories that focus on human-computer interaction (HCI) launched research projects to discover novel information models and representations of semantic knowledge as well as ground-breaking visual data description solutions. As a result, several studies proposed original methods, procedures, and techniques for the entire-data visualization of scientific, mathematical, and analytical statistics.

Data visualization was seen at the time as a unique tool (reserved for scientific use only), a way to deal with collections of raw data, and a way to recognize phenomena (because data are evidence of phenomena). A multidisciplinary research initiative first emerged ten years ago. Its goal was to establish new information representations, theories, and paradigms that would bring together various experts in computer science, information engineering, cognitive science, experimental cognitive psychology, etc.(Bihanic & Polacsek, 2012b).

The design, creation, and use of interactive, computer-generated graphics for information representation is the subject of information visualization (Chen, 2004). As a result, it is frequently inferred that information visualization mostly deals with non-spatial, abstract data. It is crucial for the discipline to convert such nonspatial data into understandable and compelling graphical representations. Designers add new meanings to graphical patterns as part of the transformation

process, which is also a creative activity. Information visualization, like art, attempts to inspire its users to make new connections and to express difficult concepts to its audience (Chen, 2010).

According to Card *et al.*, (S. K. Card *et al.*, 1999), "the visualization of information (also known as InfoVis) is the use of visual, interactive, and informed representations of abstract data to enhance cognition." The primary benefit of an InfoVis visa is the provision of cognitive processing advantages. The procedure was given priority over the final product. InfoVis was viewed as more than just a tool since it revealed the meaning of the data as well as its semantic similarities and relationships through formal representation and mapping. In other words, this revolution in information visualization aims to translate semantic networks derived from several data sources into perceptible forms in order to arrive at a fresh and useful understanding of the data (Kim *et al.*, 2016).

Information visualization "amplifies cognition" through user perception by presenting the semantic relationships between the facts. It is based on methods for connecting the model with its graphical or visual representation (Chen, 2010). The mode of reasoning and treatment provided to the user depend on these representation techniques, through potential views and viewpoints about the data, as demonstrated in the excellent paper by J. Herer and Shneiderman (Heer & Shneiderman, 2012) (offering a new taxonomy of tools that support the fluent and flexible use of visualisations today). Each method demonstrates the intricacy of the data's semantic and structural components in a certain way. Each one establishes the guidelines for the organization and visualization of data, employs multilevel exploration procedures combining various visualization techniques, and/or reflects a connection between the semiotic realities, the conditions of user appropriation, and the organization of the data.

The design, creation, and use of interactive, computer-generated graphics for information representation is the subject of information visualization (Hou *et al.*, 2018). This frequently means that information visualization focuses mostly on non-spatial, abstract data. It is crucial for the discipline to convert such nonspatial data into understandable and compelling graphical representations. Designers change graphical patterns in a creative way by giving them new interpretations. Information visualization, like art, attempts to inspire its users to make new connections and to express difficult concepts to its audience. Similar to science, information visualization demands that data and related patterns be presented with rigor, accuracy, and faithfulness (Chen, 2020). The connection between scientific visualization and information

visualization is a frequently asked subject. They are distinctive in terms of the corresponding research communities, to put it simply (Ji & Gan, 2020). They overlap, yet they also greatly diverge.

The presence or lack of data in quantitative forms and the ease with which one may convert data to quantitative forms serve as a fundamental point of differentiation between information visualization, data visualization, and scientific visualization (Chen, 2010). This is why experts stress how nonvisual data might be represented in information visualization. Information visualization can be distinguished from its closest neighbors, such as quantitative data visualization. A taxonomy of information visualization, which defines the process of information visualization in terms of data transformation, visualization transformation, and visual mapping transformation, has a more formal description of this stage. Raw data is transformed into mathematical representations (Hou *et al.*, 2018).

A visual-spatial model of the data is created through visualization transformation. The visual-spatial model's presentation to the user is determined by visual mapping transformation (Kim *et al.*, 2016). However, researchers and designers will be better equipped to take advantage of this important relationship if the data is quantitative in type. In terms of functional information visualization and aesthetic information visualization, the relationship between scientific and artistic components of information visualization is examined. While the objective of aesthetic information visualization is to offer a subjective impression of a data set by evoking a visceral or emotional response from the user, the primary aim of functional information visualization is to convey a message to the user (Chen, 2020).

The visual representation of information's semantics, or meanings, is known as information visualization. The design, creation, and use of interactive, computer-generated graphics for information representation fall under the umbrella of information visualization. However, information visualization considers designing the graphical components and their relationships as well as acquiring and processing the information that will be shown. Information may be displayed using a variety of visualization approaches, including line graphs, bar graphs, spark lines, bullet graphs, diagrams, and metaphors. According to the literature, the information visualization process includes gathering, transforming, and presenting (qualitative and quantitative) data in a visual format that encourages exploration and comprehension through the use of interaction and distortion methods (Chen, 2010).

A reference model for information visualization was introduced by Card *et al.* (S. K. Card *et al.*, 1999)and offers a high-level overview of the visualization procedure. The three essential phases of this process are user interaction, data processing, and visual transformation. Chi (Chi, 2000) has provided a technique for information visualization that follows the same stages as the one described. Data transformation, visualization transformation, and visual mapping transformation are used to represent the visualization process. Wünsche (Wünsche, 2004) added the dimension "visual perception" in addition to the conventional procedure that Card *et al.*(S. K. Card *et al.*, 1999) had previously proposed. Visual perception is the study of how much information a person can effectively see before reaching their cognitive or perceptual limits. Based on these studies, Figure 14 outlines the many phases involved in information visualization (Al-Kassab *et al.*, 2014).

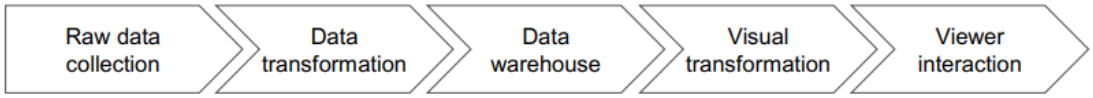


Figure 14 - Information Visualization Process adapted from (Al-Kassab *et al.*, 2014)

According to Al-Kasab *et al.* (Al-Kassab *et al.*, 2014), the set of quantitative and/or qualitative data that will be depicted is first represented as raw data. The information may be gathered from a variety of sources, but it is frequently kept in a data warehouse. A series of transformation operations must be carried out in order to visually convey this data to the end user.

The first stage involves data transformation and includes data normalization, calculation of derived data, and formatting of raw data. A set of altered data with a uniform structure is the consequence of these activities. The second stage enables the visual transformation by mapping the changed data onto a matching visual structure once it is given to the observer in a "pristine" state. Applying stacked or dense pixel displays, for example, end users may now browse across the display approaches, such as graphs, tables, or maps, using a set of views that can be derived from this visual structure.

Using interactive methods like interactive zooming, interactive connecting, or interactive brushing, the spectator may finally engage with the visual representation. At many phases of the transformation process, user activities might influence it. Users may alter the visual representation

of the data, how they perceive it, and even how it is transformed. Information visualization aids in both understanding and creating the world around us (Al-Kassab *et al.*, 2014).

3.4 Technologies for Complex Networks Constructing

According to (Shneiderman, 1996), the use of graphical data presentation maximizes the use of human vision's ability to perceive patterns, groupings, gaps, and trends in data, as well as allowing users to identify hidden qualities in enormous amounts of data. The author claims that information visualization makes it easier to explore data using a visual language. The visualization of information applied to ontology consists in graphically representing the elements that constitute ontologies: classes, relations, axioms and instances, with possibilities of interaction tasks over these elements (Netto & Lima, 2017). According to (S. Card, 2002), visualizations can increase cognitive memory and processing resources, reduce the search for information, improve pattern detection, enable perceptual inference operations as well as attention mechanisms for monitoring, encode information in a manipulable medium, allow for the examination of a large amount of data, keep an overview of the whole while pursuing details, keep track of things, and produce an abbreviated version of the whole.

In accordance with the authors (Dudás *et al.*, 2018) larger ontologies frequently do not fit on a single screen, necessitating the use of advanced visualization and filtering techniques to minimize or at least reduce clutter. A related problem is tied to the well-known visualization idea of "overview first, zoom and filter later." Another difficult problem is getting to the specifics from the overview while maintaining the context of other portions of the ontology. The third issue arises from the fact that various use cases necessitate different visualization and interaction strategies. Finally, another significant difficulty in ontology visualization is the use of cognitive and perceptual aspects capable of adding additional expressive dimensions to the display.

Graphs are widely used to represent data, as graphs are one of the fundamental data abstractions in computer science and one of the most useful structures for modelling objects and interactions (Vicknair *et al.*, 2010). Specifically, graph databases overcome the limitations imposed by traditional db-models with respect to capturing the inherent graph structure of data appearing in applications such as hypertext or geographic information systems, where the interconnectivity of data is an important aspect, with several query languages for graphs being proposed in the last

decades (Angles, 2012). In its simplest form, a graph G is a pair (V, E), where V is a finite set of nodes and E is a finite set of edges connecting pairs of nodes. Naturally, the edges can be directed or not, although we consider only the directed cases, which are the more general (Wood, 2012).

When semi-structured data and object databases became prominent in the 1990's, they provided fruitful areas for the study of graph models and query languages. In the last decade, the web semantic and also social networks have assumed themselves as possible key areas of graph-based approaches (Wood, 2012). Other areas of applications include transports networks, semantic associations as part of criminal investigations (also called linkage analyses), biological networks, program analyses, workflow and data source (Anand Bowers, S., Ludascher, B., 2010). Each of the above application areas brings its own requirements in terms of an appropriate graph model. On the other hand, the relational database model has existed since the late 1960's. It has consistently proven to provide persistence, concurrency control and integration mechanisms. Relational database management systems (RDMS) maintain tables that are defined by sets of rows and columns, where a line can be perceived as an object while the columns would be attributes/properties of these objects (Rodriguez & Neubauer, 2010b). One of the weakness of the relational model is its limited ability of explicitly capturing the semantic of requirements.

The presence of enormous graph data sets in several disciplines has made scalable big graph data mining crucial in recent years. As of December 31, 2013, the social network Facebook had 1.23 billion active users1 (i.e., vertices), whereas Twitter had 241 million monthly active users and the system as a whole transmitted more than 500 million tweets each day. De Bruijn graphs of millions (or perhaps billions) of vertices are produced by modern sequencers, which provide big graph data sets (Compeau *et al.*, 2011) for which researchers are currently looking for effective algorithms to detect Hamiltonian cycles in such enormous graph data sets.

Due to the growing number of applications that call for the storing and processing of enormous amounts of data in the form of graphs, graph data storage has received a lot of attention recently (L.-Y. Ho *et al.*, 2012). A no-sql storage approach known as the "graph database model" has shown to be more effective in applications that store and handle graph data (Vukotic *et al.*, 2015). The use of networked data model-based Graph Database Management Systems is growing in a variety of application domains, including social networks, biology, cyber security, automobile traffic modeling, etc. The difficulty of low latency online query processing on graph data sets is something that graph databases are designed to address (Shao *et al.*, 2012).

As big data and machine learning problems, involving complex interconnected information, have become increasingly common in the sciences realm, storing, retrieving and manipulating such complex data becomes costly when traditional RDMS approaches are used, as these schema-based data models, by their own definition, impose limits on how information will be stored. This implies a manual process involved to redesign the schema to adapt to new data, where RDMS are optimized for aggregate data, graph databases are optimized for highly connected data.

A graph is a data structure composed by edges and vertices (Rodriguez & Neubauer, 2010a), where a graph database technology stands as an effective tool for a project of a data model. Modelling objects and relationships between them mean that almost everything can be represented in a corresponding graph. A type of common graph supported by most of systems is a graph of properties, which can be assigned, labelled and targeted to multi-graphs systems (Miller, 2013).

As for the origins of graph databases, Angles and Gutierrez developed a research on models of databases by graphs proposed before the year of 2002. The authors later synthetized the notion of "graph databases model" and compared the available proposals until then (Angles, 2012). Particularly, the neo4j paradigm is one of the most ways to represent the graph visualization of this specific organization and context[3]. It is important to emphasize that most of studies reviewed by authors followed, above all, a theoretical interest and not so much on practical developments. Regarding the recent developments in the area, a research was carried out with the purpose of analyzing and comparing the main databases by existing graphs. Thus, based on a ranking of 31 systems by September 2022, the top 10 are selected, as shown in Table 1. This table consists of the ranking position, the name of the database management system, the model supported by each of the database technologies, and also a score.

The score is assigned taking into consideration the popularity of a system using the following parameters:

- Number of mentions of the system on websites, measured as number of results in search engine queries. At the moment we use Google and Bing for this measurement. In order to count only relevant results, we are searching for <system name> along with the term database, for example, "Oracle" and "database".

[3] https://db-engines.com/en/ranking/graph+dbms

- General interest in the system. For this measurement, the frequency of Google Trends searches is used.
- Frequency of technical discussions about the system. The number of related questions and the number of interested users on the well-known Q&A sites Stack Overflow and DBA Stack Exchange is used.
- Number of job offers, in which the system is mentioned. We use the number of offers on the leading job search engines Indeed and Simply Hired.
- Number of profiles in professional networks, in which the system is mentioned. We use the most popular professional network internationally, LinkedIn.
- Relevance in social networks. We count the number of tweets on Twitter, in which the system is mentioned.

The popularity value of a system is calculated by standardizing and averaging the individual parameters. These mathematical transformations are done in such a way that the distance of the individual systems is preserved. This means that when system A has a value twice as high in the DB Engine Ranking as system B, then it is twice as popular when averaged over the individual evaluation criteria.

In order to eliminate the effects caused by changing quantities of the data sources themselves, the popularity score is always a relative value, which should be interpreted in comparison with other systems only.

DB-Engines Ranking does not measure the number of installations of the systems, or their use within computer systems. It can be expected that an increase in the popularity of a system, as measured by the DB-Engines Ranking (for example, in discussions or job postings), will precede a corresponding broad use of the system by a certain time factor. Because of this, DB-Engines Ranking can act as an early indicator[4].

[4] https://db-engines.com/en/ranking/graph+dbms

Table 1 - Ranking of graph database technologies adapted from Engines ranking 2022

Rank (Sept 2022)	Database Management System (DBMS)	Database Model	Score (Sept 2022)
1	Neo4j	Graph DBMS	59.48
2	Microsoft Azure Cosmos DB	Multi-model	40.67
3	ArangoDB	Multi-model	6.02
4	Virtuoso	Multi-model	5.96
5	OrientDB	Multi-model	4.81
6	Amazon Neptune	Multi-model	3.19
7	JanusGraph	Graph DBMS	2.64
8	GiraphDB	Multi-model	2.52
9	TigerGraph	Graph DBMS	2.14
10	Stardog	Multi-model	1.67

As can be seen from Table 1, there is a significant difference between the first and second place in the ranking. Neo4J ranks first with a score of 59.48 points, much higher than the DBMS that occupies the second position, the Microsoft Azure Cosmos DB. In Figure 1, it can also be seen that the difference from Neo4J to the second position has greatly diminished, with Microsoft Azure Cosmos DB rising exponentially in the last years. While Azure's solution has very interesting characteristics, it also presents important limitations.

As a no-pure graph DBMS and only available through a commercial version, it could only be a viable solution if integrated in a broader Microsoft integrated system solution. Based on the comparisons made, among the database management systems identified above as one can see from Figure 15, there is a huge difference in the score attributed to these two and the rest of the graph-based DBMS, where since 2013 Neo4J is in the 1st place of the ranking.

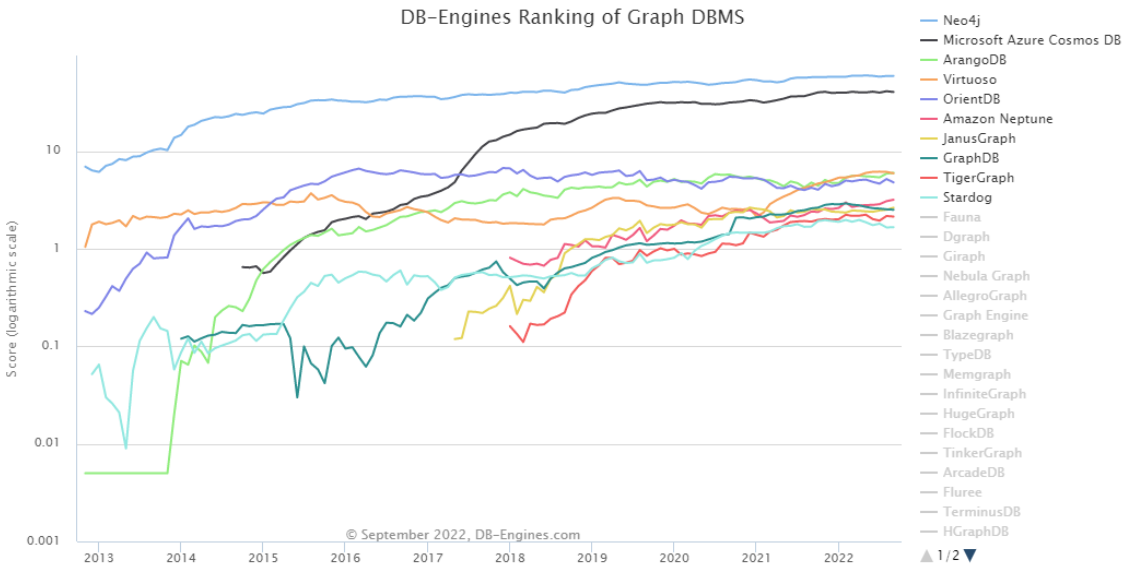


Figure 15 - DB-Engines Ranking database management systems adapted from Engines Ranking 2022

A thorough comparison of the characteristics for the top-4 solutions is further presented in Table 2 and 3, where issues as higher longevity and broadness of supported operating systems, API and programming languages, also account for Neo4J advantage regarding other tools.

Table 2 - Characteristics of graph database tools (top-4 solutions)

	Neo4J	Microsoft Azure Cosmos DB	ArangoDB	Virtuoso
Description	Neo4j is a graph database management system developed by Neo4j, Inc. Described by its developers as an ACID-compliant transactional database with native graph storage and processing.	Azure Cosmos DB is Microsoft's proprietary globally-distributed, multi-model database service "for managing data at planet-scale" launched in May 2017. It is schema-agnostic, horizontally scalable and generally classified as a NoSQL database.	Native multi-model DBMS for graph, document, key/value and search. All in one engine and accessible with one query language.	Virtuoso is a multi-model hybrid-RDBMS that supports management of data represented as relational tables and/or property graphs
Initial release	2007	2014	2012	1998
Current release	4.4.10, August 2022	-	3.9.2, June 2022	7.2.7, May 2022
License	Open Source / Commercial	Commercial	Open Source / Commercial	Open Source / Commercial
Secondary indexes	yes	yes	yes	yes

Table 3 - Characteristics of graph database tools (cont.)

	Neo4J	Microsoft Azure Cosmos DB	ArangoDB	Virtuoso
Primary Databse Model	Graph DBMS	• Document store • Graph DBMS • Key-value store • Wide column store	• Document store • Graph DBMS • Key-value store • Search engine	• Document store • Graph DBMS • Native XML DBMS • Relational DBMS • RDF store • Search engine
Server operating systems	Linux OS X Solaris Windows	hosted	• Linux • OS X • Windows	• AIX • FreeBSD • HP-UX • Linux • OS X • Solaris • Windows
SQL	No	SQL-like query language	No	Yes
APIs and other access methods	• Bolt protocol • Cypher query language • Java API • Neo4j-OGM (object graph mapper) • RESTful HTTP API • Spring Data Neo4j • TinkerPop 3	• DocumentDB API • Graph API (Gremlin) • MongoDB API • RESTful HTTP API • Table API	• AQL • Foxx Framework • Graph API (Gremlin) • GraphQL query language • HTTP API • Java & SpringData • JSON style queries • VelocyPack/VelocyStream	• ADO.NET • GeoSPARQL • HTTP API • JDBC • Jena RDF API • ODBC • OLE DB • RDF4J API • RESTful HTTP API • Sesame REST HTTP Protocol • SOAP webservices • SPARQL 1.1 • WebDAV • XPath • XQuery • XSLT
Supported programming languages	• .Net • Clojure • Elixir • Go • Groovy • Haskell • Java • JavaScript • Perl • PHP • Python • Ruby • Scala	• .Net • C# • Java • JavaScript • JavaScript (Node.js) • MongoDB client drivers written for various programming languages • Python	• C# • C++ • Clojure • Elixir • Go • Java • JavaScript (Node.js) • PHP • Python • R • Rust	• .Net • C • C# • C++ • Java • JavaScript • Perl • PHP • Python • Ruby • Visual Basic

3.5 Conclusions

Throughout this chapter, a survey of the state of the art is carried out on the topics of Ontology Development, Complex Networks and Technologies for building complex networks.

As far as Ontology development is concerned, the focus is on analyzing and detailing some of the currently existing methodologies to conceive ontologies. This analysis is extremely important and relevant because it allows us to assimilate how and what steps are necessary to build ontologies. At the end of section 3.2 we identified SAMOD and UPON lite as the methodologies to follow for future developments in this area.

About the theme of complex networks, throughout this chapter, namely in section 3.3, a survey of the state of the art is carried out in order to elucidate the reader about this subject. In this sense, the objective of this section (3.3) is to identify what complex networks are and how they can be built. In addition, an aspect associated with complex networks involving the visualization of information is also contextualized.

Finally, in section 3.4 the state-of-the-art of existing technologies for building complex networks is surveyed. Here, a survey and benchmark are made of ten graph database technologies that can effectively be used to build complex networks. This benchmark is refined to a more detailed analysis based on the top 4 technologies identified. This more detailed analysis is fundamental because it allowed us to have a range of technologies, which we can use properly and according to each project context.

3.6 References

Abdelaziz, A., Ramadan, N., & Hefny, H. (2017). Towards a Machine Learning Model for Predicting Failure of Agile Software Projects. *International Journal of Computer Applications*, *168*, 20–26.

Águila, I., Palma, J., & Túnez, S. (2014). Milestones in Software Engineering and Knowledge Engineering History: A Comparative Review. *TheScientificWorldJournal*, *2014*, 692510.

Agyapong-Kodua, K., Lohse, N., Darlington, R., & Ratchev, S. (2013). Review of semantic modelling technologies in support of virtual factory design. *International Journal of Production Research*, *51*(14), 4388–4404.

Ahmad Ghazal, Rab, T., Hu, M., Raab, F., Poess, M., Crolotte, A., & Jacobsen, H.-A. (2013). BigBench: Towards an Industry Standard Benchmark for BigData Analytics. *SIGMOD '13: Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, 1197–1208.

Ahmad, M. N., Zakaria, N. H., & Sedera, D. (2011). Ontology-based knowledge management for enterprise systems. *International Journal of Enterprise Information Systems*, *7*(4), 64–90.

Al-Kassab, J., Ouertani, Z. M., Schiuma, G., & Neely, A. (2014). Information visualization to support management decisions. *International Journal of Information Technology & Decision Making*, *13*(02), 407–428.

Anand Bowers, S., Ludascher, B., M. K. (2010). Techniques for efficiently querying scientific workflow provenance graphs. *International Conference on Extending Database Technology (EDBT)*, 287–298.

Angles, R. (2012). A comparison of current graph database models. *Proceedings - 2012 IEEE 28th International Conference on Data Engineering Workshops, ICDEW 2012*, 171–177.

Becheru, A., & Badica, C. (2014). *Complex Networks' Analysis Using an Ontology-Based Approach: Initial Steps*.

Bedini, I., & Nguyen, B. (2007). Automatic Ontology Generation : State of the Art. *PRiSM Laboratory Technical Report. University of Versailles*, 1–15.

Bihanic, D. (2015). *New challenges for data design*. Springer.

Bihanic, D., & Polacsek, T. (2012a). Models for visualisation of complex information systems. *2012 16th International Conference on Information Visualisation*, 130–135.

Bihanic, D., & Polacsek, T. (2012b). Models for visualisation of complex information systems. *Proceedings of the International Conference on Information Visualisation*, 130–135.

Bodik, P., Fox, A., Franklin, M. J., Jordan, M. I., & Patterson, D. A. (2010). Characterizing, Modeling, and Generating Workload Spikes for Stateful Services. *Proceedings of the 1st ACM Symposium on Cloud Computing*, 241–252.

Brusa, G., Caliusco, M. L., & Chiotti, O. (2006). A process for building a domain ontology: an experience in developing a government budgetary ontology. *Proceedings of the Second Australasian Workshop on Advances in Ontologies*, *72*(c), 7–15.

Card, S. (2002). Information Visualization. *IEEE Computer Graphics and Applications*, *37*(2).

Card, S. K., Mackinlay, J. D., & Shneiderman, B. (1999). *Readings in Information Visualization: Using Vision to Think* (B. (Editors) Card, S., Mackinlay, J., and Shneiderman (ed.)). Morgan Kaufmann Publishers.

Chen-Huei Chou, F. Zahedi, H. Z. (2011). Ontology for Developing Web Sites for Natural Disaster Management: Methodology and Implementation. *Computer Science IEEE Transactions on Systems, Man, and Cybernetics*.

Chen, C. (2004). *Information visualization: Beyond the horizon*. Springer Science & Business Media.

Chen, C. (2010). Information visualization. *Wiley Interdisciplinary Reviews: Computational Statistics*, *2*(4), 387–403.

Chen, C. (2020). *Introduction to Information Visualization: Transforming Data into Meaningful Information. By Gerald Benoit*. University of Toronto Press.

Chi, E. H. (2000). A taxonomy of visualization techniques using the data state reference model. *IEEE Symposium on Information Visualization 2000. INFOVIS 2000. Proceedings*, 69–75.

Compeau, P. E. C., Pevzner, P. A., & Tesler, G. (2011). How to apply de Bruijn graphs to genome assembly. *Nature Biotechnology*, *29*(11), 987–991.

Corcho, O., Fernández-López, M., & Gómez-Pérez, A. (2003). Methodologies, tools and languages for building ontologies. Where is their meeting point? *Data and Knowledge Engineering*, *46*(1), 41–64.

De Nicola, A., & Missikoff, M. (2016). A lightweight methodology for rapid ontology engineering. *Communications of the ACM*, *59*(3), 79–86.

De Nicola, A., Missikoff, M., & Navigli, R. (2009). A software engineering approach to ontology building. Information Systems, 34(2), 258-275. *Information Systems*, *34*, 258–275.

Delir Haghighi, P., Burstein, F., & Zaslavsky, A. (2013). Development and evaluation of ontology for intelligent decision support in medical emergency management for mass gathering. *Decision Support Systems*, *54*, 1192–1204.

Dudás, M., Lohmann, S., Svátek, V., & Pavlov, D. (2018). Ontology visualization methods and tools: A survey of the state of the art. In *Knowledge Engineering Review* (Vol. 33).

Femi Aminu, E., & Sunday Adewale, O. (2015). A Mechanism for Detecting Dead URLs in XTM-based Ontology Repository. *International Journal of Computer Applications*, *111*(12), 6–12.

Gavrilova, T. A., & Leshcheva, I. A. (2015). Ontology design and individual cognitive peculiarities: A pilot study. *Expert Systems with Applications*, *42*(8), 3883–3892.

Gokhale, P., Deokattey, S., & Bhanumurthy, K. (2011). Ontology Development Methods. *DESIDOC Journal of Library & Information Technology*, *31*.

Gomez-Perez, A., Fernández-López, M., & Corcho, O. (2004). *Ontological Engineering: With Examples from the Areas of Knowledge Management, E-Commerce and the Semantic Web*.

Graham, M., Kennedy, J., & Benyon, D. (2000). Towards a methodology for developing visualizations. *International Journal of Human-Computer Studies*, *53*, 789–807.

Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition*, *5*(2), 199–220.

Gruninger, M., & Fox, M. S. (1995). *Methodology for the Design and Evaluation of Ontologies*.

Guarino, N. (1995). *Ontologies and knowledge bases: towards a terminological clarification* (pp. 25-32.).

Guarino, N., Oberle, D., & Staab, S. (2009). What Is an Ontology? In *Handbook on Ontologies* (pp.

1–17).

Heer, J., & Shneiderman, B. (2012). A taxonomy of tools that support the fluent and flexible use of visualizations. *Interactive Dynamics for Visual Analysis*, *10*, 1–26.

Herman, I., Melançon, G., & Marshall, M. S. (2000). Graph Visualization and Navigation in Information Visualization: A Survey. *IEEE Transactions on Visualization and Computer Graphics*, *6*(1), 24–43.

Hlomani Hlomani, & Deborah Stacey. (2014). Approaches, methods, metrics, measures, and subjectivity in ontology evaluation: A survey. *Semantic Web Journal*, *1*(5), 1–11.

Ho, L.-Y., Wu, J.-J., & Liu, P. (2012). Distributed graph database for large-scale social computing. *2012 IEEE Fifth International Conference on Cloud Computing*, 455–462.

Ho, L. Y., Wu, J. J., & Liu, P. (2013). Data replication for distributed graph processing. *IEEE International Conference on Cloud Computing, CLOUD*, 319–326.

Hou, J., Yang, X., & Chen, C. (2018). Emerging trends and new developments in information science: A document co-citation analysis (2009–2016). *Scientometrics*, *115*(2), 869–892.

Iqbal, R., Mustapha, A., & Yusoff, Z. (2013). An experience of developing Quran ontology with contextual information support. *Multicultural Education & Technology Journal*, *7*.

Islam, N., Abbasi, A., & Shaikh, Z. (2010). *Semantic Web: Choosing the right methodologies, tools and standards*.

Ji, H., & Gan, W. (2020). Data Visualization for Making Sense of Scientific Literature. *2020 International Conference on Intelligent Transportation, Big Data & Smart City (ICITBS)*, 870–873.

Johnson, B., & Shneiderman, B. (1991). Tree-maps: a space-filling approach to the visualization of hierarchical information structures. *Proceeding Visualization '91*, 284–291.

Jones, D., Bench-Capon, T., & Visser, P. (1998). Methodologies for Ontology Development. *Proc. IT&KNOWS Conference of the 15th IFIP World Computer Congress*.

Kapoor, B., & Savita, S. (2010). A Comparative Study Ontology Building Tools for Semantic Web Applications. *International Journal of Web & Semantic Technology*, *1*.

Katifori, A., Halatsis, C., Lepouras, G., Vassilakis, C., & Giannopoulou, E. (2007). Ontology visualization methods - A survey. *ACM Computing Surveys*, *39*(4), 1–43.

Kim, M. C., Zhu, Y., & Chen, C. (2016). How are they different? A quantitative domain comparison of information visualization and data visualization (2000–2014). *Scientometrics*, *107*(1), 123–165.

Liao, J., Li, L., & Liu, X. (2013). An integrated, ontology-based agricultural information system. *Information Development*, *31*, 150–163.

Looser, D., Ma, H., & Schewe, K. D. (2013). Using formal concept analysis for ontology maintenance in human resource recruitment. *Proceedings of the Ninth Asia-Pacific Conference on Conceptual Modelling (APCCM 2013)*, *143*(Apccm), 61–68.

Lopez, M. F., Gomez-Perez, A., Sierra, J. P., & Pazos, A. (1999). Building a Chemical Ontology Using Methontology and the Ontology Design Environment. IEEE Intelligent Systems. *Intelligent Systems and Their Applications, IEEE*, *14*, 37–46.

Miller, J. (2013). Graph Database Applications and Concepts with Neo4. *Proceedings of the 2013 Southern Association for ...*, 141–147.

Munir, K., & Sheraz Anjum, M. (2018). The use of ontologies for effective knowledge modelling and information retrieval. *Applied Computing and Informatics*, *14*(2), 116–126.

Nanda, J., Simpson, T. W., Kumara, S. R. T., & Shooter, S. B. (2006). A methodology for product family ontology development using formal concept analysis and web ontology language . In *Journal of Computing and Information Science in Engineering* (Vol. 6, Issue 2, pp. 103–113).

Netto, C. M., & Lima, G. Â. (2017). Visualização de ontologias: Estudos e perspectivas. *Informacao e Sociedade*, *27*(3), 59–72.

Noy, N. (2001). *Ontology Development 101: A Guide to Creating Your First Ontology*.

Park, J., Sung, K., & Moon, S. (2008). *Developing Graduation Screen Ontology based on the METHONTOLOGY Approach*.

Rajpathak, D., & Chougule, R. (2011). A generic ontology development framework for data integration and decision support in a distributed environment. *Robotics and Computer-Integrated Manufacturing*, *24*, 154–170.

RISDEN, K., Czerwinski, M., Munzner, T., & Cook, D. (2000). Initial examination of ease of use for 2D and 3D information visualizations of Web content. *International Journal of Human-Computer Studies*, *53*, 695–714.

Rodriguez, M. A., & Neubauer, P. (2010a). *Constructions from Dots and Lines*. *X*(X), 1–11.

Rodriguez, M. A., & Neubauer, P. (2010b). *The Graph Traversal Pattern*. 1–18.

Sattar, A., Surin, E. S. M., Ahmad, M. N., Ahmad, M., & Mahmood, A. K. (2020). Comparative analysis of methodologies for domain ontology development: A systematic review. *International Journal of Advanced Computer Science and Applications*, *11*(5), 99–108.

Schwaber, K., & Sutherland, J. (2020). *Scrum Guide: The Definitive Guide to Scrum: The Rules of the Game*. *November*, 133–152.

Serna M., E., & Serna A., A. (2014). Ontology for knowledge management in software maintenance. *International Journal of Information Management*, *34*(5), 704–710.

Shao, B., Wang, H., & Xiao, Y. (2012). Managing and mining large graphs: systems and implementations. *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, 589–592.

Shneiderman, B. (1996). Eyes have it: a task by data type taxonomy for information visualizations. *IEEE Symposium on Visual Languages, Proceedings*, 336–343.

Silva-López, R. B., Silva-López, M., Bravo, M., Méndez-Gurrola, I. I., & Sánchez Arias, V. G. (2014).

GODeM: A Graphical Ontology Design Methodology. *Research in Computing Science*, *84*(1), 17–28.

Sivakumar, R., & Arivoli, P. V. (2011). ONTOLOGY VISUALIZATION PROTÉGÉ TOOLS – A REVIEW. *International Journal of Advanced Information Technology (IJAIT)*, *1*(4), 1–11.

Sofia Pinto, H., & Martins, J. P. (2001). A methodology for ontology integration. *Proceedings of the First International Conference on Knowledge Capture*, 131–138.

Tudorache, T. (2020). Ontology engineering: Current state, challenges, and future directions. *Semantic Web*, *11*(1), 125–138.

Vicknair, C., Nan, X., Chen, Y., & Wilkins, D. (2010). *A Comparison of a Graph Database and a Relational Database A Data Provenance Perspective*. 6.

Vigo, M., Bail, S., Jay, C., & Stevens, R. (2014). Overcoming the pitfalls of ontology authoring: Strategies and implications for tool design. *International Journal of Human-Computer Studies*, *72*(12), 835–845.

Vukotic, A., Watt, N., Abedrabbo, T., Fox, D., & Partner, J. (2015). *Neo4j in action* (Vol. 22). Manning Shelter Island.

Wood, P. T. (2012). Query languages for graph databases. *ACM SIGMOD Record*, *41*(1), 50.

Wünsche, B. (2004). A survey, classification and analysis of perceptual concepts and their application for the effective visualization of complex information. *Information Visualisation*, 17–24.

Yongfeng Li, Fei Tao, Ying Cheng, Xianzhi Zhang, A. Y. C. N. (2017). Complex networks in advanced manufacturing systems. *Journal of Manufacturing Systems*, *787*, 409–421.

—— This page is intentionally left blank ——

PART III – CONTRIBUTIONS

—— This page is intentionally left blank ——

CHAPTER 4

ON THE CONSTRUCTION OF TERMINOLOGIES

Summary: Issues related to interoperability, namely semantics based on ontologies, play an increasingly impactful role in today's organizations, which increasingly rely on the integration of different information systems. After an initial comparison of the 4SRS of our research groups with other proposals inspired by agile methodologies, such as UPON and AMOD, we present our proposal the 4SRS-Onto. This work is based on items related to ontologies, namely the need to perform a survey of terminologies, their cataloging through synonyms catalogs, the relationship between terms and, finally, the information visualization phase.

CHAPTER TABLE OF CONTENTS

4.1 Introduction....103
4.2 The 4-Step Rule Set Method for Terminologies Derivation....105
4.3 Demonstration Case....117
4.4 Conclusions....124
4.5 References....124

—— This page is intentionally left blank ——

CHAPTER 4

ON THE CONSTRUCTION OF TERMINOLOGIES

"Go on, then, you make the coffee while I try to bring some order to this chaos, and then the unexpected happened, for, as if giving no particular importance to the words emerging from her mouth or as if she did not entirely understand them, she murmured, 'Chaos is merely order waiting to be deciphered'"

– José Saramago, The Double (2002)

4.1 Introduction

Ontologies are semantic structures that encode concepts, relations, and axioms, providing a model for a given domain and forming the backbone of the Semantic Web, a semantic-aware version of the World Wide Web (Berners-lee & Hendler, 2001). Building an ontology is a complex task, one that requires a lot of engineering effort, discipline, and rigor, with a set of different user profiles involved in its development. Ontological engineering denotes a set of design principles, development processes and activities, supporting technologies, and systematic methodologies that facilitate ontology development and use throughout its life cycle (Gasevic *et al.*, 2006).

In the context of information systems and, more specifically, in software development, there is a growing interest in the use of ontologies as an explicit specification of a conceptualization because of their classes and properties visibility and their abstract and simplified version of the world to represent. Moreover, ontologies can be logically reasoned and shared within a specific domain, with every piece of knowledge in it committed to some explicit or implicit conceptualization. Thus, ontologies are seen as a standard

form for representing the concepts within a domain, as well as the relationships between those concepts, in a way that allows automated reasoning (Dermeval *et al.*, 2016).

The field of software engineering already conveys several ontological engineering developments that have trailed software engineering process models, as the associated community has already dealt with most of the process models and engineering challenges, adopting some of their most successful techniques, including the increasing reuse of existing ontologies by applying reference ontologies (Feilmayr & Wöß, 2016). Nevertheless, although diverse methods have been proposed over the years, no common ground has been established as these struggle to establish guided tasks and propose cycles of analysis, design, and validation inside them (Pinto & Martins, 2004).

The lack of references and difficulty in establishing a consensual scientific method has led to solutions based on the prominent UP/RUP methodology, in part due to the progressive comparison between building an ontology and building a software product (De Nicola & Missikoff, 2016). Nevertheless, the development of an ontology should always take into consideration ontological thoughts and directions (Peroni, 2018), which are heavily analyst-dependent acts. Following these recent works in ontology building and taking advantage of our work with the 4-Step Rule Set (4SRS) method for deriving an information system architecture using tabular transformations (R. J. Machado *et al.*, 2005a), we now propose an extension of this method aimed at building an ontology for an information system domain, the 4SRS Method for Ontological Design.

The original 4SRS method and its variants have proven their usefulness and adequacy in different contexts, either isolated or integrated with a V-Model approach, with its structured tasks based on the RUP methodology and its iterative cycles following the Twin-peaks approach. This method is further demonstrated in an industrial live case scenario.

This chapter follows with section 4.2, presenting background information regarding recent contributions to agile methods for building ontologies and also relating to the evolution of the 4SRS and its variants. Next, it describes in detail our proposal for the individual steps and micro-steps of the 4SRS-Onto method, while section 4.3 explores a demonstration of the application of the method in a live industrial case. Following that, section 4.4 presents a brief discussion of the work here presented and finishes with the conclusions.

4.2 The 4-Step Rule Set Method for Terminologies Derivation

Probably the most difficult task involved in the development of information systems is turning a need definition into an architectural design decision (Bosch & Molin, 1999). The issues from the previous phases are still there, but they are better understood, and the software engineer has access to a wide range of techniques, languages, and tools. Software architecture design is, by far, a less formalized process that frequently relies heavily on intuition and ad hoc work that is not well grounded in engineering concepts. Since a software system's architecture restricts the space solution, the decisions made during architectural design must be chosen carefully since they frequently have a significant impact on the effectiveness of the system that is produced.

The 4SRS method, which leverages progressive modifications of the software architecture to suit the elicited user needs, is provided as a method for architectural transformation. Its fundamental foundation is the conversion of UML use case diagrams into UML object diagrams. To ensure that the final design meets the user's needs, it is crucial that the method be iterative and that graphical representations be used. The recursive implementation of the 4SRS approach is shown in Figure 16. By using a recursive version of the 4SRS method, this work tackles the issue of determining the logic architecture of a specific platform service (referred to as a service object diagram) from a functional refinement of the platform architectural model (referred to as a platform object diagram).

One platform object diagram created by the initial 4SRS run, which matches the system's logic architecture, aids in the platform needs analysis (this first 4SRS execution is described in detail in (R. J. Machado *et al.*, 2005b)). One service object diagram produced by the second 4SRS run, which supports the service requirements analysis, relates to the logic architecture of the service that has to be described (this second 4SRS execution is the aim of this work).

The services themselves, and the end client interfaces must all be considered as user needs when defining the underlying service-oriented software architecture for mobile applications in order to properly describe the platform. Recursively using the 4SRS method, the specification of one service of the application in analysis is achieved in the demonstration instance that is provided (R.-J. Machado *et al.*, 2006).

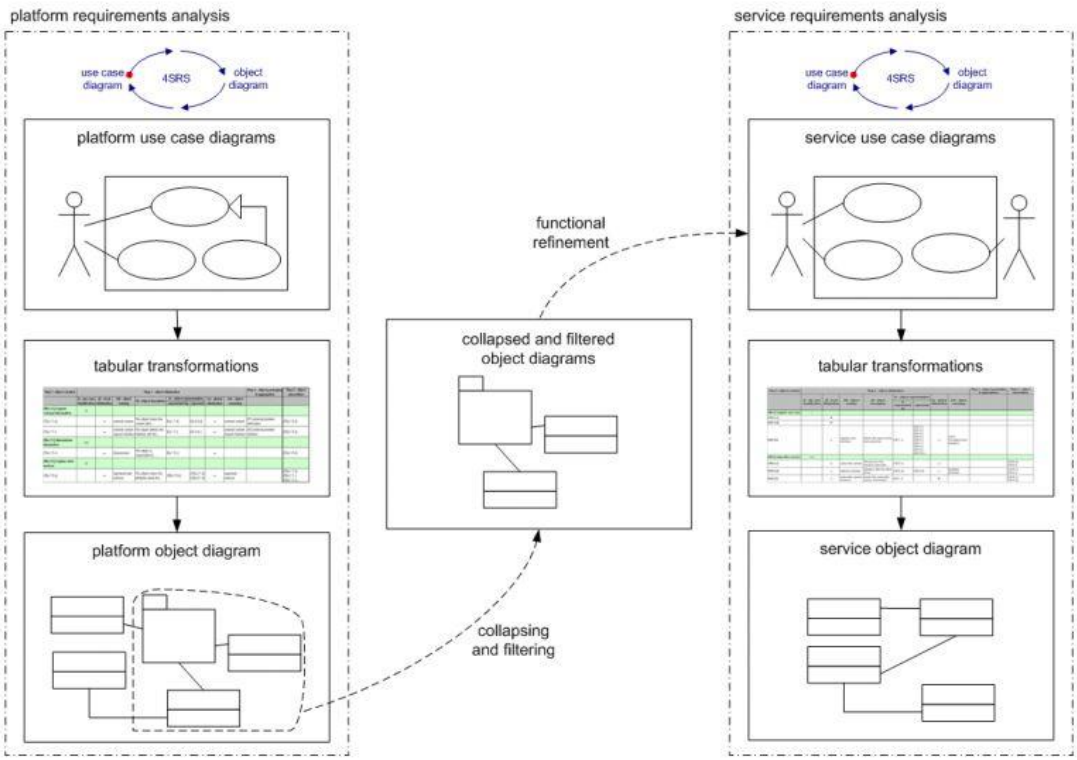


Figure 16 - Service specification with recursive 4SRS execution from (R.-J. Machado *et al.*, 2006)

The 4SRS method thus originated as a technique to aid software engineers to transform user requirements models into an initial logical architecture of the system (R. J. Machado *et al.*, 2005b). It is based primarily on the mapping of UML use-case diagrams into UML object diagrams, where UML sequence, activity, and state diagrams and other artifacts could also be considered within the transformation decisions. It uses the functional decomposition of requirements (in UML use-cases) for deriving a logical architecture composed by UML components that trace back to each elicited functionality. Further integrated in a V-Model approach, it supports and guides the design of information systems architectures by successive model derivation based on domain specific needs, promoting the alignment and traceability between the logical architecture and the requirements supporting models.

The 4SRS method has been presented as a way to convert user needs into architectural models that describe system requirements (Fernandes & Machado, 2001a). Each item discovered during the analysis phase is given a specific type, such as interface, data, or control. These categories are each connected to one of the three orthogonal dimensions—information, behavior, and presentation—that may be used to segment the analysis space (Jacobson, 1993). The 4SRS method is then briefly described for readability's sake. A detailed explanation of how to use it to derive the initial logical architecture of the sample case used in this work adopting a non-recursive method is provided in (R. J. Machado *et al.*, 2005a). Bragança

et al., (Bragança & Machado, 2005) describes an alternate method for determining the logical architecture for software product lines using the 4SRS method.

The method (as the name implies) is organized in four steps, namely: Step 1, component creation; Step 2, component elimination; Step 3, components packaging & aggregation; and Step 4, component association. The components derivation follows the Model-View-Controller (MVC) pattern. Briefly stating, Step 1 regards Components Creation, where each component found in analysis is associated to a given category: interface (i-type), data (d-type) and control (c-type), representing the different logic layers. Step 2 regards Components Elimination, where these (i-, d-, or c-type) are maintained or eliminated, considering the entire system, according to predefined rules. Step 3 deals with Component Packaging, where the remaining components should give origin to semantically consistent aggregations or packages. Lastly, Step 4 deals with Component Associations, defined whenever they are originated by the same use-case, which are also based in the flows between the use-cases during the requirements modeling phase.

Tabular representations can be used to enable the 4SRS transformation stages' execution. Additionally, the usage of tables makes it possible to create and build a set of tools that partially automate conversions. The main method for automating a variety of decision-aided model transformation stages is these tabular representations. It has been demonstrated that 4SRS is agile in assisting software developers uncover and revise architectural needs based on user requirements via application in both academia and industry [(Fernandes *et al.*, 2000), (Fernandes & Machado, 2001a), (Fernandes & Machado, 2001b)]. The transformation steps are listed in the following table: As shown in Figure 17, (1) each micro-step results in a column, and (2) each item results in a row.

The first column represents the performance of step 1. Both the reference and the use case name can be entered in the first row. For the appropriate use case, it is possible to may enter an interface, some data, and a control object in the following three rows. Since the use case for the example case has not been refined, step 1 is relevant to all use cases.

The second column represents the performance of micro-step 2i. The software engineer categorizes each use case in this micro-step into one of eight possible combinations or patterns I c, d, ic, di, cd, icd). This categorization is created with the intention of assisting in the conversion of each use case into an object. This division would offer hints as to which groups of things to employ and how to connect them.

Step 1 -object creation	Step 2 - object elimination								Step 3 - object packaging & aggregation	Step 4 - object association
	2i - use case classification	2ii - local elimination	2iii - object naming	2iv -object description	2v - object representation		2vi - global elimination	2vii - object renaming		
					is represented by	represents				
{U0.1} register new user	i									
{O0.1.c}		x								
{O0.1.d}		x								
{O0.1.i}		–	register user interface	allows the parse of the user personal...	{O0.1.i}	{O0.2.i} {O0.3.i} {O0.4.i} {O0.5.i} {O0.6.i} {O0.7.i} {O0.8.i} {O0.9.i} {O0.10.i}	–	users managemment interface		
{U0.5} subscribe service	icd									
{O0.5.c}		–	subscribe service	will process the request subscribe...	{O0.5.c}		–			{O0.5.d} {O0.5.i}
{O0.5.d}		–	defined activities	interface with the data of the...	{O0.5.d}	{O0.9.d}	–	available activities		{O0.5.c} {O0.5.i}
{O0.5.i}		–	subscribe service interface	sends the subscribe service information...	{O0.1.i}		x			{O0.5.c} {O0.5.d}

Figure 17 - Service specification with recursive 4SRS execution adapted from (R.-J. Machado *et al.*, 2006)

The 3rd column corresponds to the execution of micro-step 2ii. The objective of this micro-step is to answer whether each object created in step 1 makes sense in the problem domain, since the creation of the objects in step 1 is executed blindly, not considering the system context for the object creation. Objects that should be deleted are marked with "x" and objects that should be kept are marked with "-".

The 4th column corresponds to the execution of micro-step 2iii. In this micro-step, objects that have not been eliminated from the previous micro-step should be given a proper name that reflects both the use case from which it comes and the specific role of the object, considering its main component. The 5th column corresponds to the execution of micro-step 2iv. Each named object resulting from the previous micro-step must be described, so that the system requirements they represent are included in the object model. These descriptions should be based on the descriptions of the original use cases.

The 6th and 7th columns correspond to the execution of micro-step 2v. This is the most critical micro-step of the 4SRS method, since it supports the elimination of redundancy in the elicitation of user requirements, as well as the discovery of missing requirements. The "is represented by" column stores the object reference that will represent the object to be analyzed. If the analyzed object will be represented by itself, the corresponding "is represented by" column must refer to itself. The "represents" column stores the references to the objects that the analyzed object will represent.

The 8th column corresponds to the execution of micro-step2vi. This is a fully "automatic" micro-step, since it is based on the results of the previous one. Objects that are represented by others should be deleted, since their system requirements no longer belong to them.

The 9th column corresponds to the execution of micro-step2vii. Its purpose is to rename the objects that are not deleted in the previous micro-step and that represent additional objects. The new names should reflect the full system requirements.

The 10th column corresponds to the execution of step 3. The 11th column corresponds to the execution of step 4. For the demonstration case, the associations are solely derived from the use case classification performed in step 1.

Numerous researches have made it possible to evolve these techniques into a method that already has a considerable amount of associated work. These include several topics such as the refinement of architectures by the application of an approach (R. J. Machado *et al.*, 2005b), an extension to support the construction of a class diagram that complements the logical architecture (M. Y. Santos & Machado, 2010) and another applicable to product line architectural modeling and an adaptation for automation purposes and the study of the well-being of relationships (Azevedo *et al.*, 2012).

The context in which a corporation works is always relevant when deciding whether to integrate an ontological component. Prior to characterizing the firm, it is important to recognize and study the organizational environment, as well as the industry and related ecosystem, in which the company operates. Figure 18 proposes a traceable path between the Data Characterization, initial and detailed requirements in UML use cases, a definition of the data model to be used, specification of the data model, and finally the ontological design of the defined schema. This proposal is based on our prior research and background experience.

This approach uses the V-Model put out by Machado *et al.*, notably its left-descending branch, to build a data model that is in line with the criteria modeled in Use Cases (Ferreira *et al.*, 2013). The requirements elicitation for this V-Model left branch proposal includes a definition of the business processes for implementing the solution, sequential ordering of features, later modeling in A-type sequence diagrams (N. Santos *et al.*, 2015), and finally modeling in use case diagrams.

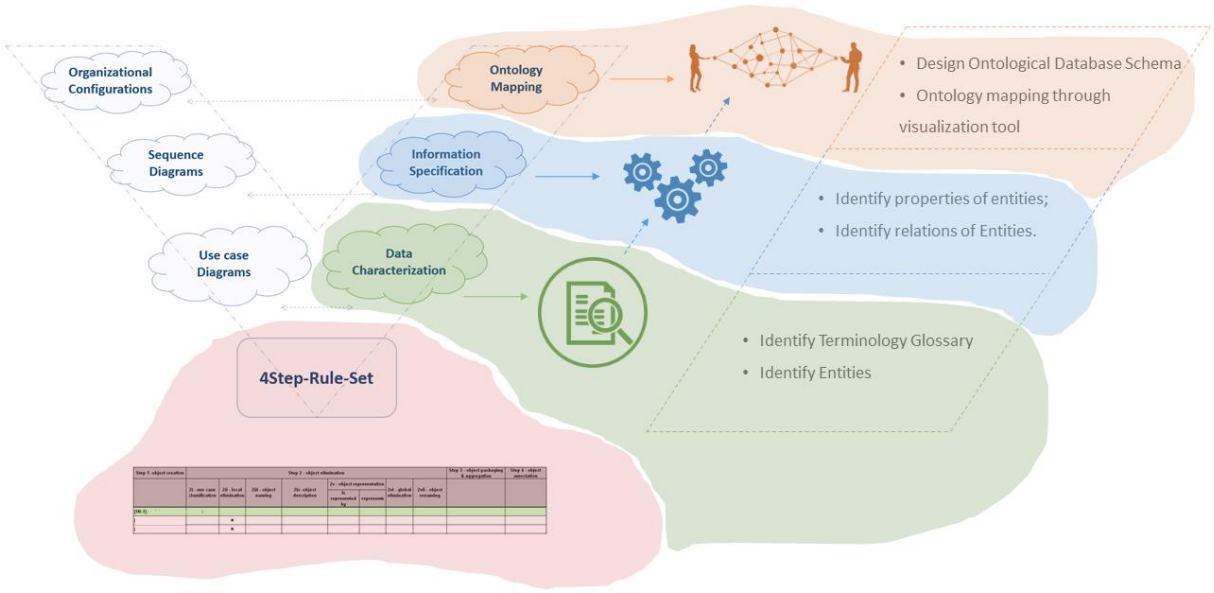


Figure 18 - 4SRS Method for Ontological Design

The way to elicit the needs of business processes ("Input from Organizational Configuration") necessary to derive the software requirements (A-type sequence diagrams and UML use cases) is described in previous work (N. Santos *et al.*, 2015); (Carlos Salgado *et al.*, 2019); (N. Santos *et al.*, 2018). It is not the purpose of this section to address the derivation of UML Use Cases. Rather, what must be retained from this process is that the requirements are elicited in UML use cases and subsequently the data model, as well as the ontological design, is aligned with each of the corresponding phases in the descending branch of the V-Model.

At the intersection of the descending and ascending branches, there is the 4SRS method. The 4SRS method (R. J. Machado *et al.*, 2005b) has proven to be versatile in deriving an architecture of components as the base structure for the development of an information system, in diverse contexts through its different versions. As the process of building ontologies is currently being associated to the development of information systems (De Nicola & Missikoff, 2016) it is then natural that the 4SRS method can be adapted to this process.

Regarding the initial conception of the method, the functional domain on which it is applied would be characterized by an initial set of UML use-cases, one of the most popular techniques for requirements

elicitation in this type of projects. These use-cases can be complemented with the association of business goals and rules in order to handle the non-functional issues in the domain (Salgado *et al.*, 2013). When building an ontology, the abstraction level of requirements is taken one-step up, so we consider that, in this case, for the characterization of the domain it can be taken in consideration a more diversified set of inputs, as business documents, use-cases, legacy relational databases, among others.

Also, this initial phase has, for some time, been the stage for important works on data preparation, which according to The Cross-Industry Standard Process for Data Mining (CRISP-DM), is one of the most important phases in knowledge discovery, counting with the help of domain experts to guide the creation of an initial ontology (Santoso *et al.*, 2011).

Overall, it is up to the analysts involved in the ontology building process to decide the origin of the elicitation inputs, depending on the available resources and the scope of the domain to cover. Moreover, as this is an iterative approach, a previous ontology built within a prior iteration of this same method can also be used as an input for the domain analysis. This new proposal for the method, the 4SRS for Ontological Design, is again composed by the four main steps initially proposed in the original method, each with their own associated micro-steps, but adapted to the reality of an ontology building process. Furthermore, this 4SRS-Onto is a derivation of the original 4SRS as with the 4SRS-SoaML version (Carlos Salgado *et al.*, 2019), an adaption of the traditional method to derive an architecture of services, where the service Participants (P) are now replaced by ontology Terms (T).

After applying the 4SRS, represented with red shading in Figure 18, and based on the outputs that result from its execution, we obtain an optimized set of terminologies for a specific domain. These terminologies that result from the execution of the 4SRS then serve as input for the next phase called the "Data Characterization" phase (green shading in Figure 18). In this phase, after the data sources have been selected, we must begin to index the terminologies present in the various sources, thus building a glossary of terminologies for the domain under study. After we finished cataloging the terminologies, we grouped and categorized the terminologies into Entities.

Next, in the Information Specification phase, represented with blue shading in Figure 18, the analysis should be oriented towards the existing data model within the domain of interest. That is, in this specification phase we use the detail from the previous phase (Data Characterization) and start identifying properties for each of the entities. After we have defined the properties, we relate the entities by creating

a data model. To perform this phase, of relating and identifying properties for each entity, we use a UML class diagram.

In the last step, Ontology Mapping Phase, with orange shading in Figure 18, we use the detailed information from the previous phase (Information Specification) to build an ontology schema using graph database technology. We then proceeded to integrate this ontology database into a visualization tool, in which stakeholders can apply filters, edit the database, or add new terminology and relationships in a more user-friendly way.

Throughout this chapter we will only focus on the description and demonstration of the 4SRS method for ontological design, with the red shading in Figure 18. The phases that are located in the right ascending arm of the V-Model (Data Characterization, Information Specification, and Ontology Mapping phases) as they are already published and validated in three other papers, will not be detailed here. Thus, in the following we will detail the 4SRS-Onto method, briefly describing its 4 steps including a case of demonstrating it in a real context.

Step 1: Terminology Creation

The initial step of the method is divided in two sequential micro-steps, the first regarding the building of the lexicon of T and the second with the construction of its associated glossary, i.e., the analysis and detail of each of the initially elicited T (Figure 19).

Step 1 -Terminology creation			
1.i)			1.ii)
Id	Term	Origin	Description

Figure 19 - Identification and analysis of Terms in a domain

Whereas in the traditional 4SRS method this step relates to the elicitation or discovery of components or participants for the architecture, now the focus is on the elicitation or discovery of terms for the ontology.

1.i) Term Lexicon: The first task in this initial step is to gather all relevant T in the domain by surveying through its input resources, where the techniques applied here can range from interviews with domain experts to data mining activities with existing legacy data, among others. Accordingly, there should be registered the resource origin for each T, which will be later useful in performing step 4. Again, we state that the dimension of the addressed problem should be adequately dimensioned regarding the availability of analysts involved in the work to perform, in order not to overwhelm their work or raise unrealistic expectations.

1.ii) Term Glossary: For each identified T in the domain, an analysis must be performed on its role in the context at work, in order to detail its description and inner works. This analysis can be performed as thoroughly as needed, also depending on the intended goals and the availability of analysts, either way, it can be further deepened or extended anytime in a future iteration of the method. The structure of this glossary is also freely decided among the analysts involved, again depending on the intended goals and objectives of the project at hand.

Step 2: Terminology Optimization

Next, the second step of the method is divided into four transversal micro-steps, each related to decisions on optimizing the representation of each of the initial identified T (Figure 20). These decisions are related to, as in the traditional 4SRS method, identifying elements that can represent the functionality of others, thus representing them in the global architecture. Now the focus is on the identification of T that can take ownership of representing other similar T in the global ontology, as for instances the identification of their synonyms.

Step 2 - Terminology Optimization				
2.i)	2.ii)	2.iii)	2.iv)	
Reference	Description	Property	Typing	Constraint

Figure 20 - Representation of Terms

Opposite to the traditional 4SRS method, there is no global elimination of the represented T, as they should remain 'alive' to be referenced to their more common representation in the domain. Added to this, there is also a task responsible for the clarification whether a term is relevant enough for the domain or if it is merely represented as a property, in which case a set of constraints should be associated to it.

2.i) Term Representation: This micro-step purpose is to identify a T whose part can be represented by another T from the global ontology. The assigned analyst signals T that can be referenced by another term without losing coherence within the ontology, while incorporating them in the chosen representing T.

2.ii) Glossary of Terms: For each representing T a further analysis on its role in the context at work must be performed, alongside the surveying of its synonyms, in order for its description to maintain coherence among the diverse represented T, avoiding the loss of any relevant information.

2.iii) Term as Property: Moreover, for each of the initially identified T (except the ones now represented by other) in the domain at hand, there should be identified if they constitute a relevant T on their own or if they can be reduced to a property, be it atomic or complex, or even a reference property (De Nicola & Missikoff, 2016).

2.iv) Property Constraint: Following, for each identified property, constraints such as typing, cardinality, granularity, or definition of a set of representative values should be detailed here. Again, the level and detail of representation for these constraints is left to the decision of the involved analysts, if they are set in a more business-oriented tone or in a more technical oriented view. Additionally, business goals and rules can be set for each term according to Business Motivation Metamodel (BMM) features (Salgado *et al.*, 2013).

Step 3: Terminology Packaging and Aggregation

In this third step, after optimizing the T, both in their representations and detail, a more structure-oriented approach is taken in order to package and aggregate T in representative groups, as in the form of an architecture of business entities using part-whole and is-a hierarchies (Figure 21). These relate to the original purpose of the third step of the 4SRS of packaging and aggregating the elements of a domain in a global architecture.

Step 3 - Terminology packaging & aggregation			
3.i)	3.ii)	3.iii)	3.iv)
Package	Description	Aggregation	Description

Figure 21 - Term Packaging and Aggregation

Now the focus is on grouping and abstracting higher-level T that can package and aggregate other T in the global ontology, thus building a taxonomy on top of the two previous knowledge levels (lexicon and glossary), while providing feedback and validating their initial construction. Another difference regarding the traditional method is that those T that package and aggregate other T are not newly created but rather selected among the previously identified T. Nevertheless, packaging is still used to introduce a very light semantic cohesion between the T, while aggregation imposes a strong semantic cohesion between T, as in the original method approach.

3.i) Term Packaging: Identify package PART-HOOD relations between T, where any T can take part on diverse packages and the T representing a package can itself be part of other packages (as is the case with UML2 Packages). Operationally, each T pertaining to a term package T is tagged accordingly in the tabular data.

3.ii) Term Packaging Glossary: For each packaging T a further description regarding its role in the context of the package representation must be performed, framing the structure of the package and its inner works.

3.iii) Term Aggregation: Identify IS-A aggregation relations between T, building the main structure of the taxonomy, establishing the diverse hierarchies commanded by the more generic concepts or abstract T. Operationally, the T ruled by an aggregation term T is tagged accordingly in the tabular data.

3.iv) Term Aggregation Glossary: For each aggregating T a further description regarding its role in the context of the aggregation representation must be performed, framing the hierarchy of the aggregate and its inner semantics.

Step 4: Terms Association

Lastly, the fourth step of the method deals with the crucial task of defining relations between the different T in the global ontology, which have direct or inferred associations according to the taxonomy previously defined (Figure 22). This is in line with the analogous step of the traditional 4SRS method that is responsible for establishing associations, direct or indirect, between elements in the global architecture.

Similarly, the former packaging and aggregation tasks already establish a type of associations between T, whenever these have a functional relation or common inner works. In addition, the work performed in the third step regarding packaging and aggregation of T implies their early association, anticipating some of the work for this current step. On the other hand, the micro-steps associated with the naming and description of SP in the 4SRS-SoaML method, which regard the aggregation of P, are not considered in this method, as this reality is not applied in an ontological view.

Step 4 - Terminology association	
4.i)	4.ii)
Direct	Infered

Figure 22 - Direct and inferred association of Terms

4.i) Direct Associations: Identify associations between the T that are linked by packaging or aggregation relations. Usually, these are domain-specific relations deriving from T originated by the same origin

(business document, use-case, database schema, etc.), these associations are depicted from the classification given in step 3.

4.ii) Inferred Associations: Identify associations between the T that, although not directly related, are implicit or explicitly linked by any business semantics. Overall, uncovering these relations is highly dependent on knowledge that resides on domain experts, attainable either from thorough analysis of business documents, or by means of interviews or enquiries, or even in some cases through their direct intervention in the process.

4.3 Demonstration Case

In this section, we present our contribution to the derivation of terminologies using the 4SRS-Onto method described earlier in this chapter (section 4.2). This contribution is used and illustrated in a real-world scenario (CCG R&D projects). This contribution is applied to textile industry demonstration cases within the STVgoDigital project. The objective of this demonstration case is to catalogue all terminologies identified in the requirements survey. To accomplish this, we use the 4SRS-Onto method to derive all of the identified terminologies, thereby creating a robust glossary.

The Project "STVgoDigital"

This project is used in this chapter as a way to demonstrate the applicability of the contributions presented, namely the applicability of 4SRS-Onto. In order to demonstrate this new version of the 4SRS method, 4SRS-Onto, we applied it to a live industrial case that concerned the reality around the supply chain of a textile factory. Samples of both raw materials and production lots come from the warehouse and the production environment, and are then forwarded to different laboratories within the company in order to perform quality tests on them.

An information system for these matters already exists, however, it is rudimentary, facing difficulties in dealing with the multiple points of view and the different languages of the various departments involved (production, warehouse, etc.). The construction of an ontology for this domain is a first step in the evolution towards the development of a more robust information system to handle all the issues involved.

The present study limits its scope to the specific subdomain of sample handling within a given department in a textile industry, in order to serve demonstration purposes and to have a clear and agile approach to the problem. In addition, in this specific case, the source for the elicitation of domain terms will be the interviews conducted with employees from different departments and a set of datasets from various information systems currently existing and functioning in the company. In Figure 23, it is possible to visualize the problematic of the STVgoDigital project.

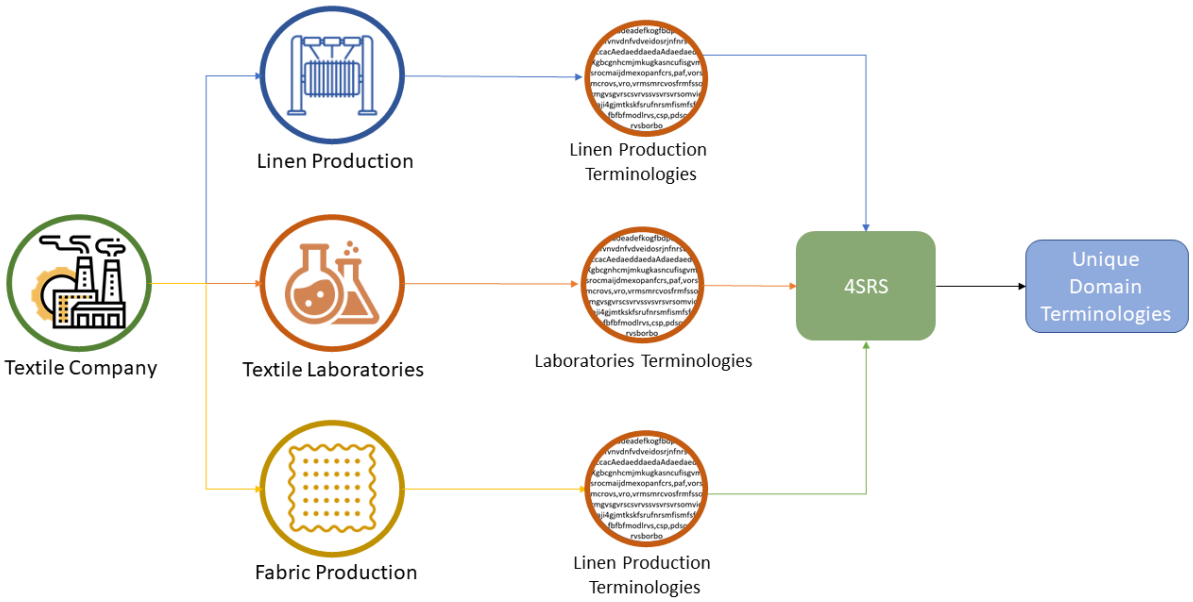


Figure 23 - Unification of terminologies process at STVgoDigital

The initial step of the method, Step-1 Terms Creation, focuses on the elicitation or discovery of terms for the ontology, divided into two sequential micro-steps, with the first regarding the building of the lexicon of terms and the second on the construction of its associated glossary. Regarding this work, three subdomains are surveyed: SAMPLES, PRODUCT and BATCH.

1.i) Term Lexicon: In this first task, a list of terms is elicited among the tables of the available dataset and complemented with associated terms referred to in the interviews with analysts and laboratory leaders. Each term has an associated identifier and an indication of where it originated. A short list of identified terms is presented in Figure 24.

1.ii) Term Glossary: Besides a short description of each term, the associated description field also refers to additional sources where further information for the current term can be found (Figure 24). In this

project, besides the documentation regarding the requirements analysis and interviews, a data catalog for each of the fields in the tables of the dataset has already been prepared.

Step 1 -Terminology creation			
1.i)			1.ii)
Id	**Term**	**Origin**	**Description**
t1	Fabric_Sample	Samples	ERP sample number; ref:s2314
t2	Sample_Type	Samples	The name of the sample type (finished cloth, linen sample)
t3	Status	Samples	Status of the sample: A- Aproved by client; R - Rejected by client; AQC - Approved by quality Control
t4	Old_Status	Samples	Previous status of the sample

Figure 24 - Identification and analysis of Terms in the domain

With regard to Step-2, Terms Optimization, the four transversal micro-steps of the second step of the method were applied consecutively, in order to optimize the representation of each of the identified terms, focusing on the identification of terms that could take ownership of representing other similar terms in the global ontology, as with the identification of synonyms. In addition, there is the clarification of whether a term is relevant enough for the domain or if it is represented as a property, in which case a set of constraints is associated with it.

2.i) Term Representation: Assign the term which will reference the surplus term in the global ontology, in this case we have the OLD_STATUS which can be represented by STATUS without losing coherence in the global ontology (Figure 25).

2.ii) Glossary of Terms: Accordingly, an additional description relating to the representing term is added (Figure 25).

2.iii) Term as Property: Terms as STATUS and OLD_STATUS where considered to be mere properties regarding the SAMPLE term, so they are typed according to their representation within the main term (Figure 25).

2.iv) Property Constraint: Moreover, a set of constraints is registered regarding their control and future handling. Associated to the constraints there are additional goals and rules described in the initial requirements documentation that can be taken in consideration (Figure 25).

Step 1 -Terminology creation				Step 2 - Terminology Optimization				
1.i)			1.ii)	2.i)	2.ii)	2.iii)	2.iv)	
Id	Term	Origin	Description	Reference	Description	Property	Typing	Constraint
t1	Fabric_Sample	Samples	ERP sample number; ref:s2314			No		
t2	Sample_Type	Samples	The name of the sample type (finished cloth, linen sample)			No		
t3	Status	Samples	Status of the sample: A- Aproved by client; R - Rejected by client; AQC - Approved by quality Control		Although in a different time frame	Yes	String	{A, C, I, P, R, U, X}
t4	Old_Status	Samples	Previous status of the sample	Status		Yes	String	{A, C, I, P, R, U, X}

Figure 25 - Representation of Terms

Step 3: Terms packaging and aggregation

As far as step 3, Terms packaging and aggregation, regarding the more structure-oriented approach of packaging and aggregating the terms in representative groups of business entities, either using part-whole or other is-a hierarchies, two main micro-steps are taken. These two group and abstract higher-level terms that package and aggregate other terms in the global ontology, thus building a taxonomy on top of the two previous knowledge levels (lexicon and glossary), while other two associated micro-steps provide textual feedback to complement the introduction of this semantic cohesion between the terms.

3.i) Term Packaging: According to the subdomains analyzed in this case, we can stand out the particular case of the definition of a TEMPLATE for guiding the planning of tests surrounding a SAMPLE. As a TEMPLATE is established according to a certain PRODUCT_TYPE, which is then associated to any production BATCH related to that product and any SAMPLE derived from it. Thus, in this particular case, these three terms should be packaged around the BATCH TEMPLATE, its main driver, as represented in Figure 26.

3.ii) Term Packaging Glossary: Any relevant additional descriptions that help explain the decisions taken should be complemented next to each of the involved terms (Figure 26).

Step 1 -Terminology creation			Step 3 - Terminology packaging & aggregation	
1.i)			3.i)	3.ii)
Id	Term	Origin	Package	Description
t15	TEMPLATE	SAMPLES	TEMPLATE	Directly associated with the BATCH
t18	PRODUCT_TYPE	PRODUCT	TEMPLATE	Through the BATCH tempate it is
t21	TEMPLATE	BATCH	TEMPLATE	

Figure 26 - Term Packaging

3.iii) Term Aggregation: Again, according to the three-subdomain analyzed in this case, the natural aggregation to perform will be around their corresponding main terms: SAMPLE, PRODUCT, BATCH. For demonstration purposes, the most relevant terms regarding the subdomain PRODUCT are listed in Figure 27, PRODUCT, PRODUCT_VERSION and PRODUCT_TYPE.

3.iv) Term Aggregation Glossary: Additional description regarding the role of the main aggregating term PRODUCT is then placed in the adjacent field, framing the hierarchy of the aggregate and its inner semantics (Figure 27).

Step 1 -Terminology creation			Step 3 - Terminology packaging & aggregation	
1.i)			3.iii)	3.iv)
Id	Term	Origin	Aggregation	Description
t16	PRODUCT	PRODUCT	PRODUCT	Aggregation for subdomain PRODUCT_TYPE
t17	PRODUCT_VERSION	PRODUCT	PRODUCT	
t18	PRODUCT_TYPE	PRODUCT	PRODUCT	

Figure 27 - Term Aggregation

Finally, in step-4 Terms association, for the task of defining relations between the different terms in the global ontology, which have direct or inferred associations according to the taxonomy previously defined, where the former packaging and aggregation tasks already established a type of association between them. Additionally, functional and non-functional relations between terms, in this particular case the ones derived from the table relations in the dataset provided and the ones inferred from the elicited requirements documentation, are taken in consideration on this step.

4.i) Direct Associations: Three direct associations are naturally identified for the subdomains involved, SAMPLE<>BATCH<>PRODUCT. These originate from the direct relations of their master tables in the available dataset (Figure 28).

4.ii) Inferred Associations: Although not directly related terms, the TYPE of a SAMPLE is explicitly related to the LOCATION associated to the BATCH, either of a production unit, a laboratory or warehouse delivery. Another indirect relation is the one existing between the BATCH TEMPLATE and the PRODUCT TYPE, as already identified in the previous step of packaging (Figure 26). A snapshot of a partial ontology for the analyzed domain is presented in Figure 28.

Step 1 -Terminology creation			Step 4 - Terminology association	
1.i)			4.i)	4.ii)
Id	Term	Origin	Direct	Infered
t1	SAMPLE_NUMBER	SAMPLES	BATCH	
t2	SAMPLE_TYPE	SAMPLES		LOCATION
t16	PRODUCT	PRODUCT	BATCH	
t18	PRODUCT_TYPE	PRODUCT		TEMPLATE
t19	BATCH_NAME	BATCH	PRODUCT; SAMPLE	
t20	LOCATION	BATCH		TYPE
t21	TEMPLATE	BATCH		TYPE

Figure 28 - Direct and inferred association of terms

Interest surrounding the task of building ontologies, especially in what regards its use within information systems development, has been increasing lately, pushing towards the proposal of methods to help analysts in this endeavor. In addition, the quest for agile methods aimed at restricted domain ontologies has also gained momentum as opposed to traditional heavy-duty and highly-disciplined methods for complete ontologies.

Recently proposed methods as UPON-Lite (De Nicola & Missikoff, 2016) and SAMOD (Peroni, 2018) have made relevant contributions both to the theoretical and practical communities around these topics, with proposals based on simple steps within iterative processes. One of their major strengths is the fact that they are based in previous established methods, with several years of work in the development of ontologies.

As far as this work is concerned, we developed an automation of the extractions from the various enterprise data sources in order to associate them with a catalog of terminologies that we defined based on the 4SRS-Onto method we have just detailed. This association is made through mapping in a graph database technology, allowing the terms of the different sources to be related and making uniform information available. This standardization is intended to allow the different information systems in the company to interoperate. For this, an architecture is developed that integrates different technologies. Initially, the datasets are loaded into a cluster, and later, using an integration platform (see chapter 6), we define a pipeline where we establish the connections between the graph database and the cluster.

4.4 Conclusions

The construction of an ontology using a method inspired by techniques used in information system development, and more specifically in agile approaches, is gaining momentum in both the research and practical communities. Following the latest trends in the topic of agile ontology construction and drawing on our long experience with the 4SRS information system development method, we proposed a new variant of the latter, 4SRS-Onto.

Thus, and since the original 4SRS method has been used within the work on Requirements Engineering at the CCG, its transformation and adaptation to the context of ontologies is idealized. The original 4SRS is based on four main steps, and each of these main steps is subdivided into microsteps. When the adaptation to 4SRS - Onto is made, the basic structure is maintained and continues to be composed of four main steps. The variation is noticed at the level of each of the steps since in this variant (4SRS-Onto) the focus is on the derivation of terminologies and not on the derivation of objects (original 4SRS).

Thus, this new method addresses the development of an ontology for a specific domain at the early stage of the development process of a full-fledged information system. The detailed specification of the method and a demonstration of its applicability within several live case studies in industrial contexts provide a clearer understanding of its inner workings, leaving room for future analysis, comparisons, and developments regarding similar methods and proposals.

While the present demonstrations are restricted to specific domains, this iterative work with the method continues to cover the full spectrum of current projects, paving the way for evolutions to meet future challenges.

4.5 References

Azevedo, S., Machado, R. J., & Maciel, R. S. P. (2012). On the Use of Model Transformations for the Automation of the 4SRS Transition Method. In M. Bajec & J. Eder (Eds.), *Advanced Information Systems Engineering Workshops* (pp. 249–264). Springer Berlin Heidelberg.

Berners-lee, B. T., & Hendler, J. (2001). *The Semantic Web*. *21*.

Bosch, J., & Molin, P. (1999). Software architecture design: evaluation and transformation. *Proceedings ECBS'99. IEEE Conference and Workshop on Engineering of Computer-Based Systems*, 4–10.

Bragança, A., & Machado, R. J. (2005). Deriving software product line's architectural requirements from use cases: An experimental approach. *2nd International Workshop on Model-Based Methodologies for Pervasive and Embedded Software, Rennes, France*.

Carlos Salgado, Machado, R. J., & Maciel, R. S. P. (2019). 4SRS-SoaML Method for Deriving a Service-Oriented Architecture From Use Cases Within a SPEM Approach. In *Modeling Methods for Business Information Systems Analysis and Design* (p. 22).

De Nicola, A., & Missikoff, M. (2016). A lightweight methodology for rapid ontology engineering. *Communications of the ACM*, *59*(3), 79–86.

Dermeval, D., Vilela, J., Bittencourt, I. I., Castro, J., Isotani, S., Brito, P., & Silva, A. (2016). Applications of ontologies in requirements engineering: a systematic review of the literature. *Requirements Engineering*, *21*(4), 405–437.

Feilmayr, C., & Wöß, W. (2016). An Analysis of Ontologies and their Success Factors for Application to Business. In *Data & Knowledge Engineering* (Vol. 101).

Fernandes, J. M., & Machado, R. J. (2001a). From use cases to objects: an industrial information systems case study analysis. In *OOIS 2001* (pp. 319–328). Springer.

Fernandes, J. M., & Machado, R. J. (2001b). System-level object-orientation in the specification and validation of embedded systems. *Symposium on Integrated Circuits and Systems Design*, 8–13.

Fernandes, J. M., Machado, R. J., & Santos, H. D. (2000). Modeling industrial embedded systems with UML. *Proceedings of the Eighth International Workshop on Hardware/Software Codesign*, 18–22.

Ferreira, N., Santos, N., Machado, R. J., Fernandes, J. E., & Gasević, D. (2013). A V-model approach for business process requirements elicitation in cloud design. *Advanced Web Services*, *9781461475*, 551–578.

Ferreira, N., Santos, N., Machado, R. J., & Gasevic, D. (2012). *Derivation of Process-Oriented Logical Architectures: An Elicitation Approach for Cloud Design. BT - Product-Focused Software Process Improvement - 13th International Conference, PROFES 2012, Madrid, Spain, June 13-15, 2012 Proceedings* (pp. 44–58).

Gasevic, D., Djuric, D., & Devedzic, V. (2006). Model Driven Architecture and Ontology Development. In *Model Driven Architecture and Ontology Development*.

Jacobson, I. (1993). *Object-oriented software engineering: a use case driven approach*. Pearson Education India.

Machado, R.-J., Fernandes, J., Monteiro, P., & Rodrigues, H. (2006). *Refinement of Software Architectures by Recursive Model Transformations*.

Machado, R. J., Fernandes, J. M., Monteiro, P., & Rodrigues, H. (2005a). *Transformation of UML Models for Service-Oriented Software Architectures*. 173–182.

Machado, R. J., Fernandes, J. M., Monteiro, P., & Rodrigues, H. (2005b). Transformation of UML Models for Service-Oriented Software Architectures. *12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'05)*, 173–182.

Peroni, S. (2018). *SAMOD : an agile methodology for the development of ontologies*. 1–11.

Pinto, H. S., & Martins, J. P. (2004). Ontologies: How can They be Built? *Knowledge and Information Systems 2004 6:4*, *6*(4), 441–464.

Salgado, C., Machado, R.-J., & Maciel, R. S. (2013). *Using Process-Level Use Case Diagrams to Infer the Business Motivation Model with a RUP-Based Approach*.

Salgado, C., Teixeira, J., Santos, N., Machado, R.-J., & Maciel, R. S. (2015). A SoaML Approach for Derivation of a Process-Oriented Logical Architecture from Use Cases. In *Lecture Notes in Business Information Processing* (Vol. 201).

Santos, M. Y., & Machado, R. J. (2010). On the derivation of class diagrams from use cases and logical software architectures. *Proceedings - 5th International Conference on Software Engineering Advances, ICSEA 2010*, *August 2015*, 107–113.

Santos, N., Rodrigues, H., Pereira, J., Morais, F., Martins, R., Ferreira, N., Abreu, R., & Machado, R. J. (2018). *Specifying Software Services for Fog Computing Architectures using Recursive Model Transformations*.

Santos, N., Teixeira, J., Pereira, A., Ferreira, N., Lima, A., Simoes, R., & Machado, R. J. (2015). A demonstration case on the derivation of process-level logical architectures for ambient assisted living ecosystems. *Ambient Assisted Living*, *June*, 103–139.

Santoso, H. A., Haw, S. C., & Abdul-Mehdi, Z. T. (2011). Ontology extraction from relational database: Concept hierarchy as background knowledge. *Knowledge-Based Systems*.

CHAPTER 5

ON THE DESIGN OF ONTOLOGIES

Summary: This chapter addresses the issues associated with ontology construction, specifically the various phases that comprise this procedure. Thus, we take into account phases that we consider fundamental to the ontology building process, such as requirements gathering, terminology cataloging, the construction of a synonym catalog, terminology categorization, and ontology schema construction. In this way, and considering these phases, we have conceived and designed a strategy to address the second objective identified in this dissertation. Thus, we propose an ontology approach that permits the construction of ontologies in three phases.

CHAPTER TABLE OF CONTENTS

5.1 Introduction....129
5.2 A Processual View of Ontologies Design....130
5.3 Demonstration Cases....140
5.4 Conclusions....163
5.5 References....164

—— This page is intentionally left blank ——

CHAPTER 5

ON THE DESIGN OF ONTOLOGIES

"I made the decision to pretend that all the things that had so far entered my mind were no more true than the illusions of my dreams."

– René Descartes

5.1 Introduction

Lately, there has been a lot of talk about ontologies and how they are beneficial when it comes to handling and organizing data. But what are ontologies anyway? Ontologies are a data model that represents a set of concepts within a domain and the relationships between them. Basically, we can characterize them as an information organization technique, mainly regarding the formal representation of knowledge (Guizzardi, 2007).

They are usually created by experts and, since their structure is based on the description of concepts and the semantic relationships between them, they allow generating a formal and explicit specification of a shared conceptualization. And why have they become so important? As it is well know, the volume of data we deal with today - on the Internet and especially at the organizational level - is very vast and diverse, and the trend is for it to become larger and larger.

Due to these high growth rates it has become necessary to use techniques that allow a better and more effective treatment and organization of data, acting mainly in its selection, processing, retrieval and dissemination - the ontologies. Ontologies can be defined by creating catalogs of terminologies, glossaries or dictionaries, by classification or categorization through taxonomies or from concepts and their relationships, with great focus on the use of semantic networks (Calero *et*

al., 2006). The use of ontologies results in a unification of the language or terminologies used in order to ensure a better understanding and modeling of the ecosystem in question.

The use of ontologies has as main advantages the possibility of communication between different types of machines, creating an interoperable network about a particular knowledge; and the formalization, which is related to the specification of the ontology, allowing the elimination of contradictions and inconsistencies in the representation of knowledge.

In terms of knowledge representation, ontologies form a consensual vocabulary that allows the representation of knowledge of a specific domain at a high level of abstraction, thus ensuring a potential for reuse. On the other hand, the most common problem that usually arises is: "which is the most correct methodology to use?" Just as in software development, in ontology design there are also several associated methodologies, some more traditional and more exhaustive, and others more recent that have an agile basis.

We have been developing our own approach and implementing it in CCG projects, with applicability in other areas and industry sectors, this approach is flexible and adjustable, taking into account the specificity of each project.

5.2 A Processual View of Ontologies Design

In order to begin the development of the method to design ontologies, a study is made about the methodologies currently available to apply to this problem. According to Haruhiko Kaiya and Motoshi Saeki (Kaiya, 2005), one of the objectives of requirements analysis is to develop a high quality requirements specification document. Thus, there are several methods to achieve this goal, for example, requirements analysis methods, scenario analysis, use case modeling techniques, and others.

One of the most crucial problems with automating requirements analysis is that requirements documents are usually written in natural language, for example, English. This way, the authors then propose a software requirements analysis method based on the domain ontology technique, where we can establish a mapping between a software requirements specification and the domain ontology that represents semantic components. Their ontology system consists of thesaurus and inference rules, and the thesaurus part comprises domain-specific concepts and relationships,

suitable for semantic processing. It allows requirements engineers to analyze a requirements specification with respect to application domain semantics.

The interdependence of requirements is being studied extensively, with greater interest in software engineering research. However, according to (Soomro *et al.*, 2014), existing tools and techniques have not adequately characterized and visualized interdependent relationships between requirements. These authors present an ontology-based representation of interdependencies between requirements and formal graphical notation for proper visualization of interdependencies between requirements. Thus, they argue that there is evidence to point out that the ontology-based approach is the best technique for managing requirements interdependence, that is, the diagrammatic representation of requirements interdependence will improve software quality and reduce failure rates.

An ontology, in the general idea, should provide appropriate workflows and the possibility to design process sequences for ontology development and maintenance defined by specific methods. Ontology building is primarily a knowledge integration process. This means that albeit, in theory, handcrafting an entire ontology by hand is possible, in practice the only feasible way to build a reasonably complex ontology is via extracting information from other sources (Wróblewska *et al.*, 2012).

Considering these aspects and the analysis done in the existing literature, one can verify that there are yet few examples of applications of agile methods for the development and building of ontologies. Two of the most relevant are the Unified Process for Ontology (UPON), an agile ontology engineering method intended to place end users at the center of the process, and the Simplified Agile Methodology for Ontology Development (SAMOD), a methodology organized in small steps within an iterative process that focuses on creating well-developed and documented models.

In a more agile approach, derived from the full-fledged UPON methodology, the UPON Lite method (De Nicola & Missikoff, 2016) assures a rigorous, systematic approach but also reflects an intuitive nature. This version is developed for a wide base of users (typically domain experts) without specific ontology expertise. It is organized as an ordered set of steps, each releasing a self-contained artifact readily available to end users. Moreover, it is progressive and differential, with each new step using the outcome of the preceding step, providing well-defined enrichment to it.

This methodology, according to its authors (De Nicola & Missikoff, 2016) is designed to enhance the role of end users and domain experts without the specific experience of the ontology engineer. UPON Lite is characterized by three main aspects of ontology engineering: a user-centric approach designed to be easily adopted by non-ontological experts, thus minimizing the role of ontology engineers; a socially oriented approach where multiple stakeholders play a central role; and an intuitive six-step ontology engineering process supported by a "familiar" spreadsheet tool.

Another interesting approach, the Simplified Agile Methodology for Ontology Development (SAMOD) (Peroni, 2018), is developed and organized in small steps within an iterative process, focusing on creating well-developed and documented models starting from exemplar domain descriptions. It uses significate exemplars of data to produce ontologies that are always ready to-be-used and easily understandable by humans in an agile way, spending the least of efforts.

Thus, after having studied and analyzed two distinct ways that allow the construction of ontologies, a first version of the approach that intends to be implemented is outlined. This approach arises from the need identified by the analysis performed on the typology of projects that have been developing within the CCG. Since these are R&D projects, it is necessary to have an agile intervention in each of the phases of the project life cycle in order to guarantee the deliveries previously defined.

Thus, to understand what characterizes the company it is necessary to identify and analyze the organizational context, as well as the sector of activity that characterizes the company. Afterwards, we must focus on the analysis of the company's internal processes, documented activities, and tasks performed in each department, as well as the actors that carry them out. In this stage, the Unified Modelling Language (UML) based use-cases are focused, as they allow to observe the organization as a whole and, together with the company, decide which area or areas make sense to be targeted through ontological implementation.

Next, the analysis must be oriented to the existing data model or reference architecture within the domain of interest. If, at least, one of these artefacts can be found, it is analyzed and detailed, while scenarios are built as a way to validate whether the ontological layer of the model covers the project's intervention areas. Otherwise, or if some specific requirement is defined by the stakeholder, a proper data model is designed to fit the needs, which is then used as a starting

point to establish an ontological database schema that is populated according to the client's specifications and terminology.

In the end, we proceed with the integration of this ontological database into a visualization tool, wherein the stakeholder can apply filters, edit the database, or add new terminology and relationships in a more user-friendly way. Figure 29, depicts the aforementioned process, which has been supporting several other projects.

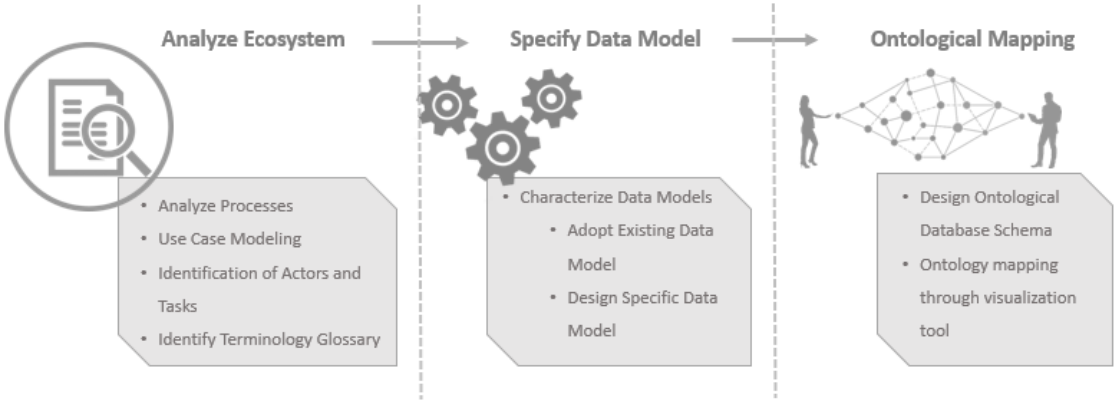


Figure 29 – Ontological Approach for Information Systematization version 1

As previously said, the primary issue that ontologies aim to address is the lack of interoperability and/or lack of internal knowledge that many businesses continue to encounter. Within organizations, information systems encompass all information flows between people, processes, and machines, including written and verbal interactions, constituting an organized system for gathering, arranging, storing, and transmitting information.

Because of all these information flows, there must be a means to standardize how a domain is handled inside the business. And at this point, obstacles to the process begin to emerge. For all of this to function correctly, there must be a uniform and cohesive business ecosystem in terms of data, information, and its applications; but, in some instances, we are confronted with unstructured data, apps that run in isolation, and excessive paper consumption.

If we want a company to have an interoperable system, we must first equip it with the appropriate resources and processes, as well as conduct a study of existing software applications and their respective terminologies in order to standardize them through a dictionary of synonyms. Ontologies

serve as a solution precisely in this instance, and a new and enhanced version of the approach we have been constructing emerges as a result (Figure 30).

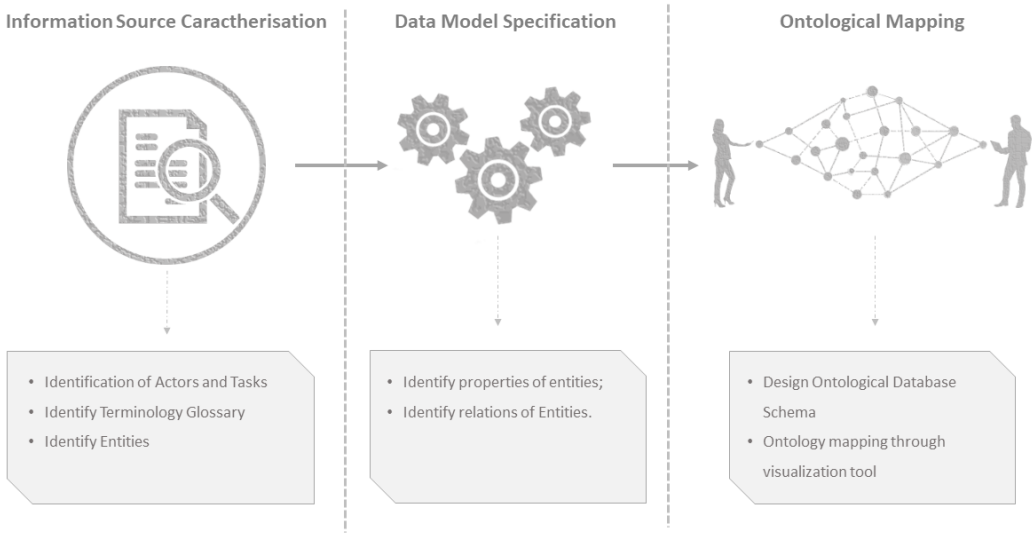


Figure 30 - Ontological Approach for Information Systematization version 2

Attending the state of the art analyzed in chapter 2, namely regarding the concepts of data, information, knowledge and wisdom, we verified that the approach presented in Figure 29 is a little bit misaligned with the vision presented in the DIKW pyramid. In this way, and as a way to align this approach with the concepts of the DIKW pyramid, we understand as necessary to refine our approach giving rise to a new version (Figure 30).

Thus, this new version of the ontological approach to systematize information includes the following phases:

- First phase - Data Characterization;
- Second phase – Information Specification;
- Third phase - Ontological Mapping.

Initially, we should focus on the analysis of the company's internal processes, documented activities, and tasks by each department, as well as the actors that perform them to make it easier to define which data sources to select and analyze (Data Characterization phase). In this phase, after the data sources have been selected, we must begin to index the terminologies present in the various sources, thus building a glossary of terminologies for the domain under study. After we finished cataloging the terminologies, we grouped and categorized the terminologies into Entities.

This phase ends with the definition and construction of a catalog of synonyms, where each terminology previously identified must have a set of synonyms associated.

Next, in the Information Specification phase, the analysis should be oriented towards the existing data model within the domain of interest. That is, in this specification phase we use the detail from the previous phase (Data Characterization) and start identifying properties for each of the entities. After we have defined the properties, we relate the entities by creating a data model. To perform this phase of relating and identifying properties for each entity, we use a UML class diagram.

In the final phase, Ontology Mapping Phase, we use the detailed information from the previous phase (Information Specification) to build an ontology schema using graph database technology. We then proceeded to integrate this ontology database into a visualization tool, in which stakeholders can apply filters, edit the database, or add new terminology and relationships in a more user-friendly way.

In addition to these modifications relative to the prior version, as seen in Figure 31, there are iterations between each phase. These iterations are created so that if the user desires and it is required, he may go back through the phases and make the needed adjustments.

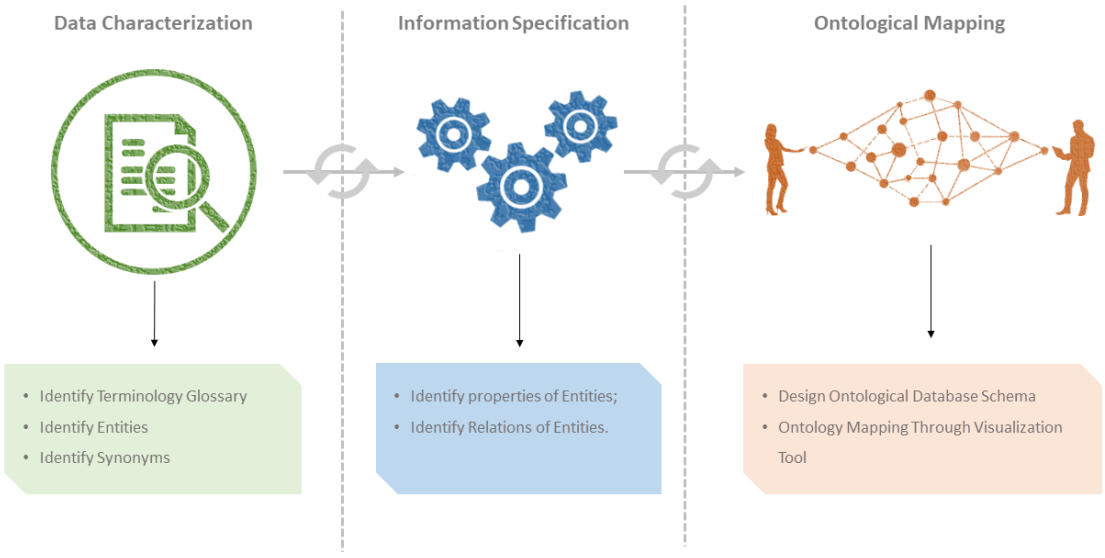


Figure 31 - Ontological Approach for Information Systematization (version 3)

Based on our previous research, background experience, and taking into consideration the contributions presented in chapter 4, Figure 32 proposes a traceable path using an adaptation to the systems development V-Model. This is done by aligning the data characterization, initial and detailed requirements in UML use cases, a definition of the information specification and the data model, and finally the ontological design of the defined schema.

This method follows the V-Model proposed by Machado *et al* to derive a data model aligned with the requirements modelled in Use Cases (Ferreira *et al.*, 2013), more specifically on its left-descending branch. This V-Model left branch proposal uses models in a successive way, where a previous model is inserted into the next one, that is, requirements elicitation includes a definition of the business processes for executing the solution, sequential ordering of features, afterwards modeled in A-type sequence diagrams (Santos *et al.*, 2015) and, finally, modeling in Use Case diagrams.

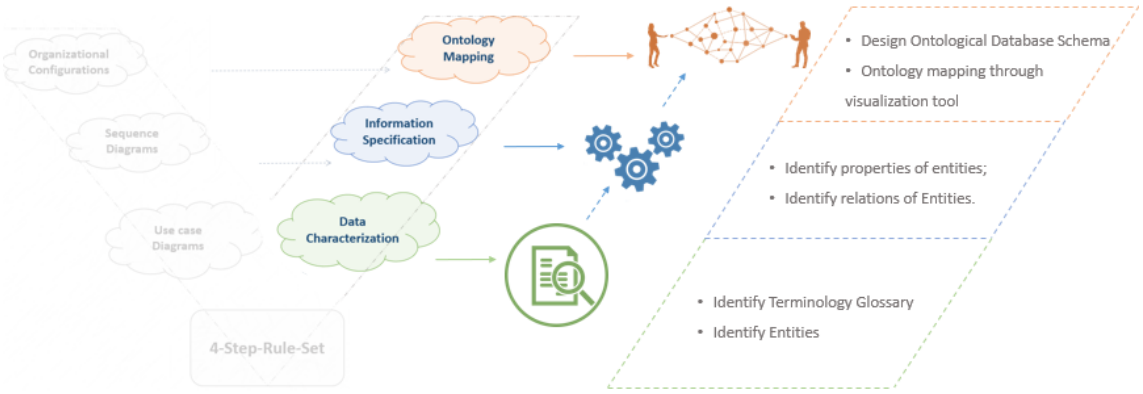


Figure 32 – Ontological Approach for Information Systematization (version 3) included into V-Model

The way to elicit the needs of business processes ("Input from Organizational Configuration") necessary to derive the software requirements (A-type sequence diagrams and UML use cases) is described in previous work [(Santos *et al.*, 2015), (Carlos Salgado *et al.*, 2019), (Santos *et al.*, 2018)].

This section is not concerned with the development of UML use cases. Rather, it should be noticed that requirements are expressed in UML use cases and that the data model and ontology design are then aligned with each of the appropriate phases in the descending branch of the V-model. At the junction of the top-down and bottom-up branches there is the 4SRS-Onto method, which, as described in Chapter 4, enables to catalog terminology by aggregating them trough use cases. It

is designed specifically for complex situations with a high number of use cases, helping to filter, aggregate, and link the future ontology's several components. It is another input to the right ascending branch, where we begin our new proposal, which is structured in three successive phases, as indicated previously.

Below, we describe these three steps in detail illustrating the outcome of each of the phases.

Data Characterization Phase

As far as this case is concerned, the demonstration that will be carried out concerns the right edge of the "V". Thus, the initial iterations down to the vertex of the model result from the application of the V-Model + 4-Step Rule Set (4SRS) method. As discussed in other articles [(Ferreira *et al.*, 2013), (Santos *et al.*, 2015), (Carlos Salgado *et al.*, 2019), (Salgado *et al.*, 2015)], 4SRS allows for the derivation of a logical architecture. It is precisely from this moment that we start the application and demonstration of this method (4SRS Method for Ontological Design).

Speaking specifically about the "Data Characterization" phase, at first, it is necessary to characterize the actors as well as the tasks in which each one participates. This description will help to understand in detail the field of action as well as the details of the tasks that are performed. Afterwards, we proceed to the identification and construction of a glossary of terminologies.

Each terminology glossary identifies a specific domain, as such should be constructed through an analysis in the context of the project, in order to detail the identified terms through a description. These terms, as previously mentioned, result from the intersection between the "Use Case" phase and the logical architecture resulting from the application of the 4SRS. This terminology analysis and cataloging should be carried out in as much detail as necessary, and it can be deepened or extended at any time in a future iteration of the method.

For a better understanding and contextualization of the identified terms, at this stage we define a table-like structure where each term is characterized by the following set of properties:

- ID: Term reference identification. Identification must follow the next rule:
  - Information model terms: TMIx, where "TMI" represents the information model term and "x" indicates the term number. Example: TMI1.
- Terminology: Term name.

- Description: detailed description of the identified term.
- Term Source: this property allows us to identify the sources of the identified term, manufacturing order, datasheets, Subcontractors, Contractors, among others.
- Interfaces: This property is composed of the End Users, which correspond to the multiple user profiles of the platform to be developed, and also the External Components / Applications, identifying the need to develop interfaces for communication with other systems that already exist or that can be deployed in the future.
- Synonyms: Identify synonyms for each terminology.
- Dependencies: Dependencies allow the identification of: a) Stakeholders who are directly or indirectly interested in and/or related to the identified terms; b) what data sources are needed to collect (serve as input) the data that interact with the defined term.
- Domain: Information Owner.
- Security Level: Public/Private

Information Specification Phase

In this Information specification phase, previously identified terminologies are used, and the main objective of this phase of the method is to map the terminologies through a UML class diagram. When mapping the terminologies, we must associate each one of them with the corresponding synonyms that were listed in the previous phase.

Thus, to elaborate the mapping, we follow the design rule of a class diagram composed by:

- Class (the class itself, this element is used when we want to visually demonstrate the class in the diagram). Association (Association - connector without tips - It is a type of relationship used between classes. Applicable to classes that are independent (they live without dependence on each other), but that at some point in the ontology may have some conceptual relationship).
- Generalization (Heritage - connector with an arrow at one end - It is a type of relationship where the generalized class (where the “arrowhead” of the connector is) provides resources for the specialized class (heir). It adopts more advanced concepts, that is,

everything that the mother (generalized) class has, the daughter (specialized) will have. We are talking here about a concept of property inheritance).

- Compose (Composition - connector with a "diamond" hatched at the tip - It is a type of relationship where the composite class depends on other classes to “Exist”. For example, the “Car” class has a composition with the “Motor” class. Without the "Motor" class, the "Car" class cannot functionally exist) and Aggregate (Aggregation - connector with a leaked "diamond" on the tip - It is a type of relationship where the aggregate class uses other classes to "exist", but you can live without it. For example, the “Car” class has an aggregation with the “Roof” class. Without the “Rooftop” the “Car” class can exist).

As mentioned before, considering the mapping of the class diagram, we now focus on the entities that compose it. Thus, an analysis and description of each of the previously identified classes is performed in order to allow the definition of the schema for the ontological database.

Ontological Mapping Phase

The last phase of the method, called Ontological Mapping, aims to map the schema of the ontological database. For that, the information existing in the previous phase is used, namely the class diagram and its constituents. This mapping should be done using a graph database technology, and as a rule, we have been using two possibilities: Neo4j or Cosmos DB Azure Gremlin API. Our choice has fallen mainly on Neo4j, since it has open source versions and provides very interesting features such as importing .csv and .xml files, allowing us to automate and facilitate the mapping and integration of prototypes with the client.

In addition to this, we often use a plugin, Bloom, which we configure within Neo4J and which allows us to manage the mapping, including creating, editing and deleting nodes and relations in the database. An example of a mapping in Neo4J integrated with a backoffice dashboard through its Bloom plugin can be seen below in Figure 33. An advantage of this component is its user-friendly interface, which allows it to be used by any user without scripts or queries.

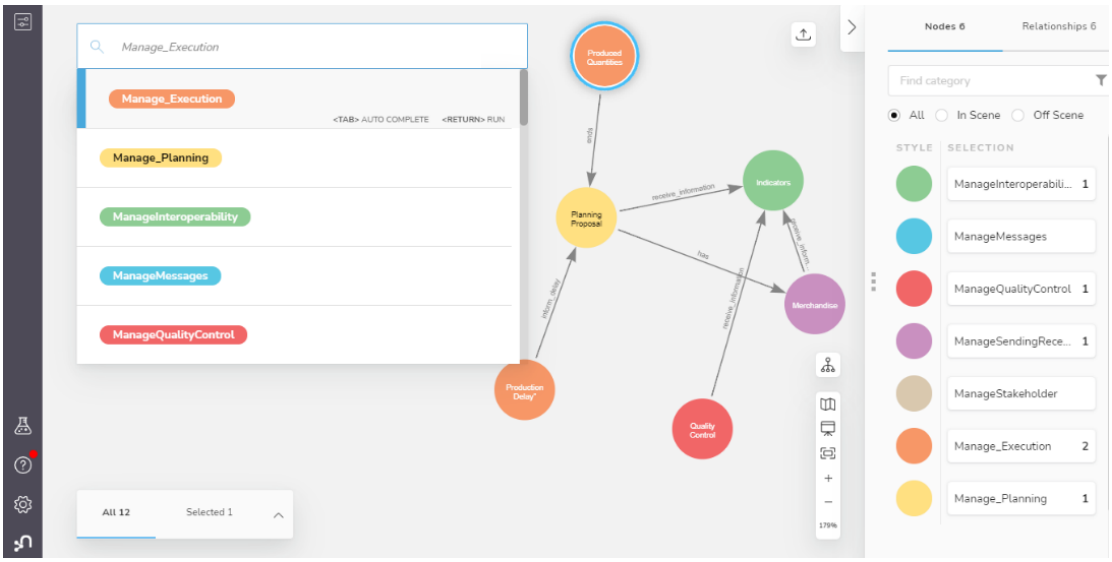


Figure 33 - Ontological Mapping Schema. - Bloom Interface

5.3 Demonstration Cases

In this section, contributions to ontology construction using the methodology we defined and described earlier in this chapter (section 5.2) are presented. These contributions are used and demonstrated in actual projects (CCG R&D projects). This contribution is applied to software (PHC and F3M) and smart cities (CityCatalyst) industry demonstration cases. In the case of PHC, the objective is to develop a virtual assistant that enables users to interact verbally with PHC software in order to access its various features. To achieve this objective, the entire software structure is mapped, and a synonyms catalog is assigned to each terminology, creating an ontology for PHC's management software. In the case of F3M, the objective is to map, using graph database technologies, the existing terminologies in the information systems of textile companies in order to develop a network of knowledge and interdependencies between the various terminologies used in the industry. Regarding the smart cities demonstration case, the objective is to compile a catalog of municipal terminologies and associate them with one another. Using graph database technologies, this terminology association is created to construct an ontology for the smart cities industry.

The Project "PHC - Voice Interaction Framework"

For the vast majority of today's computer workstations, the main interfaces for human information input are the keyboard and mouse peripherals. The introduction of additional information input interfaces (such as audio) can prospectively increase the productivity of users interacting with an information system (IS). Technologies such as voice activity detection (VAD), speech-to-text (S2T) and an Ontological Component (OC), which map the structure and catalog data of interest, have the potential to unlock advanced user-system interaction through human-intrinsic communication means - also known as Natural User Interfaces (NUI) - in particular, voice.

Voice Activity Detection is a term used for signal processing methods that allow the detection of speech contained in an audio signal. In speech processing, the discrimination between speech and non-speech is, still to this day, a relevant issue affecting web applications for real-time speech recognition. Speech processing algorithms are often significantly demanding of computing resources and, as speech is naturally discontinuous; the inclusion of VAD methods in these algorithms is a design strategy to alleviate unnecessary processing (Barry & Crowley, 2012). In the proposed framework, the VAD implementation integrates a Front-End module, where the audio signal acquisition is performed from a microphone device. The signal is analyzed for speech detection, segmented, and finally, speech only signals streamed to a Back-End processor. The implemented VAD must be capable of detecting speech for typical background noise, a feature that is commonly referred to as noise-robust.

Nowadays, speech recognition (SR) implemented into voice assistants represents a supplementary input source for devices like mobile phones, tablets and virtual assistants, allowing user interaction with these devices. Benefiting from this widespread demand, SR technology has achieved a level of maturity justifying its implementation in web systems as an additional input source of information.

Relative to the Ontological Component, Ontologies can be considered a data model that represents a set of concepts within a domain and the relationships between them. Fundamentally, we can characterize it as a technique for organizing information, especially with regard to the formal representation of knowledge. These are generally created by specialists and, as their structure is based on the description of concepts and the semantic relationships between them, they make it possible to generate a formal and explicit specification of a shared conceptualization.

In this demonstration case, it is intended to build a virtual assistant that allows a user to interact with the ERP through voice commands. Thus, in Figure 34, it is demonstrated how the concepts of VAD, S2T and OC are used and how they relate to each other in order to build this system.

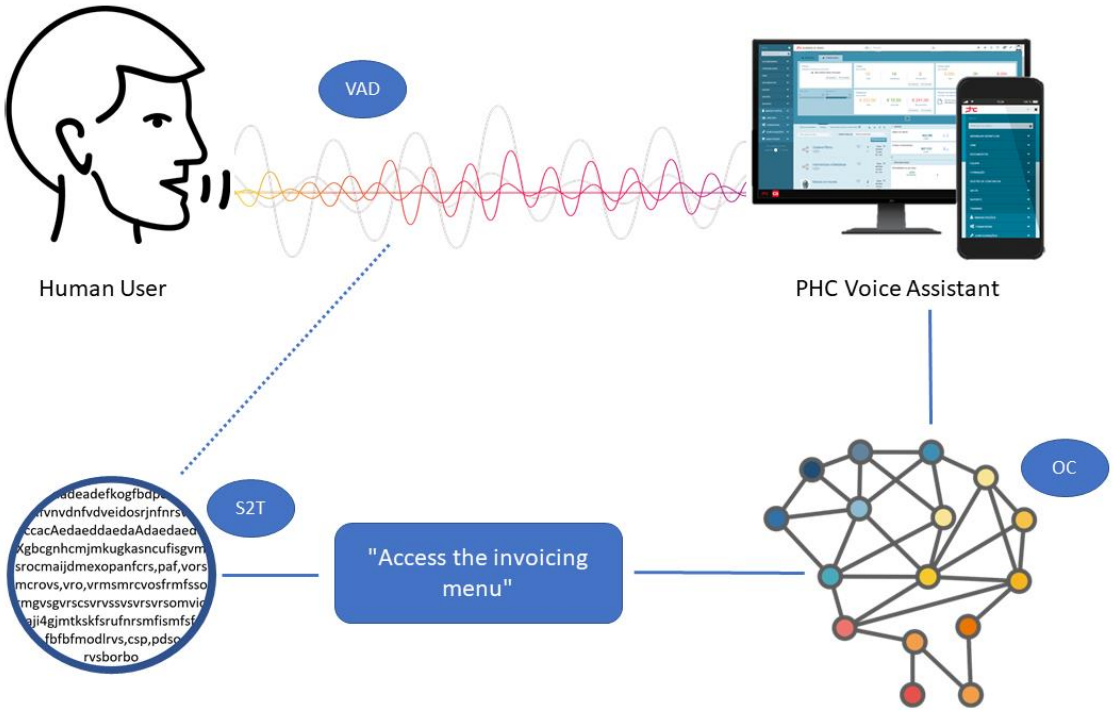


Figure 34 - PHC Voice Assistant

Considering that the focus of this doctoral work concerns Ontologies, only the ontology component (OC) is described in detail in this demonstration case. In this context, an analysis and demonstration of the adopted approach are performed. To this end, an ontological architecture is defined where the entire ERP is mapped, specifically all of its menus, features, and expressions involved. This ontological systematization allows to catalogue the terminologies inherent to an ERP as well as the attribution and definition of synonyms for expressions and functionalities associated.

The rapid progress in efficiency that has been seen in the three afore-mentioned technologies and their combination, create the opportunity for an increasingly natural and faster user interaction with potentially complex IS, therefore permitting an improvement in productivity.

According to the different solutions envisioned for the diverse parts of the proposed approach, a framework connecting the three main parts has been discussed among the team and validated

with the customers involved. Initially a trigger would be set at the client ERP, detecting a valid voice command, which would then be interpreted and processed for transformation into a text command. Next, the resulting text is semantically validated and mapped through the ontological database in order to, in case of success, return a valid command for the ERP within its present context. So, the proposed framework (Figure 35) is then divided into three modules: 1) Voice Activity Detection (VAD) Module; 2) Speech Recognition Module 3) and Ontological Module, which is subsequently described.

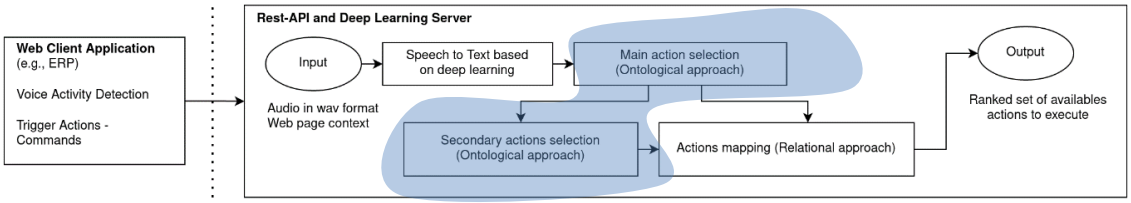


Figure 35 - Voice Interaction Framework architecture proposed

As described above, the Ontological Component (or ontological layer) is a developed module that comprises the entire process related to the semantic mapping of the ERP and its implementation in companies is always related to the context in which the company operates. Thus, to understand what characterizes the company, in first place it is necessary to identify and analyze the organizational context, as well as the sector of activity that characterizes the company.

Afterwards, we must focus on the analysis of the company's internal processes, documented activities and tasks performed in each department, as well as the actors that carry them out. Thus, it is possible to observe the organization as a whole and, together with the company, decide which area or areas make sense to be targeted through ontological implementation. All this contextualization is very important to understand the range of products that exist and all their specificities, something that would not be possible just observing the functioning of the ERP in particular.

Next, the analysis should be oriented towards the specification of the data model within the domain of interest. In this specific case, the PHC software is analyzed in order to identify its data model and facilitate the identification of the hierarchical structure of the software. By hierarchical structure, we mean the existing dependencies between the menus, functionalities, and

expressions, which are the possible actions to execute. This step is done by building a class diagram in order to identify each of the software's terminologies and relate them to each other. This step is extremely important because it is used as a starting point to establish an ontological database schema that is populated according to previously identified specifications and terminology.

In the end, we proceed with the integration of this ontological database into a visualization tool, where the stakeholder can apply filters, edit the database or add new terminology and relationships in a more user-friendly way. Figure 36, depicts the aforementioned process, which has been supporting several other projects.

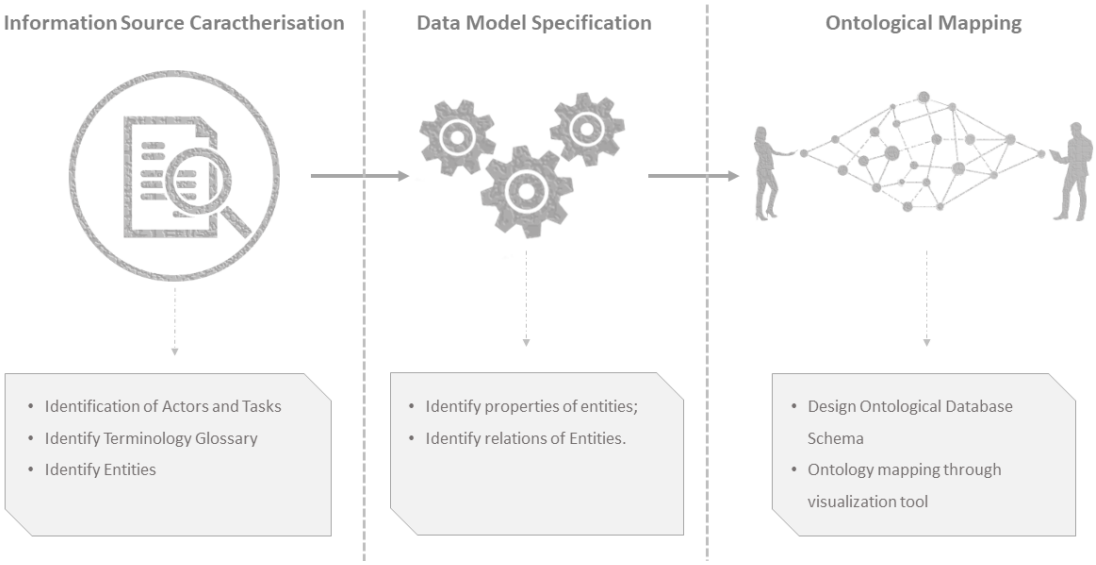


Figure 36 - Process Used in Ontology Design.

The ontological processing and mapping are performed by one of the most used graph database engines, Neo4J (Zhu *et al.*, 2019), which facilitates the development of ontological components. The language used by this tool is Cypher (Query Language) and its syntax allows to combine node patterns and relationships in graphs, visually and intuitively.

The component that works offline concerns the ontological mapping of the entire context present in the ERP. The process to be carried out to the ontological mapping is dependent on the Speech

Recognition module. With respect to the generation of expressions and words that “Speech to Text" makes available, a mapping of these same expressions is made, associating them to a range of synonyms. The mapping is done in a hierarchical way, that is, in order to ensure that all expressions that are present in the same menu can have access to common commands.

The ontological database contains the mapping of the entire ERP interface, namely its menus, the existing functionalities in each one of the menus and even the expressions that each page of functionalities contains. In order to overcome the use of different words for a same meaning and compose all this mapping, it became necessary to add a set of synonyms for each of the expressions existing in the ERP. This allows for the optimization of the search engine and assure that whatever search is carried out, by voice command, the result will be returned correctly. On the one hand, this method avoids the existence of expressions with exactly the same name and, on the other hand, it allows for a better and faster execution of each command. Thus, the objective presented here is the development of a virtual assistant integrated in an ERP, enabling the end user to search and fill out forms through voice commands.

The ontological database proposed in this work has very specific characteristics where it is intended to organize the database through a combination of language, geography and ERP version (PT-Portugal-V1, ES-Spain-v1 or EN-English-V1). In addition, each of these databases must have a hierarchical structure with the following nodes: Module (page); Functionalities; Expressions and Synonyms. In addition to these structural features of the ontological database, nodes are composed of properties or attributes such as Name, ID, CommandID (this one with the intention of redirecting to a script or url in ERP). The "Expressions" nodes also have the "ERP Product Edition" and "Geography" (country where ERP is used) properties available. In Figure 37 is possible to visualize an excerpt of the code, in Cypher (Neo4J), that allows the creation of the nodes related to the ERP Pages, the creation of the nodes that represent the Functionalities, and also to assign the functionalities to the CRM page.

```
5   //Create Pages PHCVoiceFramework
6   CREATE (colaborador :Page {name: 'employee'})
7   CREATE (vencimentos :Page {name: 'salaries'})
8   CREATE (contabilidade :Page {name: 'accounting'})
9   CREATE (imobilizado :Page {name: 'real estate'})
10  CREATE (crm :Page {name: 'crm'})
11
12
13  //Create Functionalities of CRM Page
14  CREATE (clientespotenciais :Funcionalidades {name: 'leads', stamp: 'script'})
15  CREATE (negocios :Funcionalidades {name: 'businesses', stamp: 'script'})
16  CREATE (vendedores :Funcionalidades {name: 'salespersons', stamp: 'script'})
17  CREATE (contactos :Funcionalidades {name: 'contacts', stamp: 'script'})
18  CREATE (pessoas :Funcionalidades {name: 'people', stamp: 'script'})
19  CREATE (produtosvendidos :Funcionalidades {name: 'products sold', stamp: 'script'})
20
21
22  //Associate the Functionalities to Pages
23  CREATE
24  (clientespotenciais) -[:funcionalidade_de]->(crm),
25  (negocios) -[:funcionalidade_de]->(crm),
26  (vendedores) -[:funcionalidade_de]->(crm),
27  (contactos) -[:funcionalidade_de]->(crm),
28  (pessoas) -[:funcionalidade_de]->(crm),
29  (stockseservicos) -[:funcionalidade_de]->(crm)
```

Figure 37 - Excerpt of the PHC demonstration case code

Thus, tests are performed on the developed prototype in order to ensure full alignment with the customer's needs. These tests are performed in a real context, where company employees are selected to test the queries on the ontology database. These tests allow us to evaluate the quality of the response times and the quality of the information that is returned to the end user. Finally, tests are also performed to validate the integration between the three modules (VAD, Speech Recognition and Ontology). That said, in the following Figure 38 it is possible to visualize the final result of this mapping.

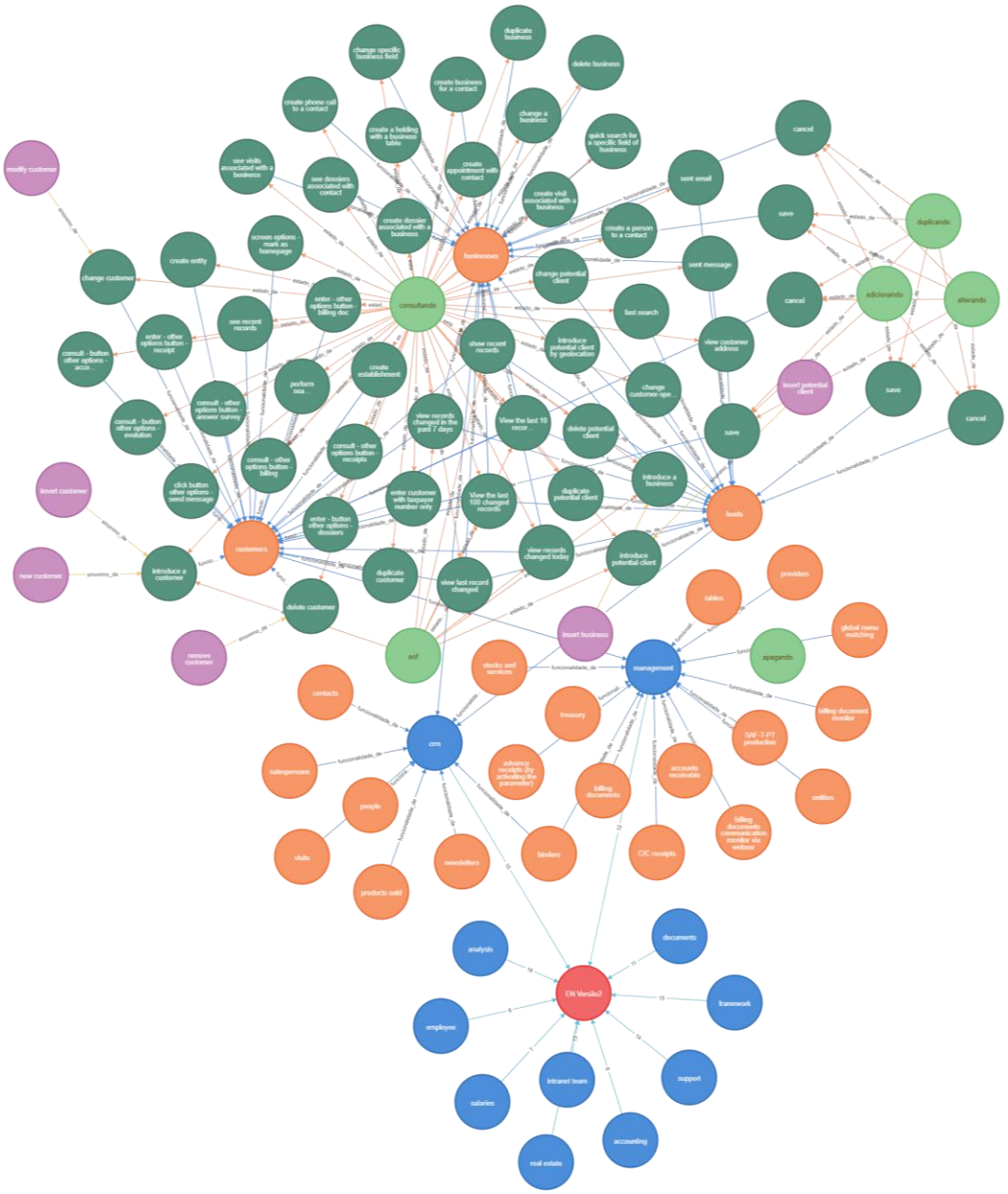


Figure 38 - Excerpt from the ontological mapping performed in Neo4J.

The Project "F3M: Interoperability and Digital Thread domain for a More Competitive Textile Industry"

To face some business challenges in the Textile and Clothing Industry, the development of the Interoperability and Digital Thread domain for a More Competitive Textile Industry (IDT4CTI project - in partnership with F3M) will be supported on a platform based on the Industry 4.0 paradigm, adopting an ontological model that allows loading data models from different ERPs and interoperating different subcontractors' systems and exchanging messages with each other.

As referred earlier, the main problem to tackle with ontologies is the lack of interoperability that organizations still face. Information systems encompass all information flows within organizations, involving people, processes, and machines, including paper and oral communications, constituting an organized system of information recollection, organization, storage, and communication. Due to all these information flows, it is paramount to unify the handling of the different domains inside an organization.

This is where some obstacles to the process begin to appear. For all this to work properly it is necessary to have a uniform and coherent business ecosystem in terms of data and its applications, but in some cases, we are faced with unstructured data, with applications that operate in isolation and, also, with the excessive use of paper. So, if we want a company to have an interoperable system, first we have to provide it with the necessary means and processes, carry out a survey of the existing computer applications and terminologies used so that we can proceed to standardize them through a catalog of synonyms. It is precisely here that ontologies act as problem solvers.

Focusing on this specific scenario, company X (which operates in the textile industry) is growing exponentially and feels the need to become more efficient in terms of its processes and more technologically developed. After the requirements gathering phase, we came across the following reality: the company has two different information systems (Information System 1 and Information System 2), one for managing orders, stock, purchase orders, and related items, and other for everything that is related to invoicing, which do not communicate with each other. We also found that the company's workers still handle several important documents on paper and that they use different terms to describe the same purpose.

In addition to this, Company X subcontracts different types of work, namely garment makers, printers, and companies to apply accessories to textile products. Each of these subcontractors also has its own information system (Information System 3, Information System 4, and Information System 5). Thus, as illustrated in Figure 39 we see that in this context there are 5 or more information systems, all of them different and working independently where each one has defined different terminologies for the textile sector.

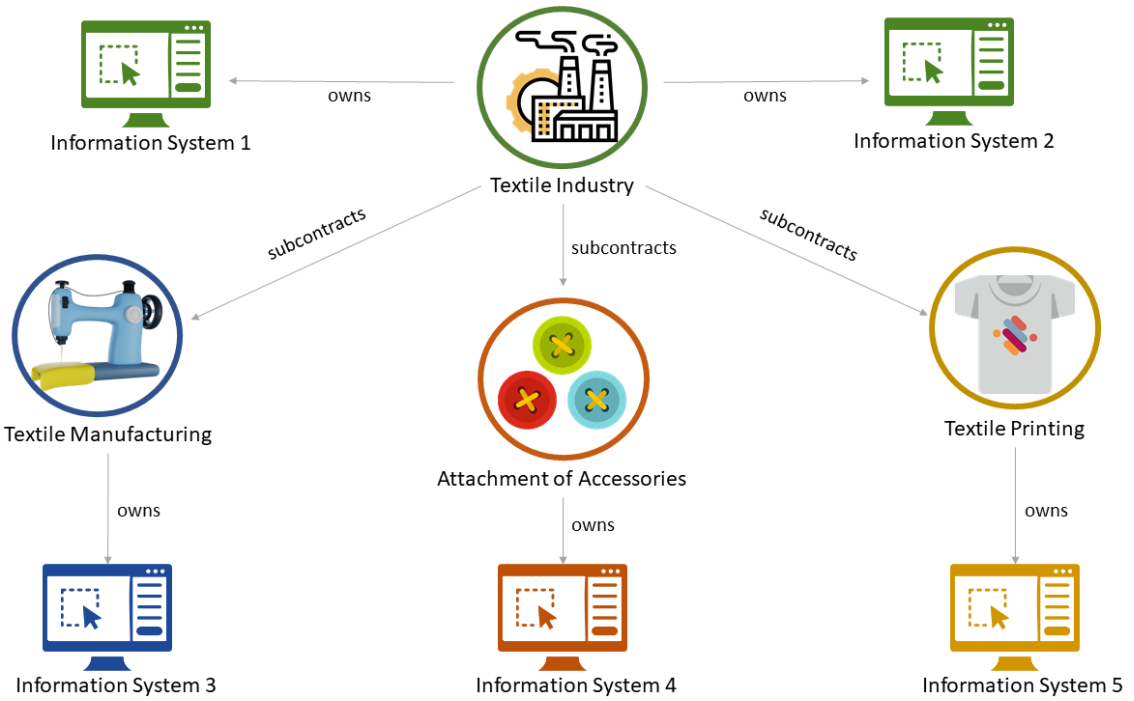


Figure 39 - Representation of the context of the demonstration case scenario

After observing and detailing this demonstration case we have chosen to implement an ontology component in company X. This way it is possible to produce a single terminology catalog containing a list of synonyms mapped to the ontology built. In this way, terminologies such as "purchase order" and "purchase order", which are exactly the same thing but in different languages, are able to be related through ontology mapping, ensuring that every entry or query made in the management software returns the same data, regardless of whether you are looking for one or the other. This eliminates the restriction of employees having to search for only a single term, thus streamlining their work.

On the other hand, company X can begin to interoperate its different software components so that its users can interpret the data from the ontology components uniformly and communicate with each other. In this way, we eliminate the constraints for employees in the financial domain, regarding the manual transcription, for example, of the values of a purchase order entered in the management software to the invoicing software, so that they can invoice it effectively.

Thus, considering this realistic scenario and in a project context with our business partners, we applied the Ontological Approach to Information Systematization that we developed to meet the needs identified above. Next, we will explain in detail this approach already applied and with real

results. In this chapter we also describe the three phases that make up the right-hand side of the model, where the focus is on ontological design, Figure. 40, (Data Characterization, Information Specification, and Ontology Mapping).

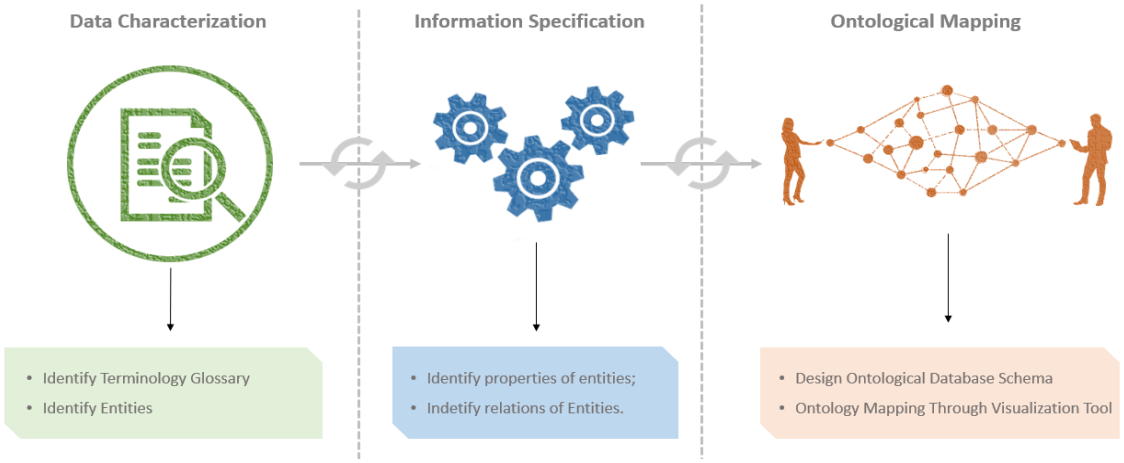


Figure 40 - Characterization of the ascending phases of the right edge of the V-Model for Ontological Design

Regarding the Data Characterization phase, it is important to point out that it is fully aligned with the use case diagrams (UML) modeling phase. The specification phase of information sources aims to define and characterize, based on the use case diagrams, which actors are an integral part of the process we intend to detail, and also catalog the domain terminologies.

The definition of information model terminologies is a recurring task in this phase, where they are initially defined and periodically revised and updated according to the characteristics of the platform/modules to be implemented. This approach to the cataloging and description of the terms to be used by the platform allows us to characterize in detail all the terminologies inherent to the domain. The cataloging and characterization of terms section contains the identification and description of the terminologies to be used. This will define the semantic component of the information model of the technology platform.

For a better understanding and contextualization of the identified terms, at this stage we define a structure where each term is characterized by the following set of properties:

- ID: Term reference identification. Identification must follow the next rule:

- Information model terms: TMIx, where "TMI" represents the information model term and "x" indicates the term number. Example: TMI1
- Terminology: Term name.
- Description: detailed description of the identified term.
- Term Source: this property allows us to identify the sources of the identified term, manufacturing order, datasheets, Subcontractors, Contractors, among others.
- Interfaces: This property is composed of the End Users, which correspond to the multiple user profiles of the platform to be developed, and also the External Components / Applications, identifying the need to develop interfaces for communication with other systems that already exist or that can be deployed in the future.
- Dependencies: Dependencies allow the identification of: a) Stakeholders who are directly or indirectly interested in and/or related to the identified terms; b) what data sources are needed to collect (serve as input) the data that interact with the defined term.
- Domain: Information Owner
- Security Level: Public/Private

The definition and cataloging of terms is, as a rule, a recurrent task where they are initially defined and periodically revised as the platform's ideal features become increasingly clear to stakeholders, also in order to deal with the evolving needs of users and the innovations brought about by research and technologies. Then, in Table 4, we can visualize the identified terms as well as the Use Case (UC) to which each one corresponds.

Table 4 - General Terminology Identification (excerpt).

ID	Terminology	Use Case
TMI1	Contractor	1, 2, 3, 6
TMI2	Subcontractor	1, 2, 3, 6
TMI3	Planning Proposal	1
TMI4	Raw Material	2
TMI5	Quantity Produced	3
TMI6	Production Delay	3

The Information Specification phase receives information directly from the previous phase of the model (Data Characterization) and is also aligned with the sequence diagram modeling phase (present in the descending phase of the method). Considering all the Use Cases defined above as well as the sequence diagrams related to the processes identified when gathering requirements, at this stage, a Class Diagram (UML) is created where the entities and their attributes are mirrored and methods to be used in each of the entities (Figure 41). At the top of each group of classes, the Use Case to which these classes relate is identified. In each group of Use Cases we can visualize the identified classes and the existing relationships between them. By analyzing the context of this data model, we identify the use case "Manage Planning" where entities "Planning proposal", "OF operation", "Expected consumption" and "Expected output" are identified. Next, in "Manage Delivery/Reception Flow", is represented the entity "Merchandise". The class "Manage Execution", contains the entities "Production Delay" and "Produced Quantities". The class "Manage Messages" contains only one entity "Messages". The class "Manage Quality Control" is also represented in this model and contains three entities: "Quality Control", "Quality Control Defects" and "Quality Control Received". At class "Register Stakeholder" there are four mapped entities: "Contractor", "Subcontractor", "Entities" and "Invitations to Entities". Lastly, we represent one class "Manage Interoperability" which contains the entities "History", "Information Model", "Interoperability Services" and "Indicators".

In addition to all this information, there are also the relationships between each of the identified entities. These relationships aim to integrate each of the related entities in order to understand the interactions between them, but also those when their programmatic component can have an associated interoperability.

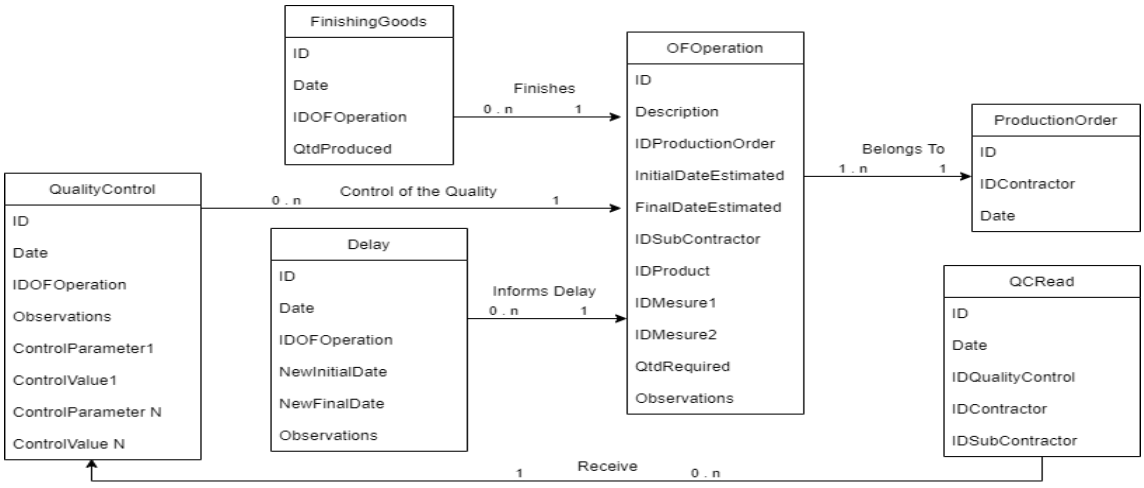


Figure 41 - Excerpt of the Class Diagram

Bearing in mind all the entities, attributes, and methods identified in the Class Diagram (UML), we performed a Neo4J modeling of the ontological database schema to be used by the Enterprise system. In Figure 42 is possible to see an excerpt of the code, in Cypher (Neo4J), that allows the creation of the nodes related to the terminologies and actions identified in the demonstration case of a textile industry.

```
//Criar Planeamento
CREATE (propostaplan :Manage_Planning {nodename: 'Planning Proposal', nome: 'nome', datainicio: 'date', datafim: 'date',
aceitado: 'bolean', rejeitado: 'bolean' })


//Criar Gerir Stakeholder
CREATE (entidade :ManageStakeholder {nodename: 'Entities', nome: 'nome', telemovel: 'int', email: 'string', Nif: 'int',
Observacoes: 'string'})
CREATE (contratante :ManageStakeholder {nodename: 'Contractor', nome: 'nome'})
CREATE (subcontratado :ManageStakeholder {nodename: 'Subcontractor', nome: 'nome'})
CREATE (convites :ManageStakeholder {nodename: 'Invitations to Entities', nome: 'nome', Nif_Convidado: 'int', telemovel_convidado: 'int',
email_convidado: 'string', data: 'date'})


//Creation of Relationships
CREATE
(contratante) -[:is_an]->(entidade),
(subcontratado) -[:is_an]->(entidade),
(propostaplan) -[:has]->(mercadoria),
(propostaplan) -[:is_proposal_a]->(subcontratado),
(propostaplan) -[:is_proposed_by]->(contratante)
```

Figure 42 - Excerpt of the F3M demonstration case code

This schema, codified in Neo4J, translates the semantic model of the Ontological component. As can be seen in Figure. 43, at the right of the image, there are labels, where each of these concerns a Use Case identified in the requirements document of the project.

At the center of the image, we can visualize the existing nodes and relationships. Each node refers to an entity identified in the class diagram, and its color is associated with the label, which in this case indicates the respective Use Case that is associated with it.

Thus, analyzing the context of this semantic model, with the orange color we identify the use case "Manage Planning" where there are identified the nodes "Planning Proposal", "OF Operation", "Expected Consumptions", and "Expected Output". Following, with the blue label, "Manage sending/receiving flow", has represented the node "Merchandise". The yellow label presents, "Manage Execution", which contains the nodes "Production Delay" and "Produced Quantities". With pink it is illustrated the label "Manage Messages" which contains the node "Message".

The label "Manage Quality Control" is represented by the red color and contains three nodes: "Quality Control", "Quality Control Defects" and "Quality Control Received". With the purple color there is the label "Manage Stakeholder" where four nodes are mapped: "Contractor", "Subcontractor", "Entities" and "Invitations to Entities". Lastly, represented by the green color, the label "Manage Interoperability" contains the nodes "Historic", "Information Model", "Interoperability Services" and "Indicators". In addition to all this information, there are also relationships between each of the identified nodes. These relationships are intended to represent the way each of them communicates, as well as the dependencies between them. Each of the nodes here represented (Figure 43), contain unique information that characterizes them.

This information concerns the attributes that are collected from each of the entities in the class model. As can be seen by way of example, in Figure 38, the node "OF operation" possesses the following attributes: "ID"; "Order Quantity"; "Cod_Operation_Contractor"; "Description"; "Dimension1"; "Dimension2"; "Instruction_Work" and "NodeName". Each of these attributes has a type of parameters which can be, for example, string, date, boolean, etc.

Thus, it is also possible to unify the form and type of data collection for each of the attributes. This information allows us to store the respective data for each of the entities. It is in this way, based on all this information and relationships, that interoperability between the different applications becomes viable.

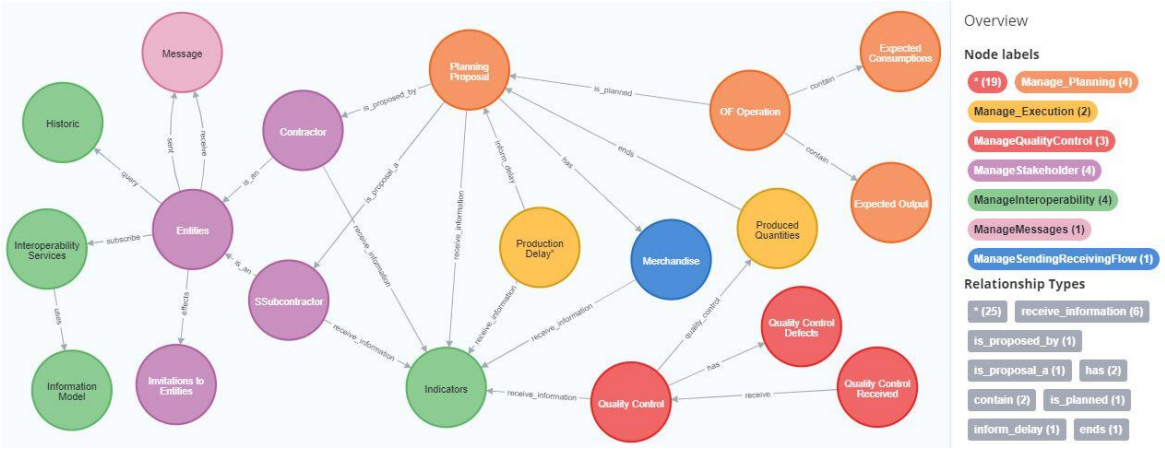


Figure 43 - Ontological Database Schema mapped in Neo4J

After having this schema defined and mapped in Neo4j, the loading of information relative to purchase orders for tissue manufacturers is performed. This information is stored in an xml and the xml document is imported to be associated with the "OF Operation" node. Next, in Figure 44 it is possible to see the code used to load the purchase order information through the xml and also the part of the code that allows eliminating null properties (Figure 45).

```
//importar TEXOrder53 através de url
CALL apoc.import.xml(
  "https://www.moda-ml.org/moda-ml/repository/istanzenoxsl/v2018-1/en/TEXOrder.xml",
  {relType:'NEXT_WORD53', label:'XmlWord53'}
)
YIELD node
RETURN node;
```

Figure 44 - Code to import purchase orders for fabric manufacturer

```
//Apagar propriedades nulas
MATCH (n:XmlWord53)
where trim(n.text) = ""
detach delete n
```

Figure 45 - Code for eliminating null properties

Figure 46 shows the result of the mapping that is imported using the code in Figure 44. This process is replicable not only for purchase orders but also for production planning.

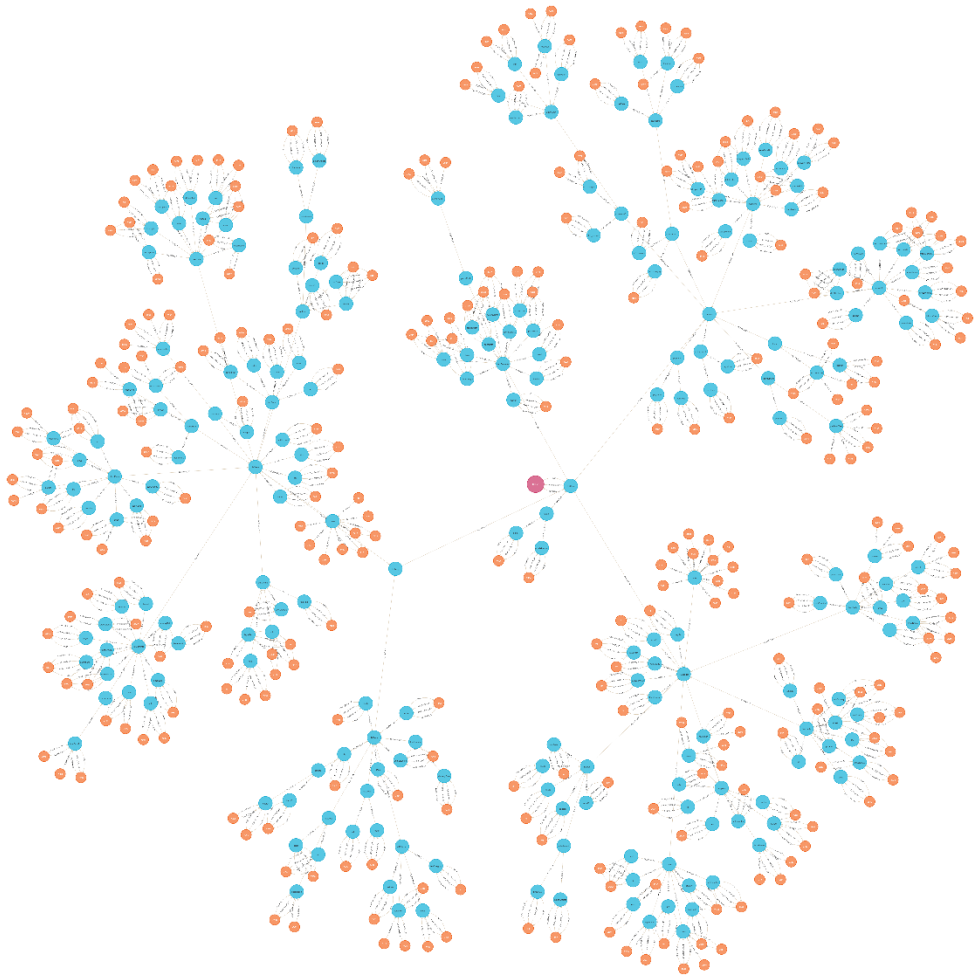


Figure 46 - Mapping of purchase order information

The Project "CityCatalyst: Normalized City Analytics"

Currently, there are still some gaps related to the lack of semantic interoperability, which are in line with what happens in many organizations in different sectors. Here we refer mainly to the excessive use of verbal communication to transmit information between departments, the excessive use of Excel resources as a way to store information about the management and planning of the organizational process, and the manual export of application logs that are used in other applications, among other aspects. This lack of interoperability results, in many cases, from a delay in the work process and a failure of communication/knowledge about what the other organizational areas are developing.

Therefore, all these interdependencies associated with the phenomenon of the new industrial revolution and Smart Cities will have a strong impact on the relationship with business processes,

as they change perspectives from a centralized to a decentralized paradigm. This will require the widespread adoption of machine and system interoperability, not only at the same production site/city neighborhood, but also across the entire ecosystem.

Cities are currently faced with a very diverse set of challenges, resulting from the complexity of the urban experience of citizens, who are active in different segments. The ambition of the City Catalyst project[5] is, therefore, to address these challenges through research, development, and validation, in a real-world context, of innovative technological solutions and services that promote integrated, more efficient, and effective urban management and catalyze innovation and sustainable development through specific contributions to the implementation and interoperability of urban platforms.

In the context of the present project, more specifically PPS3 "Normalized City Analytics" aims to promote the normalization of data models and interfaces in order to avoid silos between domains, solutions or cities, through the use of Minimal Interoperability Mechanisms (MIM). Considering this main objective, it will also serve as a basis to allow the implementation and validation of an AutoML platform (Auto Machine Learning), with different analytical capabilities (batch, stream and AutoML algorithms) based on a data lake and different modules integrated as microservices, available as a service to be used in Smart Sustainable Cities in the areas of governance/sustainability, energy, and mobility. This platform will be based on a data aggregator solution of urban platforms, to be developed, that will allow the integration, processing, and distribution of data from the platforms of the various stakeholders.

To assure the solution's replicability and scalability, it is necessary to choose and use a data model open standard, to define and develop an ontological model and to agree on a normalized city information model that the existing platforms comply with, for context-sensitive access to their standardized data, facilitating the work of data analysis tools and AI/ML (artificial intelligence/machine learning).

The definition of the semantic model aims to support the semantic interoperability layer. Thus, an approach for building semantic models is used, supporting a unified lexicon and glossary, as well as the relationships between their terms, for the different domains of cities. The integration layer

[5] City Catalyst - Catalisador para Cidades Sustentáveis, http://citycatalyst.efacec.com/

will include the design and implementation of a semantic model. The semantic model (and its mapping rules) will be designed based on data extraction and subsequent centralization of information in order to unify the terminology of the data catalogues. With this semantic model will also be possible to identify the relationships between data from different sources and the mapping rules that will facilitate the definition of queries to an ontological database.

The development of scenarios is one of the instruments used for decreasing uncertainty. Four scenarios are defined from different cities: *Porto*, *Vila Nova de Famalicão*, *Aveiro* and *Guimarães*. In each one, a description is made that makes it possible to understand the action focus in each activity area and builds the semantic model. To connect all the terms, a table is built with the terminologies present in the different use cases that are described, including the NGSI-LD model. In Figure 47 it is possible to have a visual perception of the work to be done in this project, essentially at the level of data normalization for the four municipalities.

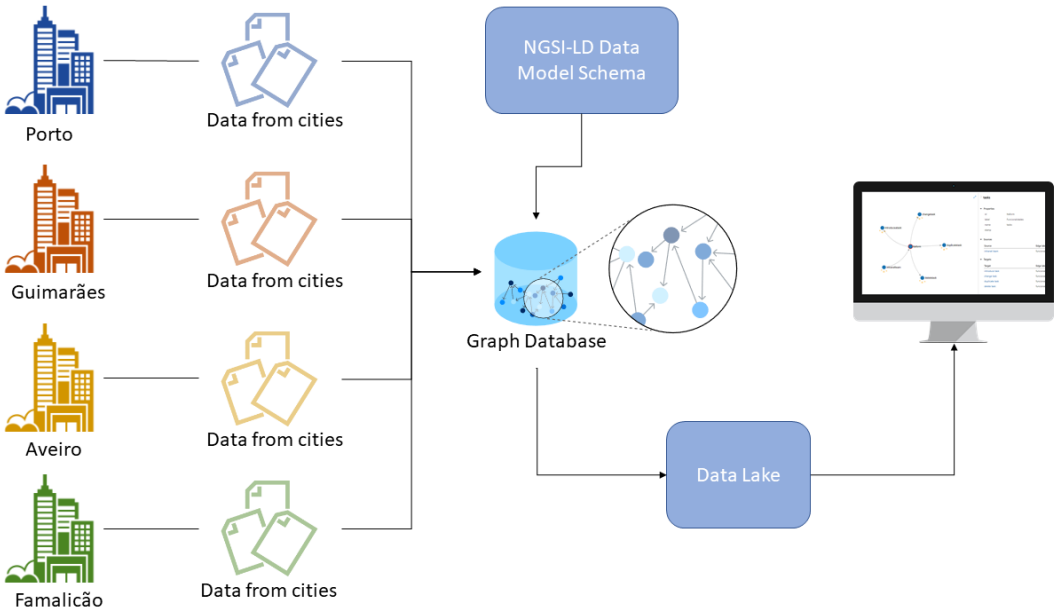


Figure 47 - Representation of the context of the demonstration case scenario

To address the problem identified, it is initially necessary to focus on the analysis of the cities' internal processes and activities, as well as the actors that carry them out (Data Characterization phase). In this phase, the Unified Modelling Language (UML) based use-cases are focused, as they allow to observe the city as a whole and, together with the demonstrator´s responsible, decide which area or areas make sense to be targeted through ontological implementation.

Next, in the Information Specification phase, the analysis must be oriented to the existing data model or reference architecture within the domain of interest. If, at least, one of these artefacts can be found, it is analyzed and detailed, while scenarios are built as a way to validate whether the ontological layer of the model covers the project's intervention areas. Otherwise, or if some specific requirement is defined by the stakeholder, a proper data model is designed to fit the needs, which is then used as a starting point to establish an ontological database schema that is populated according to the client's specifications and terminology.

In the end, Ontology Mapping Phase, we proceed with the integration of this ontological database in a visualization tool, wherein the stakeholder can apply filters, edit the database or add new terminology and relationships, in a more user-friendly way (Sokolov *et al.*, 2019).

Speaking specifically about the "Data Characterization Phase", at first, it is necessary to characterize the actors as well as the tasks in which each one participates. This description will help to understand in detail the field of action as well as the details of the tasks that are performed. Afterwards, we proceed to the identification and construction of a glossary of terminologies.

Each terminology glossary identifies a specific domain and, as such, should be constructed through an analysis in the context of the project in order to detail the identified terms through a description. These terms, as previously mentioned, result from the intersection between the "Use Case" phase and the analysis of the environment and context of the city. This terminology analysis and cataloging should be carried out in as much detail as necessary, and it can be deepened or extended at any time in a future iteration of the method. For the cataloging of terminologies, a previously defined structure must be followed, as shown in Table 5.

Table 5 - Identification of terminologies in the CityCatalyst Project

ID	Terminology	Use Case
TMI1	Citizen	Guimarães, Aveiro and Porto
TMI2	Sensor	Guimarães
TMI3	Sustainability indicator	Guimarães
TMI4	Air quality stations	Guimarães
TMI5	Machine learning platform	Guimarães
TMI6	Parameter	Aveiro
TMI7	Smart lamp posts	Aveiro

TMI8	Rainfall	Aveiro
TMI9	Solar radiation	Aveiro
TMI10	Wireless access points	Porto
TMI11	LoRa gateways	Porto
TMI12	Manipulate data collected	Porto
TMI13	Urban platform	Porto
TMI14	Temporal	NGSI-LD
TMI15	Zone	NGSI-LD
TMI16	Location	NGSI-LD

For a better understanding and contextualization of the identified terms, at this stage is further defined a table-like structure where each term is characterized by an ID which will allow to identify the term, the name of the term, a detailed description of the identified term and the term Source property that allows us to identify the sources of the identified term (e.g., City of *Guimarães*, City of *Aveiro*, City of *Porto*, City of *Famalicão* or NGSI LD data model). Are still identified synonyms for each terminology and its dependencies.

In the Information Specification phase, previously identified terminologies are used, where the main objective of this phase of the method is to map the terminologies through a UML class diagram. When mapping the terminologies, we must associate to each one of them the corresponding synonyms that were listed in the previous phase.

Thus, to elaborate the mapping, we follow the design rule of a class diagram composed by:

- Class: the class itself, this element is used when we want to visually demonstrate the class in the diagram;
- Association: connector without tips - It is a type of relationship used between classes. Applicable to classes that are independent (they live without dependence on each other), but that at some point in the ontology may have some conceptual relationship;
- Generalization: Heritage - connector with arrow at one end - It is a type of relationship where the generalized class (where the "arrowhead" of the connector is) provides resources for the specialized class (heir). It adopts more advanced concepts, that is,

everything that the mother (generalized) class has, the daughter (specialized) will have. We are talking here about a concept of property inheritance;

- Composition: connector with a "diamond" hatched at the tip - It is a type of relationship where the composite class depends on other classes to “Exist”. For example, the “Car” class has a composition with the “Motor” class. Without the "Motor" class, the "Car" class cannot functionally exist and Aggregation - connector with a leaked "diamond" on the tip - It is a type of relationship where the aggregate class uses other classes to "exist", but you can live without it. For example, the “Car” class has an aggregation with the “Roof” class. Without the “Rooftop” the “Car” class can exist.

As mentioned before, considering the mapping of the class diagram, we now focus on the entities that compose it. Thus, an analysis and description of each of the previously identified classes are performed to allow the definition of the schema for the semantic database (Figure 48).

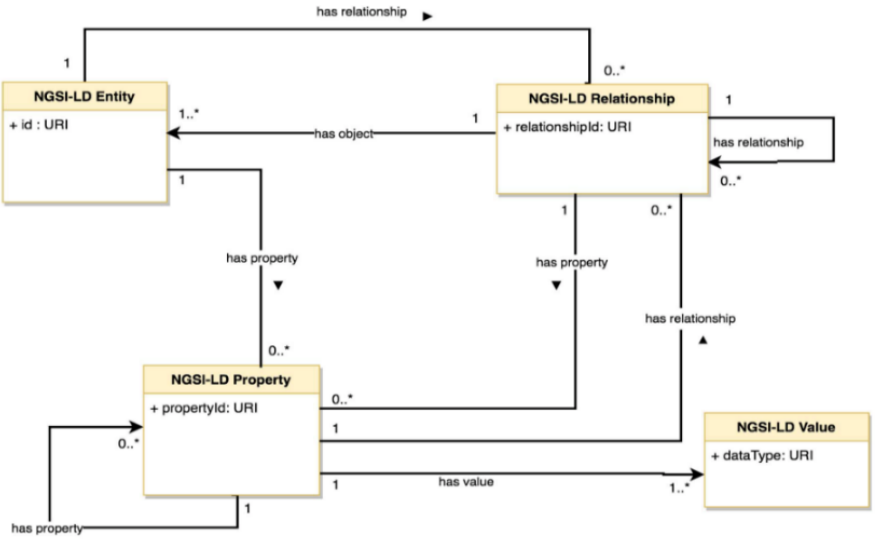


Figure 48 - Mapping of the class diagram

Considering all entities, attributes and methods identified in the Class Diagram (UML), we performed a Neo4J modeling of the ontological database schema to be used by the CityCatalyst system. This is the Ontological Mapping phase. Thus, this scheme translates the semantic model of the Ontological component. Each node refers to a terminology previously identified in the glossary terminology, and its color is associated with a label, which in this case indicates the respective defined hierarchical level. In the center we define the project name "CityCatalyst" in red,

in green we identify the municipalities where the project will be applied, the blue color refers to the NGSI-LD data model and finally the orange color identify all the terminologies raised in the course of the project associated with smart cities.

In addition to all this information, there are also relationships between each of the identified nodes. These relationships are intended to represent the way each of them communicates, as well as the dependencies between them. Each of the nodes here represented contains unique information that characterizes them (Figure 49).

Each of these pieces of information can be represented by attributes that have a type of parameter, which can be, for example, string, boolean, date, etc. Thus, it is also possible to unify the form and type of data collection in each of the attributes. This information allows us to store the respective data for each of the entities. It is in this way, based on all this information and relationships, that interoperability between the different applications becomes viable.

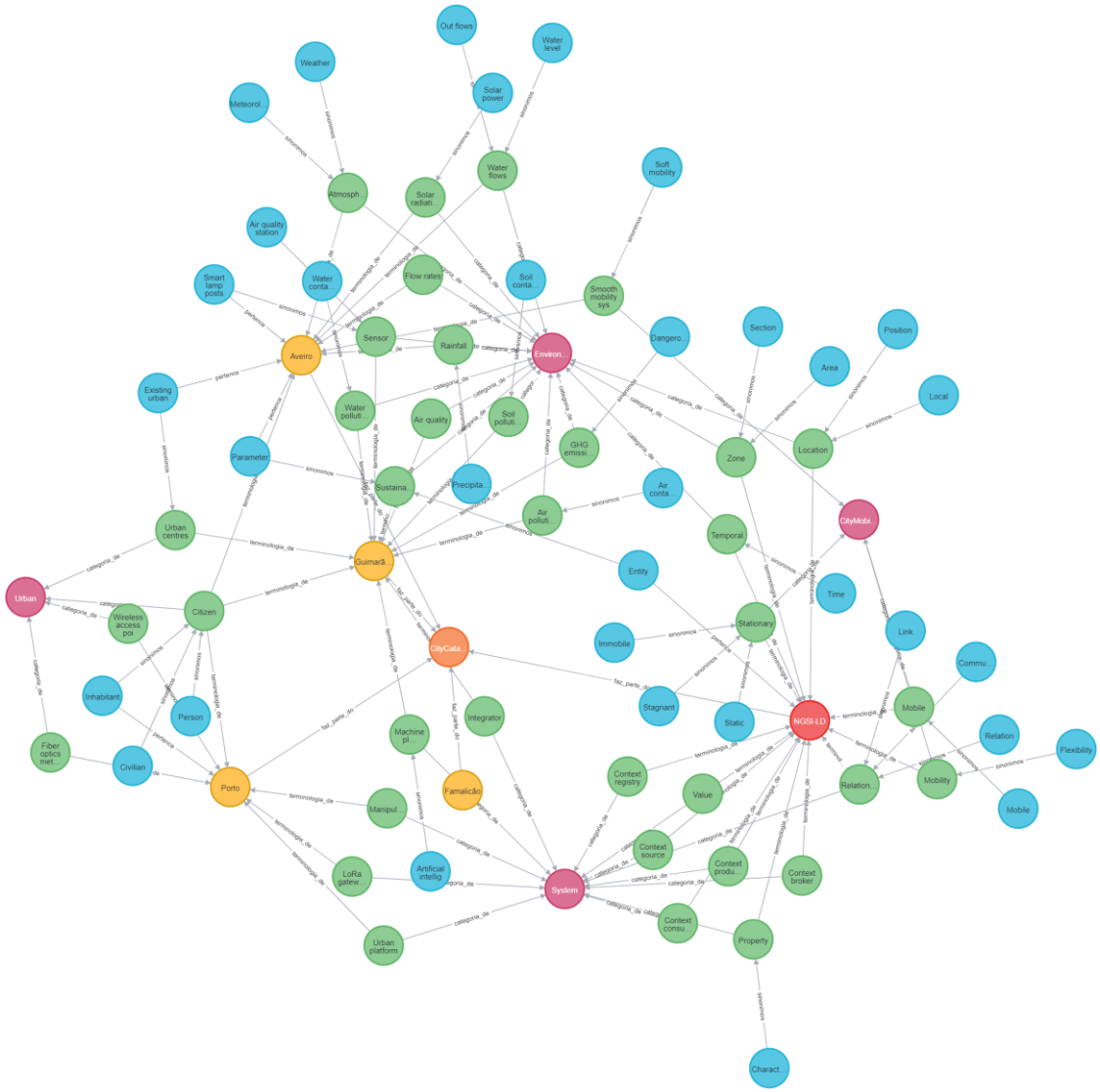


Figure 49 - Schema for the semantic database mapped in Neo4J

5.4 Conclusions

The work presented in this chapter, revolving around the approach that we propose, encompasses a dual perspective. On the one hand, following the aforementioned work on ontology development, and on the other hand, the ease with which this method enables the chained and systematic definition of an ontology mapping scheme. By analyzing the present literature, we have identified the need to establish an agile strategy for ontology mapping that enables rapid and effective responses to scientific initiatives throughout the development of our projects.

The work documented here, through the explanation of the approach and the framing with real-world demonstration cases, is primarily based on a survey of the terminologies to be used by Enterprise Information Systems. This allows the standardization of these terminologies to enable

Enterprise System interoperability with other enterprise information systems. The terminologies obtained are mapped from various Information Systems, namely at the interface level (menus, features, and expressions), correlating them with a library of terminologies and synonyms that are also mapped in the ontology database.

This worldwide database enables the unification of terminologies, the optimization of search engine actions, and the integration of company information systems with their applications. Thus, throughout this chapter 5 it is possible to see different types of applicability of ontologies using the approach we propose for ontology construction.

5.5 References

Barry, P., & Crowley, P. (2012). *Modern embedded computing: designing connected, pervasive, media-rich systems*. Elsevier.

Calero, C., Ruiz, F., & Piattini, M. (2006). Ontologies for Software Engineering and Software Technology. In *Springer*.

Carlos Salgado, Machado, R. J., & Maciel, R. S. P. (2019). 4SRS-SoaML Method for Deriving a Service-Oriented Architecture From Use Cases Within a SPEM Approach. In *Modeling Methods for Business Information Systems Analysis and Design* (p. 22).

De Nicola, A., & Missikoff, M. (2016). A lightweight methodology for rapid ontology engineering. *Communications of the ACM*, *59*(3), 79–86.

Ferreira, N., Santos, N., Machado, R. J., Fernandes, J. E., & Gasević, D. (2013). A V-model approach for business process requirements elicitation in cloud design. *Advanced Web Services*, *9781461475*, 551–578.

Guizzardi, G. (2007). On ontology, ontologies, conceptualizations, modeling languages, and metamodels. *Frontiers in Artificial Intelligence and Applications*, *155*, 18–39.

Kaiya, H. (2005). Ontology Based Requirements Analysis : Lightweight Semantic Processing Approach. *Fifth International Conference on Quality Software (QSIC'05)*, 8.

Peroni, S. (2018). *SAMOD : an agile methodology for the development of ontologies*. 1–11.

Salgado, C., Teixeira, J., Santos, N., Machado, R.-J., & Maciel, R. S. (2015). A SoaML Approach for Derivation of a Process-Oriented Logical Architecture from Use Cases. In *Lecture Notes in Business Information Processing* (Vol. 201).

Santos, N., Rodrigues, H., Pereira, J., Morais, F., Martins, R., Ferreira, N., Abreu, R., & Machado, R. J. (2018). *Specifying Software Services for Fog Computing Architectures using Recursive Model Transformations*.

Santos, N., Teixeira, J., Pereira, A., Ferreira, N., Lima, A., Simoes, R., & Machado, R. J. (2015). A

demonstration case on the derivation of process-level logical architectures for ambient assisted living ecosystems. *Ambient Assisted Living*, *June*, 103–139.

Sokolov, A., Veselitskaya, N., Carabias, V., & Yildirim, O. (2019). Scenario-based identification of key factors for smart cities development policies. *Technological Forecasting and Social Change*, *148*, 119729.

Soomro, S., B, A. H., & Shaikh, A. (2014). Ontology Based Requirement Interdependency Representation and Visualization. *Springer*, *2*, 259–270.

Wróblewska, A., Podsiadły-Marczykowska, T., Bembenik, R., Protaziuk, G., & Rybiński, H. (2012). Methods and tools for ontology building, learning and integration - Application in the SYNAT project. *Studies in Computational Intelligence*, *390*, 121–151.

Zhu, Z., Zhou, X., & Shao, K. (2019). A novel approach based on Neo4j for multi-constrained flexible job shop scheduling problem. *Computers & Industrial Engineering*, *130*, 671–686.

—— This page is intentionally left blank ——

CHAPTER 6

ON THE CONSTRUCTION OF ONTOLOGIES REPRESENTATION

Summary: The objective of this chapter's technological contribution is the building of complex networks. This contribution is applied to two demonstration scenarios in two diverse industries: textile and robotics. The goal of the two demonstration cases is similar, as they both aim to perform the mapping of two frameworks through complex networks. In the case of the textile industry, the objective is to map, using graph database technology, the existing terminologies in the information systems of textile companies, as well as the association of two frameworks (eBiz and IDS), in order to obtain a network of knowledge and dependencies among the various terminologies used in the industry. Regarding the robotics industry demonstration example, the goal is to map two frameworks, CMMI and OpenUp, in order to obtain a mapping of the frameworks' coverage and common points.

CHAPTER TABLE OF CONTENTS

6.1 Introduction....169
6.2 Complex Networks for Ontologies Representation....170
6.3 Demonstration Cases....182
6.4 Conclusions....204
6.5 References....205

—— This page is intentionally left blank ——

CHAPTER 6

ON THE CONSTRUCTION OF ONTOLOGIES REPRESENTATION

"t has to be admitted that it is true that knowing something does not imply thinking about it - but doesn't someone who knows something have to be able to doubt it? And to doubt means to think."

– Ludwig Wittgenstein

6.1 Introduction

Throughout this chapter, our study on the feasibility of developing a mechanism to automate data entry in a graph database is given. Thus, a first architectural approach is defined which, as can be seen throughout section 6.2 allows two paths.

On the one hand, and in the case of complex contexts with large volumes of data, these must be stored in a cluster which is our data repository. Then, through an integration platform (Talend), a pipeline is defined that allows the connection to the cluster and also to the graph database technology. In this same pipeline the terminologies and relations that we want to have as a result in the visualization technology are defined.

The information in the dataset is mapped directly to the graph database and the results are presented to the end user through a user interface. In order to optimize this whole extraction and mapping process, we are working on an integration with other visualization tools. This way we can define a more detailed architecture that responds to the different needs of R&D projects.

6.2 Complex Networks for Ontologies Representation

After collecting requirements and modeling business processes in the organizations and projects under study, we found that many of the previously defined processes are not executed in their entirety and are often not executed correctly. Thus, and taking into account all the interactions with various stakeholders of the companies, it is important to point out that one of the points where it is possible to intervene and help would be the definition and implementation of a technological architecture for visualization of information using complex networks.

This definition of an architecture will allow, above all, that employees have an intuitive visualization tool that can be parameterized through interactions in the user interface. These visualization components also allow the visualization of dynamic data catalogs where there is a unification of all the terms used.

This is because, in the vast majority of observations made to the internal contexts of organizations, it appears that within the same areas and departments, there are different ways to designate a specific term. Thus, we developed a first version of the technological architecture to visualize information, see Figure 50, composed of three main steps: Information Source; Generation Method and Ontology.

In the first phase, Information Source, it concerns the existing data already handled by an expert user of the domain, which are agglomerated in one or several datasets. The quantity of datasets varies according to the quantity of existing information sources.

Next, we understood it is necessary to define a "Generation Method" that would allow us to work with the data and relate them to each other in order to have consistency and meaning mapped through a graph database technology.

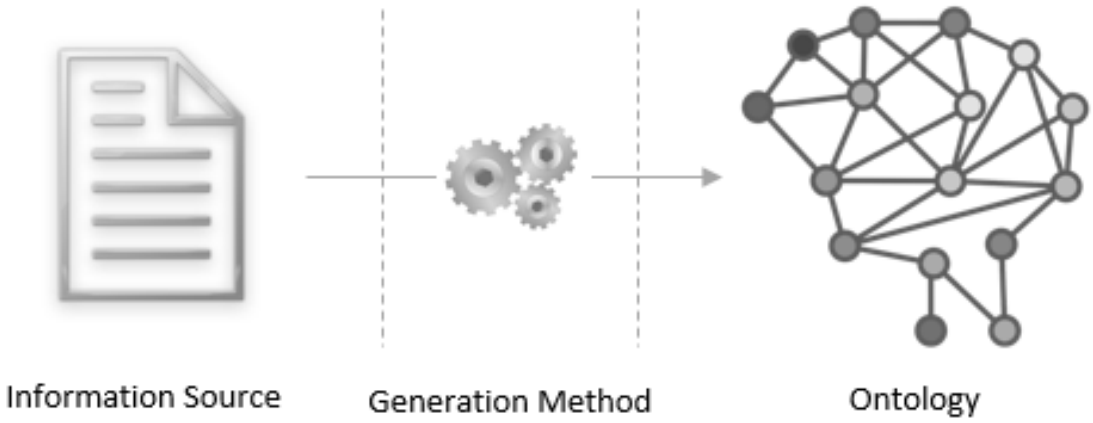


Figure 50 - Technological architecture for visualizing complex networks

Considering all the specificities found in the different projects where we have participated, and in order to analyze the different types of data that are provided to us, it is necessary to create a repository where to store these data and perform a cleaning and treatment of them. So, all the data sets are stored in a cluster that we have in the CCG. At the same time, a way to represent the data under analysis is studied and analyzed, and after surveying and benchmarking several graph mapping technologies, we arrived at Neo4J as one of the good solutions identified.

This process required some study of the tool, both in terms of configuration and use, and of its own programming language (cypher). Overcoming these initial difficulties, the first tests and mappings are performed with a restricted set of data. As seen in the previous figure, the defined architecture is quite simple. In Figure 51, we can see some details of a more detailed version of the "Generation Method" component.

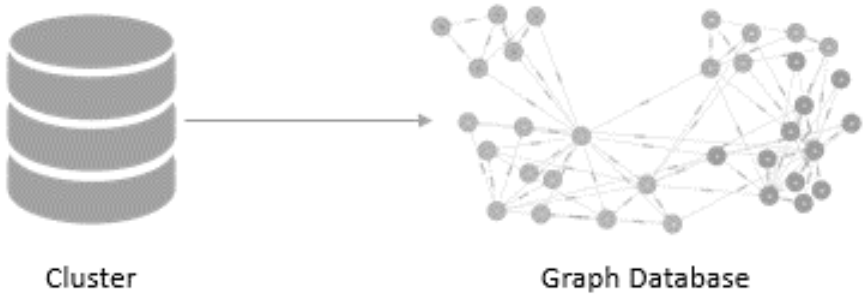


Figure 51 – "Generation Method" - Extractions and mapping

As a way to test this technological architecture, some analyses have been carried out in pilot companies within the different projects where we are inserted in the CCG. The technological configurations of this architecture, as well as the first analyses have led us to promote some changes as to how this process is being carried out. At this time, all this mapping is done very manually, and thus and taking into account that the vast majority of stakeholders of the companies with which we have been working have no knowledge of programming, we decided to try to automate this process a little to get around this limitation. Regarding the work in progress, an automation of the extraction of different data from the databases is being developed, in order to also allow the automatic mapping of the same.

Thus, we have been evolving the initial architecture to one that integrates different technologies (Figure 52). Initially, the datasets are loaded into the cluster, and later, using an integration platform, we establish a connection between the graph database and the cluster. In this way, mappings are made in our graph database using the information that is in the cluster. In addition to these mappings, and beyond the analytical component that visualization and graph queries allow to study the phenomena of the projects, it is also noteworthy that this extraction and loading by graphs allows to continue the implementation of an ontological database using complex networks.

Having done all the integration between the cluster and the graph database, we can represent the mappings in the user machines, or, on the other hand, if it is more convenient, represent the mappings on a server where all employees have access to the same information. At the moment, these demonstrations have already been carried out both locally and remotely (accessing the server) where some search queries have already been defined which allow, on the one hand, to filter the nodes, but also to show all the relationships that are associated with them. This process allows you to automate the mappings based on the data produced by the company's information systems.

As can be seen in Figure 52, number one (1) refers to mapping through a graph database, number two (2) represents the integration platform, which Finally, the number three (3) represents the user interface which is detailed in Visualization Component Analysis.

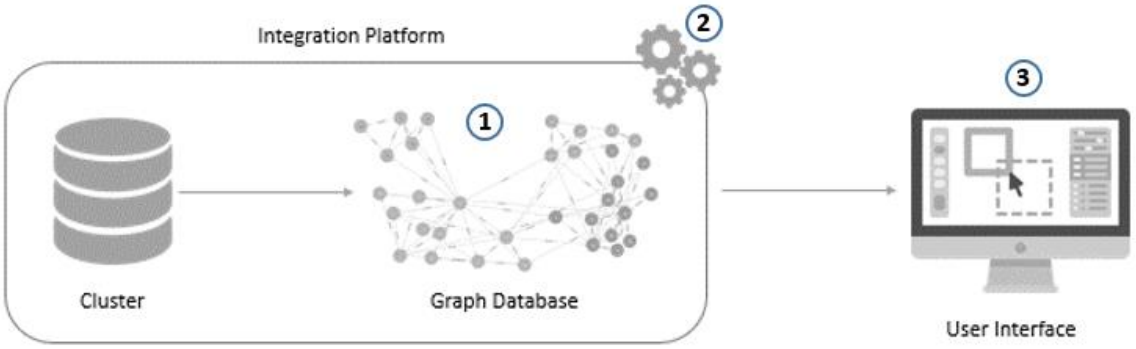


Figure 52 - Improved architecture

In order to optimize this entire extraction and mapping process, we are working on an integration with other visualization components. This will allow users to have more search functionality, filtering and attribute selection. In addition to these advantages it also allows to use some algorithms that do the proximity and proximity calculation for the mapped nodes, as well as a user-friendly interface.

Regarding the work developed about the visualization of information through complex networks, after conducting some workshops on this theme in different projects (called demonstration cases) in which we have been working at CCG, where demonstrations are made about the functionalities and practical cases of use of graph databases, it was concluded that this process would require some programming.

Since not everyone who wants to use visualization tools has the skills or aptitude for software development, we agreed that the ideal would be to have a way to automate the input of data into the graph database. Thus, a study is started on the feasibility of developing a mechanism to circumvent these issues that had been raised. We then arrived at the definition of an architecture that solves this problem (Figure 53).

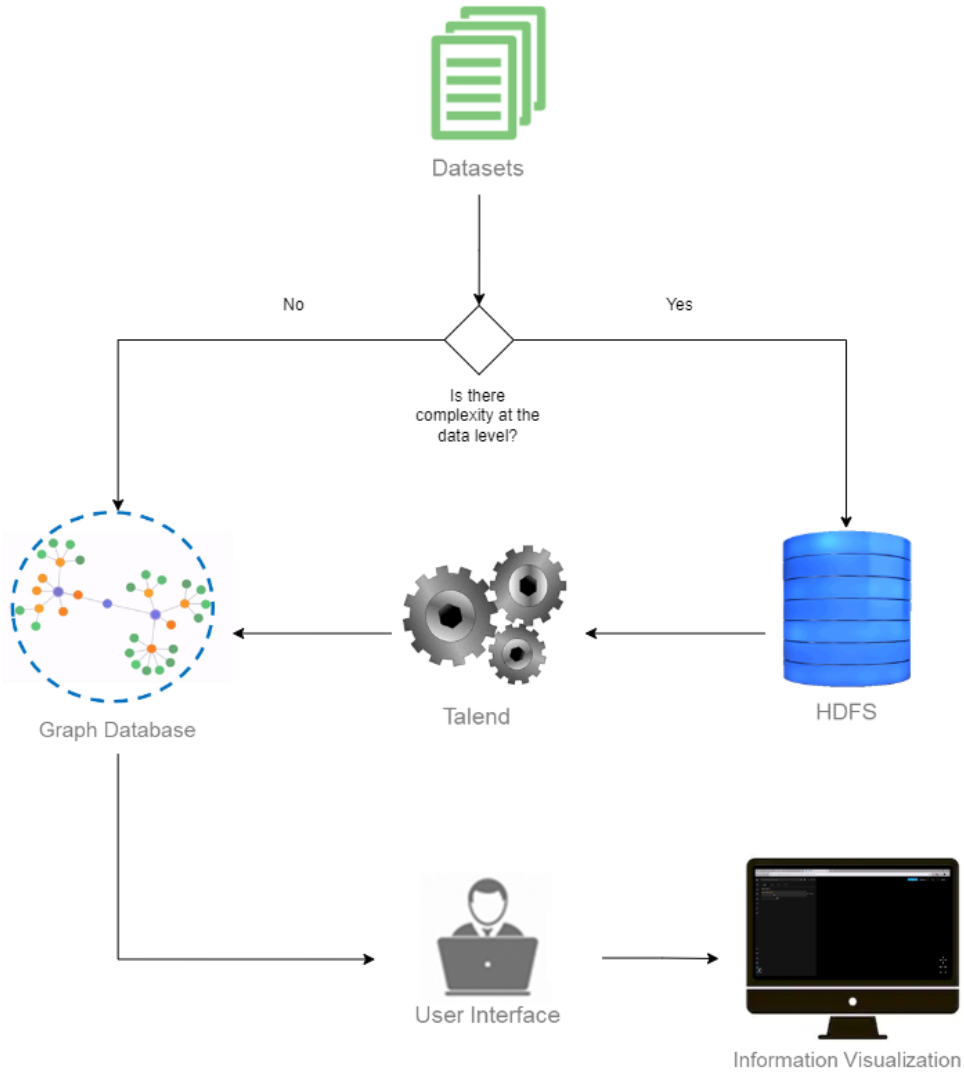


Figure 53 – Technological Architecture for Information Visualization

The architectural approach is then defined which, as can be seen in Figure 53 allows two paths. On the one hand, and in case of data complexity (more than one dataset, more than 1000 lines of information in each dataset) the data should be stored in a cluster. Then, through Talend, a pipeline must be defined that allows the connection to HDFS and also to the graph database technology. In Talend we have also developed a pipeline that allows us to define which terminologies and relationships we want to have as output in the visualization technology. Once this step is defined, the pipeline must be executed and the information is mapped to the graph database and presented to the end user through a user interface (depending on the graph database technology used). Appendix A contains the complete pipeline as well as descriptions for its entire configuration.

On the other hand, the other possible path, is intended for cases with less complexity at the level of data present in the datasets. So, for cases where we only have one dataset, and there are less than 1000 rows of data in that dataset, it is not necessary to use the HDFS repository, nor the Talend pipeline. The information in the dataset is mapped directly into the graph database and the results are presented to the end user through a user interface (depends on the graph database technology used).

In the end, we will have as a final product a complex network stored in a graph database technology that will be gradually fed. Therefore, initially it is necessary to study and treat the data sets that are being collected and provided by the domain being studied. From this point on, the datasets are stored in a cluster, which in this case is our data repository.

At the same time, as a graph database technology we used neo4j and also Azure Cosmos DB to perform the mappings and build the complex network. It should also be noted that these two technologies result from the analysis of technologies to implement complex networks previously identified in section 3.4 of this document. That said, we chose to use an integration platform (Talend) where we connect to both the cluster and the Graph Database Technology. Here all the necessary rules are defined and which dataset is intended to be mapped through a complex network.

At this point, the user just has to execute the process, in Talend, that we have defined for the mapping to take place. In addition, and given the visualization and parameterization components, which are understood to be necessary, it has also been integrated with four different data visualization platforms (Y-files, Graph XR, Bloom and Graph Algorithms Playground). Each of these visualization tools provides a wide range of parameterization, filters, and data visualization algorithms, such as centrality, community detection, and others.

In order to optimize this whole extraction and mapping process, we are working on an integration with other visualization tools. This will allow users to have more search, filtering and attribute selection functionalities. In addition to these advantages, it also allows the use of some algorithms that calculate the proximity and centrality of the mapped nodes, as well as a user-friendly interface.

Next is an analysis of some tools for visualizing information. For connecting to a graph database technology, which in this specific case and for the purposes of projects in the CCG we used Neo4J.

Thus, as tools to visualize information, we highlight the analysis of yfiles Neo4J explorer, Graph XR by Kineviz, Bloom, and Graph Algorithms Playground. This analysis is performed based on real project data.

yfiles Neo4J Explorer

Neo4j Explorer is a free online tool developed using the commercial yfiles for HTML diagramming library. With this tool, we can connect to the remote or local Neo4j database and interactively and visually explore the database schema and data without having to write Cypher queries. In addition, it offers a broad set of visualization features, ranging from interactive exploration to automatic layout and graph analysis. In the following figure (Figure 54) the data schema to be mapped can be visualized. In this test case five different attribute types are used, “product”, “receivedate”, “location”, “typeofsample” and “status”.

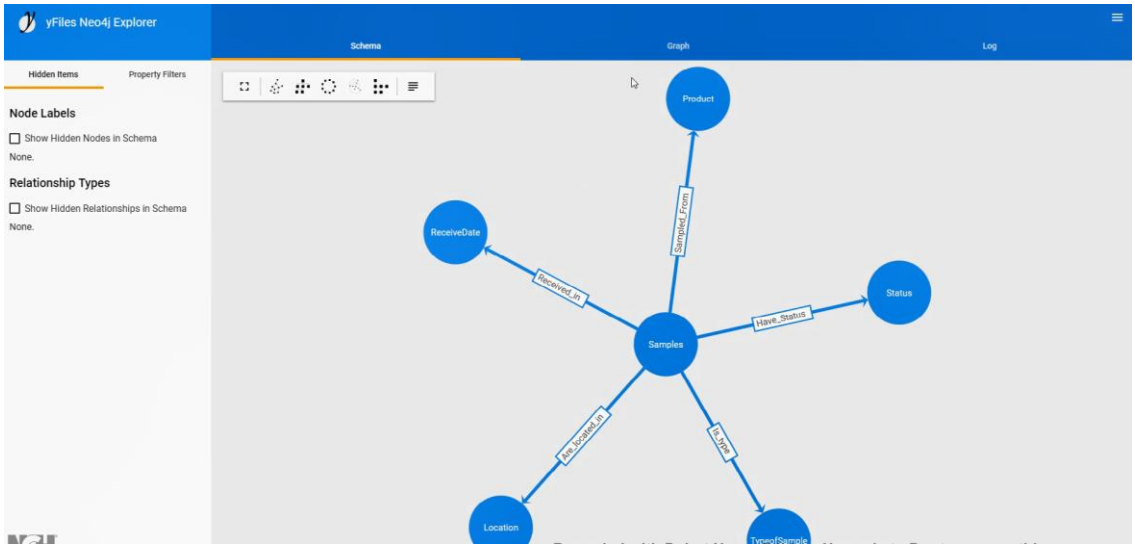


Figure 54 - yFiles visualization of schema

Apart from this visualization of the ontological database schema, this visualization tool has some very useful and intuitive features from the user's point of view. As we can see in Figure 55, it is possible to filter by existing terms to map only those same terms. Given the large volume of data from organizations, and given that the employees' goal depends on their area of expertise internally, this option allows them to focus on the data that is relevant to them.

After applying the desired filters, by clicking on “import all”, we can see the result of our search, Figure 55 and Figure 56. If the visualization seems confusing, there is still the possibility to switch within the possibility of algorithms provided by the tool. It is possible to represent the terms and their relationships in a hierarchical way, considering the neighborhood, the centrality, among others.

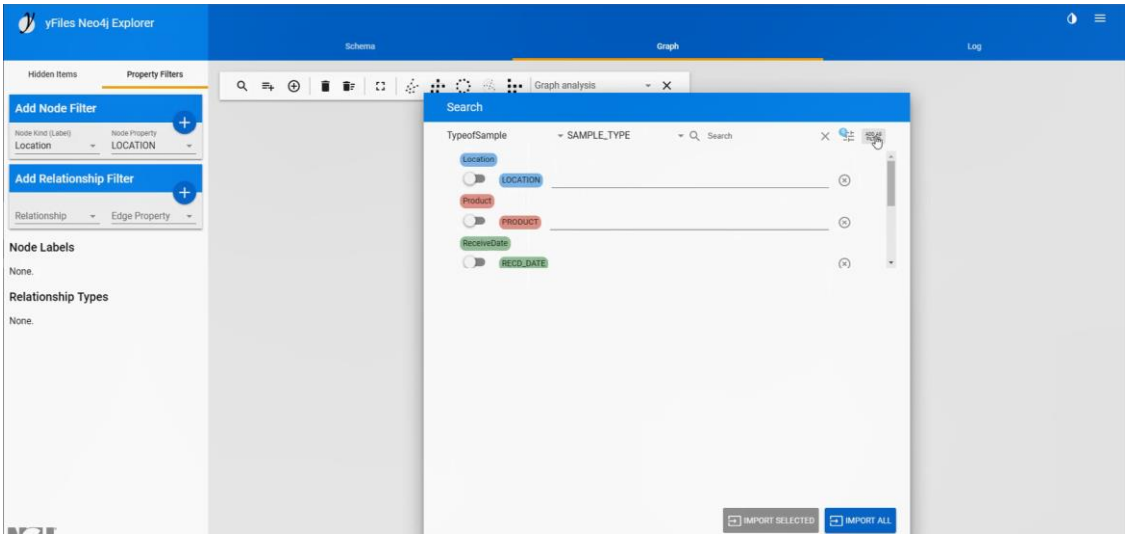


Figure 55 –yfiles type of filters

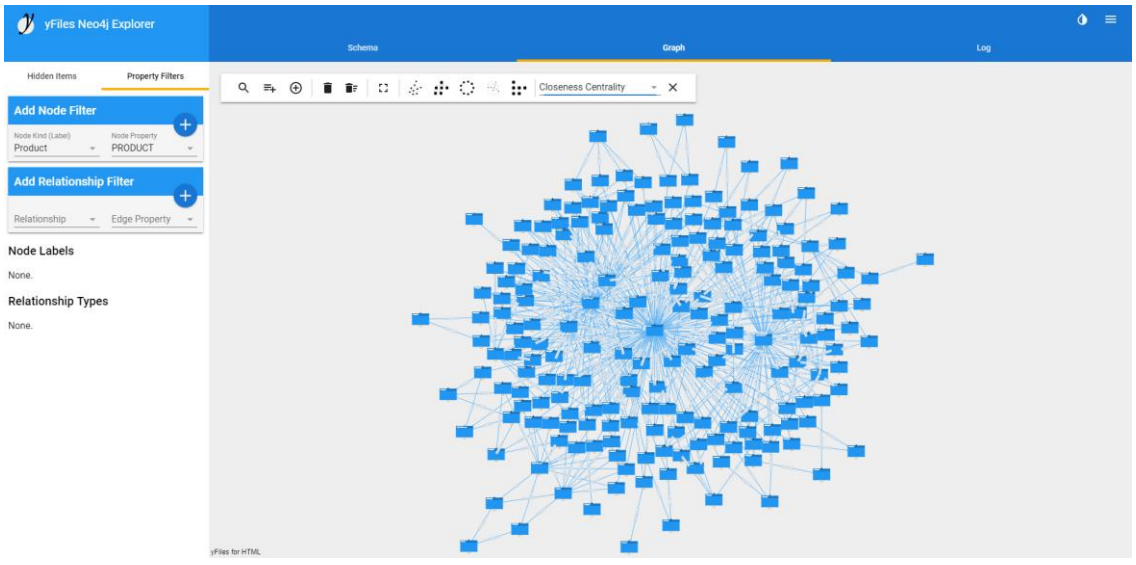


Figure 56 - yFiles visualization of Neo4J mapping

Graph XR by Kineviz

GraphXR is launched in December 2018 and allows the exploration of graphics in virtual reality. However, most GraphXR users work in traditional 2D computing environments. The advantages of using GraphXR are the ease of performing statistical link analysis, geospatial visualization, timeline filtering, and embedding rich content such as portrait and video images in the application.

Regarding the content used, the data remains in the database and is processed in memory. So the only thing that gets to the Graph XR server is just the login credentials. The Explorer edition - which supports Neo4j Community Edition is free for individual users. Figure 57 shows a mapping performed in this tool, using data from one of the projects under study. As in the example shown above, the features are similar and also allow you to make various types of filters by applying them to each specific term. In the figure we can see that the default visualization type allows us to differentiate the type of relations by color and also assign an icon or image to each group of terms.

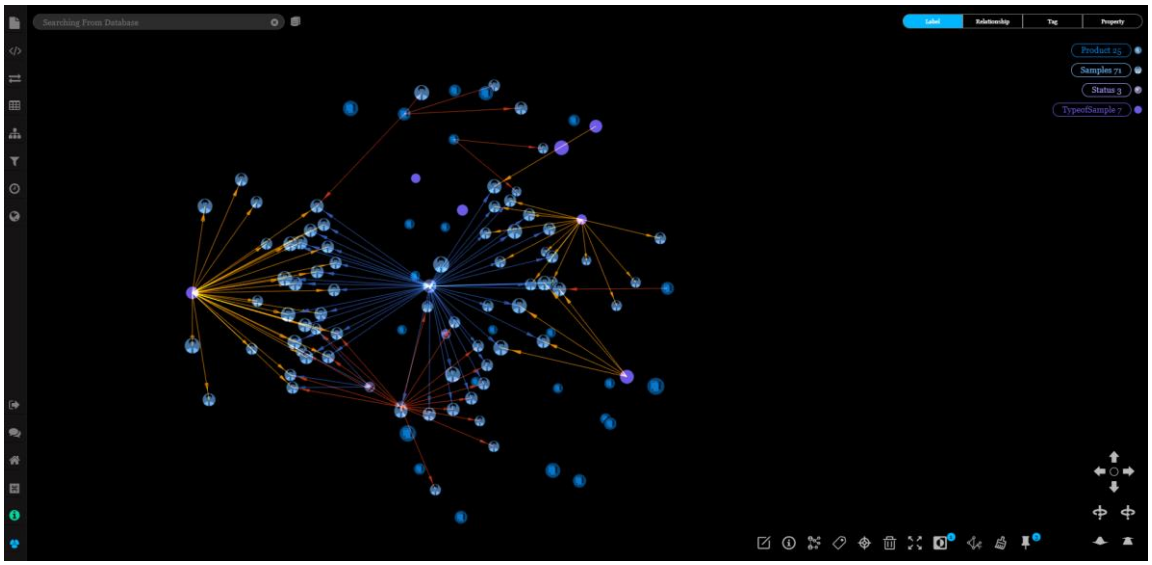


Figure 57 - Graph XR visualization of Neo4J mapping

Neo4J Bloom

Bloom is a visualization and editing component developed by Neo4J and can be installed as a plugin within Neo4J itself. Initially, Bloom was only available in commercial versions of Neo4J, however, in mid-2020, this paradigm was inverted, and the company (Neo4J) released Bloom in all versions of Neo4J, whether commercial or open-source. This is an interesting technology both for developers and end-users, since it allows a wide range of features.

In this technology it is possible to visualize all the mapping done in Neo4J, make edits to the nodes and relations inserted, create new nodes and relations, create new labels, and still customize each of the nodes with icons and a wider range of colors relative to the Neo4J interface. In summary, it is a robust technology that allows the User to customize each node with the icon we want and also offers a wide range of filter types. In figure 58 it is possible to see an example of a mapping in Bloom using data from one of the projects of the demonstration cases.

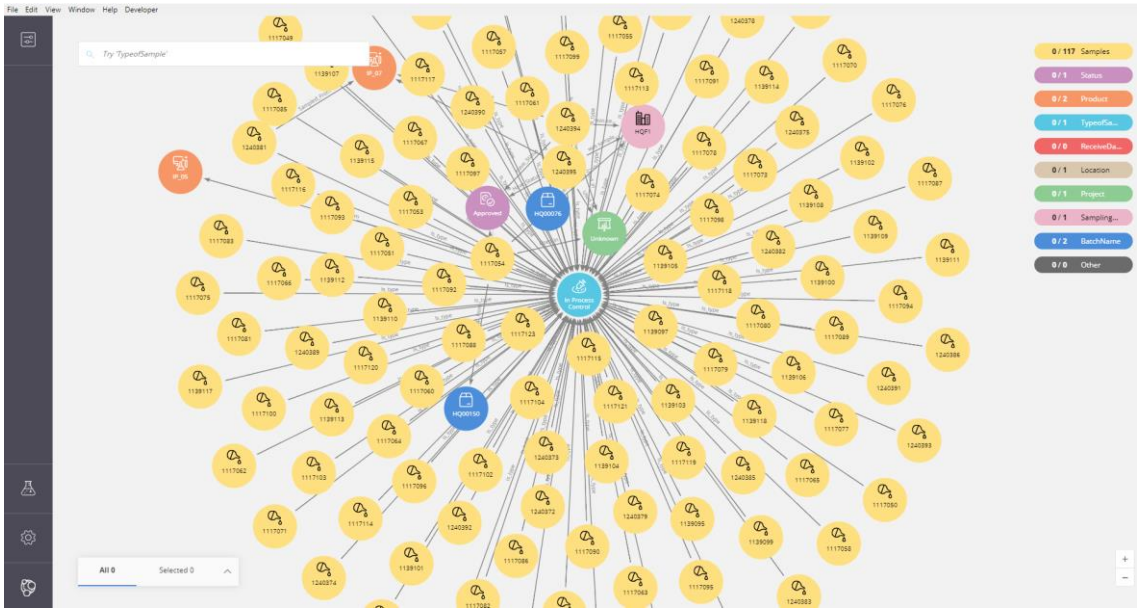


Figure 58 - Bloom Visualization

In addition to the most common type of filter in this type of tooling, it is also possible to perform ratio filtering and neighborhood filtering (see Figure 59). These are some of the aspects that facilitate the usability of the User and can also be customizable by adding more features.

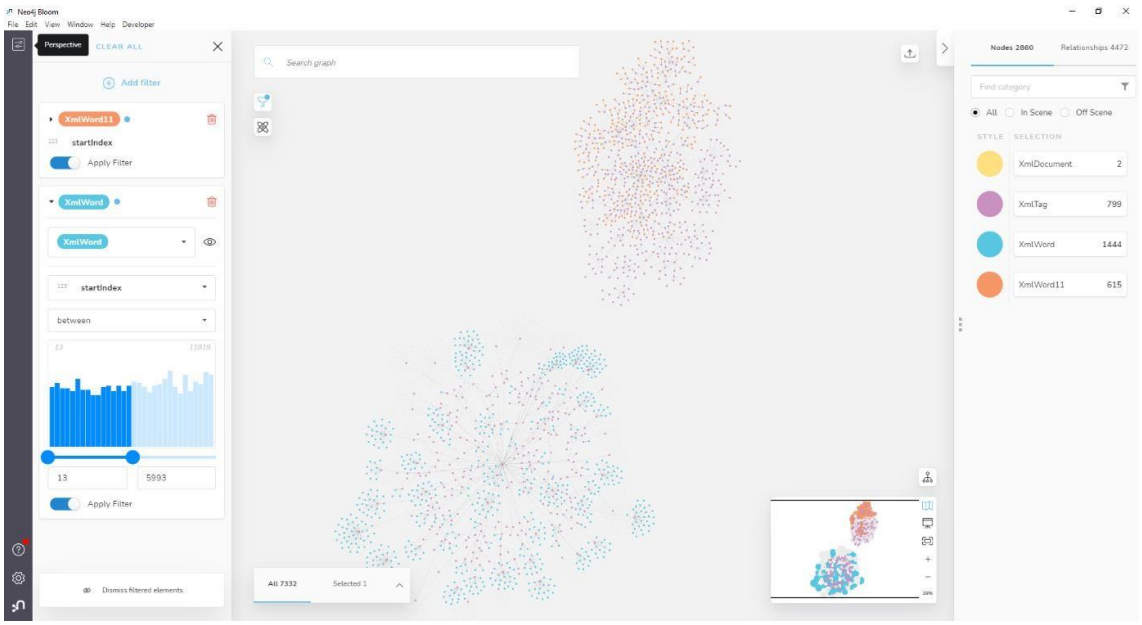


Figure 59 – Bloom filters

Graph Algorithms Playground

Graph Algorithms Playground is another tool used and tested with data provided to us over the last few months. This tool has basically the same functionality as described above, however with the connection made loses a little compared to the previous ones. In this mapping, the tool takes a while to respond, and sometimes there are features that do not run correctly. When connecting to the ontological database, the mapping is performed based on the similarity algorithm as shown in Figure 60.



Figure 60 - Graph Algorithms Playground - centralities visualization

The main features of this tool are then:

- Path Finding – these algorithms help find the shortest path or evaluate the availability and quality of routes;
- Centrality – these algorithms determine the importance of distinct nodes in a network;
- Community Detection – these algorithms evaluate how a group is clustered or partitioned, as well as its tendency to strengthen or break apart;
- Similarity – these algorithms help calculate the similarity of nodes.

Then, in Figure 61, it is possible to visualize the centrality algorithm already applied to the organization data. The concept of centrality is important because it allows the visualization of the nodes in focus and from different perspectives. In this specific case it allows the highlight in the center, in yellow, the different types of products and the relationships that are associated with them represent the samples of each of the products.

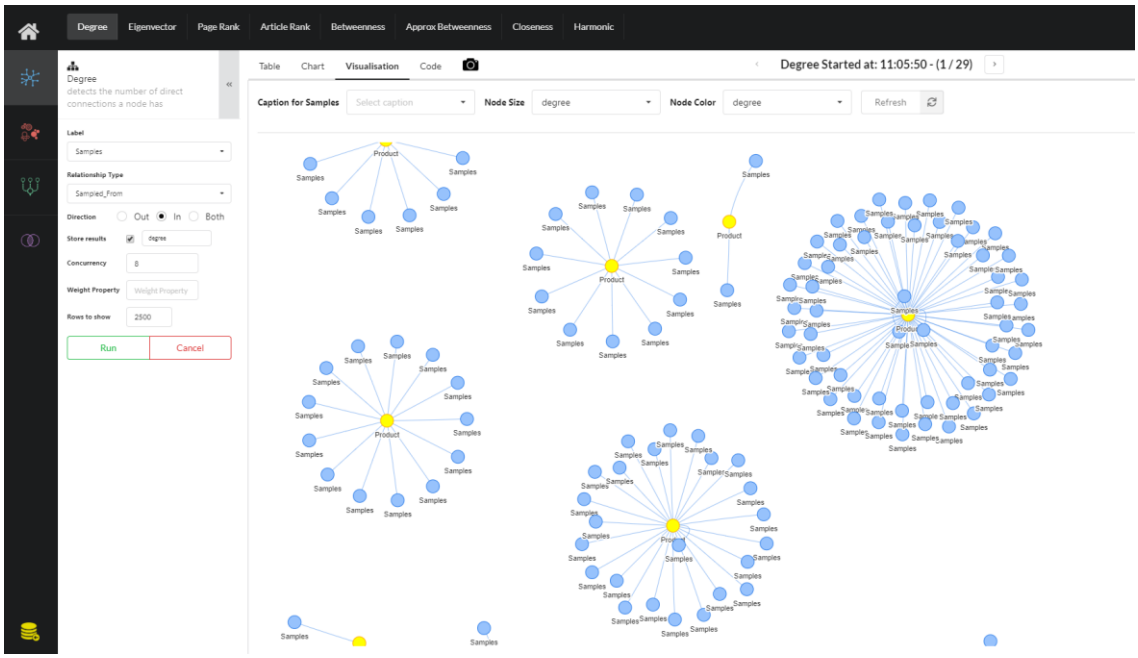


Figure 61 - Graph Algorithms Playground - "samples are sampled from"

6.3 Demonstration Cases

In this section, contributions to the design of a technological architecture for mapping information using graph database technologies are presented. These contributions are used and demonstrated in actual projects (CCG R&D projects). They are applied to demonstration cases in the textile and robotics industry. In the textile industry case, the goal is to map, using graph database technologies, the existing terminologies in the information systems of textile companies, as well as the association of two frameworks, eBiz and IDS, in order to obtain a network of knowledge and dependencies between the various terminologies used in the industry. Regarding the demonstration case of the robotics industry, the goal is to map two frameworks, CMMI and OpenUp, in order to obtain a map of the frameworks' overlap and common points.

The Project "STVgoDigital: Information Systems, eBIZ and IDS Representation"

This work, intends to contribute to the solution of the problem of language uniformity in the textile area, through the development of an ontological model. As a final result, we have a system capable of integrating various data sources for a generic ontological model.

This way, it will be possible to take advantage of the advantages that ontologies bring in terms of semantics and the explanation of concepts through relations, thus reducing the possibility of data conflicts. Thus, the entities that intend to use the data will be able to easily access the queries that are intended to be performed according to the existing description of the data sources in the general ontology (Seliverstov, 2015).

Next, the applicability of the technological architecture that we defined in Chapter 6.2 is presented in order to meet the needs of this demonstration case. The characterization of terminology catalogs needed to build the ontological model has already been demonstrated in chapters 4 (through the application of 4SRS-Onto) and 5 through the implementation of the approach we propose to develop ontologies.

Thus in this chapter, a framework is made about the characterization of terminology catalogs in order to contextualize the work done. Thus, the focus is essentially on how the technological architecture for visualizing information through complex networks is applied in this project context.

Next, we present the steps we take to map several catalogs of terminologies as well as their integration with a visualization tool.

To develop the ontological model through complex networks, it is necessary to collect and organize the terminology catalogs and respective synonyms catalogs related to a specific company in the textile and clothing sector. In addition to these catalogs, we also cataloged two frameworks: eBIZ and International Data Spaces (IDS). Below, you can get to know eBIZ and IDS in a little more detail.

eBIZ

Collaboration between IT systems can be achieved through at least two different paths (Chen *et al.*, 2008): integration, where concepts such as coordination, coherence, and standardization of elements of IT systems are closely linked, and interoperability, where concepts are related to coexistence, autonomy, and a federated environment.

The interoperability approach intends to affect only the interfaces between systems, so that they can remain independent systems, able to operate within open systems (Brutti *et al.*, 2012). This is the case in the textile and clothing industry where the European industry of industry associations, as well as other technology associations and standardization actors have been promoting initiatives on interoperability, both from a technological and scientific point of view and from a standardization point of view.

In order to address this problem, in 2008 the European Commission launched a first standards-based interoperability initiative to "harmonize e-commerce in the European Textile Apparel and Footwear Industry", the eBIZ-TCF project. This project provided the possibility to collect all the results of previous initiatives, identifying a reference architecture and its large-scale deployment in more than 150 organizations from 20 European countries. This activity highlighted the benefits that can be achieved when there is a critical mass of users (Bindi *et al.*, 2016).

The eBIZ 4.0 initiative aims to digitally connect at least 100 fashion companies across Europe. It aims to deliver IT solutions combining the benefits of the eBIZ digital language with Radio Frequency Identification (RFID) or Near Field Communication (NFC) technologies. This solution will enable increased product traceability, improved time-to-market and warehouse management, and

reduced data exchange barriers with external suppliers by shortening distances along the supply chain.

The goals of eBIZ are essentially to digitize the industry by providing reliable, real-time data connections for companies and to help set up voluntary traceability, fighting counterfeiting and unauthorized distribution channels. This will be possible by reducing the distance at each step of the supply chain with the integration of information flows between different departments (through RFIDs) and companies (through eBIZ and RFIDs).

The goal is to bring benefits to Small and Medium Enterprises (SMEs) with the IT tools most commonly used in large, fashionable businesses to significantly improve their process management and their connection with suppliers and customers (EBIZ, 2017).

International Data Spaces

The International Data Spaces (IDS)[6] Reference Architecture is an initiative created in 2006, specified and managed by a private, non-profit association called the International Data Spaces Association. It is a virtual data space that is based on existing standards, technologies, and governance models accepted by the data economy. It facilitates the exchange of data in a secure and standardized way, linking data in a trusted business ecosystem.

The main strategic requirements that underpin the initiative are essentially trust. For data exchange and sharing to take place, trust is required based on certification not only of human resources but also of the technical components of data exchange, security, and data sovereignty, which essentially comprises restrictions on use.

At the data ecosystem level, the fundamental principles are the non-centralization of the data repository, the architecture is decentralized and the data sources must be described allowing entities to access data through search services available from brokers. In the strategic requirement of normative interoperability, the most important component of the standard, the IDS connector, is the most technical component. It allows one entity to publish data and another to consume it, and the exchange and sharing of data is done through these connectors.

[6] https://internationaldataspaces.org/

The standard provides for several implementations and variations of these connectors, and it is possible to purchase these same connectors by adapting them to your needs. The strategic requirements related to applications add value and are complementary to the connectors, as they are integrated into them and appear as services in most cases for data processing and data conversion to implement specific communications protocols. Finally, the data market is related to the economic value assets that data services bring along with the creation of new associated business models (Otto *et al.*, 2018).

Characterization of Terminology Catalogs

At this stage, it is necessary to characterize the tasks, activities, and actors involved in the textile processes. This description will help to understand in detail the field of action, as well as the details of the tasks that are performed. Subsequently, a glossary of terminologies is identified and constructed.

This glossary of terminologies identifies a specific domain and, as such, should be built through an analysis in the context of the project in order to detail the terms identified through a description. This terminology analysis and cataloguing should be carried out in as much detail as necessary, and may be deepened or extended at any time in a future iteration of the method. For the cataloging of terminologies, a predefined structure should be followed, as shown in Table 6, where you will find an excerpt from the table constructed within this project. Appendix B contains the table with all the terminologies collected as part of this demonstration case.

Table 6 - Excerpt from the General Catalogue of Terminologies

ID	Terminologies	Data Source
TMI1	Textile Sector	eBIZ
TMI2	Footwear Sector	eBIZ
TMI3	Brand Owner function	eBIZ
TMI4	Producer function	eBIZ
TMI5	Data Consume	IDS
TMI6	Data Provider	IDS
TMI7	Space	IDS

ID	Terminologies	Data Source
TMI8	Entity	IDS
TMI9	Sample	Textile Information System
TMI10	Batch	Textile Information System
TMI11	Textile Finished Product	Textile Information System
TMI12	Location	Textile Information System

Process for Building Complex Networks

After cataloging the terminologies, they are separated into different documents, each for a specific source (textile company terminologies, eBIZ terminologies, and IDS terminologies). Each of these documents (.csv files) contains the terminologies as well as a synonym catalog for each of the terminologies identified. Once this phase is complete, these catalogs are loaded into HDFS (Figure 62).

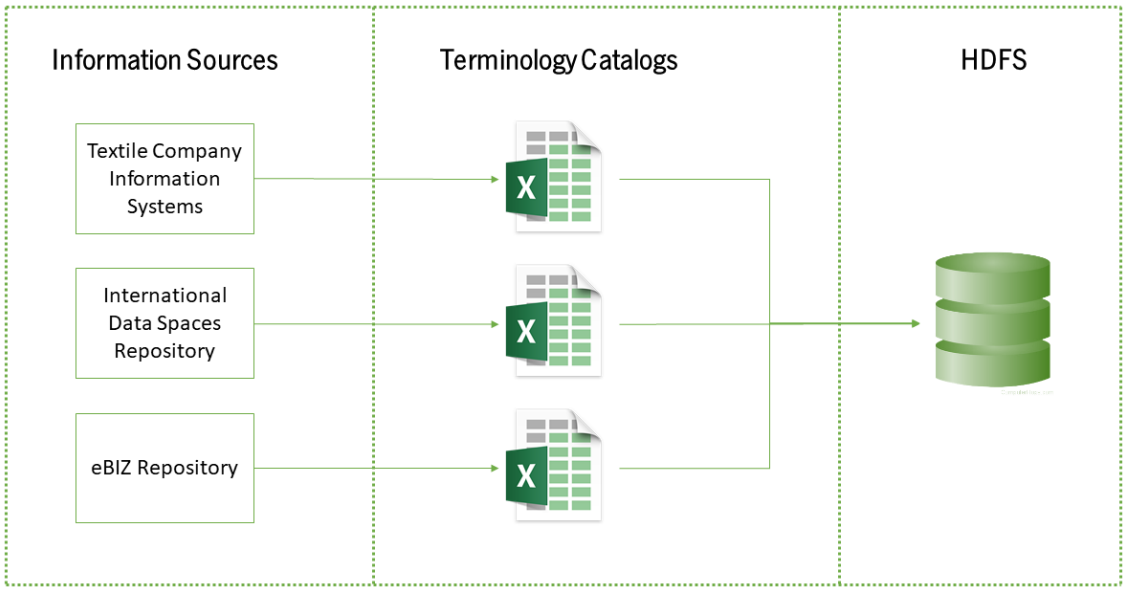


Figure 62 - Building and storing the terminology catalog

Once the terminology catalogs are hosted in HDFS a pipeline is then defined, in this specific case in Talend. This pipeline (Figure 62) consists of the following steps:

- A - Insert the link with the port number that you see in the step 7 from Neo4J and insert the user name and password that you create at step 4 from Neo4J.

- B - Select “Use an existing connection” and insert the query "Match (n) Detach delete n”. This query deletes all nodes from neo4j to assure that the new execution occurs correctly.
- C – This step is where you set the HDFS connection.
- D – This step associates the files we want to access. Therefore, it is necessary to select “Use an existing connection” (which is defined in the previous step) and then place the link where the .csv file is allocated in HDFS.
- E – The “tSampleRow” component is used here to work with only a sample of the data. In this case, we select lines 1,5 the range between lines 10 and 70, plus the range between lines 2000 to 2100.
- F – Double clicking on “tmap” will open a new window where you can then select how many outputs we want to represent in the future mapping. The left-hand table shows the attributes of the entire dataset, while the right-hand side are the defined outputs.
- G - In this step, double click on “tUniqRow”. Each output defined in the previous step is now chosen in this component as illustrated in the following Figure 63.
- H – By clicking on “tNeo4jOutput” only once, we select “Use an existing connection”. This connection concerns the connection to the Neo4J set at the beginning (Neo4JConnection). In addition, we also define the label we want to identify this type of output in neo4J visualization.

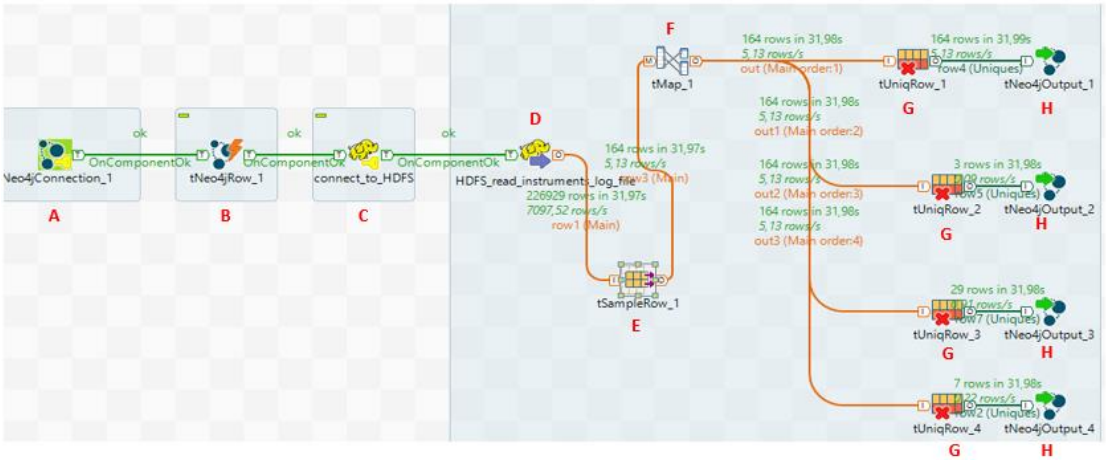


Figure 63 - Talend Pipeline

Regarding the definition of relationships between terminologies, we have also defined a pipeline in Talend which we illustrate below (Figure 64).

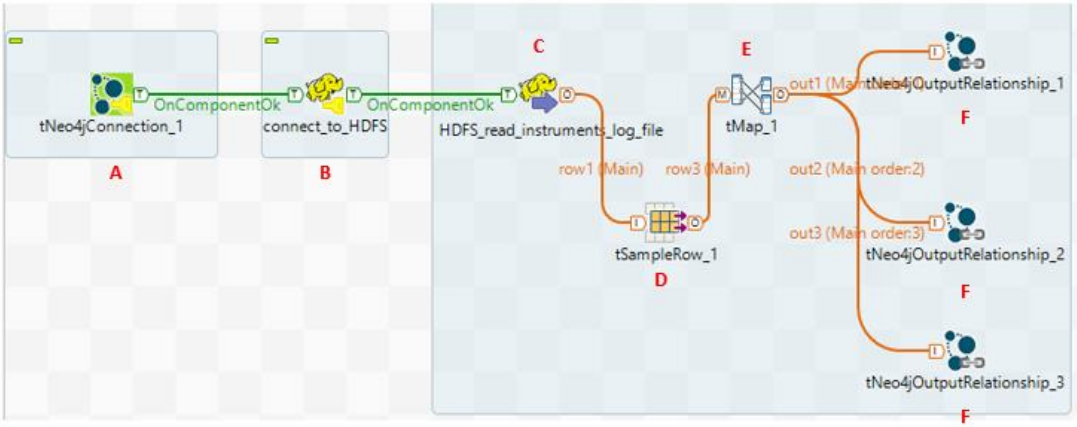


Figure 64 - Visualization of the architecture of all Talend jobs - relations

Here we will also demonstrate the process for associating the relationships between their outputs. Thus, the steps marked in the image with the letters "A", "B", "C" and "D" are filled in exactly the same way as in the previous section.

- E – double clicking on "tMap", we click on the icon "+" to create the associations between attributes.

  In this particular case we want to associate:

  - "STATUS" with "SAMPLE_NUMBER";
  - "SAMPLE_NUMBER" with "PRODUCT";
  - "SAMPLE_NUMBER" with "SAMPLE_TYPE";
  - "SAMPLE_NUMBER" with "RECC_DATE";
  - "PRODUCT" with "LOCATION".
  - "TEXTILE_COMPANY" with "LOCATION"

- F – double-clicking on "tNeo4jOutputRelationship" we select "Use an existing connection". Then:

After running these two pipelines, the information is mapped into the graph database by complex networks (Figure 65).

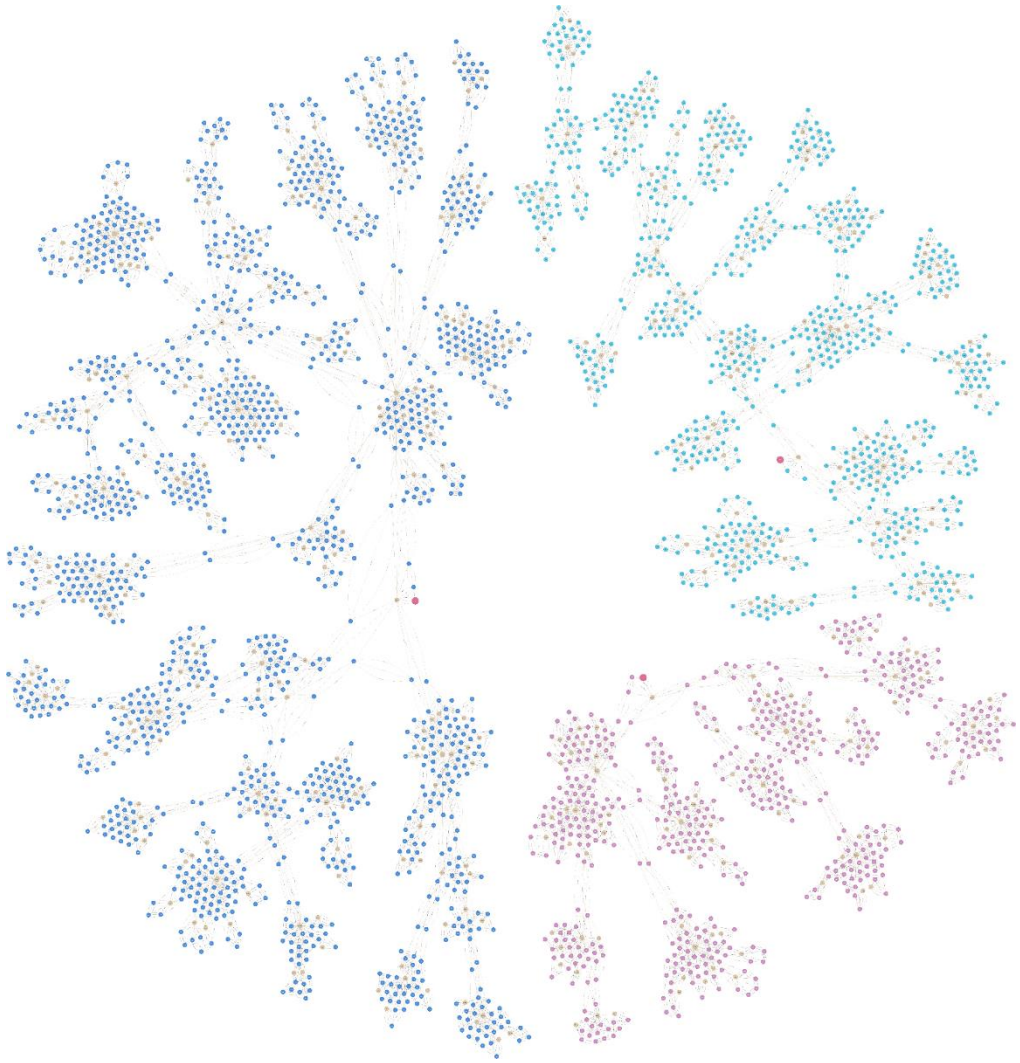


Figure 65 - Complex Network STVgoDigital Project

The ontological model presented, through a complex network, gathers the terms used in the context of textile production as well as the system responsible for the exchange of information. Thus, it is possible to analyze how these terms are related and to understand the information present in the different documents that, in the format in which they are found, are not easy to read. Using the visualization technologies presented previously (section 6.2), in this specific case, the choice falls on Neo4j Bloom. This tool enhances team collaboration through code-free search, simplifies complex queries using custom Cypher-based search phrases, and uses natural language search functions. The complex networking provided by this technology is similar to what is presented earlier.

However, through this tool, the user is able to create filters so that only relevant information can be accessed, as well as change colors and properties of nodes. Furthermore, it is possible to select a property associated with the node and, this way, analyze which data is associated with it. If the value is numeric, the tool shows a bar graph with the value intervals and respective counts. Figure 66 shows the ontological model represented in the Neo4j Bloom tool.

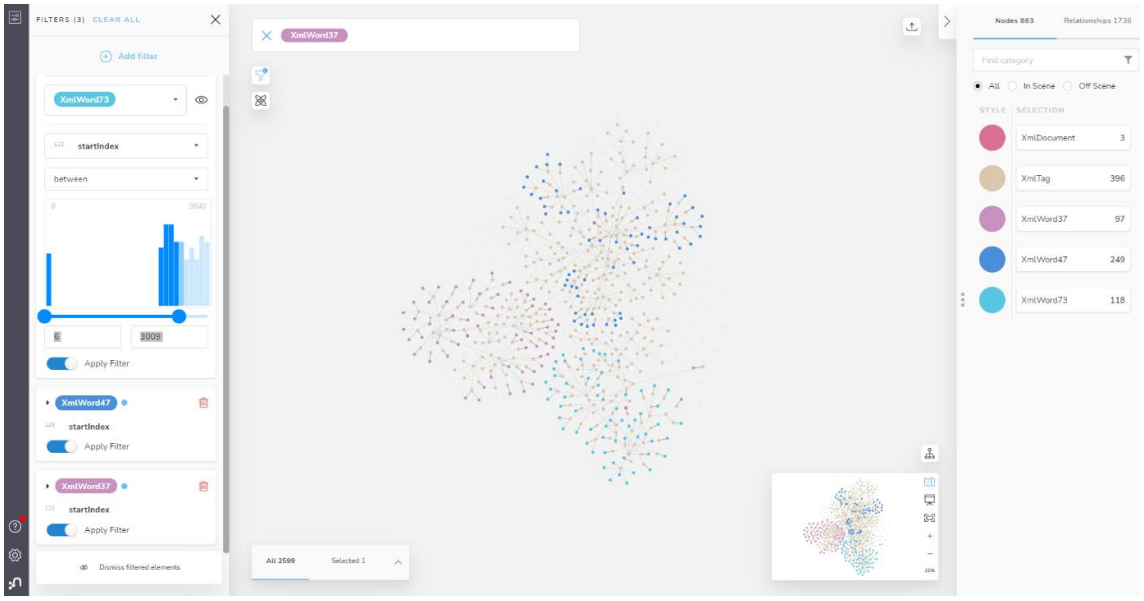


Figure 66 - Applying filters to the ontological model

The Project ITEC: "A Guideline to Software Development"

In order to create a best practices guide for future use at ITEC, the OpenUp and CMMI frameworks are analyzed, and their information is organized in datasets using cross tables so that the two can be related. After this step is defined, the information present in the tables is used to create a complex network with the related information from CMMI and OpenUP. In Figure 67 it is possible to visualize the problematic of the ITEC project.

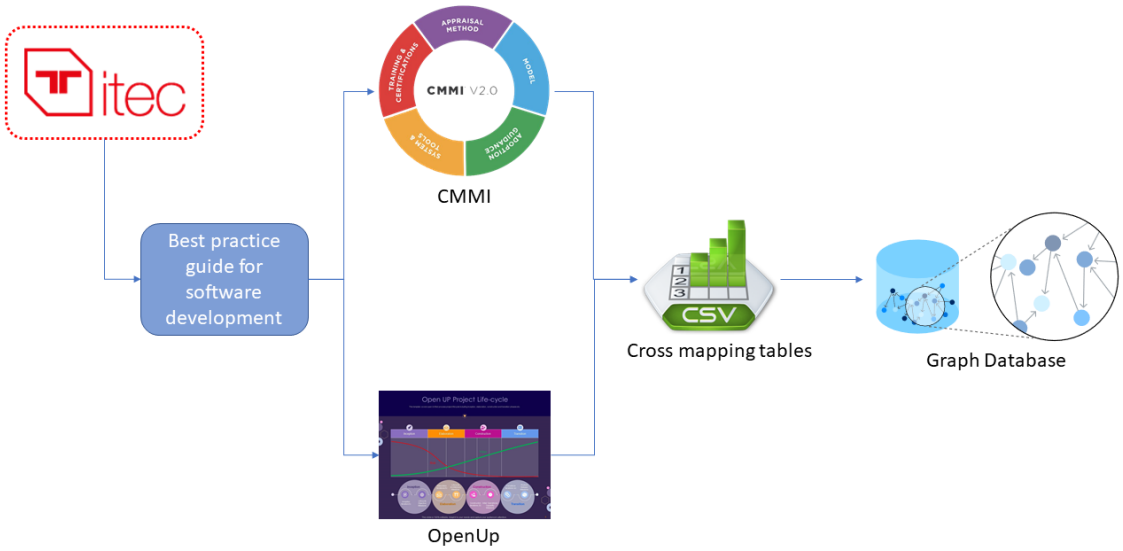


Figure 67 - Demonstratio case ITEC

In order to contextualize these two frameworks (CMMI and OpenUp), we believe it is necessary to provide a brief description of each.

The Capability Maturity Model Integration (CMMI) is a well-known Software Process Improvement (SPI) model developed by the Software Engineering Institute and currently managed by ISACA[7] and its main concern is to help organizations to improve their processes, (Hoda *et al.*, 2017). This model emerged in 2000 as an improvement of the Capability Maturity Model (CMM) (1991)[8]. Since then, it was implemented by several organizations, improving productivity and performance (Hoda *et al.*, 2017). According to (Staples & Niazi, 2008) the most frequent reasons given by organizations for adopting a SPI model based on CMMI, are improvements in quality, development time, costs

[7] Information Systems Audit and Control Association (ISACA), *CMMI Model 2.0*. 2021.

[8] C. Product Development Team, 'CMMI for Systems Engineering/Software Engineering, Version 1.02, Staged Representation (CMMI-SE/SW, V1.02, Staged)', 2000. [Online]. Available: http://www.sei.cmu.edu/publications/pubweb.html

and productivity, while customer satisfaction and staff motivation are also mentioned, a few times by stakeholders, as a key factor.

CMMI consists of an integrated set of practices that recommended what allows companies to improve the performance of their main business processes. This model is developed by production teams grounded on a collaboration between industry and the CMMI institute.

CMMI 2.2 is divided in 4 categories, Doing - referring to capability areas for producing and delivering quality solutions; Managing - for planning and managing the implementation of solutions; Enabling – with capabilities areas that supports the implementation and delivery of solutions; and Improving - to sustain and improve performance. Each of these categories is divided into capability areas, which are finally distributed in several practice areas (PAs).

The PAs, are organized in 5 levels of maturity, and are applied according to the intended goals. Each PA is divided in a set of Practices, which one representing a different maturity level that can be achieved by the PA. Thus, at each maturity level, the difficulty and implementation costs increase as they approach the maximum level (Ariffin & Ahmad, 2021).

At Maturity Level (ML) 1, the results are unpredictable and the production is reactive, in this level despite the work being completed, it is often delayed and goes over budget, as it does not have defined processes, and also fails to carry out adequate plans (Teixeira *et al.*, 2020). While at ML2, production and development are managed at project level, planning, implementation, measurement and control of projects are present, however, at this level companies still carry out initial processes, but with the crucial difference that they easily control costs and project deadlines for delivering products and services (Singh & Gill, 2020). Whereas the key point of ML3 is the definition of process standards for the entire organization[9]. Maturity level 4 is considered a high ML, at this level organizations define and rely on complex quantitative and statistical analyzes to determine, identify and manage the trend. Finally, ML5 is presented as the highest level of maturity, in which organizations focus on continuous improvement to achieve flexible processes capable of responding to opportunities and changes (Ariffin & Ahmad, 2021), in fact, there are few organizations that reach this level, due to their high costs and demanding process (Grossi *et al.*, 2014).

[9] Information Systems Audit and Control Association (ISACA), *CMMI Model 2.0*. 2021.

The OpenUp is an opensource project, which is currently supported by the Eclipse Foundation, and it is defined as a lean unified process that applies iterative and incremental approaches within a structured lifecycle[10]. OpenUp is initially developed by the International Business Machines Corporation (IBM) as a subset of RUP (Rational Unified Process) it is initially named as BUP (Basic Unified Process) but in 2005 it is transitioned to the Eclipse Foundation and renamed as Open Unified Process (Purnama *et al.*, 2020).

This process framework is referred as a pragmatic agile philosophy that emphasizes the collaborative nature of software development (Cossentino *et al.*, 2014). In fact, it has a minimum and sufficient philosophy, meaning that it provides the necessary guidelines but does not provide guidelines for all the possible elements that are handled in a project, it is a tools-agnostic, low-ceremony process that can be relate to a wide variety of projects [(Cossentino *et al.*, 2014),(Ruiz *et al.*, 2018)]. However, it preserves some necessary formalisms, mainly regarding the documentation, requirements and architectures (Borg *et al.*, 2007).

The Framework is based on an iterative and incremental cycle, project iteration and micro-iteration, focusing on creating an agile and precise model[11]. Henceforth the project lifecycle provides visibility and decision points that assists the team manage their work through micro-increments (Cossentino *et al.*, 2014).

The OpenUP framework divides the project lifecycle into four phases: inception, elaboration, construction, and transition, and it is composed by roles, disciplines, activities, tasks, artifacts and process and guidelines. Regarding the tasks, they define how to perform the work and therefore this work focuses on cross mapping OpenUP tasks with the CMMI PAs.

Therefore, and taking into account the contextualization made about CMMI and OpenUp, we decided to define different levels of coverage that will serve to make a traceability and relationship between the two frameworks. In the scope of the project we categorized four levels of coverage (Table 7) that, we believe are the most appropriate to represent our study, having as a goal the continuous improvement of the company.

[10] Eclipse, 'OpenUP V1.5.2'. 2018.

[11] Ricardo Balduino, 'Introduction to OpenUP (Open Unified Process)', 2007. Accessed: Nov. 11, 2022. [Online]. Available: https://www.eclipse.org/epf/general/OpenUP.pdf

The H code is assigned to tasks with high coverage and significance; these tasks must be completed to ensure the correct delivery of the product. The H+ code signifies the same, but with increased relevance. Medium-coverage tasks are assigned with the letter M, and these tasks must be completed, but they only encompass a portion of the aligned Practice. Lastly, the letter L indicates that these tasks have low coverage and at same time low importance; they cover a very small portion of the Practice in question and have a weak relationship with the Practice (CMMI) and the task (OpenUP).

Table 7 - Coverage Levels Categorization

H+	Super High Coverage
H	High Coverage
M	Medium Coverage
L	Low Coverage

The next table contains information about CMMI and OpenUp. This table was created in order to map CMMI and Open and register their coverage levels using the levels presented in Table 7. In order to show how the tables are structured, the information present in each one of the columns is presented, with the goal of facilitating its understanding. In Figure 68, we can see an example of a table with a CMMI "Practice Area", containing the information of the "Category", "Capability Area" and "Practice", with the information of the respective maturity level. We made 29 similar tables for each CMMI Practice Area. Thus, by studying CMMI, we concluded that these maturity levels have what is necessary to implement in the company and achieve the desired goals in the company's daily procedures.

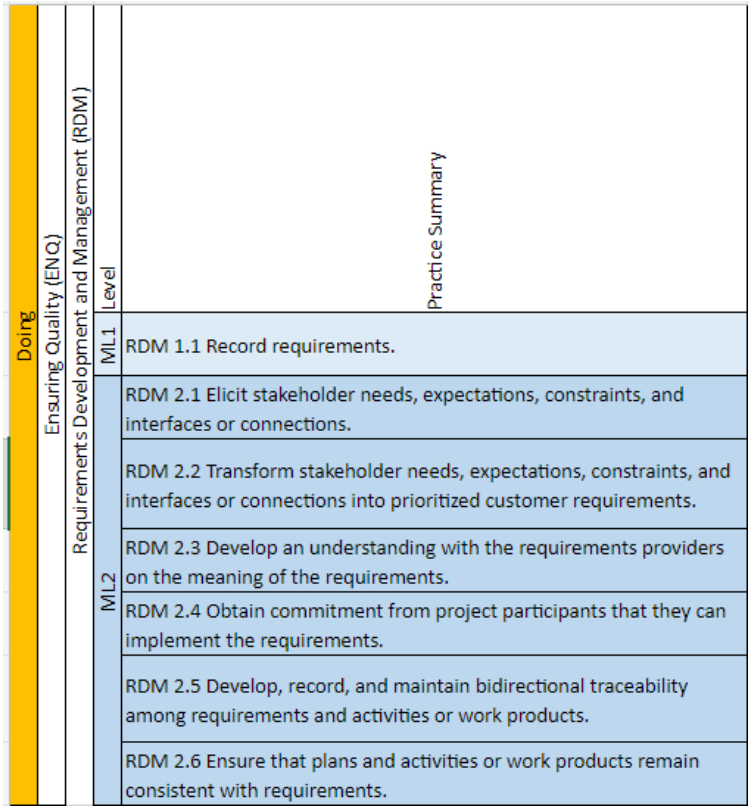


Figure 68 - Example of a Practice Area (CMMI)

In order to perform the mapping between CMMI and OpenUp, the Disciplines presented in OpenUp are introduced in each of the tables mentioned above, where the separation by tasks is performed, as shown in Figure 69. The Disciplines are divided by colors to facilitate the distinction of the areas presented in this framework.

Deployment	Review and Conform to Release Controls
	Plan Deployment
	Install and Validate Infrastructure
	Develop Release Communications
	Develop Backout Plan
	Verify Successful Deployment
	Package the Release
	Execute Deployment Plan
	Execute Backout Plan (if necessary)
	Deliver Release Communications
	Develop Training Materials
	Deliver Support Training
	Deliver end user Training
	Develop Support Documentation
	Develop User Documentation
	Develop Product Documentation
Test	Run Tests
	Implement Tests
	Create Test Cases
Environment	Verify Tool Configuration and Installation
	Set Up Tools
	Tailor the Process
	Deploy the Process
Development	Design the Solution
	Integrate and Create Build
	Run Developer Tests
	Implement Solution
	Implement Developer Tests
Requirements	Develop Technical Vision
	Detail System-Wide Requirements
	Detail Use-Case Scenarios
	Identify and Outline Requirements
Project Management	Request Change
	Plan Project
	Plan Iteration
	Manage Iteration
	Assess Results
Architecture	Envision the Architecture
	Refine the Architecture

Figure 69 - OpenUp Tasks by Category

Considering the information that has been previously mentioned, some examples of relationships will be presented, with reference to the coverage levels defined in Table 7. We can observe in Table 8, the relationship between the two frameworks under study. In the last column we present, as an example, the CMMI Practice Area "Requirements Development and Management (RDM), and its Practices for ML1 e ML2, that belongs to the Capability Area "Ensuring Quality (ENQ) from the Category "Doing". and we will have the coverage level between the OpenUP taks and CMMI Practices. Table 8 is only illustrative of an example of how the mapping tables are structured, because the same procedure was performed for all 29 CMMI Practice Areas.

Table 8 - Relationship between CMMI and OpenUp

		Doing						
		Ensuring Quality (ENQ)						
		Requirements Development and Management (RDM)						
		CMMI Practice Summary						
		ML1	ML2					
OpenUP Disciplines	OpenUp Tasks	RDM 1.1 Record requirements.	RDM 2.1 Elicit stakeholder needs, expectations, constraints, and interfaces or connections.	RDM 2.2 Transform stakeholder needs, expectations, constraints, and interfaces or connections into prioritized customer requirements.	RDM 2.3 Develop an understanding with the requirements providers on the meaning of the requirements.	RDM 2.4 Obtain commitment from project participants that they can implement the requirements.	RDM 2.5 Develop, record, and maintain bidirectional traceability among requirements and activities or work	RDM 2.6 Ensure that plans and activities or work products remain consistent with requirements.
Architecture	Refine the Architecture							
	Envision the Architecture							
Project Management	Assess Results							H
	Manage Iteration						M	M
	Plan Iteration							
	Plan Project	M	M			H		
	Request Change							
Requirements	Identify and Outline Requirements	H	H					
	Detail Use-Case Scenarios			H				
	Detail System-Wide Requirements			H				
	Develop Technical Vision			H	H	H		
Development	Implement Developer Tests							
	Implement Solution							
	Run Developer Tests							
	Integrate and Create Build							
	Design the Solution			H	M	H	H	H
Enviroment	Deploy the Process							H
	Tailor the Process				H	H		
	Set Up Tools							
	Verify Tool Configuration and Installation							
Test	Create Test Cases							M
	Implement Tests							M
	Run Tests							H
Deployment	Develop Product Documentation						H	
	Develop User Documentation						H	
	Develop Support Documentation						H	
	Deliver end user Training							
	Deliver Support Training							
	Develop Training Materials							
	Execute Backout Plan (if necessary)						M	H
	Execute Deployment Plan							
	Package the Release							
	Verify Successful Deployment							H
	Develop Backout Plan							
	Develop Release Communications							
	Install and Validate Infrastructure							
	Plan Deployment						H	
	Review and Conform to Release Controls							H

Once the mapping tables with information related to CMMI and OpenUp have been defined, and the technological architecture, the next step involves converting the two frameworks through a graph database technology. The technological architecture previously defined, in this specific case the execution of the left path will be demonstrated in Figure 70.

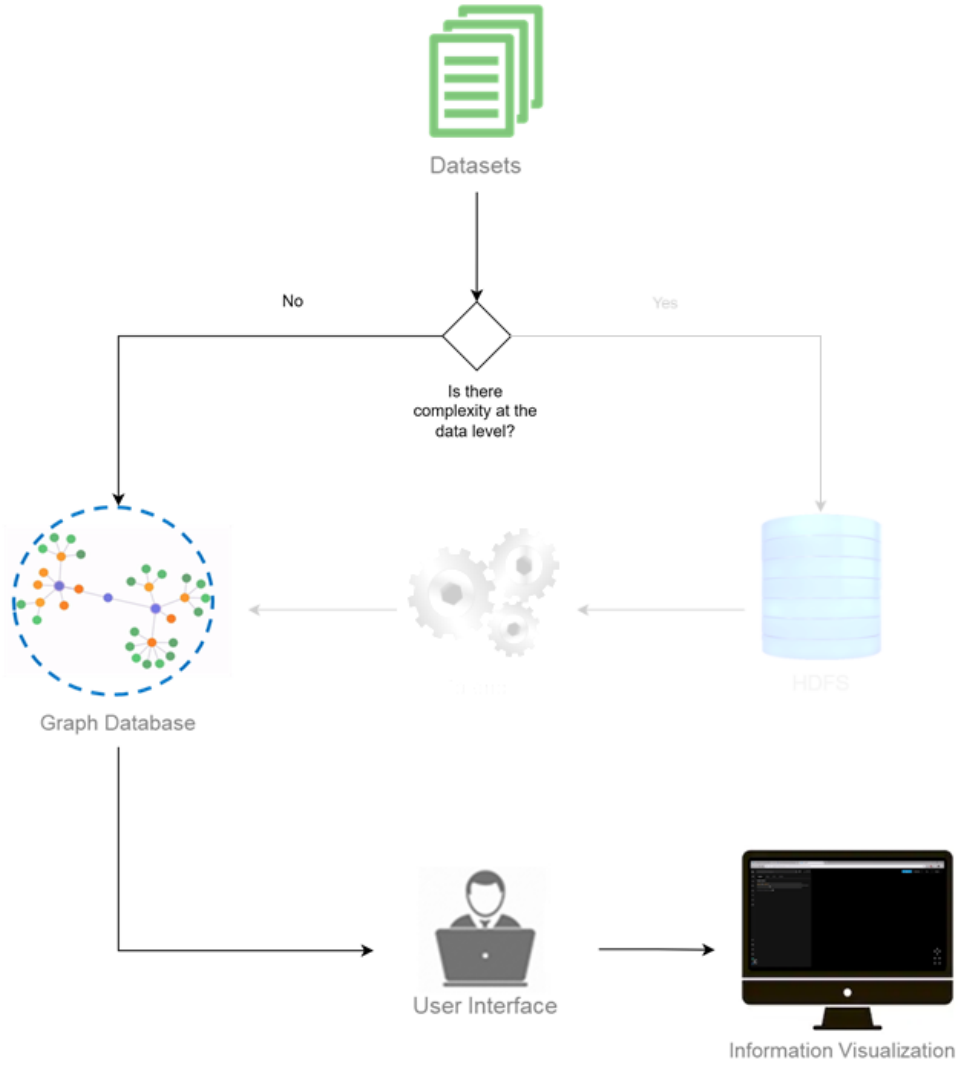


Figure 70 - Technological Architecture for mapping information in graph databases

Therefore, based on the analysis made to ITEC it was decided, in the project context, that regarding CMMI we will map the maturity levels 1 and 2. In Figure 71, we can see the graph mapping of CMMI maturity levels 1 and 2, which are used during this work and will help the company perform its daily tasks with the best possible quality.

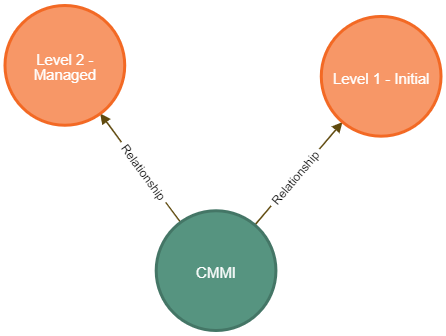


Figure 71 - CMMI Maturity Levels Mapping

After converting in graph the two maturity levels that are used in this demonstration case, the relationships between the CMMI Categories are presented next. CMMI is divided into four distinct categories, composed of different areas of capability. The four categories are related to each other because there is a dependency between the procedures that each one contains. Thus, it is possible to observe this relationship in Figure 72.

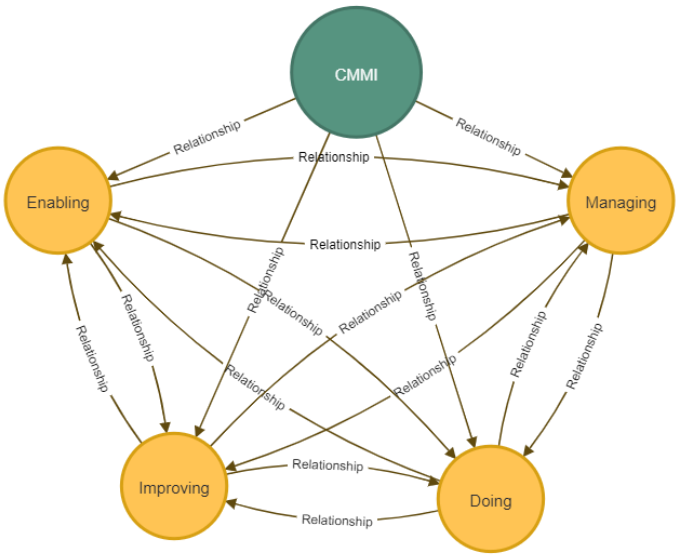


Figure 72 - Mapping between the different CMMI categories

After defining the relationships between the categories, the mappings performed for each of the four existing categories are shown, bearing in mind the Capability Areas, maturity levels and practices. Firstly the "Doing" Category and all its relationships are presented.

In Figure 73, you can see the mapping, in the graph database, of the "Doing" category, which is divided into four Capability Areas. We also have their respective Practice Areas and Practices. That

said, and in order to better understand Figure 73, the green color represents CMMI, the yellow color represents the CMMI categories, the blue color represents the four capability areas in the Doing category, the red color represents the Practice Areas of each capability area, the brown color represents the Practice and finally the orange color represents the different maturity levels. Since this procedure is performed in a similar way for all four CMMI categories, we will only illustrate as an example the Doing category.

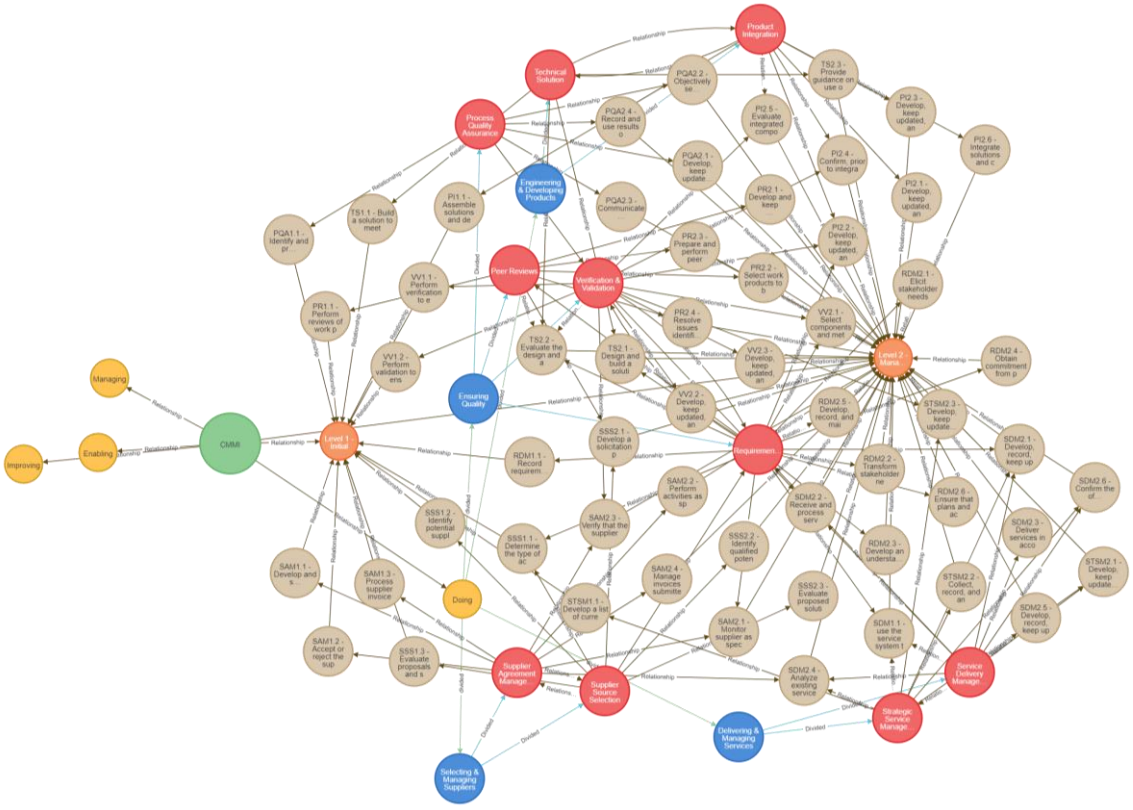


Figure 73 - Mapping the Practice Summary of CMMI's Doing category

For OpenUp we mapped its tasks, categorized by disciplines, in a graph database technology and proceeded in a similar way as previously described in relation to CMMI. Thus, in the case of OpenUp, and considering the study of this demonstration case, the goal is to map the tasks present in this framework relating them later with the information related to CMMI.

The OpenUp tasks, which will be mapped, are divided into disciplines, with the goal of being easier to use and understand by the user. This way, the relationship between this framework and CMMI also becomes more perceptible and intuitive. In Figure 74, we can see in red the OpenUp, in yellow the reference to the OpenUp Disciplines, with the dark brown color we present the different groups

of disciplines and finally, with the dark blue color we represent the names of the tasks that are performed in each discipline.

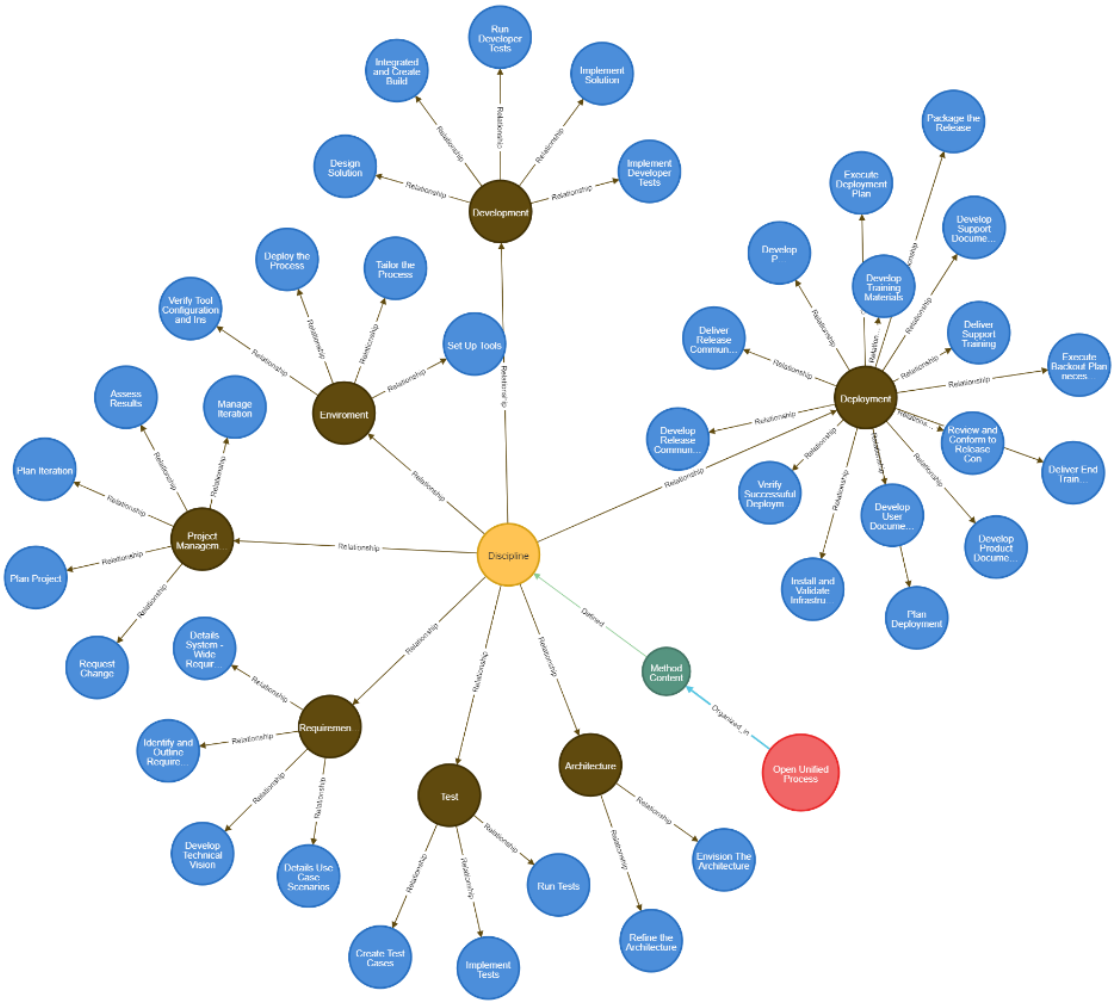


Figure 74 – OpenUP tasks and disciplines mapping

Finally, the data from CMMI and OpenUP are cross mapped, supported by information previously classified in tables (Table 8). For a better visualization and understanding, the mapping will be presented below, divided by the CMMI 2.0 categories. The mapping is divided by groups (Figure 75), meaning, brown is the OpenUP disciplines and blue the OpenUP tasks, while in orange is the CMMI maturity levels, beige the CMMI category, in blues are the CMMI capability areas, pink the practice areas and in green the Practices. The Managing category is divided in 3 Capability Areas identified as: Managing Business Resilience (MBR) divided in 3 PAs; Managing the Workforce (MWF) divided by 2 PAs; and Planning & Managing Work (PMW) represented by 3 PAs. Since this procedure is done in a similar way for all OpenUp disciplines, we will only be presenting the Managing Discipline as an example.

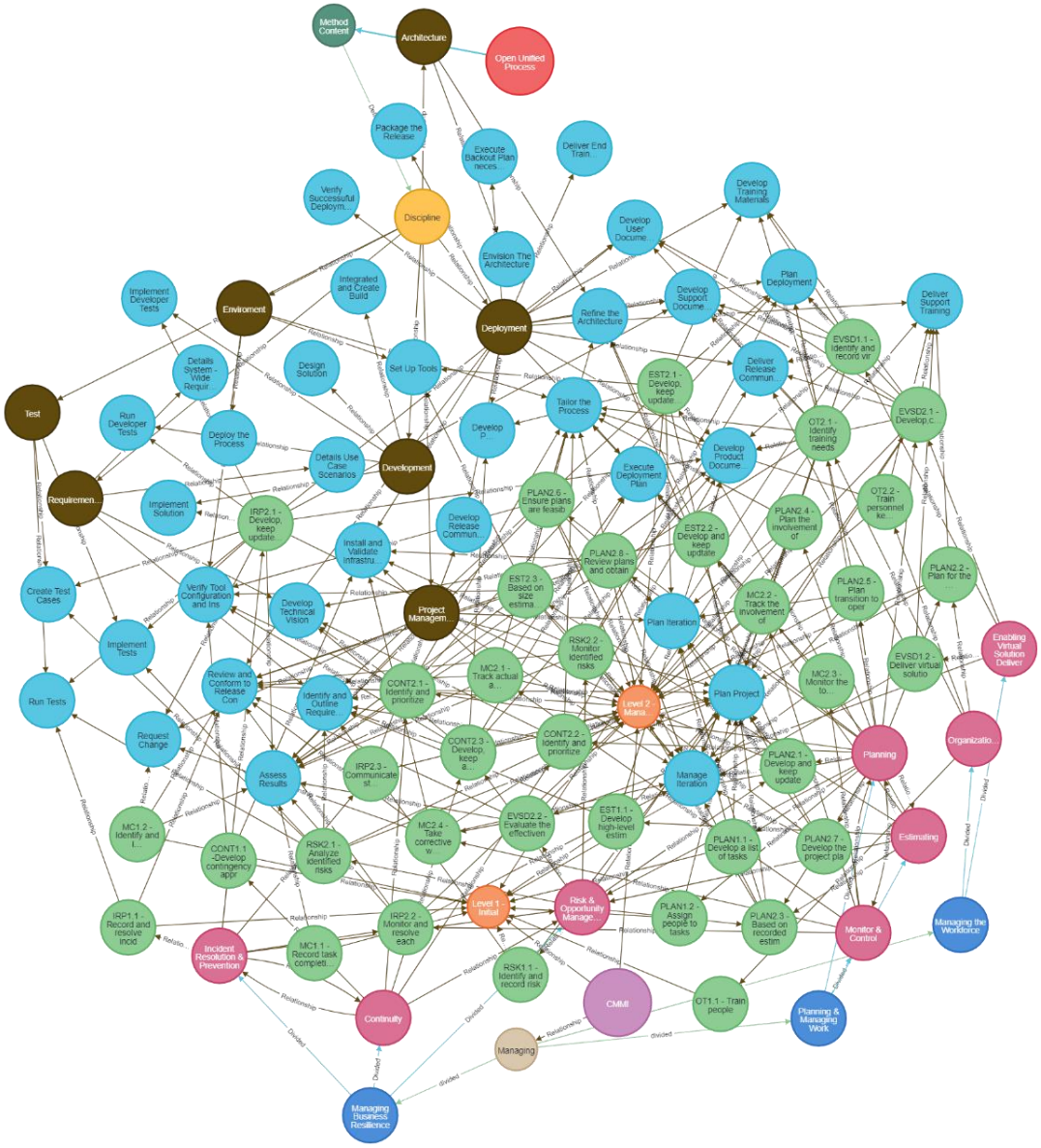


Figure 75 – Mannaging Category Mapping

In order to allow an easy visualization, the mapping results are presented by categories to make the consequent understanding of the mapping more intuitive. However, the final result is a large set of connection information charted in one graph dataset as presented in Figure 76.

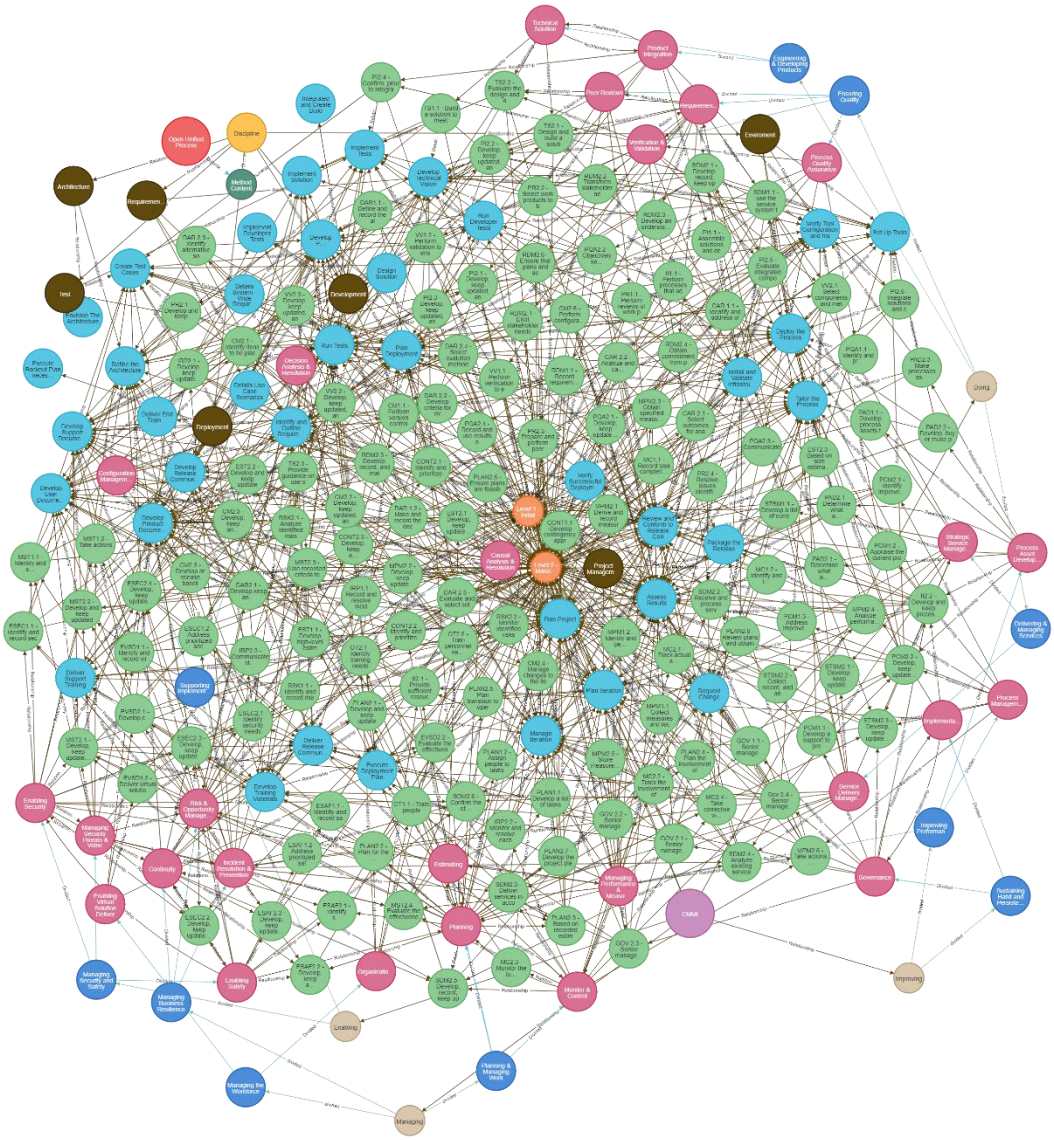


Figure 76 – Full alignment mapping

6.4 Conclusions

The work presented in this chapter focuses on presenting a technological architecture that we developed in order to represent ontologies using complex networks.

Thus, in a first moment the different instances necessary for the conception of the technological architecture are made known. Throughout each instance tests are made to understand if the development of the architecture would meet the needs of the projects where it is being applied and validated. In this way, in addition to contextualizing and explaining the technological architecture for representing ontologies through complex networks, we also validate its applicability in

demonstration cases that we call research projects. In this chapter 6, the demonstration cases used to test and validate the technological architecture are the STVgoDigital project, related to the textile sector and industry, and also the ITEC project, related to a robotics industry. As the technological architecture has two paths that can be followed, each of the projects follows one of these two paths. In this way, it becomes possible to test the technological architecture in its entirety.

However, we would like to point out that this technological architecture allows more than just the representation and visualization of ontologies. The architecture allows, through the definition of a set of parameters, to automate information mappings from various information sources and systems.

6.5 References

Ariffin, K. A. Z., & Ahmad, F. H. (2021). Indicators for maturity and readiness for digital forensic investigation in era of industrial revolution 4.0. *Computers & Security*, *105*, 102237.

Bindi, B., Bandinelli, R., & Rinaldi, R. (2016). RFID and eBIZ implementation in the Textile and Clothing Industry: Evidence of a pilot project. *Proceedings of the 21st Summer School Francesco Turco*, 156–160.

Borg, A., Sandahl, K., & Patel, M. (2007). Extending the openup/basic requirements discipline to specify capacity requirements. *15th IEEE International Requirements Engineering Conference (RE 2007)*, 328–333.

Brutti, A., De Sabbata, P., Scalia, M., Krzystek, K., Aguado, J. I., Ricchetti, M., & Baker, M. (2012). eBusiness in fashion industry: Interoperability standardisation meets industry supply chain. *Proceedings of EChallenges E-2012 Conference*.

Chen, D., Doumeingts, G., & Vernadat, F. (2008). Architectures for enterprise integration and interoperability: Past, present and future. *Computers in Industry*, *59*(7), 647–659.

Cossentino, M., Hilaire, V., Molesini, A., & Seidita, V. (2014). *Handbook on agent-oriented design processes*. Springer.

EBIZ. (2017). *eBIZ 4.0: the third phase of the eBIZ Initiative*.

Grossi, L., Calvo-Manzano, J. A., & San Feliu, T. (2014). High-maturity levels: achieving CMMI ML-5 in a consultancy company. *Journal of Software: Evolution and Process*, *26*(9), 808–817.

Hoda, R., Salleh, N., Grundy, J., & Tee, H. M. (2017). Systematic literature reviews in agile software development: A tertiary study. *Information and Software Technology*, *85*, 60–70.

Otto, B., Lohmann, S., Steinbuß, S., & Teuscher, A. (2018). IDS reference architecture model. *Industrial Data Space. Version*, *2*.

Purnama, I. D. C., Mursanto, P., Hazmi, F., & Oriza, I. I. D. (2020). Development of Android-Based Counseling Application Using Open Unified Process (OpenUP). *2020 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, 457–462.

Ruiz, P., Agredo, V., Camacho, C., & Hurtado, J. (2018). A canonical software process family based on the Unified Process. *Scientia et Technica*, *23*(3), 369–380.

Seliverstov, A. (2015). *An ontological approach for modeling spatiotemporal information with applications in transportation*.

Singh, A., & Gill, S. S. (2020). Measuring the maturity of Indian small and medium enterprises for unofficial readiness for capability maturity model integration-based software process improvement. *Journal of Software: Evolution and Process*, *32*(9), e2261.

Staples, M., & Niazi, M. (2008). Systematic review of organizational motivations for adopting CMM-based SPI. *Information and Software Technology*, *50*(7–8), 605–620.

Teixeira, D., Pereira, R., Henriques, T., Silva, M. M. Da, Faustino, J., & Silva, M. (2020). A maturity model for DevOps. *International Journal of Agile Systems and Management*, *13*(4), 464–511.

PART IV - CONCLUSIONS

—— This page is intentionally left blank ——

CHAPTER 7

CONCLUSIONS

Summary: This chapter concludes this thesis. It describes the overall focus of the conducted work. Additionally, it summarizes the research efforts as well as the scientific results of this thesis.
Finally, this chapter ends with a set of proposed future work.

CHAPTER TABLE OF CONTENTS

7.1 Focus of the Work 211
7.2 Scientific Outputs 213
7.3 Critical Analysis and Future Work 218
7.4 References 221

—— This page is intentionally left blank ——

CHAPTER 7

CONCLUSION

"Two things that fill my soul with growing admiration and respect, the more intensely and frequently my thought occupies it: the starry sky above me and the moral law within me."

– Immanuel Kant

7.1 Focus of the Work

The use of models throughout the software development lifecycle (SDLC) – typically beginning with the identification of business needs or opportunities, followed by requirements, design, implementation, testing, and deployment – reflects the knowledge that stakeholders possess at a given phase or stage regarding the solution being developed. As the SDLC advances, models often incorporate more information about the software solution's behavior than the business context in which it will be deployed.

By supporting iterative and incremental development and gathering feedback and learning for ongoing adaptation, ontological development approaches reduce the level of abstraction of data modeled as they advance in a gradual and continuously updated manner. Tracing model evolution is necessary for them to achieve their intended function, which is to assist teams in developing ontologies and complex networks.

Organizations have increasingly embraced ontologies in order to facilitate organizational interoperability or as a means of expressing organizational context information. As detailed in Chapter 3, their increasing usage in larger contexts has led to the development of approaches for

implementation at scale, which are still the topic of study, with a focus on how to specify and implement them.

While acknowledging the significance of methodology for developing ontologies, particularly in software development contexts, these approaches lack a systematic framework for the use of such knowledge in scientifically-driven initiatives (R&D projects).

Thus, this thesis aims to answer some identified gaps reflected in the research question '' How to design ontologies to support the analysis of requirements engineering problems in information systems projects?"''.

In order to answer the previously mentioned research question, we developed three types of contributions that also answer the three objectives identified in chapter 1:

1. To evolve the 4-Step Rule Set Method for deriving domain terminologies;
2. To elaborate an approach for designing ontologies based on domain terminologies;
3. To construct a technological architecture for supporting the visualization of ontologies by using complex networks.

These three contributions are organized and detailed in chapters four, five and six, and in a certain way the application of these three contributions is related. In this way, we adapted the VModel so that it can respond to R&D projects and where the focus is on the development of ontologies. Thus, in chapter 4, we evolve the 4SRS method in order to derive terminologies from a specific domain (VModel vertical), in chapter 5 we build an approach that allows the construction of an ontology and in chapter 6 we define a technological architecture to represent the ontology through complex networks.

7.2 Scientific Outputs

This thesis aims to contribute to both the knowledge base space (i.e., scientific theories and methods added to "scientific" knowledge) and the environmental space in organizations (i.e., practices for adoption by organizations with software development teams). Due to the nature of the research conducted, namely through the development of scientific design research based on demonstration cases, the designed artifacts, methods, and processes are validated in both theoretical and practical ways.

As for the contributions to the environmental space, this thesis proposed a technological architecture in order to produce complex network mappings using graph database technologies. As for contributions to the knowledge base space, this thesis proposed adapting the original 4SRS method, deriving a new version called 4SRS-Onto. This method is composed of "steps" that software engineers can adopt to derive a catalog of terminologies guiding for the subsequent design of an ontology. In this thesis is also produced a new approach to designing ontologies. As stated in Section 7.1, the development of the approach for designing ontologies included research on topics such as currently existing methodologies and approaches for designing ontologies in order to understand how these developments are carried out and which phases constitute each of the current approaches.

Furthermore, it should be noted that although this thesis proposed the approach for designing ontologies, it is composed of a set of phases that can be adopted in the SDLC independently, allowing flexibility in defining the most appropriate ontology development process for a given domain. The contributions are (1) 4SRS Method for Ontological Design (2) Ontological Systematization Approach (3) Technological Architecture for Visualizing Complex Networks.

Contribution 1 (C1): 4SRS Method for Ontological Design

Numerous researches have enabled the evolution of VModel+4SRS into a method that already requires a substantial amount of labor. These include the improvement of architectures by applying an approach (Machado *et al.*, 2005), an extension to support the construction of a class diagram that complements the logical architecture (Santos & Machado, 2010), and another applicable to product line architectural modeling and an adaptation for automation purposes, as well as the

study of the health of relationships (Azevedo *et al.*, 2012). There is also substantial process-based work for product elicitation (Ferreira *et al.*, 2012) and the usage of SOA participants as opposed to conventional architectural elements to match business needs with services (Salgado *et al.*, 2015).

When selecting whether or not to incorporate an ontology component, a company's operational environment is always crucial. Before defining the firm, it is essential to identify and investigate its organizational environment, as well as its industry and connected ecosystem. Thus, we offer a traceable route between Data Characterization, initial and comprehensive requirements in UML use cases, a description of the to-be-used data model, specification of the data model, and ontological design of the defined schema. This suggestion is based on our past research and expertise with 4SRS, and the V-model proposed by Machado *et al.* is adopted for this approach.

At the junction of the VModel's descending and ascending branches lies the 4SRS method. Different variants of the 4SRS method (Machado *et al.*, 2005) have demonstrated their adaptability in generating a component architecture as the foundation for constructing an information system in various circumstances. As the ontology building process is currently connected with information system development (De Nicola & Missikoff, 2016), we chose to apply this method to the ontology design process.

Contribution 2 (C2): Ontological Systematization Approach

In general, an ontology should provide suitable workflows and the ability to design process sequences for the development and maintenance of ontologies defined by certain methods. The creation of ontologies is largely a process of integrating knowledge. In fact, the only viable approach to construct a decently complicated ontology is through extracting information from other sources (Wróblewska *et al.*, Podsiady-Marczykowska, Bembenik, Protaziuk, and Rybiski, 2012).

We propose to align our approach with the principles of the DIKW pyramid in light of these considerations and the analysis conducted in chapters 2 and 3 of the current literature, specifically addressing the concepts of data, information, knowledge, and wisdom.

Therefore, the following phases comprise an ontological approach for systematizing information:

- First phase: Data Characterization;

- Second phase: Information Specification;
- Third phase: Ontology Mapping

Initial emphasis should be placed on studying the company's internal processes, recorded activities, and departmental duties, as well as the actors doing them, in order to assist the selection and analysis of data sources (Data Characterization Phase). After selecting the data sources, the next phase is to begin indexing the terminologies existing in the various sources, therefore creating a glossary of terms for the subject under investigation. After the terminologies have been catalogued, we arrange and classify them into Entities. This phase concludes with the development and building of a synonyms catalog, where each previously discovered phrase must be paired with a collection of synonyms.

Next, during the Information Specification phase, the analysis should be aligned with the current data model for the domain of interest. In other words, in this phase of the specification, we use the information gathered in the previous phase (Data Characterization) to begin determining the attributes of each entity. After defining the entities' attributes, we create a data model to establish their relationships. We use a UML class diagram to accomplish this phase of linking and identifying characteristics for each object.

In the third phase, Ontology Mapping, we use the information specified in the preceding phase (Information Specification) to construct an ontology schema adopting graph database technologies. Then, we incorporate this ontology information into a user-friendly visualization tool so stakeholders may apply filters, change the database, and add new terms and connections.

Contribution 3 (C3): Technological Architecture for Building Complex Networks

After gathering requirements and modeling business processes in the companies and projects under investigation, we discovered that many of the previously described processes are not carried out in their entirety and are frequently not carried out correctly. Thus, and taking into account all interactions with the various stakeholders of the companies, we identified the definition and implementation of a technological architecture for the visualization of information using complex networks as one of the areas where we could intervene and provide assistance.

This architecture specification will allow employees to have access to an intuitive visualization tool that can be customized via user interface interactions. These visualization components also provide the visualization of dynamic data catalogs in which all terms are standardized.

As seen in chapter 6, the architectural approach is then specified, and it permits two paths. The data should be kept in a cluster if the data is complex (more than one dataset and more than 1000 lines of information in each dataset). Then, a Talend pipeline must be established to enable the connection to both HDFS and graph database technologies. In Talend, we also create a pipeline that enables us to specify the terminologies and relationships we desire for the visualization technology's output. Once this phase has been designed, the pipeline should be executed and the data should be mapped to the graph database and given to the end user via an interface.

The alternative potential option, on the other hand, is designed for scenarios with less data complexity included in the datasets. In situations when there is just one dataset containing fewer than one thousand rows of data, neither the HDFS repository nor a Talend pipeline is required. The dataset's information is translated directly to the database using graphs, and the results are displayed to the user via an interface.

As a final outcome, we will have a complex network maintained in graph database technology, which will be gradually fed back.

The contributions of these research work result from different demonstration cases. The way the contributions of the demonstration cases are organized for the research contributions is summarized in Table 9. As described in the column "Observations" in Table 9, each project can contribute differently to the research contribution. For example, the STVgoDigital project provides two distinct contributions. The first contribution is to implement the 4SRS-Onto method with the aim of building an ontology that supports interoperability in the textile industries. The second contribution is to develop a technological architecture to visualize information using complex networks where the main focus is to visualize the dependencies between the two reference architectures used (eBIZ and IDS).

The PHC, F3M and CityCatalyst projects, despite having different objectives, contribute to apply and validate the ontological systematization approach that we idealized and developed. Finally, the ITEC project provides another contribution distinct from STVgoDigital at the level of technological architecture to visualize information based on complex networks (Contribution 3).

Table 9 - Demonstration case's inputs towards the research contributions

Demonstration case	C1	C2	C3	Observations
STVgoDigital	x		x	For contribution 1 in STVgoDigital, 4SRS-Onto is used to derive terminologies for the textiles and clothing industry, enabling the development of a comprehensive terminology catalog. In the same project, the mapping of terminologies and eBIZ and IDS frameworks across complex networks is also conducted (contribution 3).
PHC: Voice Interaction Framework		x		The PHC project uses contribution 2, as we followed our own methodology to develop the Ontology that forms the basis of the PHC voice assistant.
F3M: Interoperability and Digital Thread domain for a More Competitive Textile Industry		x		The F3M project uses contribution 2 because we followed our own methodology for developing an Ontology to connect and standardize textile industry terminologies.
CityCatalyst: Normalized City Analytics		x		The Citycatalyst project uses contribution 2, as we developed an Ontology to standardize and relate smart city terminology using our own methodology. In this particular instance, we employ terms from four Portuguese municipalities (Guimarães, Porto, Famalicão, and Aveiro). The purpose of this standardization is to enable municipalities to exchange data in a single, standardized format in the future.
ITEC: A Guideline to Software Development			x	In the ITEC project, the ontology is constructed using complex networks. This project's objective is to establish a relationship between the CMMI and OpenUp frameworks in order to analyze and comprehend the level of overlap between them.

Each of the demonstration cases relate to a funded R&D project. Which is to say that this thesis work includes scientific outputs from STVgoDigital, CityCatalyst, F3M, PHC and ITEC projects. As mentioned, the DSR cycles are performed in parallel, where a set of demonstration cases contribute for the development and evaluation of the artefacts. In general, a demonstration case is not specific to one DSR cycle. The projects are used within different contributions for each research objective, but they overlapped in time between each other.

7.3 Critical Analysis and Future Work

Throughout this section, we conduct an analysis of the contributions made within the scope of this dissertation, and we also identify some questions that we believe remain unanswered. These same points may also serve to improve our current contributions in the future, as well as point to and envision new contributions based on our techniques and methods for constructing ontologies. Then, we perform a critical analysis of our contributions, analyzing each chapter of the contribution. Chapter 4 adapts the 4SRS method to the design of ontologies, Chapter 5 develops a methodology for designing ontologies, and Chapter 6 develops an architecture for constructing ontologies using complex networks.

Concerning chapter 4, are focused on the 4SRS technique. Thus, we modified the original method so that it is possible to derive terminologies based on specifications. This modification to 4SRS results in a new variant to aid in the creation of Ontologies. To use this method, which we refer to as 4SRS-Onto, we add it to the vertex of a V-Model. As a result, the left branch of the "V" retains its original configuration, and work continues on the requirements gathering and modeling component. Regarding this aspect, it is our understanding that, in the future, each component of the left branch of the "V" should be reviewed in order to facilitate a better alignment with the remaining phases.

Analyzing the Rational Unified Process (RUP), i.e., its phases (Inception, Elaboration, Construction, and Transition), as well as the Disciplines (Business and Requirements Modeling, Analysis and Design, Implementation, Testing, Deployment, Configuration and Change Management, Project Management and Environment), is something that has not been done, but we believe it will be

important in the future. This analysis will allow us to frame and align the V-Model Onto with the RUP so that we can determine in which phases and disciplines it is applicable.

Regarding chapter 4, our primary objective is to identify a method that would permit us to derive terminologies based on the identified requirements and use cases in a given context. As stated previously, we used the STVgoDigital project for this purpose, where we applied and validated the method.

In the fifth chapter, we develop and present a contribution to the construction of ontologies. This contribution is based on an approach consisting of three phases (Data Characterization, Information Specification, and Ontology Mapping) that are executed sequentially and can be comprised of one or more iterations based on the requirements of the project. This proposed approach can be executed with the execution of 4SRS-Onto as its antecedent, or it can be executed even if the project lacks sufficient use cases to be considered complex. Regarding this contribution, and taking into consideration the demonstration cases that we used to validate the proposed approach, we believe that it would be beneficial to associate this ontology building approach with a later stage of the software development process, namely the deployment level. These future developments will allow us to determine how our approach behaves and whether it retains its value and utility in later stages.

The chapter 6 contributions represent the conclusion of the ontology design. As demonstrated in Chapter 6, ontologies are constructed using complex networks, and we used graph database technology for this purpose. Nevertheless, we distinguish two options for doing so, taking into account the level of complexity and/or potential demands of the project in which the ontology construction is applied. Thus, it is possible to construct the ontology in a "manual" manner using queries in the graph database to create nodes and relationships between each node, and it is also possible to construct the ontology in a more automated manner. In this second perspective, we employ three distinct technologies to construct ontologies.

In the first instance, the terminology catalogs are stored using a technology, typically HDFS. Talend is the second technology we adopt; it enables us to create a pipeline in which we define the rules for terminology creation and relationship definition. Finally, we employ a graph database technology, primarily Neo4J, where the mapping is performed according to the Talend-defined

pipeline. As a means of applying and validating this contribution, we used two projects that are distinct in both context and level of complexity. The projects are ITEC and STVgoDIgital.

As previously stated, chapter 6 represents the culmination of the ontology design by using complex networks. Nonetheless, we note that it would be advantageous to continue this work by integrating and demonstrating the integration typology with technologies that generate dashboards from the mapped ontology. However, not everything is positive, so we denote that it becomes difficult to manage the visualization and comprehension of complex networks when faced with enormous volumes of information. In other words, highly complex contexts are difficult to analyze when complex networks are present. In addition to these characteristics, our use of complex networks differs somewhat from the typical use of complex networks, as we did not use the typical weights and metrics that characterize complex networks in the work presented throughout this dissertation. Thus, we recognize the use of complex networks with weighted relationships between nodes as a challenge for future projects.

Considering the critical analysis performed as well as the clarifications provided in this section 7.3 regarding future work, we identify some points that should be expanded upon in the future in order to evolve and clarify the work described and presented throughout this document.
Thus, at the VModel level, particularly with regard to the 4SRS-Onto method, we believe that in the future an analysis of the RUP must be conducted to determine in which phases and disciplines of the RUP this VModel fits.

In addition, it would be appropriate to consider and develop a method for automating the 4SRS-Onto procedure. We highlight this development as significant because, when defining and cataloging terminologies, we are often confronted with large volumes of data, and there is a way to facilitate their derivation.

Last but not least, we have identified another topic that we believe will warrant future research. Consequently, we deem it essential to focus on the alignment between the left and right branches of the V-Model. Thus, it will be possible to obtain a fully-framed V-model at the ontology design level.

7.4 References

Azevedo, S., Machado, R. J., & Maciel, R. S. P. (2012). On the Use of Model Transformations for the Automation of the 4SRS Transition Method. In M. Bajec & J. Eder (Eds.), *Advanced Information Systems Engineering Workshops* (pp. 249–264). Springer Berlin Heidelberg.

De Nicola, A., & Missikoff, M. (2016). A lightweight methodology for rapid ontology engineering. *Communications of the ACM*, *59*(3), 79–86.

Ferreira, N., Santos, N., Machado, R. J., & Gasevic, D. (2012). *Derivation of Process-Oriented Logical Architectures: An Elicitation Approach for Cloud Design. BT - Product-Focused Software Process Improvement - 13th International Conference, PROFES 2012, Madrid, Spain, June 13-15, 2012 Proceedings* (pp. 44–58).

Machado, R. J., Fernandes, J. M., Monteiro, P., & Rodrigues, H. (2005). Transformation of UML Models for Service-Oriented Software Architectures. *12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'05)*, 173–182.

Salgado, C., Teixeira, J., Santos, N., Machado, R.-J., & Maciel, R. S. (2015). A SoaML Approach for Derivation of a Process-Oriented Logical Architecture from Use Cases. In *Lecture Notes in Business Information Processing* (Vol. 201).

Santos, M. Y., & Machado, R. J. (2010). On the derivation of class diagrams from use cases and logical software architectures. *Proceedings - 5th International Conference on Software Engineering Advances, ICSEA 2010*, *August 2015*, 107–113.

—— This page is intentionally left blank ——

APPENDICES

—— This page is intentionally left blank ——

Appendix A – Information Visualization: Talend Pipeline.....226

Appendix B – General Terminologies Catalog STVgoDigital.....232

Appendix A – Information Visualization: Talend Pipeline

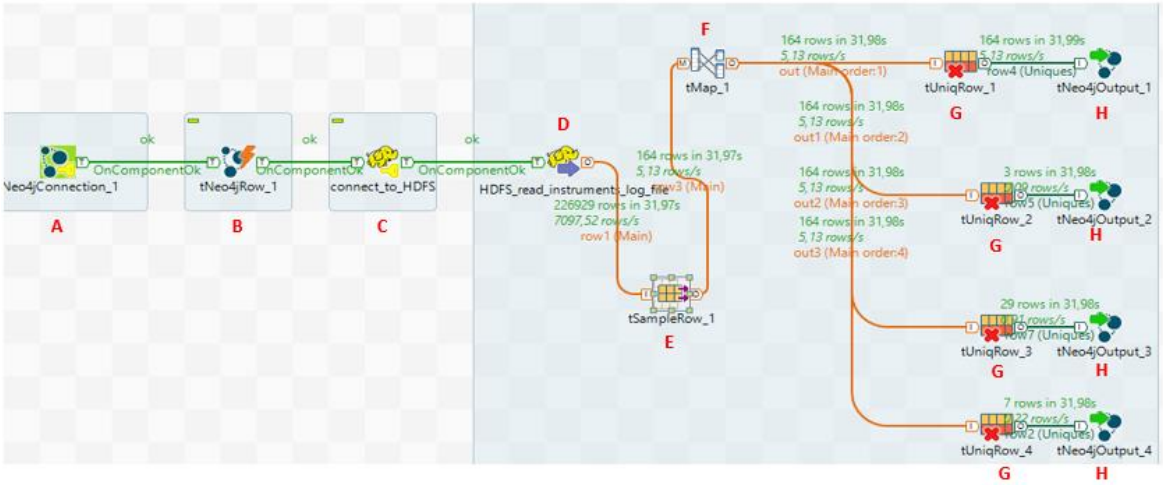


Figure 77 - Terminologies Output Pipeline

- A - Insert the link with the port number that you see in the step 7 from Neo4J and insert the Neo4J user name and password.

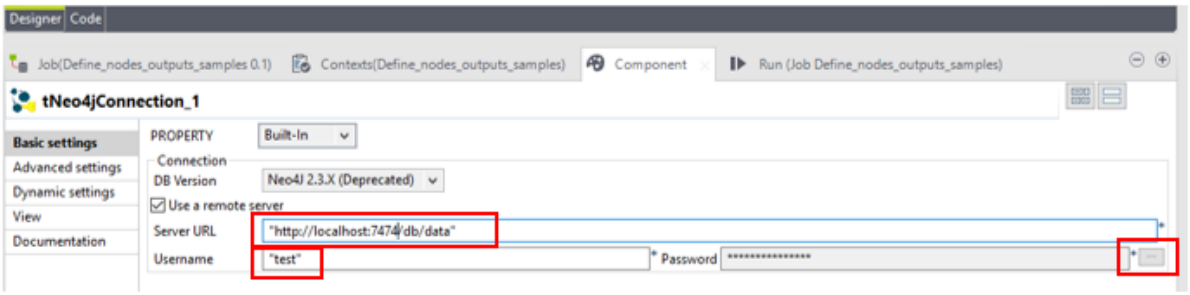


Figure 78 - Insert Neo4J configurations into Talend

- B - Select “Use an existing connection” and insert the query "Match (n) Detach delete n”. This query delete all nodes from neo4j to assure that the new execution occur correctly.

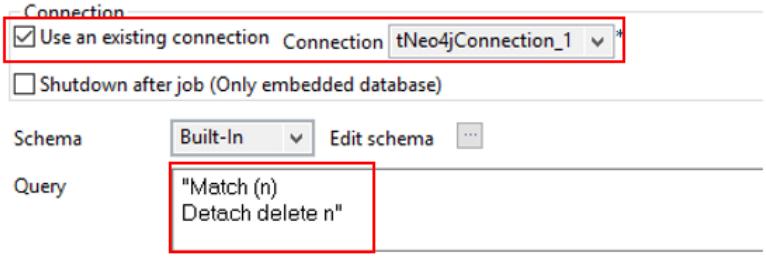


Figure 79 - Delete nodes from Neo4J

- C – This step is where you set the HDFS connection.

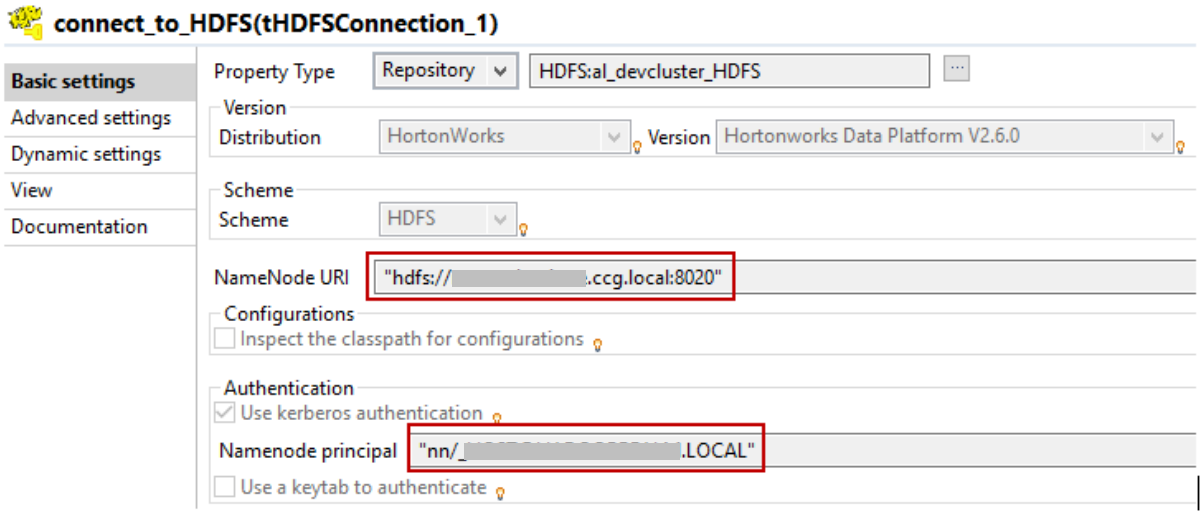


Figure 80 - HDFS connection into Talend

- D – This step associates the file we want to access.
- Therefore, it is necessary to select "Use an existing connection" (which is defined in the previous step) and then place the link where the .csv file is allocated in HDFS.

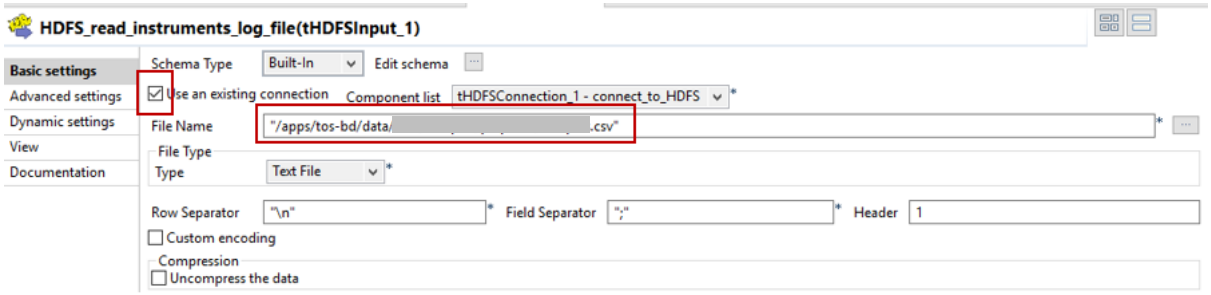


Figure 81 - Select file from HDFS to connect

- E – The "tSampleRow" component is used here to work with only a sample of the data. In this case, we select lines 1,5 the range between lines 10 and 70, plus the range between lines 2000 to 2100.

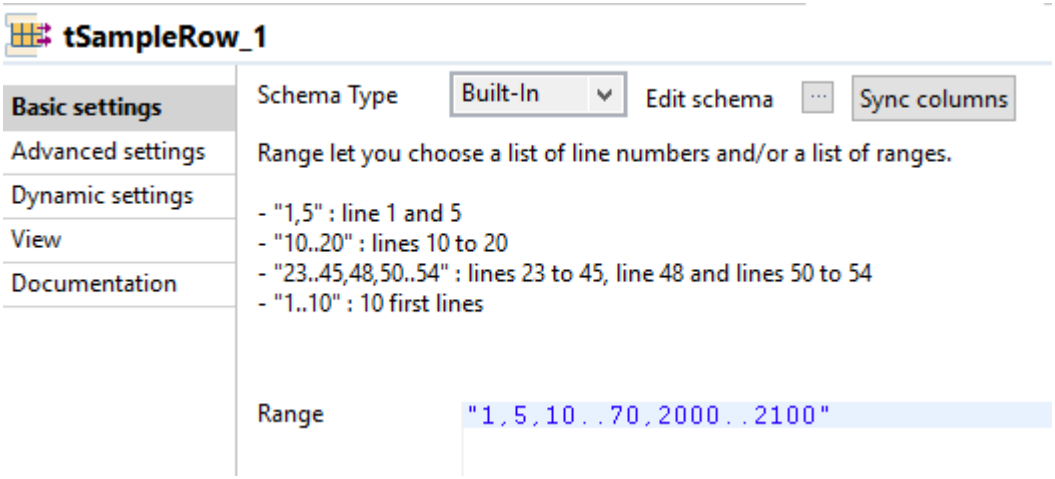


Figure 82 - tSample Row

- F – Double clicking on “tmap” will open a new window where you can then select how many outputs we want to represent in the future mapping.
- The left-hand table shows the attributes of the entire dataset, while the right-hand side are the defined outputs.

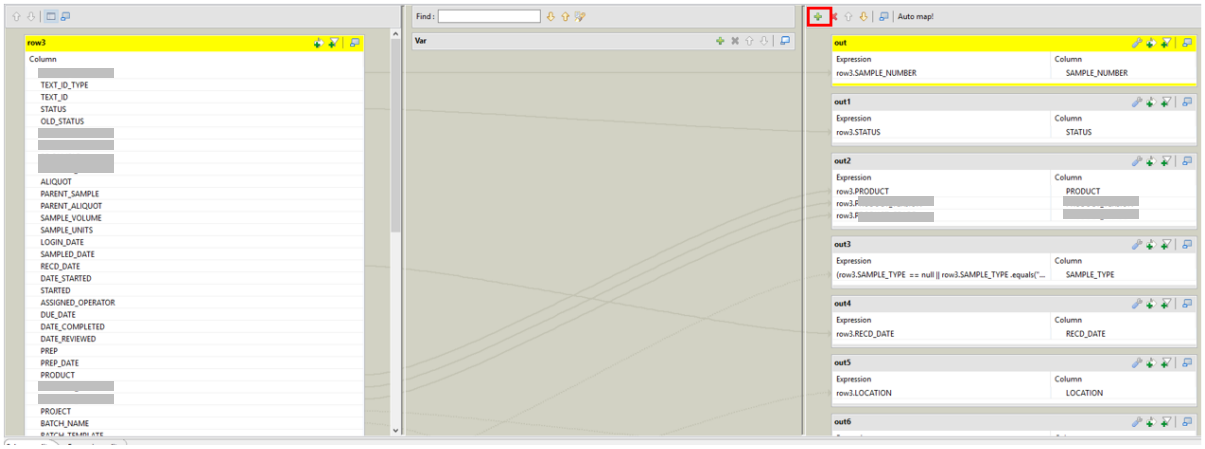


Figure 83 - Attributes mapping

  - 1° - click icon “+” to create new output.
  - 2° - make an association between the left table of which attributes you want for each output.
- G - In this step, double click on “tUniqRow”. Each output defined in the previous step is now chosen in this component as illustrated in the following Figure.

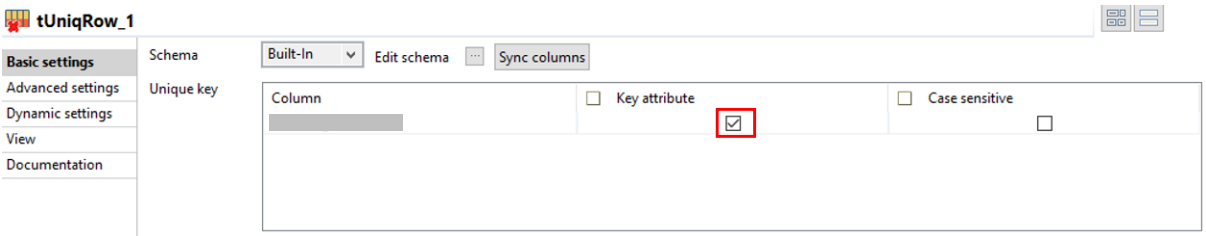


Figure 84 - tUniqRow component

- H – By clicking on “tNeo4jOutput” only once, we select “Use an existing connection”. This connection concerns the connection to the Neo4J set at the beginning (Neo4JConnection). In addition, we also define the label we want to identify this type of output in neo4J visualization.

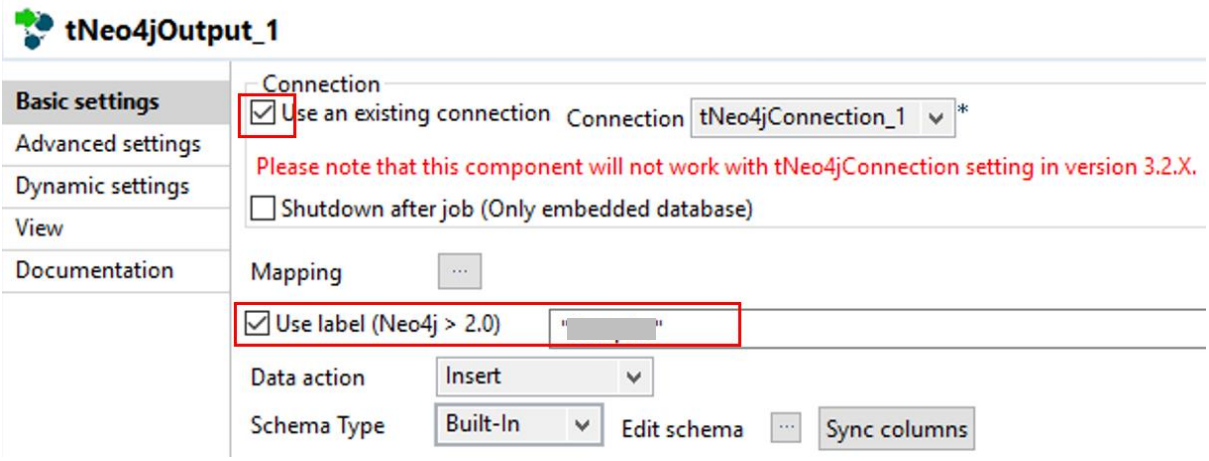


Figure 85 - tNeo4jOutputs connection definitions

Then double click on the “tNeo4jOutput” component where it will open a new window. Then we select the attribute, in this case example is called “SAMPLE_NUMBER”. After selecting the attribute, we select “creation index”. Here we click on the icon “+” to add a name and a key.

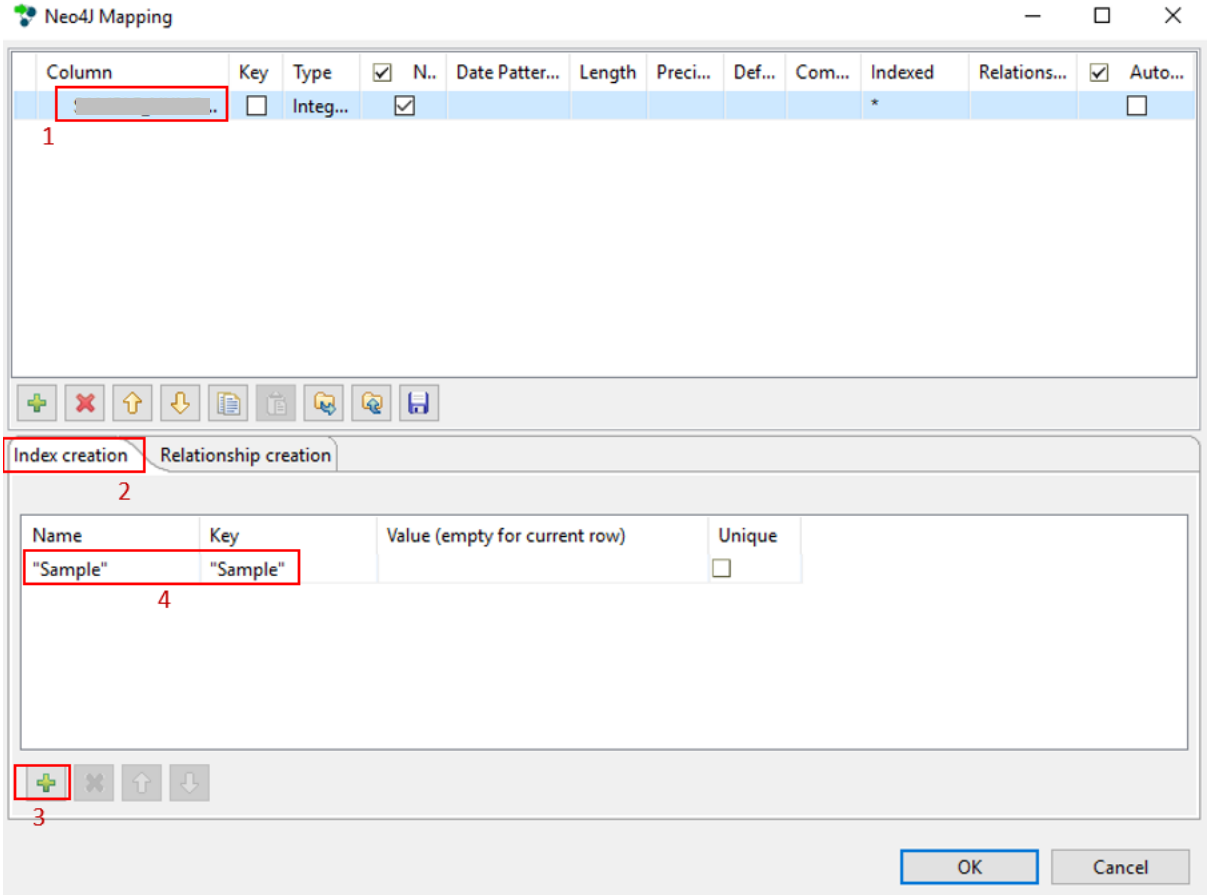


Figure 86 - Neo4J mapping

Regarding the definition of relationships between terminologies, we have also defined a pipeline in Talend which we illustrate below (Figure 87).

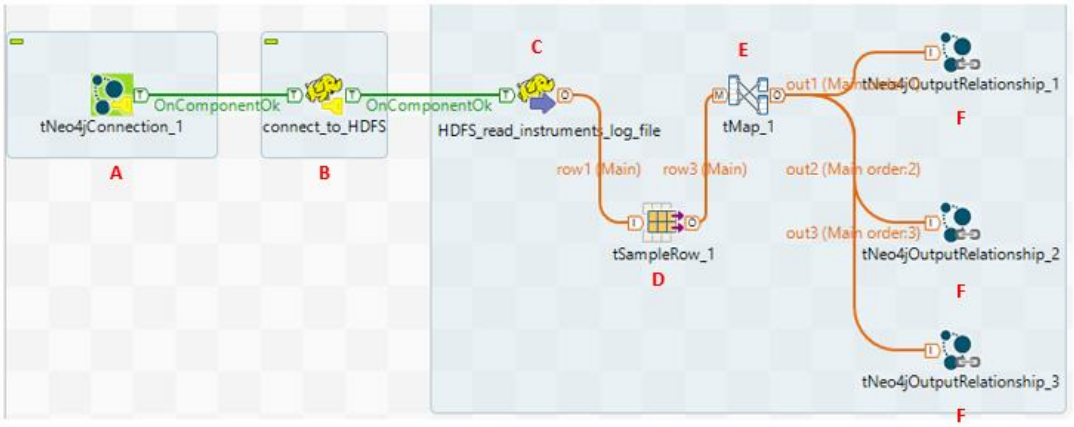


Figure 87 - Visualization of the architecture of all Talend jobs - relations

As in the previous section where we define outputs, here we will also demonstrate the process for associating the relationships between their outputs. Thus, the steps marked in the image with the letters "A", "B", "C" and "D" are filled in exactly the same way as in the previous section.

- E – double clicking on "tMap", we click on the icon "+" to create the associations between attributes.
  In this particular case we want to associate:
  - "STATUS" with "SAMPLE_NUMBER";
  - "SAMPLE_NUMBER" with "PRODUCT";
  - "SAMPLE_NUMBER" with "SAMPLE_TYPE";
  - "SAMPLE_NUMBER" with "RECC_DATE";
  - "PRODUCT" with "LOCATION".

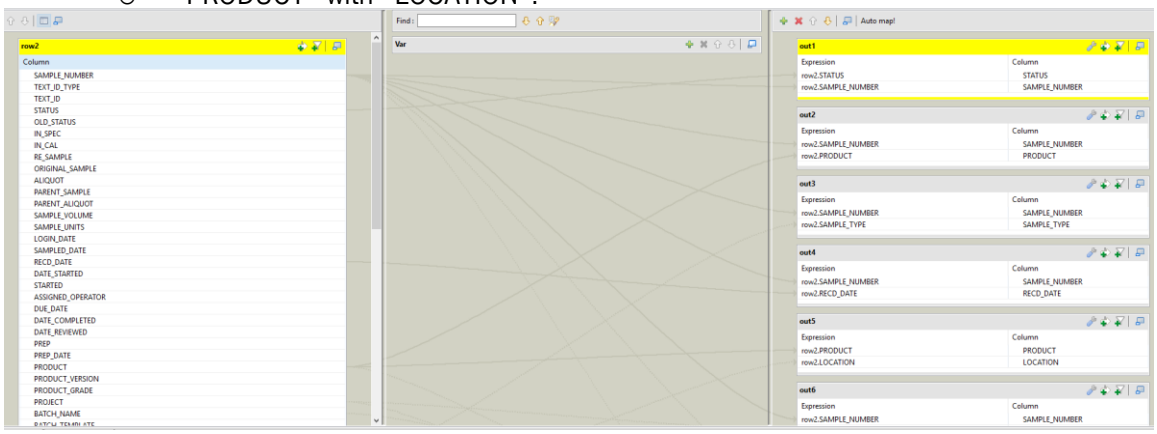


Figure 88 - Mapping relations

- F – double-clicking on “tNeo4jOutputRelationship” we select “Use an existing connection”. Then:
  - 1 – insert the name of relation;
  - 2 – select the value;
  - 3 – insert “index name” and “index key”. These indexes are used in Figure 77 (step H – section 8.3 Talend configuration).

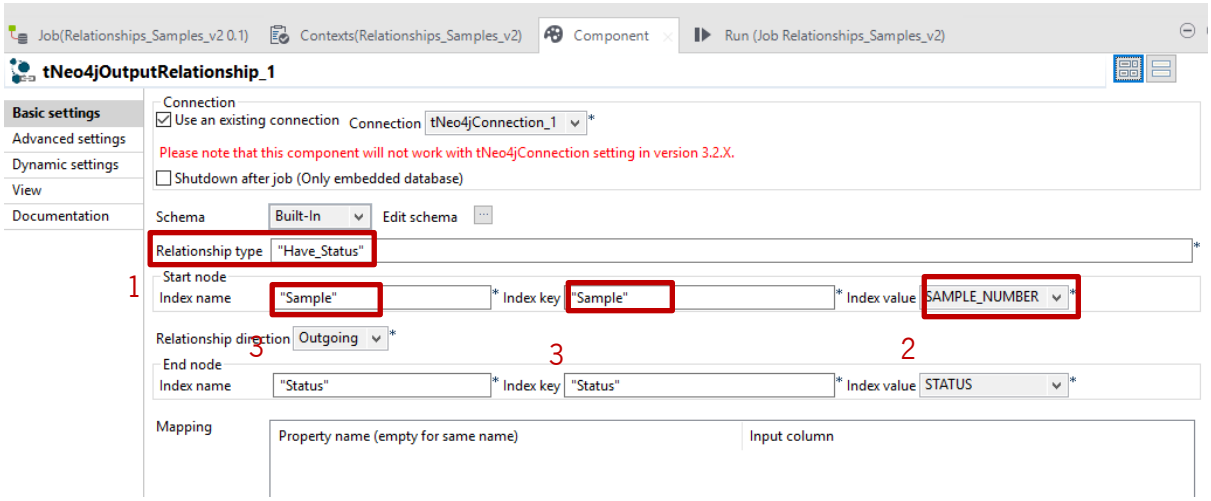


Figure 89 - Relationships configuration

Appendix B – General Terminologies Catalog STVgoDigital

ID	Terminologia	UC
TMI1	Textile Sector	eBIZ
TMI2	Footwear Sector	eBIZ
TMI3	TCF Sectors	eBIZ
TMI4	Brand Owner function	eBIZ
TMI5	Producer function	eBIZ
TMI6	Manufacturer function	eBIZ
TMI7	Retailer function	eBIZ
TMI8	Consumer	eBIZ
TMI9	Supplier function	eBIZ
TMI10	Downstream	eBIZ
TMI11	Upstream	eBIZ
TMI12	eBusiness	eBIZ
TMI13	Formal Standard	eBIZ
TMI14	Local Standard	eBIZ
TMI15	Proprietary Standard	eBIZ
TMI16	Traditional EDI Messages	eBIZ
TMI17	Vendor	eBIZ
TMI18	Stock risk	eBIZ
TMI19	Assortment	eBIZ
TMI20	Return of goods	eBIZ
TMI21	Exchange of goods	eBIZ
TMI22	Handling Unit	eBIZ
TMI23	Product	eBIZ
TMI24	Pre-product	eBIZ
TMI25	Component	eBIZ
TMI26	Material	eBIZ
TMI27	Stock	eBIZ
TMI28	Depot	eBIZ
TMI29	BM inventory	eBIZ
TMI30	VM inventory	eBIZ
TMI31	Distribution Centre Supplier	eBIZ
TMI32	Distribution Centre Service Supplier	eBIZ
TMI33	Distribution Centre Manufacturer	eBIZ
TMI34	Central	eBIZ

TMI35	Decentral	eBIZ
ID	**Terminologia**	**UC**
TMI36	By production order	eBIZ
TMI37	Collective delivery	eBIZ
TMI38	Seasonal delivery	eBIZ
TMI39	DC	eBIZ
TMI40	CD1	eBIZ
TMI41	CD2	eBIZ
TMI42	Bill at delivery	eBIZ
TMI43	Bill per use	eBIZ
TMI44	Credit note	eBIZ
TMI45	Allocation	eBIZ
TMI46	Floor-Management system	eBIZ
TMI47	Direct	eBIZ
TMI48	Connector	IDS
TMI49	Broker	IDS
TMI50	Data App	IDS
TMI51	Data Source	IDS
TMI52	Data Sink	IDS
TMI53	Data Consumer	IDS
TMI54	Data Provider	IDS
TMI55	Time	IDS
TMI56	Space	IDS
TMI57	Entity	IDS
TMI58	Keyword	IDS
TMI59	Term	IDS
TMI60	Type	IDS
TMI61	Message	IDS
TMI62	Connector Available Message	IDS
TMI63	Connector Update Message	IDS
TMI64	Connector Unavailable Message	IDS
TMI65	Resource Available Message	IDS
TMI66	Resource Update Message	IDS
TMI67	Resource Unavailable Message	IDS
TMI68	Description Request Message	IDS
TMI69	Query Message	IDS
TMI70	Request Message	IDS
TMI71	Message Processed Notification	IDS
TMI72	Rejection Message	IDS
TMI73	Result Message	IDS
TMI74	Response Message	IDS

ID	Terminologia	UC
TMI75	Certification	IDS
TMI76	Contract	IDS
TMI77	Participant	IDS
TMI78	Resource	IDS
TMI79	Instance	IDS
TMI80	Representation	IDS
TMI81	Provenance	IDS
TMI82	Quality	IDS
TMI83	Policy	IDS
TMI84	Pricing	IDS
TMI85	Operation	IDS
TMI86	Endpoint	IDS