REPUBLIQUE DU CAMEROUN
Paix – Travail – Patrie
****
UNIVERSITE DE YAOUNDE I
FACULTE DES SCIENCES
DEPARTEMENT DE INFORMATIQUE
****
CENTRE DE RECHERCHE ET DE FORMATION DOCTORALE EN SCIENCES, TECHNOLOGIE ET GEOSCIENCES
LABORATOIRE D'INFORMATIQUE ET APPLICATIONS

REPUBLIC OF CAMEROUN
Peace – Work – Fatherland
****
UNIVERSITY OF YAOUNDE I
FACULTY OF SCIENCE
DEPARTMENT OF COMPUTER SCIENCE
****
POSTGRADUATE SCHOOL OF SCIENCE, TECHNOLOGY & GEOSCIENCES
LABORATORY OF COMPUTER SCIENCE AND APPLICATIONS

SEMANTIC-AWARE EPIDEMIOLOGICAL SURVEILLANCE SYSTEM: APPLICATION TO TUBERCULOSIS IN CAMEROON

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy in Computer Science

Par : **JIOMEKONG AZANZI FIDEL**

Sous la direction de
Professor Maurice TCHUENTE
University of Yaoundé I, Cameroon
Professor Gaoussou CAMARA
Université Alioune Diop de Bambey, Sénégal

Année Académique : 2020

Updated protocol list of the faculty of science

UNIVERSITY OF YAOUNDE I

FACULTY OF SCIENCE

Department of Academic Affairs and Cooperation

UNIVERSITÉ DE YAOUNDÉ I

FACULTÉ DES SCIENCES

Direction des Affaires Académiques et de la Coopération

LIST OF PERMANENT TEACHING STAFF	LISTE DES ENSEIGNANTS PERMANENTS

ACADEMIC YEAR 2019/2020
(By Department and by Grade)

UPDATING DATE: THE 19ST OF FEBRUARY, 2019

ADMINISTRATION

DEAN: TCHOUANKEU Jean Claude, Professor
VICE-DEAN / DPSAA: ATCHADE Alex de Théodore, Professor
VICE-DEAN DSSE: AJEAGAH Gideon AGHAINDUM, Professor
VICE-DEAN DRC: ABOSSOLO Monique, Professor
Head of the Division of Academic Affairs, Scholarity and Research: MBAZE MEVA'A Luc, Professor
Head of the Administrative and Financial Division: NDOYE FOE Marie C. F., Professor

1- DEPARTMENT OF BIOCHEMISTRY (BC) (40)			
N^o	NAMES AND SURNAMES	GRADE	OBSERVATIONS

1	FEKAM BOYOM Fabrice	Professor	In service
2	MBACHAM FON Wilfried	Professor	In service
3	MOUNDIPA FEWOU Paul	Professor	Head of Department
4	NINTCHOM PENLAP V.	Professor	In service
5	OBEN Julius ENYONG	Professor	In service
6	ACHU Merci BIH	Associate professor	In service
7	ATOGHO Barbara Mma	Associate professor	In service
8	BELINGA NDOYE FOE M. C. F.	Associate professor	Chef DAF / FS
9	BIGOGA DIAGA Jude	Associate professor	In service
10	BOUDJEKO Thaddée	Associate professor	In service
11	EFFA NNOMO Pierre	Associate professor	In service
12	FOKOU Elie	Associate professor	In service
13	KANSCI Germain	Associate professor	In service
14	NANA Louise	Associate professor	In service
15	NGONDI Judith Laure	Associate professor	In service
16	NGUEFACK Julienne	Associate professor	In service
17	NJAYOU Frédéric Nico	Associate professor	In service
18	AKINDEH MBUH NJI	Lecturer	En poste
19	AZANTSA KINGUE GABIN BORIS	Lecturer	En poste
20	BEBOY EDZENGUELE Sara Nathalie	Lecturer	In service
21	DAKOLE DABOY Charles	Lecturer	In service
22	DJOKAM TAMO Rosine	Lecturer	In service
23	DJUIDJE NGOUNOUE Marcelline	Lecturer	In service
24	DJUIKWO NKONGA Ruth Viviane	Lecturer	In service
25	DONGMO LEKAGNE Joseph Blaise	Lecturer	In service
26	EWANE Cécile Anne	Lecturer	In service
27	FONKOUA Martin	Lecturer	In service
28	BEBEE Fadimatou	Lecturer	In service
29	KOTUE KAPTUE Charles	Lecturer	In service
30	LUNGA Paul KEILAH	Lecturer	In service
31	MANANGA Marlyse Joséphine	Lecturer	In service
32	MBONG ANGIE M. Mary Anne	Lecturer	In service
33	MOFOR TEUGWA Clotilde	Lecturer	Inspector, in Service MINESUP

34	PACHANGOU NSANGOU Sylvain	Lecturer	In service
35	Palmer MASUMBE NE-TONGO	Lecturer	In service
36	TCHANA KOUATCHOUA Angèle	Lecturer	In service
37	MBOUCHE FANMOE Marceline Joëlle	Assistant lecturer	In service
2- DEPARTMENT OF BIOLOGY AND ANIMAL PHYSIOLOGIES (B.P.A.) (44)			
1	BILONG BILONG Charles-Félix	Professor	Head of Department
2	DIMO Théophile	Professor	In service
3	DJIETO LORDON Champlain	Professor	In service
4	ESSOMBA NTSAMA MBALA	Professor	Vice Dean/FMSB/UYI
5	FOMENA Abraham	Professor	In service
6	KAMGANG René	Professor	C.S. MINRESI
7	KAMTCHOUING Pierre	Professor	In service
8	NJAMEN Dieudonné	Professor	In service
9	NJIOKOU Flobert	Professor	In service
10	NOLA Moïse	Professor	In service
11	TAN Paul VERNYUY	Professor	In service
12	TCHUEM TCHUENTE Louis Albert	Professor	Inspecteur de service Co-ord.Progr./MINSANTE
13	AJEAGAH Gideon AGHAINDUM	Associate professor	VICE-DOYEN / DSSE
14	DZEUFIET DJOMENI Paul Désiré	Associate professor	In service
15	FOTO MENBOHAN Samuel	Associate professor	In service
20	JATSA BOUKENG Hermine	Associate professor	In service
16	KEKEUNOU Sévilor	Associate professor	In service
17	MEGNEKOU Rosette	Associate professor	In service
18	MONY Ruth	Associate professor	In service
19	NGUEGUIM TSOFACK Florence	Associate professor	In service
21	TOMBI Jeannette	Associate professor	In service
22	ZEBAZE TOGOUET Serge Hubert	Associate professor	In service
23	ALENE Désirée Chantal	Lecturer	In service
24	ATSAMO Albert Donatien	Lecturer	In service

25	BELLET EDIMO Oscar Roger	Lecturer	In service
26	BILANDA Danielle Claude	Lecturer	In service
27	DJIOGUE Séfirin	Lecturer	In service
28	DONFACK Mireille	Lecturer	In service
29	GOUNOUE KAMKUMO Raceline	Lecturer	In service
30	KANDEDA KAVAYE Antoine	Lecturer	In service
31	LEKEUFACK FOLEFACK Guy B.	Lecturer	In service
32	MAHOB Raymond Joseph	Lecturer	In service
33	MBENOUN MASSE Paul Serge	Lecturer	In service
34	MOUNGANG LucianeMarlyse	Lecturer	In service
35	MVEYO NDANKEU Yves Patrick	Lecturer	In service
36	NGOUATEU KENFACK Omer Bébé	Lecturer	In service
37	NGUEMBOK	Lecturer	In service
38	NJUA Clarisse Yafi	Lecturer	Chef Div. UBA
39	NOAH EWOTI Olive Vivien	Lecturer	In service
40	TADU Zephyrin	Lecturer	In service
41	YEDE	Lecturer	In service
43	ETEME ENAMA Serge	Assistant lecturer	In service
44	KOGA MANG DOBARA	Assistant lecturer	In service
3- DEPARTMENT OF BIOLOGY AND VEGETAL PHYSIOLOGY (B.P.V.) (27)			
1	AMBANG Zachée	Professor	Chef Division/UYII
2	BELL Joseph Martin	Professor	In service
3	MOSSEBO Dominique Claude	Professor	In service
4	YOUMBI Emmanuel	Professor	Head of Department
5	ZAPFACK Louis	Professor	In service
6	ANGONI Hyacinthe	Associate professor	In service
7	BIYE Elvire Hortense	Associate professor	In service
8	DJOCGOUE Pierre François	Associate professor	In service
9	KENGNE NOUMSI Ives Magloire	Associate professor	In service
10	MALA Armand William	Associate professor	In service

11	MBARGA BINDZI Marie Alain	Associate professor	CT/UDs
12	MBOLO Marie	Associate professor	In service
13	NDONGO BEKOLO	Associate professor	CE / MINRESI
14	NGONKEU MAGAPTCHE Eddy L.	Associate professor	In service
15	TSOATA Esaïe	Associate professor	In service
16	GOMANDJE Christelle	Lecturer	In service
17	MAFFO MAFFO Nicole Liliane	Lecturer	In service
18	MAHBOU SOMO TOUKAM. Gabriel	Lecturer	In service
19	NGALLE Hermine BILLE	Lecturer	In service
20	NGOUO Lucas Vincent	Lecturer	In service
22	NOUKEU KOUAKAM Armelle	Lecturer	In service
23	ONANA JEAN MICHEL	Lecturer	In service
24	NSOM ZAMO Annie Claude	Lecturer	Expert national/UNESCO
25	TONFACK Libert Brice	Lecturer	In service
26	DJEUANI Astride Carole	Assistant lecturer	In service
27	NNANGA MEBENGA Ruth Laure	Assistant lecturer	In service
4- DEPARTMENT AND INORGANIC CHEMISTRY (C.I.) (35)			
1	AGWARA ONDOH Moïse	Professor	Head of Department
2	ELIMBI Antoine	Professor	In service
3	Florence UFI CHINJE épouse MELO	Professor	Recteur Univ.Ngaoundere
4	GHOGOMU Paul MINGO	Professor	Ministre Chargé deMiss.PR
5	NANSEU Njiki Charles Péguy	Professor	In service
6	NDIFON Peter TEKE	Professor	CT MINRESI/Chef de Departement
7	NDIKONTAR Maurice KOR	Professor	Vice-Doyen Univ. Bamenda
8	NENWA Justin	Professor	In service
9	NGAMENI Emmanuel	Professor	DOYEN FS UDs
10	BABALE DJAM DOUDOU	Associate professor	Chargée Mission P.R.
11	DJOUFAC WOUMFO Emmanuel	Associate professor	In service
12	KAMGANG YOUBI Georges	Associate professor	In service
13	KEMMEGNE MBOUGUEM Jean C.	Associate professor	In service

14	KONG SAKEO	Associate professor	In service
16	NGOMO Horace MANGA	Associate professor	Vice Chancelor/UB
17	NJIOMOU C.	Associate professor	In service
18	NJOYA Dayirou	Associate professor	In service
19	YOUNANG Elie	Associate professor	In service
20	ACAYANKA Elie	Lecturer	In service
21	BELIBI BELIBI Placide Désiré	Lecturer	CS/ ENS Bertoua
22	CHEUMANI YONA Arnaud M.	Lecturer	In service
23	EMADACK Alphonse	Lecturer	In service
24	KENNE DEDZO GUSTAVE	Lecturer	In service
24	KOUOTOU DAOUDA	Lecturer	In service
25	MAKON Thomas Beauregard	Lecturer	In service
26	MBEY Jean Aime	Lecturer	In service
27	NCHIMI NONO KATIA	Lecturer	In service
28	NDI NSAMI Julius	Lecturer	In service
29	NEBA nee NDOSIRI Bridget NDOYE	Lecturer	Inspecteur de Service MINFEM
30	NYAMEN Linda Dyorisse	Lecturer	In service
31	PABOUDAM GBAMBIE A.	Lecturer	In service
32	TCHAKOUTE KOUAMO Hervé	Lecturer	In service
5- DEPARTMENT OF ORGANIC CHEMISTRY (C.O.) (33)			
1	DONGO Etienne	Professor	Vice-Doyen
2	GHOGOMU TIH Robert Ralph	Professor	Dir. IBAF/UDS
3	NGOUELA Silvère Augustin	Professor	Head of department UDS
4	NKENGFACK Augustin Ephreïm	Professor	Head of Department
5	NYASSE Barthélemy	Professor	Directeur/UN
6	PEGNYEMB Dieudonné Emmanuel	Professor	Directeur/ MINESUP
7	WANDJI Jean	Professor	In service
8	Alex de Théodore ATCHADE	Associate professor	DEPE/ Rectorat/UYI
9	EYONG Kenneth OBEN	Associate professor	Chef Service DPER
10	FOLEFOC Gabriel NGOSONG	Associate professor	In service
11	KEUMEDJIO Félix	Associate professor	In service
12	KEUMOGNE Marguerite	Associate professor	In service

13	KOUAM Jacques	Associate professor	In service
14	MBAZOA DJAMA Céline	Associate professor	In service
15	MKOUNGA Pierre	Associate professor	In service
16	NGO MBING Joséphine	Associate professor	Sous/Direct. MINERESI
17	NOUNGOUE TCHAMO Diderot	Associate professor	In service
18	TABOPDA KUATE Turibio	Associate professor	In service
19	TCHOUANKEU Jean-Claude	Associate professor	Doyen /FS/ UYI
20	TIH NGO BILONG E. Anastasie	Associate professor	In service
21	YANKEP Emmanuel	Associate professor	In service
22	AMBASSA Pantaléon	Lecturer	In service
23	FOTSO WABO Ghislain	Lecturer	In service
24	KAMTO Eutrophe Le Doux	Lecturer	In service
25	MVOT AKAK CARINE	Lecturer	In service
26	NGOMO Orléans	Lecturer	In service
27	NGONO BIKOBO Dominique Serge	Lecturer	In service
28	NOTE LOUGBOT Olivier Placide	Lecturer	Chef Service/MINESUP
29	OUAHOUO WACHE Blandine M.	Lecturer	In service
30	TAGATSING FOTSING Maurice	Lecturer	In service
31	ZONDENDEGOUMBA Ernestine	Lecturer	In service
32	NGNINTEDO Dominique	Assistant lecturer	In service
6- DEPARTMENT OF COMPUTER SCIENCE (IN) (28)			
1	ATSA ETOUNDI Roger	Professor	Chef Div.MINESUP
2	FOUDA NDJODO Marcel Laurent	Professor	CD Info ENS/Chef IGA.MINESUP
3	NDOUNDAM Réné	Associate professor	In service
4	AMINOU Halidou	Lecturer	Head of Department
5	DJAM Xaviera YOUHEP KIMBI	Lecturer	In service
6	KOUOKAM KOUOKAM E. A.	Lecturer	In service
7	MELATAGIA YONTA Paulin	Lecturer	In service
8	MOTO MPONG Serge Alain	Lecturer	In service
9	TAPAMO Hyppolite	Lecturer	In service

10	ABESSOLO ALO'O Gislain	Lecturer	In service
11	KAMGUEU Patrick Olivier	Lecturer	In service
12	MONTHE DJIADEU Valery M.	Lecturer	In service
13	OLLE OLLE Daniel Claude Delort	Lecturer	C/D Enset. Ebolowa
14	TINDO Gilbert	Lecturer	In service
15	TSOPZE Norbert	Lecturer	In service
16	WAKU KOUAMOU Jules	Lecturer	In service
17	BAYEM Jacques Narcisse	Assistant lecturer	In service
18	DOMGA KOMGUEM Rodrigue	Assistant lecturer	In service
19	EBELE Serge	Assistant lecturer	In service
20	HAMZA Adamou	Assistant lecturer	In service
21	JIOMEKONG AZANZI Fidel	Assistant lecturer	In service
22	KAMDEM KENGNE Christiane	Assistant lecturer	In service
23	MAKEMBE. S . Oswald	Assistant lecturer	In service
24	MEYEMDOU Nadège Sylvianne	Assistant lecturer	In service
25	NKONDOCK. MI. BAHANACK.N.	Assistant lecturer	In service
7- DEPARTMENT OF MATHEMATICS (MA) (31)			
1	BITJONG NDOMBOL	Professor	In service
2	DOSSA COSSY Marcel	Professor	In service
3	AYISSI Raoult Domingo	Associate professor	Head of Department
4	EMVUDU WONO Yves S.	Associate professor	Chef division MINESUP
5	NKUIMI JUGNIA Célestin	Associate professor	In service
6	NOUNDJEU Pierre	Associate professor	In service
7	TCHAPNDA NJABO Sophonie B.	Associate professor	Directeur/AIMS Rwanda
8	AGHOUKENG JIOFACK Jean Gérard	Lecturer	Chef Cellule MINPLAMAT
9	CHENDJOU Gilbert	Lecturer	In service
10	DJIADEU NGAHA Michel	Lecturer	In service
11	DOUANLA YONTA Herman	Lecturer	In service
12	FOMEKONG Christophe	Lecturer	In service
13	KIANPI Maurice	Lecturer	In service
14	KIKI Maxime Armand	Lecturer	In service
15	MBAKOP Guy Merlin	Lecturer	In service

16	MBANG Joseph	Lecturer	In service
17	MBEHOU Mohamed	Lecturer	In service
18	MBELE BIDIMA Martin Ledoux	Lecturer	In service
19	MENGUE MENGUE David Joe	Lecturer	In service
20	NGUEFACK Bernard	Lecturer	In service
21	NIMPA PEFOUNKEU Romain	Lecturer	In service
22	POLA DOUNDOU Emmanuel	Lecturer	In service
23	TAKAM SOH Patrice	Lecturer	In service
24	TCHANGANG Roger Duclos	Lecturer	In service
25	TCHOUNDJA Edgar Landry	Lecturer	In service
26	TETSADJIO TCHILEPECK M. E.	Lecturer	In service
27	TIAYA TSAGUE N. Anne-Marie	Lecturer	In service
28	MBIAKOP Hilaire George	Assistant lecturer	In service
8- DEPARTMENT OF MICROBIOLOGY (MB) (13)			
1	ESSIA NGANG Jean Justin	Professor	Head of Department
2	ETOA François Xavier	Professor	Head of Department/FS/UYI, Recteur Université de Douala
3	BOYOMO ONANA	Associate professor	In service
4	NWAGA Dieudonné M.	Associate professor	In service
5	NYEGUE Maximilienne Ascension	Associate professor	In service
6	RIWOM Sara Honorine	Associate professor	In service
7	SADO KAMDEM Sylvain Leroy	Associate professor	In service
8	ASSAM ASSAM Jean Paul	Lecturer	In service
9	BODA Maurice	Lecturer	In service
10	BOUGNOM Blaise Pascal	Lecturer	In service
11	ESSONO OBOUGOU Germain G.	Lecturer	In service
12	NJIKI BIKOï Jacky	Lecturer	In service
13	TCHIKOUA Roger	Lecturer	In service
9- DEPARTMENT OF PHYSICS (PH) (41)			
1	BEN- BOLIE Germain Hubert	Professor	In service
2	ESSIMBI ZOBO Bernard	Professor	In service

3	KOFANE Timoléon Crépin	Professor	In service
4	NDJAKA Jean Marie Bienvenu	Professor	Head of Department
5	NJANDJOCK NOUCK Philippe	Professor	Sous Directeur/ MINRESI
6	NJOMO Donatien	Professor	In service
7	PEMHA Elkana	Professor	In service
8	TABOD Charles TABOD	Professor	Doyen Univ/Bda
9	TCHAWOUA Clément	Professor	In service
10	WOAFO Paul	Professor	In service
11	BIYA MOTTO Frédéric	Associate professor	DG/HYDRO Mekin
12	BODOBertrand	Associate professor	In service
13	DJUIDJE KENMOE épouse ALOYEM	Associate professor	In service
14	EKOBENA FOUDA Henri Paul	Associate professor	Chef Division. UN
15	EYEBE FOUDA Jean sire	Associate professor	In service
16	FEWO Serge Ibraïd	Associate professor	In service
17	HONA Jacques	Associate professor	In service
18	MBANE BIOUELE César	Associate professor	In service
19	NANA ENGO Serge Guy	Associate professor	Director/Students/Affairs. UB
20	NANA NBENDJO Blaise	Associate professor	In service
21	NOUAYOU Robert	Associate professor	In service
22	SAIDOU	Associate professor	Sous Directeur/Minresi
23	SIEWE SIEWE Martin	Associate professor	In service
24	SIMO Elie	Associate professor	In service
25	VONDOU Derbetini Appolinaire	Associate professor	In service
26	WAKATA BEYA Annie	Associate professor	Sous Directeur/ MINESUP
27	ZEKENG Serge Sylvain	Associate professor	In service
28	ABDOURAHIMI	Lecturer	In service
29	EDONGUE HERVAIS	Lecturer	In service
30	ENYEGUE A NYAM	Lecturer	In service
31	FOUEDJIO David	Lecturer	Chef Cell. MINADER
32	MBINACK Clément	Lecturer	In service
33	MBONO SAMBA Yves Christian U.	Lecturer	In service
34	MELI'I Joelle Larissa	Lecturer	In service
35	MVOGO ALAIN	Lecturer	In service
36	NDOP Joseph	Lecturer	In service

37	OBOUNOU Marcel	Lecturer	DA/Univ Inter Etat/Sangmalima
38	WOULACHE Rosalie Laure	Lecturer	In service
39	CHAMANI Roméo	Assistant lecturer	In service
10- DEPARTMENT OF EARTH SCIENCES (S.T.) (44)			
1	BITOM Dieudonné	Professor	Doyen / FASA / UDs
2	FOUATEU Rose	Professor	In service
3	KAMGANG Pierre	Professor	In service
4	MEDJO EKO Robert	Professor	Conseiller Technique/UYII
5	NDJIGUI Paul Désiré	Professor	Head of Department
6	NKOUMBOU Charles	Professor	In service
7	NZENTI Jean-Paul	Professor	In service
8	ABOSSOLO ANGUE Monique	Associate professor	Vice-Doyen / DRC
9	GHOGOMU Richard TANWI	Associate professor	CD/UMa
10	MOUNDI Amidou	Associate professor	CT/ MINIMDT
11	NDAM NGOUPAYOU Jules-Remy	Associate professor	In service
12	NGOS III Simon	Associate professor	DAAC/Uma
13	NJILAH Isaac KONFOR	Associate professor	In service
14	ONANA Vincent Laurent	Associate professor	In service
15	BISSO Dieudonné	Associate professor	Directeur/Projet Barrage Memve'ele
16	EKOMANE Emile	Associate professor	In service
17	GANNO Sylvestre	Associate professor	In service
18	NYECK Bruno	Associate professor	In service
19	TCHOUANKOUE Jean-Pierre	Associate professor	In service
20	TEMDJIM Robert	Associate professor	In service
21	YENE ATANGANA Joseph Q.	Associate professor	Chef Div. /MINTP
22	ZO'O ZAME Philémon	Associate professor	DG/ART
23	ANABA ONANA Achille Basile	Lecturer	In service
24	BEKOA Etienne	Lecturer	In service
25	ELISE SABABA	Lecturer	In service
26	ESSONO Jean	Lecturer	In service
27	EYONG JOHN TAKEM	Lecturer	In service
28	FUH Calistus Gentry	Lecturer	Sec. D'Etat/MINMIDT
29	LAMILEN BILLA Daniel	Lecturer	In service
30	MBESSE CECILE OLIVE	Lecturer	In service

31	MBIDA YEM	Lecturer	In service
32	METANG Victor	Lecturer	In service
33	MINYEM Dieudonné-Lucien	Lecturer	CD/Uma
34	MOUAFO Lucas	Lecturer	In service
35	NGO BELNOUN Rose Noël	Lecturer	In service
37	NGO BIDJECK Louise Marie	Lecturer	In service
38	NGUEUTCHOUA Gabriel	Lecturer	CEA/MINRESI
39	NOMO NEGUE Emmanuel	Lecturer	In service
36	NTSAMA ATANGANA Jacqueline	Lecturer	In service
40	TCHAKOUNTE J.	Lecturer	Chef.cell / MINRESI
41	TCHAPTCHET TCHATO De P.	Lecturer	In service
42	TEHNA Nathanaël	Lecturer	In service
43	TEMGA Jean Pierre	Lecturer	In service

Numbered repartition of permanent teachers by Department
(The 15th Jully of 2016)

Department	**Number of teachers**				
	Pr	**AP**	**L**	**ASS**	**Total**
BCH	5 (1)	12 (6)	19 (11)	1 (1)	**37 (19)**
BPA	12 (1)	10 (5)	20 (7)	2 (0)	**44 (13)**
BPV	5 (0)	10 (2)	9 (4)	2 (2)	**26 (9)**
CI	9 (1)	9 (2)	14 (3)	0 (0)	**32 (6)**
CO	7 (0)	14 (4)	10 (4)	1 (0)	**32 (8)**
IN	2 (0)	1 (0)	13 (0)	10 (3)	**26 (3)**
MAT	2 (0)	4 (0)	19 (1)	2 (0)	**27 (2)**
MIB	2 (0)	5 (2)	5 (1)	0 (0)	**12 (3)**
PH	10 (0)	17 (2)	11 (3)	1 (0)	**39 (5)**
ST	7 (1)	15 (1)	21 (5)	1 (0)	**43 (7)**
Total	**61 (4)**	**97 (25)**	**141 (39)**	**19 (6)**	**318 (75)**

A total of: 340 (70) whose
- Professors 61 (4)
- Associate professors 97 (25)
- Lecturers 141 (39)
- Assistant lecturers 18 (5)
- ( ) = Number of women.

The dean of the Faculty of science

Dedication

To Sorel my wife,
For her patience and support.

To my parents Dongmo Fabien and Voufack Claire,
Thanks for all that you have done for us.

Acknowledgements

I sincerely thank:

My advisers, Pr. Maurice Tchuente of the University of Yaounde I in Cameroon and Pr. Gaoussou Camara of Université Alioune Diop de Bambey in Sénégal for their continuous support in the course of my PhD and research. Their patience, motivation and guidance were a source of strength to me. Their office doors were always open whenever I knocked for a question about my research, being it on the methodologies, experimentations, results analysis, articles and thesis writing.

All the members of the jury, especially the president Pr. Fouda Ndjodo Marcel of the University of Yaounde I and the members Pr. Malo Sadouanouan of Université Nazi Boni de Bobo-Dioulasso, Pr. Georges Edouard Kouamou of the University of Yaounde I, Pr. René Ndoundam of the University of Yaounde I and Pr. Yves Emvudu of the University of Yaounde I. I am deeply grateful for agreeing to read the manuscript and to participate in the defense of this thesis.

Pr. Moussa Lo of Université Gaston Berger in Sénégal. I discovered the domain of Semantic Web through him during his seminar in Cameroon in 2012 and he received me for two-months as PhD internship in Sénégal in 2013.

Pr. Daniel Hagimont and Dr. Laurent Broto of ENSEEIHT de Toulouse in France. For their great offer-a two months PhD internship in 2011. This internship developed in me a passion in software engineering.

Dr. Nolna Désiré and Abena Jean from the National Tuberculosis Control Program in Cameroon, Dr. Texier Gaetan from Centre Pasteur du Cameroon, Dr. Tapamo Hippolyte, Iloga Sylvain and Melatagia Paulin from UMMISCO in Cameroon, Mr. Braak Laurent and Mr. Dupouy Julien from MEDES in France and all those who contributed to the EPICAM and the MABO projects.

The former heads of Computer Science department Dr. Kamgnia Emmanuel, Dr. Louka Basile, Pr. Atsa Roger, Pr. Emvudu Yves; the current head of Computer Science Department Dr. Halidou Aminou for all the facilities and advices they gave to me.

All the teachers of the Computer Science Department of the University of Yaoundé I and the University of Dschang for the training and supervision during my studies.

Folefac Martins, Mbanwei Prodencia and Fah Moise for helping me to fix the shells that were in my work.

My brothers and sisters for their unconditional support during my studies.

All the persons who contributed from far and near to this thesis.

CONTENTS

Abstract

Epidemiological surveillance systems implement complex processes for the collection, analysis and interpretation of health data, for the planning or the evaluation of public health practices and policies. These systems must be able to provide up-to-date, precise and complete information to stakeholders whose interests are diverse and evolve with time. There are many difficulties faced when putting in place such systems. For example, the ability for such systems to collect, transmit and manage structured data, while taking into consideration the security, authentication and confidentiality of the data is crucial. To this end, a software editor known as "IMOGENE", based on Model Driven Architecture (MDA) was developed by MEDES in Toulouse. A joint project led by UMMISCO, MEDES, Centre Pasteur du Cameroun and National Program to Fight against Tuberculosis, made use of the latter platform to develop and deploy a tuberculosis surveillance system named EPICAM. This project showed that the absence of semantic links between data only allowed the exploitation of information explicitly defined in a database. The idea presented in this thesis to solve this problem is the use of information contained in source code, to infer new knowledge and integrate them in a domain ontology. To be precise, we propose a solution based on Hidden Markov Models (HMMs), which as opposed to other existing techniques that are limited to extraction of terminologies, concepts and properties also enables learning of axioms and rules. The implementation on the source code of the EPICAM platform has allowed us to describe in a clear, precise and succinct manner what we consider as principal information obtained, which has been evaluated and validated by domain experts.

Keywords: Epidemiological surveillance, Model-Driven architecture, Ontology, Machine learning, Ontology learning, Hidden Markov Models, Source code.

Résumé

Les systèmes de surveillance épidémiologique mettent en œuvre des processus complexes de collecte, d'analyse et d'interprétation de données de santé, en vue de la planification ou de l'évaluation des pratiques et politiques de santé publique. Ces systèmes doivent être capables de fournir des informations actualisées, précises et complètes aux différents acteurs dont les intérêts sont divers et évoluent au cours du temps. Les difficultés auxquelles on est confronté dans leur mise en œuvre sont nombreuses. Par exemple, la capacité à collecter, transmettre et gérer une information structurée, prenant en compte les besoins de sécurité, d'authentification et de confidentialité des données est cruciale, et le MEDES de Toulouse a développé à cet effet un éditeur d'applications "IMOGENE" basé sur le principe de l'Ingénieurie Dirigée par les Modèles (IDM). L'utilisation de cette plateforme pour développer et déployer un système de surveillance de la tuberculose au Cameroun, dans un projet collaboratif impliquant UMMISCO, le MEDES, le Centre Pasteur du Cameroun et le Programme National de Lutte Contre la Tuberculose a montré que l'absence de liens sémantiques entre les données ne permet pas d'exploiter que les informations explicitement stockées dans les bases de données. L'idée présentée dans cette thèse pour répondre à cette insuffisance est l'exploitation des informations contenues dans le code source, pour inférer de nouvelles connaissances et les intégrer dans une ontologie de domaine. Plus précisément, nous proposons une approche basée sur les Chaînes de Markov Cachées (CMC) et qui, contrairement aux techniques existantes qui se limitent à l'extraction des terminologies, concepts et propriétés, permettent aussi l'apprentissage des axiomes et des règles. La mise en œuvre sur le code source de la plateforme EPICAM a permis de dire de manière claire, précise et brève ce qu'on considère comme principale information obtenue qui a été évaluée et validée par les experts du domaine.

Mots clés : Surveillance épidémiologique, Ingénierie Dirigée par les Modèles, Ontologie, Apprentissage automatique, Apprentissage des ontologies, Chaînes de Markov Cachées, Code source.

1 Introduction

Over centuries, infectious diseases have always been one of the main problems for human health because they spread quickly and result in high mortality rates. Effective management of these diseases necessitates Hospital Information Systems, Laboratory Information and Management systems, epidemiological surveillance systems which can provide timely, accurate, relevant, comprehensive and updated information to stakeholders [17]. To relieve users of the manual tasks of collecting and exploiting data in order to obtain information, the Semantic Web seeks to model data sources and gives the possibility to reason automatically on them. In this thesis, we are seeking how to design semantic-aware epidemiological surveillance systems. Then, in the following sections, we present epidemiological surveillance systems, the problems generally encountered during the development of these types of systems and the management of epidemiological data (section 1.1); the domain of semantic Web (section 1.2) and the thesis objectives, contributions and structure (section 1.3).

1.1 Epidemiological surveillance systems

Epidemiological surveillance systems enable the collection, analysis, and interpretation of data, together with the dissemination of these data to public health practitioners, clinicians, decision makers and the general population for preventing and controlling diseases [27, 110]. It should support timely, efficient, flexible, scalable and interoperable data acquisition, analysis and dissemination. These information are essential to the planning, implementation and evaluation of public health practices [27, 50]. Epidemiological surveillance can be done manually or automatically/semi-automatically.

1.1.1 Manual approach

Regarding manual management, data is collected using paper-based collection tools (forms, register), analysed on the spot or transmitted to data analysis centers. The analysis is done by hand, counting manually the number of cases, examinations, etc. and building statistical tables. Software is often used to improve data collection and analysis. For example in many countries, the data is collected in paper form and typed in input masks built using tools such as Excel, EPIINFO, EPIDATA, CSPRO, etc. These tools are then used to analyze these data. These are heavy tasks because the health workers register data twice: on paper form and in a computer input mask. The statistics obtained after data analysis are interpreted and transmitted to the decision-making centers by land transport [12].

Mapatano et al. [84] have identified the main problems caused by manual management of health data from the Health Information Systems in DR Congo. The main problems are low rates of promptitude and completeness, the production of basic statistics are generally late, poor statistics with some rates more than 100%, some rates less than 0% yet cases have been recorded, poor dissemination of information and the absence of feedback to the data producers. In addition, lost sight patients are difficult to identify. These problems may cause some damages. For example, large delay or missing laboratory results of a communicable disease (for example tuberculosis, measles or cholera) could not only have a clinical impact on the patient, but also keep them in an infectious state. This can promote transmission of the strain and an epidemic can occur [21]. In the case of TB, the patient may become multidrug-resistant (MDR) or extensively drug resistant (XDR).

1.1.2 Automatic approach

With the rapid advancement of Information and Communication Technologies (ICT), the software approach has proven to be more efficient. For example, Joaquín et al. [21] have proven that a Web-based system to transmit laboratory reports decreases results delivery times, reduces redundancy in resource utilization, provides faster and more complete notification for public health purposes, decreases the number of reporting errors to health centers, and improves monitoring of patients. In effect, with these systems, clinicians have greater access to their history and laboratory data. ICT offers remarkable enhancement opportunities for epidemiological surveillance systems. The software used may be developed from scratch or existing ones (e.g., District Health Information Software-DHIS[1] [37], OpenMRS[2] [128]) may be adopted. In developed countries, the IT ecosystem of epidemiological surveillance is a heterogeneous network of applications of different design and developers [16]. For example, generally, there is a system for laboratory management, another one for patient management and the third one for disease surveillance (in some cases, one per disease). The multitude of heterogeneous systems can slow the response to new requirements of the healthcare landscape [117]. In addition, epidemiological surveillance systems evolve rapidly (new drugs, new treatment protocols, etc.), leading to software updates which can take time (while waiting for new versions) and be expensive [27]. On the other hand, depending on the data gathered on

[1] https://www.dhis2.org/
[2] http://openmrs.org/

the field, the epidemiologists may need to collect additional data to evaluate a new parameter (for example, the height of the patients in order to calculate their body mass index). These tasks may be done by using additional materials such as paper or spreadsheet (which can lead to a problem of data integration). Another solution is to introduce new requirements and the software updated (which can lead to the problem of software regression).

The problem of failed software developed for epidemiological surveillance is often the result of an unsystematic transfer of business requirements to the implementation [117]. This problem can be avoided if the system is established using a well-defined framework/architecture permitting the rapid development/update of the surveillance software by experts such as health workers who do not master software engineering.

Epidemiological surveillance systems are generally implemented according to the Client-Server architecture in which:

- the server runs one or more server programs which share their resources with one or many clients;
- the clients are Web browsers which interpret and display information provided by the server.

This architecture assumes an open world in which information can be explicitly defined, shared or distributed. Moreover, information can also be interchanged and used to make deductions or queries. At the server side, information is generally stored in a relational database which is based on a relational data model. To query and maintain the database, the standard Structured Query Language (SQL) is used. Relational databases have proven in the last few decades their efficiency, flexibility and performance for representing and managing data. However, the dramatic increase in the use of knowledge discovery applications in the health domain requires the users/developers to write complex database queries to retrieve information. Such users are not only expected to grasp the structural complexity of complex databases but also the semantic relationships between data stored in these databases. In order to overcome such difficulties, researchers have been working on semantic annotation of data [60, 85, 89, 122].

1.2 Semantic Web

According to Tim Bernee Lee [129], "The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation". The meaning of information is made explicit with the help of the formal (structured) and standardized representation of knowledge. Semantic Web permits us to: **(1)** automatically process information before returning it to users; **(2)** describe and integrate heterogeneous data sources; **(3)** automatically deduce implicit data or non-obvious data. To this end, each information source is extended with a semantically structured representation. The most popular way to include these semantics in the systems is the use of ontologies which complement the relational databases with semantic annotations [85].

Studer et al. [125] defined an ontology as "A formal, explicit specification of a shared conceptualization". In the context of domain ontologies, conceptualization refers to the abstract model of the domain which is machine readable and where all the elements are explicitly defined and accepted by the members of a group. Several domain ontologies define and organize relevant knowledge about activities, processes, organizations and strategies, in order to facilitate information exchange between machines and between a human and a machine [58, 75]. Building domain ontologies requires the access to domain knowledge owned by domain experts or contained in knowledge sources [58, 127]. However, domain experts are not always available for interviews, and when they are available, the knowledge provided is often incomplete and subjective. In addition, as the domain evolves, the knowledge provided by the experts is likely to be out of date. Therefore there is a lot of added value in creating domain ontologies from existing knowledge sources such as structured and unstructured documents of the domain: texts [4, 5, 30, 55], databases [29, 32, 66, 144], XML files [59], existing ontologies [79, 104, 124], UML/Meta-model diagrams [24, 42, 54, 141], and source code [11, 13, 14, 23, 26, 144].

Although source code is often used to extract concepts and relations, its full potential is not exploited to extract for example, axioms and rules [13, 14]. Actually, source code is any fully executable description of a software designed for a specific domain such as medical, industrial, military, communication, aerospace, commercial, scientific, etc. It can be used for the collection, organization, storage and communication of information. It is designed to facilitate repetitive tasks or to process information quickly. In a software design process, a set of knowledge related to the domain is captured and integrated in the source code.

The extraction of knowledge from structured (relational databases, XML) and unstructured (text, documents, images) sources is also known as ontology learning [7, 121, 132]. It consists of applying statistical techniques, symbolic techniques or both to (semi-)automatically extract the ontological knowledge from knowledge sources. Several authors have proposed the use of symbolic techniques [11, 51, 144] and statistical techniques [23, 77] to extract generally concepts and properties from source code. In this thesis, we propose an approach for extracting ontological knowledge from Java source code using Hidden Markov Models (HMMs). This approach is used to extract knowledge from EPICAM source code. After the extraction process, the knowledge obtained is combined with the knowledge extracted from existing biomedical ontologies to construct the Ontology for Tuberculosis Surveillance System (O4TBSS).

1.3 Thesis Positioning

The main goal of this thesis is to propose models, methods, and tools for stakeholders involved in disease surveillance. The results are experimented in the epidemiological surveillance of tuberculosis (TB) in Cameroon. Indeed, 1.4 million deaths worldwide are caused by TB between 2011 and 2016 globally [56, 57]. The WHO reported 1.4 million people died from TB[3] in 2019 (including 208 000 people with HIV). Worldwide, TB is one of the top 10 causes of death and the leading cause from a single infectious agent (above HIV/AIDS) [138]. In the next paragraphs, we will present the objectives, the contributions and the thesis structure.

[3] https://www.who.int/news-room/fact-sheets/detail/tuberculosis

1.3.1 Objectives

Our main objective is to design methods, models and tools used to build semantic-aware epidemiological surveillance systems. The two specifics objectives to achieve this goal are:

- The proposition of solutions for efficient development of data collection and management systems;
- The proposition of solutions for the development of knowledge management systems.

1.3.2 Contributions

In this thesis, we propose a model for the generation of epidemiological surveillance systems and another one for knowledge extraction from source code. The knowledge extracted is used to build a domain ontology for the annotation of tuberculosis data. The overall works gave three main contributions:

- A Model Driven Architecture (MDA) based approach for the development of epidemiological surveillance systems: To solve the failed software development problem caused by an unsystematic transfer of business requirements to the implementation [71, 112, 117], we propose an approach based on MDA. This approach is used to develop EPICAM, a platform for epidemiological surveillance of tuberculosis.
- A Hidden Markov Model (HMM) based approach for knowledge extraction from source code: The development of the EPICAM platform allowed us to collect epidemiological data. To facilitate the access to this data, we propose to use an ontology. The second contribution of this thesis is a method that permits us to extract knowledge (or learn ontology) from Java source code.
- An Ontology for Tuberculosis Surveillance System (O4TBSS): The third contribution of this thesis is the Ontology for TB Surveillance System (O4TBSS). This ontology is built using the knowledge learned from the EPICAM source code, EPICAM database and existing biomedical ontologies.

1.3.3 Thesis structure

The rest of this document consists of five chapters and a conclusion:

1. **Chapter 2** presents a review of epidemiological surveillance. This involves the presentation of the main components of epidemiological surveillance systems, approaches and tools generally used to develop these systems, and the limits of these approaches and tools.

2. **Chapter 3** makes a review of Model-Driven Architecture and agile methodology. Thereafter, we present an approach based on MDA to successfully develop epidemiological surveillance systems. At the end, we show how this approach was used to develop EPICAM, an epidemiological surveillance platform of tuberculosis.

3. **Chapter 4** presents a review of ontology engineering. Firstly, we present how ontologies are modelled, built and validated. Thereafter, we review ontology learning data sources, techniques and evaluation.

4. **Chapter 5** details the approach we propose in this thesis for knowledge extraction from Java source code. Considering that HMMs are probabilistic models, we start this chapter by presenting probabilistic models and Hidden Markov Models. Thereafter, the source code is presented. Finally, we present the HMM-based approach for knowledge extraction from Java source code and we apply this approach to learn knowledge from EPICAM source code.

5. **Chapter 6** presents an ontology constructed using knowledge extracted from EPICAM source code, databases and some biomedical ontologies. In this chapter, we present the methodology used to construct this ontology, the ontology constructed and use cases.

6. The **conclusion** summarizes the entire work and presents the direction of our future research.

2 A review of epidemiological surveillance systems

Epidemiological surveillance systems examine the factors that determine the occurrence and the distribution of diseases or other pathological manifestations [50, 134]. They are usually put in place for diseases of epidemic potential; when one is faced with a new problem and would like to know its extent (for example, if one discovers a case of Ebola virus disease); when one wants to eradicate a disease and would like to follow the process (for example epidemiological surveillance of tuberculosis); or when one is faced with a problem and the Health Information Systems (HIS) do not provide the expected information [134]. To have a good surveillance system, it is important to focus on the collection, analysis, interpretation and dissemination of data. Data used in surveillance systems are in large quantities (census, survey, patient information, services and resource data) and come from different data sources (figure 1).

The objective of this chapter is to make a review of epidemiological surveillance systems, which is the domain of application of this thesis. Then, the section 2.1 presents the epidemiological surveillance process. Sections 2.2 and 2.3 present the architectures and tools generally used and the section 2.4 presents the epidemiological surveillance of tuberculosis in Cameroon.

2.1 Epidemiological surveillance process

This section presents the main activities of epidemiological surveillance systems. In the next paragraphs, we will present the data collection (section 2.1.1), data analysis (section 2.1.2), information interpretation and dissemination to stakeholders (section 2.1.3).

2.1.1 Data collection

Data is at the center of any system because it is used to make rational decisions. Epidemiological surveillance systems provide the Health System with the necessary information for decision-

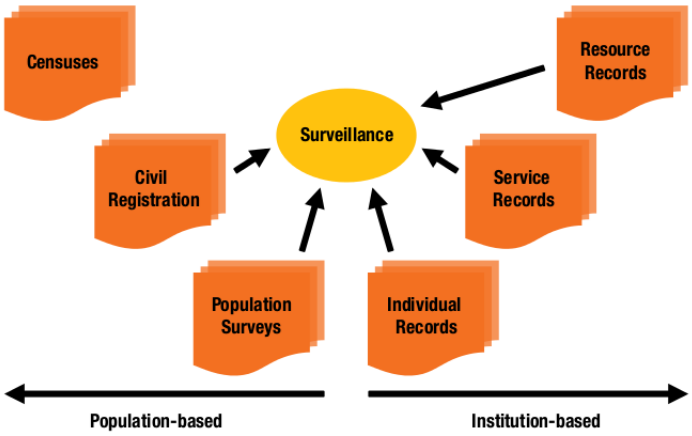


Figure 1: Multiple sources of information [65]

making and action taking. This section presents the epidemiological data collection and the main functionalities of a data collection system.

Epidemiological surveillance systems permit us to collect data that will be used to answer the questions Who? Where? When? [27, 133, 134].

- **Who?** It must provide information on targeted and at-risk populations (stratified by sex, age, etc.);
- **Where?** It must provide information about the locations, where the different disease cases are reported and equally gives the statistics on the spatial distribution of these diseases;
- **When?** It must specify the periods when the disease cases are usually reported. It should be possible to identify the periods where there are more or less reported cases.

In epidemiological surveillance systems, data collection is a continuous and regular activity closely linked to health decisions and the implementation of health programs. Specific objectives are developed according to the needs of health programs and the priorities of each country [17, 133, 134]. However, data helps Health System to [27]:

1. **Detect issues and outbreaks as early as possible:** Data collection will help to determine the health profile of the population and detect any changes, identify new emerging problems, recognize cases or clusters of cases to trigger interventions to prevent transmission or reduce morbidity and mortality, detect epidemics, identify risk factors and high risk population groups, etc.;
2. **Monitor trends of health status of the population and consider priorities on a continuous basis:** For example, the monitoring of medical consultation data may help us to assess the need of the population for health care services, set priorities, guide public health policy strategies and make informed decisions related to resource allocation;

3. **Assessing public health impacts and trends of new emerging health problems:** For example, the identification of economic and social impact that influence the health of a population;
4. **Assessing the effectiveness of the interventions and the health services offered:** For example, the data will be analyzed and used to evaluate the effectiveness of health interventions;
5. **Make sure that resources are directed to the needed sectors and groups:** From data, places and people at risk will be determined;
6. **Support health research:** For example, data will help researchers to formulate hypotheses that lead to analytic studies about disease causation, propagation, or progression;

To achieve these objectives, the content of the systems are grouped according to three key axes including standards and indicators, tools and instructions, coordination and supports [133]:

- Indicators are variables, reproducible in time that allow decision makers to estimate objectively the size of a health problem and monitor the processes, the products, or the effects of an intervention on the population. They enable us to follow the evolution of the achievement of the objectives. To define them, the question about the categories of information gathered by the surveillance system is posed.
- A minimum standard is used to determine the minimum acceptable level in the achievement of each indicator. The standards are adapted to the specific environment in which they are used and must be based on the validity of each operation. In the absence of country-specific data, international standards can be used. Thus, the tools to be used for data collection, the people in charge of this task, the timing of data collection and how data should be collected should be clearly defined.
- Coordination and support should help to avoid time wastage when the same data is collected severally by different people.

The health system of the majority of sub-Saharan African countries is organized in a pyramidal form (figure 2) and data is produced at all levels of the system [98, 99]. Data from lower levels are sent to upper levels to report on the overall health situation. Data collectors can be health workers in health centers, community relay agents in the community, any person or organization that can provide health information. Data is generally collected in different formats: paper, data collection forms, registers (consultation register, hospitalization), databases, Excel files, Word files, SMS, etc.

The main functionalities for data collection systems are [27, 62, 65, 97, 133, 134]:

- **Data recording:** The system must be able to record all data entered by users in a consistent way;

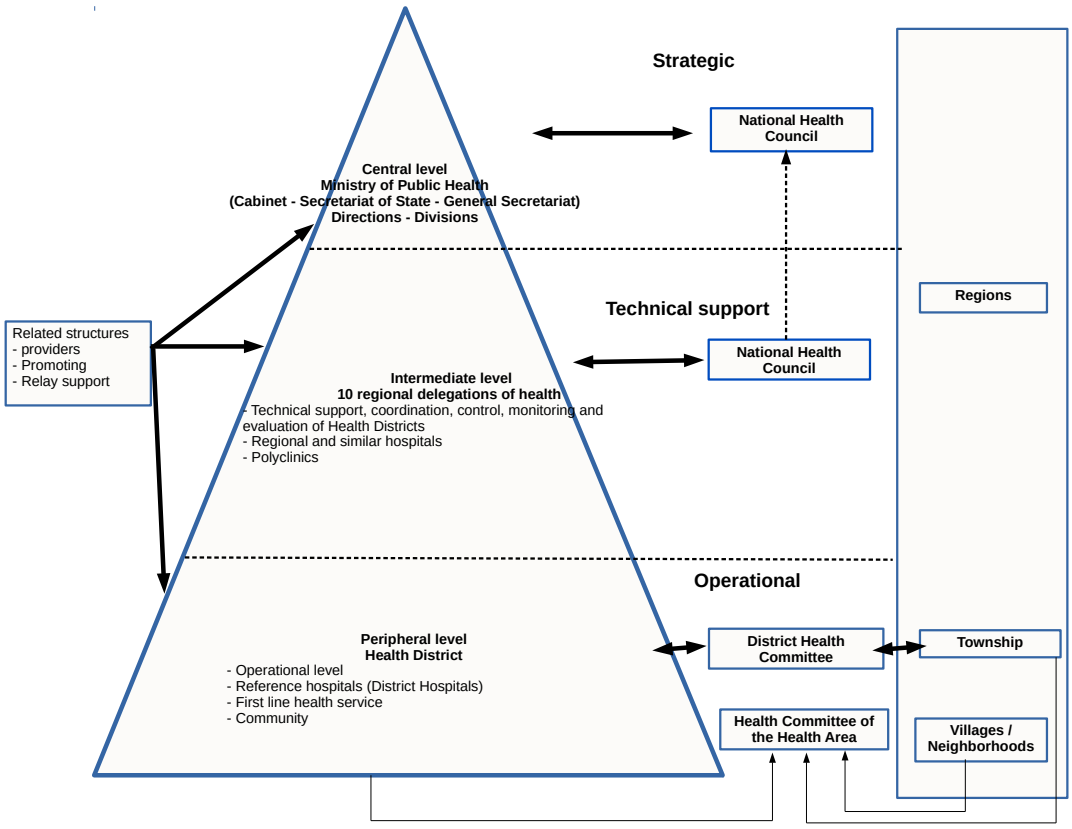


Figure 2: Organization of the health system [98]

- **Data Control:** The system must provide a check module when saving data to avoid missing or erroneous data;
- **Data transmission on time:** A data transmission system must allow the collected data to be transmitted in a timely manner to the stakeholders;
- **Data Access Security:** The system should grant access to data to entitled users;
- **Data pre-processing:** The system must provide a minimal analysis of the data;
- **Feedback of information:** The system must inform data collectors on the use of data;
- **Data Integration:** The system must allow the integration of other data sources;
- **Data Backup:** The system must have a data backup mechanism to ensure that the recorded data will be retrieved when it is needed;
- **Platform evolution:** To facilitate its evolution, the system must evolve in order to meet users needs.

2.1.2 Data analysis

After the collection of data, its analysis will enable the extraction of useful information, by reducing the data to a few easy-to-understand indicators, tables and graphs [27]. These are called statistics. The data collected has many values when integrated with other information (figures 3). For example, combining malaria epidemiological data with climate data can be used to track changes in incidence of malaria.

The analysis permits us to determine the occurrence of a health concern and the characteristics and behaviours of people with a health concern as well as changes over time. The analysis of data should be done in terms of time, place, and person by looking at time trends, geographic distributions and comparing age, sex, and population groups. More advanced data analysis include space-time clustering, time-series analysis, geospatial analysis, linear/logistic regression trends and small area analysis, mathematical models to study the dynamic of infections within communities of people. Methods for the forecast of epidemics based on data may be done when necessary [27].

The information obtained after the analysis of data is communicated to decision-makers in a form that changes their understanding of health issues and needs (indicators, tables, graphs, cards). Thus information becomes evidence to justify decisions made by decision-makers. The real impact on health of the measures taken is controlled by the information system by measuring the evolution of health indicators (figures 3) [65]. During data analysis, the description and presentation of data is made by identifying their interesting aspects and revealing their structures. It will enable, for example, a health district to determine priorities for better resource management, provide knowledge about the disease, risk behaviors, health service coverage, indicator trends, and system performance. Table 2.1 gives an example of a priority table that the district can use and the table 2.2 an example of using this table after analysis of disease data.

	Score		
Score Value	**+**	**++**	**+++**
Importance of the disease	Low	Medium	Strong
Efficiency of the intervention	Fairly efficient	Efficient	Very efficient
Cost of the intervention	High	Medium	Low

Table 2.1: A model of priority table in the District [134]

The analysis of data goes through the definition of the flow of information within the system and their circulation. The demand and supply of health information varies at each level of the health system. Thus the data must be analyzed according to the frequency of control, the different volumes of information, and especially the different information needed by the users at each level. Information obtained after the analysis of data may allow at the level of the individuals and communities, to ensure effective clinical management and assess the extent to which a community's

Disease	Relative importance (morbidity, mortality)	Efficiency	Cost	Priority (score)
Measles	+++	+++	+++	9
Diarrhea	+++	+	+++	7
Malaria	+++	+	+	5
Tuberculosis	++	+	+	4
Stroke	+	+	+	3
Leukemia	+	+	+	3

Table 2.2: Example of using the priority table [134]

needs and demands are met. At the level of health partners, to make decisions about the actual functioning of health services and resource allocation; at higher levels, such as headquarters, donors, and Ministries of Health, to help develop strategic policies, recommendations, ensure development and for resource mobilization.

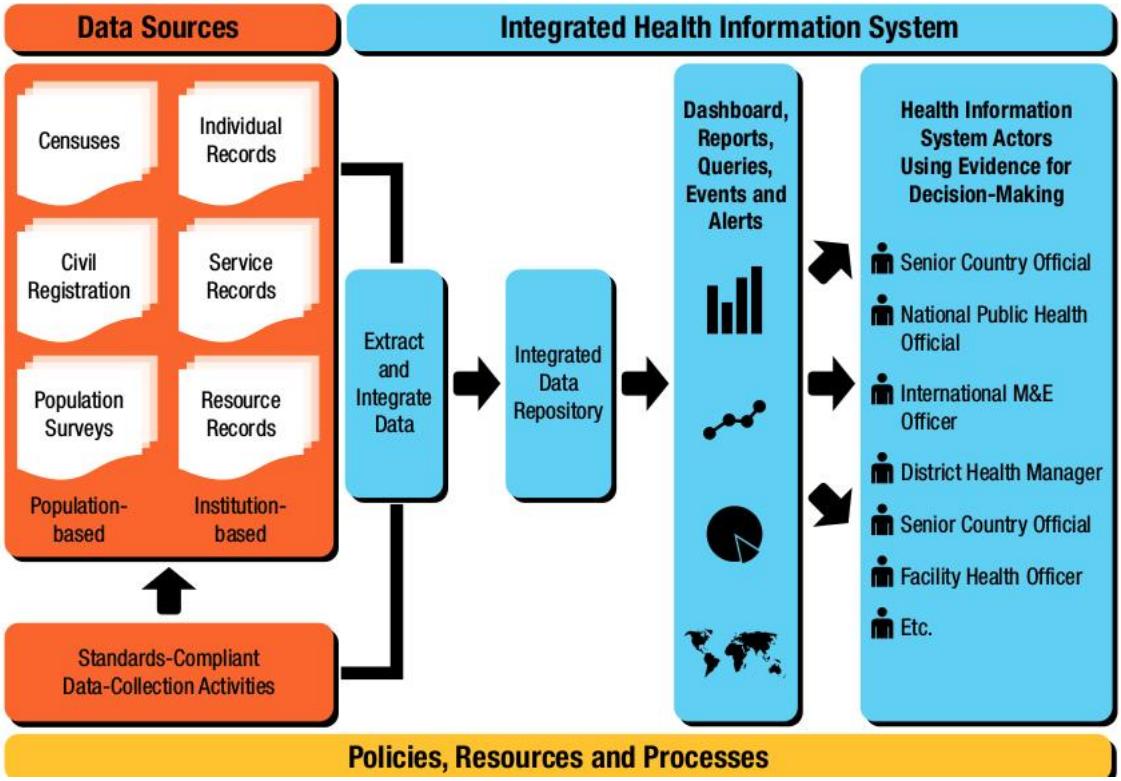


Figure 3: Integration of different information sources [65]

The analyzed data is made available to decision makers via user dashboards, reports, queries and alerts (an illustration is given by the figure 3). Such formatting is an essential function of the system in order to demonstrate the value of the data it contains.

Data analysis may face some problems: Mapatano and Piripiri [84], asserted that data is generally un-analyzed or poorly analyzed. From the evaluation of some health information systems in low income countries, we noticed that many systems are rich in data and low in information [84, 98, 99]. However, for data to be used, raw data must be analysed and disseminated to dedi-

cated users.

2.1.3 The interpretation and dissemination of information

Data analysis must be followed by its interpretation. Interpretation will help for example to know whether the apparent increases in disease occurrence within a specific population at a particular time and place, represents the reality. For instance, it can be due to the migration of population.

Effective communication strategies must address the needs of a heterogeneous group of users, and overcome challenges in relaying timely information to even the most peripheral sites. In fact, the final stage of the surveillance process is the timely communication of information to users. These users need to know about the sensitization, program planning and decision-making. They include public health practitioners, health planners, epidemiologists, researchers, and policy-makers as well the general public and media houses. In addition, recipients should include those who provide reports and those who collect the data [27].

The dissemination of information is two-way, following the health system pyramid presented by the figure 2: lower levels to higher levels and higher levels to lower levels (Regular feedback to health workers enhances data quality as well as effort and interest in the surveillance). Data is collected in health centers and sent to the district to report on the health situation in the health area. Once in the district, the data from the various health centers are compiled to know the situation in the district and the data is sent to the region. Data from districts of each region are synthesized and forwarded to the central level. At the central level, information will be used to know the national health situation of the population. From the central level, health information is disseminated to stakeholders: The National and regional health authorities; health district and health center officials; local community organizations; the general population; non-governmental health organizations; and health partners. However, the presentation of information depends on the users. For example, the tables and graphs are used to present the health situation during health district coordination meetings, reports are sent to higher levels using secure mail, SMS are used to raise awareness in the population. At each level of the pyramid, information is used:

1. **Peripheral level:** It is the base of the pyramid and is considered as the operational level. In Cameroon, it consists of the health districts. Each health district covers a well-defined geographical health area with several health centers [99]. At this level, information can be used to know the needs of the population in terms of health care;
2. **Intermediate level:** It is the center of the pyramid and is the technical support level. In Cameroon, it is made up of the regional delegations of public health. Each regional delegation of public health covers a well defined health area, and is made up of Health Districts. At this level, information is used, for example, to transform health strategies into technical programs to be applied at the health district level;
3. **Central Level:** It is the top of the pyramid and is the strategic level. It covers the entire country. It consists of health services and central structures of the Ministry of Public Health. At the central level, information can be used to design strategic frameworks, develop and mobilize resources.

To be useful, each level of the health system must have information for decision-making resulting from data collected by the different structures of the level [98, 99, 134]. These information will be used to answer the following questions [134]:

- **"Here?"** Where are we now? The goal is to assess the health situation of the population and to determine the resources available;
- **"There?"** Defining a health policy with clear objectives and well-defined targets;
- **"From here to there?"** How do we get there? It aims at determining the targets (locality, population) and actions to put in place corresponding to the needs of these targets.

These questions are posed at all the levels of the health system in order to determine the use of the information:

1. **At the central level:** The information enables the definition of health objectives and strategies, the development and mobilization of health resources, the improvement of management, etc.
2. **At the intermediate level:** Information enables the selection and adaptation of techniques for the implementation of health policies in health districts, technical coordination of health structures, synthesis and administrative management of staff, etc.
3. **Peripheral level** The peripheral level is the operational level of the health system. The information is used to know the epidemiological profile of the population of the health area, thus, making it possible to prepare for an advocacy; to inform the population about health problems and how to protect themselves against diseases. At this level, the integration of resources to make effective the actions of the health system in the community is made.

2.2 Epidemiological surveillance systems architectures

Epidemiological surveillance is generally implemented according to the client-server architecture [12]:

- **The client:** The client uses a data collection tool that may be a paper form or an electronic form (in a computer or a mobile phone). In the case of epidemiological surveillance, the client is generally the doctor who is treating the patients, the laboratory technician who does the laboratory exams, or the pharmacist who gives the medication to the patient;
- **A communication medium:** It connects the client terminal to the server. This medium can be the Internet network, the mobile network (GSM, GPRS etc.), or a USB key in a car in the case of data transmission using roads, rivers, etc;

- **Multiple Protocols:** These can be used to send data collected from the client to the server: paper, SMS, MMS, HTML, WML, i-HTML. The protocol for conveying the data is generally dependent on the transmission medium. For example, for the transmission by surface, paper are used;
- Once at the server level, the data is saved in a database and used for decision making.

Depending on their implementation and data management process, surveillance systems can be categorized by: Form-Route-Server (section 2.2.1), Mobile-MobileNetwork-Server (section 2.2.2), Computer-Internet-Server/Computer-MobileNetwork-Server (section 2.2.3) and multi-strategy approaches (section 2.2.4) [12].

2.2.1 Form-Route-Server architecture

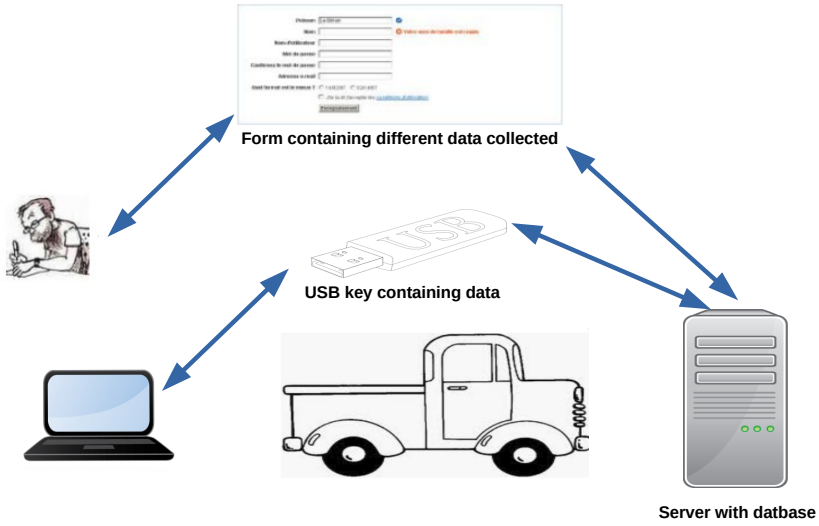


Figure 4: Form-Road-Server architecture

Form-Route-Server architecture uses forms (paper or electronic) for data collection and the road for data transmission (Figure 4). The client is a person who collects data either with a paper form, or an electronic form in a computer (for example, the data entry forms of EpiInfo). These tools are used to record data and store on removable media [12]. The removable media are transmitted via the road to decision-makers. This architecture is the most common in developing countries who do not have an electronic data collection system [98, 99, 88, 105] or in areas where the network is broken [143].

The Form-Route-Server architecture will ensure the reliability and completeness of data because the information collected can be typed. It is robust because it can use many communication

mediums (road, air, sea). On the other hand, the promptitude of data is not always guaranteed because the distance between the server and the client is often long and the quality of roads is poor. Data security can be ensured in the context of numeric files (encryption) but it is more difficult in the context of paper. There is no control to ensure that the forms will be completely filled and to avoid missing data. This architecture is not versatile because an update of paper forms or the electronic form on a computer disconnected from the network is a heavy operation [12].

2.2.2 Mobile-MobileNetwork-Server architecture

Mobile-MobileNetwork-Server architecture uses mobile phones for data collection and the mobile network for data transmission (Figure 5). In this architecture, the mobile phone is the client and the mobile network is the medium of communication. The data collector enters the data in a SMS or MMS draft and the transmission is done via the available mobile network (GSM, GPRS etc.) to the server [12]. Because mobile networks are the most widespread in the world, this architecture is often implemented in the management of health crises in developing countries [96] or countries affected by natural disasters [143]. Examples of software implementing Mobile-MobileNetwork-Server architecture are RapidSMS, ODK, Magpi, DHIS.

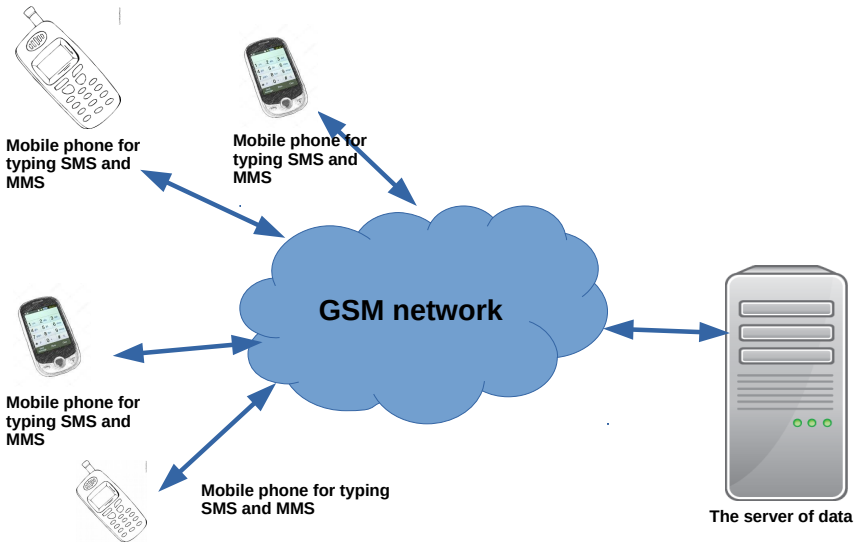


Figure 5: Mobile-MobileNetwork-Server architecture

The Mobile-MobileNetwork-Server architecture ensures the promptitude of the data because SMS and MMS are transmitted immediately to decision makers. But completeness and consistency is not always guaranteed because the data is entered in SMS or MMS. It is possible that the data collectors forget to enter certain data. Security is not guaranteed because the data is difficult to encrypt. Finally, it is not versatile because the modification of an element can disturb the data collector and require training it again. On the other hand, it is robust because mobile networks are

easy to deploy, resilient to natural disasters and is the most widespread in many countries in the world [12].

2.2.3 Computer-Internet-Server architecture

This architecture (shown in Figure 6) uses a standard Web application for data collection and management. The client is a computer or a smartphone and the medium of communication is the Internet. The data exchange protocol is usually HTML or an assimilated language [12]. Examples of software implementing the Computer-Internet-Server architecture are DHIS, OpenMRS, RapidSMS, Damoc, webTBS, etc.

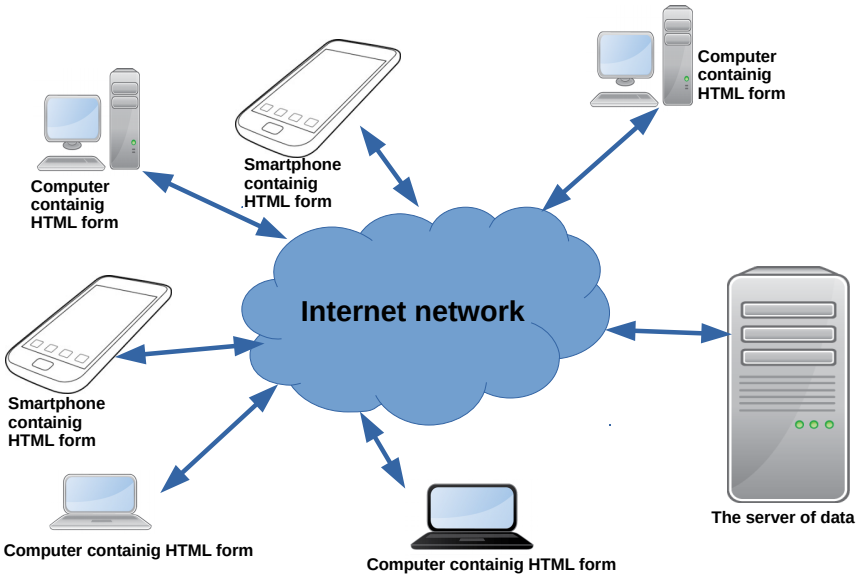


Figure 6: Computer-Internet-Server architecture

Computer-Internet-Server guarantees promptitude because the Internet is fast enough compared to the time required by the health system. Thanks to the data exchange protocols used, the chances of obtaining reliable data are great. In fact, the information collected can be verified and the transmission errors avoided thanks to the TCP/IP protocol. The TCP/IP protocol will also provide a high level of security through data encryption. This system is also versatile, only the server must be updated if the application changes. Note that the Internet is not always available in some rural areas. Because it is difficult to deploy (more than mobile network), this system may be difficult to put in place in some contexts.

2.2.4 Multi-strategy architecture

Some surveillance systems used a combination of several architectures. For example, for the epidemiological surveillance of tuberculosis in Cameroon, the $Form - Route - Server$ and the $Computer - Internet - Server$ architectures were combined. In order to improve the collection and management of tuberculosis surveillance data, the National Program to Fight against Tuberculosis (NTCP) in Cameroon has put in place a pilot project for electronic surveillance of tuberculosis. The software developed in this project was deployed in 25 health centers and the remaining 115 centers continue to use paper forms for data collection and the road for data transmission. After the 2010 Sichuan Earthquake in China, the electronic data collection and management system was damaged by the earthquake. The Chinese government has combined the Phone-NetworkMobile-Server system with the computer-Internet-Server system for the transmission of the data [143].

2.3 Epidemiological surveillance tools

Epidemiological surveillance can be done manually, automatically or semi-automatically. This section presents the tools generally used for epidemiological surveillance.

2.3.1 Data collection tools

Data collection is done using the tools built according to the information needs of the different stakeholders. When epidemiological surveillance is manual, the tools comprise paper sheets, registers, medical notebooks, etc. These tools are filled by the health workers to keep records of patients' health problems. However, this system may lead to loss of data, unreadable data and poorly inputted data [84]. In addition, these data are generally transmitted by land transport, air or water, which can lead to the problems of promptitude and physical data security.

Developed countries [16] have de facto adopted electronic data collection tools. In fact, these tools provide real-time data to different stakeholders and facilitate the automatic production of basic statistics and reports. In these countries, the data collection tools are generally integrated in an epidemiological surveillance platform permitting the collection, but also the analysis and the automatic generation of basic statistics and reports. Given the problems of the manual data collection in certain situations (e.g., the outbreak of cholera in Cameroon in 2010), like the low-resource settings, they generally use mobile phones to improve the data collection system [62]. With this system, data is inputted in a preformatted SMS and sent to the decision centers. This permits us to obtain data in real time in the decision centers. However, sometimes errors may occur during the typing of the SMS [12].

With the software approach, depending on the data gathered on the field, the epidemiologists may need to collect additional data in order to explain a phenomenon (for example, the height of the patients in order to calculate their body mass index). This task may be done by using supplementary materials such as paper form or spreadsheets, which can lead to a problem of data integration.

New requirements can be introduced and the software updated, which can lead to the problem of software regression.

2.3.2 Data analysis tools

Data analysis is done manually or using dedicated tools. When the analysis is manual, health professionals proceed by hand counting. Then, the manual analysis is tedious and usually error prone. In addition, some useful statistics such as Principal component analysis, linear and logistic regression, etc. are rarely applied and some relevant statistics may be lost.

Software is often used to improve data analysis. For example, in many developing countries, data is collected using paper form and inputted in an input mask built using tools such as EPIINFO[1] or EPIDATA[2]. Thereafter, these tools are used to analyze these data. This is a heavy task because the health worker registers the data twice: on paper form and in a computer input mask. The statistics obtained after data analysis are interpreted and transmitted to the decision-making centers by land transport, air or water.

In developed countries, epidemiological surveillance tools generally contain data collection and analysis modules. The data analysis module permits the production of basic statistics. For example, in the US, CDC (Centers for Disease Control and Prevention) has built a tool to improve the ability to understand health statistics through visualization; most timely estimates of births, deaths, and infant deaths can be visualized with this tool [62, 110].

2.3.3 Tools for information dissemination

The statistics obtained after data analysis is interpreted and the information is disseminated to different stakeholders. These actors can be divided into three broader groups: health workers, epidemiologists, decision makers and the population. With the manual management, reports are sent to the health authorities by email or using postal services. In some cases, a person is designated to transport these data to the centers where they will be used. However, with this system, physical data security and access security cannot be guaranteed. In addition, the feedback to the data producer (health workers) on the use of the information is rare. The electronic system usually allows for the production of basic reports in time and which can be used as a starting point for decision making.

The dissemination of information to the public (health information/advice/warning) is usually done through the mass media, local community meetings, health communities, or door-to-door outreach. However, it can be difficult to reach certain villages. In the big cities, people go out to work during the day and can miss important health information [134]. To overcome these problems, given the high rate of the use of mobile phones, SMS are increasingly used.

[1] https://www.cdc.gov/epiinfo/index.html
[2] https://www.epidata.dk/

According to the importance of the disease/outbreak, reports are produced daily, weekly, or monthly and are made available to decision makers, epidemiologists, scientists, the press and the public. Examples of Polio in 1955 (daily reports), influenza in 1961 (weekly reports) in the US [62].

In developed countries, the basic reports are automatically generated by the surveillance systems and made available to the different stakeholders. In some cases, special tools (e.g., mobile phones) are made available to improve access and visualization for different purposes [62].

2.3.4 Electronic surveillance tools

To overcome the problem of manual management of epidemiological surveillance, many countries have adopted the use of electronic tools. These tools can be developed using a traditional approach for software development (Figure 7). It follows the complete development cycle such as specifications, analysis, design and implementation. This approach generally uses languages like Unified Modelling Language (UML) and methods like Unified Process (UP) or Merise in the analysis and design steps. The conceptual model is usually designed at once and is used to develop the software [20].

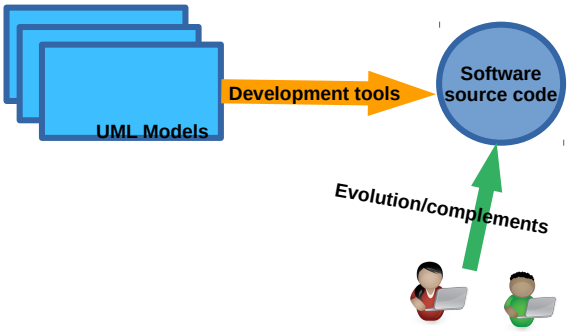


Figure 7: Classic approach for software development

In many developed countries, national communicable diseases surveillance systems (NCDSS) [16] such as CDC in the US are being developed. These systems are composed of the different features of the epidemiological surveillance system (data collection, analysis, and dissemination) [17]. Currently, these systems are not accessible to the general public. In some cases (mainly in resource poor settings or when an outbreak occurs) different tools to achieve different features of epidemiological surveillance are developed and used. Then, a tool may be used for data collection, another one for data analysis and another for information dissemination.

A review of tools used for epidemiological surveillance:

- **RapidSMS**[3]**:** It is an Open Source framework based on the Django framework and developed in 2009 by UNICEF. It can be used to develop epidemiological surveillance platforms accessible by mobile phone[4]. RapidSMS was used to monitor malnutrition Nigeria in 2009[5], Ethiopia in 2009[6] and Senegal in 2010[7].
- **District Health Information Software (DHIS)**[8]**:** It is an open source software whose first version was developed at the University of Oslo in Norway. It has been used to put in place health information systems in more than 30 countries in Africa, Asia and Latin America. Kenya, Uganda, Rwanda, Ghana, Liberia and Bangladesh have adopted it as their Health Information System.
- **Damoc**[9]**:** It is a software developed by EpiConcept for centers to fight against tuberculosis in France. It allows each health center to collect data and produce statistics and reports of activities on the performance of tuberculosis treatment. Damoc is currently used for epidemiological surveillance of tuberculosis at Ile de St. Martin in France.
- **BK4**[10]**:** It was developed in France by Epiconcept in collaboration with the Institut de Veille Sanitaire (InVS). It is used in the ARS (Agence régionale de la santé) in France for the surveillance of tuberculosis.
- **OpenMRS**[11]**:** It is used to support healthcare delivery in developing countries. It was born out of the need to intensify the follow-up of patients living with HIV in Africa, but, from the outset, it was designed for the management of patients' medical records, thus enabling them to support a full range of medical care. OpenMRS is used in several continents in a variety of contexts: patient medical records, epidemiological surveillance and research. It has been used for the monitoring of multidrug-resistant tuberculosis in Peru.
- **WEB-TBS:** It is used in Djibouti for tuberculosis surveillance[12]. With WEB-TBS, users use online forms and reports which are identical to that of the WHO standard indicator's registration and identification systems.

In spite of the advantages of electronic reporting systems and despite the fact that national communicable diseases surveillance systems have been in place, the results revealed that the electronic surveillance systems of the studied countries are far from the desirable state at both local and state levels. The experience and profiles of developed countries indicate that such systems could not be initiated so long as the proper infrastructures are not implemented for data exchanges across different centers. Poor communication among the centers and organizations related to the management of communicable diseases at different levels was one of the drawbacks identified in the studies carried out in these countries [17].

[3]http://www.rapidsms.org
[4]http://www.rapidsms.org
[5]http://www.rapidsms.org/case-studies/nigeria-monitoring-supplies-in-a-campaign-setting
[6]http://www.rapidsms.org/case-studies/supply-chain-management-during-food-crises/
[7]http://www.rapidsms.org/case-studies/senegal-the-jokko-initiative
[8]https://www.dhis2.org
[9]http://www.epiconcept.fr/en/produit/damoc
[10]http://www.invs.sante.fr/applications/bk4
[11]http://openmrs.org/
[12]http://www.emro.who.int/fr/dji/djibouti-events/atelier-formation-tuberculose-lutte-surveillance.html

In developed countries, the IT ecosystem of epidemiological surveillance is a heterogeneous network of applications of different designs and developers. For example, generally there is a system for laboratory, patient management, disease surveillance (and in some cases, one per disease). The multitude of heterogeneous systems can slowly react to new requirements that are brought up by changes of the care landscape [117]. In addition, epidemiological surveillance systems evolve faster (new drugs, new treatment protocols, etc.), leading to software updates which can take time (while waiting for a new version) and are costly [27]. On the other hand, depending on the data gathered on the field, the epidemiologists may need to collect additional data in order to explain a phenomenon (for example, the height of the patients in order to calculate their body mass index). This task may be done by using supplementary materials such as paper or spreadsheets (which can lead to a problem of data integration) or new requirements can be introduced and the software updated (which can lead to the problem of software regression).

The problem of failed software used for epidemiological surveillance are often the result of an unsystematic transfer of business requirements to the implementation [117]. This problem can be avoided if the system is established using a well-defined framework/architecture permitting the rapid development/update of the surveillance software by non-informatics experts such as health workers. Chapter 3 presents the approach we propose for this problem.

2.4 Epidemiological surveillance of tuberculosis

The recognition of Tuberculosis' (TB) substantial burden has kept TB control at the top of the international public health agenda since the 1990s. In this section, after the presentation of tuberculosis, we will present how the National Tuberculosis Control Program (NTCP) in Cameroon has been designed to fight against TB in Cameroon.

2.4.1 Tuberculosis

Tuberculosis (TB) is an infectious and contagious disease caused by *Mycobacterium tuberculosis* or *Koch Bacillus (BK)*. One third of the world's population is estimated to be latently infected with TB bacteria; about 1.4 million deaths caused by TB between 2011 and 2016 [56, 57]; 8.7 million new cases in 2011 [56]; 10.4 million new cases in 2016 [57]; 1.3 million deaths underlying caused by TB among HIV patients. The WHO reported 1.4 million people died from TB[13] in 2019. Tuberculosis remains a global scourge. It is the top killer amongst infectious diseases globally [57, 138].

Developing countries are those who are most affected. It is a disease of poverty that thrives where social and economic determinants of ill health prevail, and it affects mostly young adults in their most productive years; 95% of TB deaths being in the developing countries. The proportion of TB cases coinfected with HIV was the highest in countries in the African Region; overall, 39% of TB cases were estimated to be coinfected with HIV in this region, which accounted for 79% of

[13] `https://www.who.int/news-room/fact-sheets/detail/tuberculosis`

TB cases among people living with HIV [56, 57].

Most recently, the global concerns about the emergence of multidrug-resistant and extensively drug resistant TB (MDR and XDR-TB) caused by the bacterium's resistance to the usual drugs complicates the management of TB and represents one of the most important emerging challenges in the control of TB worldwide. In fact, the MDR and XDR-TB treatment is long, costly and less effective. Also, resistant strains can propagate to other individuals. For example, among patients with MDR-TB who started their treatment in 2009, only 48% were successfully treated. Among a subset of 200 patients with XDR-TB in 14 countries, treatment success was only 33% overall and 26% died [56, 57]. Then, MDR and XDR-TB are global concerns in the world of today. The management of drug resistance is based on the prevention of resistant strains and the early detection and proper treatment of patients.

Conscious of that, ambitious targets for reduction of the epidemiological burden of TB have been set within the context of the Sustainable Development Goals (SDGs) and the End TB Strategy. Achieving these targets is the focus of national and international efforts. Global targets for reduction in the epidemiological burden of TB have been set for 2015 and 2050 within the context of the Millennium Development Goals (MDGs) and by the Stop TB Partnership [56, 57]. The goal being that the prevalence and death rates should be halved by 2015 compared to their levels in 1990, and that TB should be eliminated as a public health problem by 2050 (defined as less than one case per million population) [56]. To reach these targets, data must guide decision making. To prevent susceptible individuals from becoming infected or developing the disease, an epidemiological surveillance system of the disease must be put in place and reinforced by other activities that can help to identify and treat the patients in time. To strengthen the epidemiological surveillance, additional activities including population sensitization, drug management, recall of patient for drug taking, etc. are generally integrated [17, 27, 62, 110]. For example, in Cameroon to efficiently fight against tuberculosis, the Cameroonian National Tuberculosis Control Program (NTCP) in addition to epidemiological surveillance manages anti-TB drugs, follow-up appointments of patients, sensitization of patients, etc.

Tuberculosis is transmitted by air from a patient with pulmonary tuberculosis. Contamination occurs through droplets, loaded with tubercle bacilli from the patient's lungs. These fine droplets are produced when the patient sneezes, coughs, speaks or laughs. They dry quickly and can remain suspended in the air for several hours. Other modes of transmission of TB bacilli, such as manual contact with contaminated objects or the accidental introduction of the bacillus through the skin are very rare and of no epidemiological significance [105].

The main reservoir of the tubercle bacillus is the pulmonary TB patients with positive microscopy (TPM+). TB patients with negative microscopy (TPM-) rarely transmit the disease. Patients with extra-pulmonary TB are non infectious [105]. The screening and treatment of pulmonary TB is the main goal of the NTCP.

The treatment protocol of every patient is defined according to their bacteriological status (pulmonary TB or negative pulmonary TB), the location of the disease (pulmonary or extra-pulmonary), the patient's therapeutic history (never been treated or already treated for TB). Patients are classified in: new cases (that have never been treated) and cases to retreat (in case of relapses or failures). Many reasons may cause the failure of the TB treatment: the non-intake of the medi-

cation regularly, the non completion of the TB medication as prescribed by the health personnel. Roughly, if the treatment protocol is neglected, it will fail and this may lead to multidrug-resistant and extensively drug resistant TB [95].

TB control is based on preventing susceptible individuals from becoming infected, infected individuals from developing the disease and individuals with TB from contaminating others. To do this, surveillance data will permit us to measure the burden of the disease and thus, serve as the basis to inform decisions regarding the planning and targeting of health care interventions at the national and international levels [28].

2.4.2 Tuberculosis surveillance in Cameroon

To fight against TB, the government of Cameroon has put in place the National Tuberculosis Control Program (NTCP). In 2002, it was recognized as a priority program of the Ministry of Health (central level). It is now attached to the Ministry of Health following the reorganization of the program in 2002 with the creation of the National Committee to Fight Against Tuberculosis, its Central Technical Group and its decentralized units. This testifies the determination of the Cameroon government to effectively fight against this disease.

The goal of the NTCP is to detect, treat patients with TB and prevent the disease in persons at risk. Persons at risk are children who have been in contact with an adult with TB or persons living with HIV. To do this, NTCP has 248 Centers for Diagnosis and Treatment of Tuberculosis, a national reference laboratory (Centre Pasteur du Cameroun - CPC), four reference laboratories and has put in place a system to monitor the disease. To treat patients and prevent the disease, NTCP has put in place an epidemiological surveillance system.

To detect and treat patients, NTCP uses passive screening of symptomatic patients by microscopic examination of sputum (bacilloscopy). The health personnel who take care of the patient must identify the closest hospital to the patient's home for his/her treatment. At the hospital level, patient's data is collected and recorded in the TB Registry and treatment is initiated. The patient's treatment data are recorded each time he/she goes for an appointment at the hospital using different data collection tools, which are paper based (treatment card, tuberculosis register and patient's card). To treat all children below 5 years of age, who have had contact with a TB case, health personnel must sensitize the patient to bring children who have been exposed for prevention. To this end, all patients must complete a form for all children exposed to the disease. This form allows, depending on the child's history (illness, contact with a TB case) and laboratory tests to determine whether the child should follow or not a treatment. At the end of the treatment, the patient's issue (healing, treatment completed, failure, death, lost sight, transfer) must be recorded in the tuberculosis register. When screening or treating a patient, he/she may be transferred or referred:

- **Transfer:** a patient is transferred if he/she initiated his/her treatment in a hospital (*hospital A*) and decides to continue in another hospital (*hospital B*). *Hospital A* delivers a certificate allowing him to continue the treatment in *hospital B*. Thus, the patient no longer needs to register in *hospital B*. At the end of the treatment, *hospital A* must be informed of the treatment issue.

- **Referral:** a patient is referred if he/she initiated his treatment in a hospital (*hospital A*) and the doctor decides for some reasons that he/she must continue in another hospital (*hospital B*). In this case, the patient is not registered in *hospital A*, but in *hospital B*. The *hospital A* must have feedback to know if the patient has arrived at the referral center and is really supported.

Once detected, new TB cases are treated with a standardized therapeutic regimen lasting six months. Relapses, reinstatements and failures benefit from a standard 8 month reprocessing plan. MDR-TB patients are treated in specialized services with a short-term (12-month) second-line regimen. The patient's treatment data are recorded each time during his visit to the hospital on cards.

Chemoprophylaxis targets children under 5 years of age with TPM + contacts and Persons Living with HIV (PLHIV). Health staff should sensitize the detected adult patient to bring in target children who have been exposed for prevention. Health personnel must complete a scorecard for all children suspected of being exposed to the disease. This card allows, depending on the child's history (illness, contact with a TPM +) and laboratory tests to determine whether the child should follow or not chemoprophylaxis.

Reports are transmitted following the health system pyramid as described by the figure 2. At the end of each quarter, the hospital reports (sometimes obtained by a manual analysis of data) are sent to the district level, where all the statistics are compiled and sent to the regional level, and the regional level compiles the statistics of all districts and sent to the NTCP.

To improve on the TB surveillance network, the NTCP regularly organizes the training sessions for health workers involved in the management of TB and it ensures that every health personnel must participate. NTCP has put in place many strategies to make people adopt behaviors that can reduce TB morbidity. This consists of informing people about the curability of the disease, organizing information campaigns for early detection, tracing contacts (asking patients to bring children under the age of five that are exposed for consultation), promote adherence to TB treatment and integrate cured patients in the community.

Epidemiological surveillance at the NTCP is done manually. It uses paper forms for data collection (registers, notification forms, statistics sheets) and the road for data transmission. At the level of the districts, collected data from hospitals are compiled and statistics are transmitted to regional levels. At the regional level, district data is aggregated and transmitted to the NTCP. At the end of each year, the data is archived in a closet. The figure 8 presents the archiving of data at the Jamot Hospital. Several types of reports are produced: quarterly reports for tuberculosis case detection, reports of the treatment results of pulmonary tuberculosis cases between 9 to 11 months, reports of the distribution of drugs and consumables. The manual management of data poses a number of problems. The most important ones reported in a survey that we conducted on the staff of the NTCP are: a poor rate of completude and promptitude; late production of basic statistics; insufficient awareness of the population; under use of laboratory data; difficulties to identify the personnel already trained by the NTCP; difficulties to geolocate hospitals in order to better guide a patient; difficulties to manage lost sight patients.

Figure 8: Archiving data at the Jamot Hospital in Yaounde

2.5 Conclusion

In this chapter, we presented a review of epidemiological surveillance in 3 main points: epidemiological surveillance process, epidemiological surveillance systems architectures and epidemiological surveillance tools. The epidemiological surveillance process involves the main activities of epidemiological surveillance which are data collection, data analysis, information interpretation and dissemination to stakeholders. The data are collected at all levels of the health system and the data from lower levels are sent to upper levels to report the overall health situation. The data obtained after the data collection process is analysed in order to extract useful information. Information obtained is sent to stakeholders in a form that changes their understanding of health issues and needs. To provide relevant information to stakeholders, epidemiological surveillance systems are implemented according to Form-Route-Server, Mobile-MobileNetwork-Server, Computer-Internet-Server/Computer-MobileNetwork-Server, and multi-strategy architectures. The Form-Route-Server architecture uses forms (paper or electronic) for data collection and the road for data transmission. Even if this architecture may ensure the reliability and completeness of data, the promptitude of data, data security, and the versatility of the system are not guaranteed. The Mobile-MobileNetwork-Server architecture uses mobile phones for data collection and the mobile network for data transmission. This architecture is robust and the promptitude of the data is ensured. However, the security, the versatility, the completeness and the consistency are not always guaranteed. The Computer-Internet-Server architecture uses a standard Web application for data collection and management. Although it allows us to guarantee the promptitude and security of data, and the versatility of the system, it is difficult to deploy because the Internet is not available in many areas. To improve the epidemiological systems, Multi-strategy architectures combining the previous architectures are generally used. The architectures presented in this chapter are implemented in many epidemiological surveillance tools such as DHIS, OpenMRS. At the end of the chapter, we presented the epidemiological surveillance of TB in Cameroon as the domain of application of the thesis.

We noted in this chapter that the unsystematic transfer of business requirements of epidemiological surveillance systems to the implementation causes the problem of failed software. In chapter 3, we propose a solution to this problem.

3

A MDA-based approach for the development of epidemiological surveillance systems

Software developers of today face the problems of increasingly complex softwares as clients demand richer functionalities in shorter timescales. This has given rise to the software development questions and dilemmas: how to provide the customer with the software and software artefacts that fulfil their expectations in time and within the budget? To respond to these questions, many approaches and tools are proposed in the domain of software engineering [102]. In this chapter, we will present the agile software development methodologies (section 3.1), the Model-Driven Architecture approach (section 3.2.2), our approach which is the combination of agile methodology and MDA approach for the development of epidemiological surveillance systems (section 3.3) and the application of our approach to develop EPICAM, an epidemiological platform of TB in Cameroon (section 3.4).

3.1 Agile methodologies

The use of the traditional software development methodologies to develop health information systems may lead to software failure because the needs are not clear and are constantly changing [3, 100]. This section presents an alternative to traditional software development methodologies which is the agile methodology. Section 3.1.1 presents an overview of the agile methodology, section 3.1.2 presents the scrum methodology and section 3.1.3 presents the use of agile methodologies to develop health information systems.

3.1.1 Agile overview

Agile software development methodologies started officially when the agile movement presented the agile manifesto in 2001. The agile manifesto contains the central values which proposes to place more emphasis on people, interaction, working software, customer collaboration, and change,

rather than on processes, tools, contracts and plans during software development [3, 40, 68, 115]. It defines the twelve principles of agile software development methodologies: customer satisfaction by early and continuous delivery of valuable software; welcome changing requirements, even in late development; deliver working software frequently (weeks rather than months); close, daily co-operation between business people and developers; projects are built around motivated individuals, who should be trusted; face-to-face conversation is the best form of communication (co-location); working software is the primary measure of progress; sustainable development, able to maintain a constant pace; continuous attention to technical excellence and good design; simplicity-the art of maximizing the amount of work not done is essential; best architectures, requirements, and designs emerge from self-organizing teams; regularly, the team reflects on how to become more effective, and adjusts accordingly.

Agile methodologies rely on an iterative, incremental and adaptive development cycle which considers that the needs cannot be fixed and proposes to adapt the development to the changes. It's intended to support early and quick production of working code. In the particular case of health information, it allows fast deployment and adoption [9]. It consists firstly of setting a short-term goal and of getting into the project without delay. Once the first objective is attained, we take a short break and adapt the path according to the situation of the moment, and so on, until we reach the final destination. This helps us to give more visibility, involving the client from the beginning to the end of the project and adopting an iterative and incremental process. In a broader sense, an agile methodology can be seen as a process consisting of an initialization step, iterative steps and incremental steps [3, 78].

With the agile methodologies, the project is divided into modules (each module contains a list of deliverables) of shorter spans. A project control list containing all tasks that need to be performed is also created. The project control list is constantly being revised as a result of the software development. During the development, the following process is performed:

- **Initialization step**: During the initialization, the base version (prototype) of the application is developed. This base version contains a solution to the problem that is simple enough to understand. This first version is evaluated by a focus group not associated with the software development team in order to obtain unbiased opinions. The information obtained from the focus group will be incorporated into the next iteration.
- **Incremental step:** After the delivery of the first version of the software, the developers complete the list of deliverables given the users' feedback and start a new module, which is a new increment, and an iterative process starts on this increment.
- **Iterative step:** The development of each module is done by one or many iterations until its delivery. During the iterations, the analysis (structure, modularity, usability, reliability, efficiency, and achievement of goals) of users' feedbacks (focus group) and of the developed application will be used to redesign (adding/removing functionalities, introduction of new technologies, etc.) and a new version of the software will be implemented. At the end of each iteration, the partial but usable product is validated by the client by ensuring that it aligns with the needs.

Agile is team-based and user-focused means that:

- **Team-based:** with the agile methodology, the development group comprises both software developers and customers representatives. During the development, everything is made public and transparent. The clients and the customers are actively involved in the development process and have frequent and early possibilities to see the work being delivered, in order to make decisions and changes if necessary. In a broader sense, the agile software methodology relies on teamwork, collaboration, time boxing tasks, and the flexibility to respond to changes as quickly as possible. Team members share ownership of the project and each of them play an active role in completing the module within the estimated time.
- **User-focused:** after each iteration, the delivery version containing the features developed can be used by the users.

The success of agile software development methodologies has given rise to a large number of research work. The scientific papers [40, 68, 115] present a few. There are a large number of agile software methodologies adopted in many software engineering, requirements engineering, and consulting organizations: Extreme Programming (XP) [18], Dynamic Systems Development Method (DSDM), Rational Unified Process (RUP) [40], Scrum [119], crystal family of methodologies [3], feature driven development (FDD) [114], Open Source Development (OSD) [3]. In this thesis, Scrum is adopted to develop software and software artefacts.

3.1.2 Scrum

The first reference of the term "Scrum" originates from Japan and refers to an adaptive, quick, and self-organizing product development. Scrum describes how the team members should be organized in order to develop a system using agile principles [119]. In this section, the scrum methodology is presented in two main points: major Scrum concepts and Scrum process.

3.1.2.1 Major concepts

Using the Scrum methodology entails the mastery of some major concepts. The roles that can be identified in Scrum are: Scrum Master, Product Owner, Scrum Team, Customer, User and Management. The main meetings identified in Scrum are: Sprint Planning Meetings, Scrum Weekly Meeting, Scrum Daily Meeting and Scrum Review Meeting. The working tools of the team are Product Backlog, Backlog Items and Sprint Backlog.

- **Product Backlog** defines the work to be done in the project. It is created at the beginning of the project, is constantly updated as the project goes on and contains the prioritized list of businesses and technical requirements of the system to be built or enhanced. Backlog items can include, features, functions, bug fixes, defects, requested enhancements and technology upgrades;

  The Product Backlog List, contains all the requirements that are currently known. These requirements are prioritized and the effort needed for their implementation is estimated. It is reviewed by the Scrum Team and updated if necessary at the start of each increment.

- **Sprint** is an iterative work cycle during which the development is completed and made available for review. During a Sprint, the Scrum Team organizes itself to produce a new executable version of the product. The elements involved in the development from the specification to the implementation may evolve.
- **The Scrum Master** is the person in the Scrum team who interacts with all the others during the project. His/her role is to ensure that the project is carried through according to the practices, values and rules of Scrum and that it progresses as planned;
- **The Product Owner's** role is to control and make visible the Product Backlog List. He/she is appointed by a mutual agreement with the Scrum Master, the customers and the manager;
- **The Scrum Team** is composed of the software developers and the customers. They work together and define the effort estimation, the creation and the update of the Product Backlog List and suggest impediments to be removed from the project. The Scrum Team decides and organizes itself in order to achieve the goals of each Sprint;
- **The customer**, in accordance together with the Sprint Team create and update the Product Backlog Items for the system being developed;
- **Management** is responsible for the final decision on the product. They set the goals and requirements of the project;
- **A Sprint Planning Meeting** is a two-phase meeting organized by the Scrum Master. The first phase involves the customers, users, management, Product Owner and Scrum Team and the second phase involves the Scrum Master and the Scrum Team. During the first phase, participants decide upon the goals and the functionalities of the next Sprint. The second phase focuses on how the product increment is implemented during the Sprint;
- **Sprint Backlog** is the starting point of each Sprint involving the selection of a list of Product Backlog Items to be implemented. Unlike the Product Backlog, the Sprint Backlog is stable until the Sprint (one week, two weeks or one month) is completed. When all the items in the Sprint Backlog are completed, a new iteration of the system is delivered;
- **Sprint Review Meeting** is the meeting held at the end of the sprint during which the Scrum Master presents the results of the Sprint to the management, customers, users and Product Owner. The different parties assess the product obtained after the increment and decide about the product and the activities that follow (e.g., change the direction of the product under development).
- **Daily Scrum Meetings (approximately 15 minutes)** conducted by the Scrum Master, are organized to keep track of the progress of the Scrum Team continuously and serve as planning meetings. During the Daily Scrum meeting, the main questions to be answered are: what has been done since the last meeting and what is to be done before the next one? Any problems, deficiencies or impediments in the system development process or engineering practices are looked for, identified and the solution is put in place.

3.1.2.2 Process

Scrum process includes three phases: pre-development, development and post-development.

- **Pre-development:** The pre-development phase involves the planning and the design. The planning is done based on the Product Backlog List, in collaboration with the customer. The requirements are prioritized, the resources (team, tools, etc.) needed for their implementation are estimated, risks assessed, identification of training needs and verification management approvals made. The design phase is done by defining the architecture based on the Product Backlog List; preliminary plans for the contents of releases are proposed. The design meeting permits us to make decisions about the implementation and preliminary plans for the contents of releases are prepared.
- **The development:** During the development phase, the system is built in Sprints during which the functionalities are developed or enhanced to produce new increments.
- **Post-development:** This phase is reached when the requirements are completed. The system is ready for the release and tasks such as system integration and documentation are made.

3.1.3 The use of agile in the medical domain

The use of agile software development methodologies to develop health information systems is not new. In fact, due to the complexity of the processes in health care, changes in the requirements may introduce a need for a correction of the implemented system, and can lead to regression [9]. This problem can be avoided by involving end users to the development process and making an adapted methodology for iterative and gradual development. Many authors [8, 100, 87] presented the adoption of agile methodology for the design and implementation of health information systems.

3.2 A review of Model Driven Architecture

The problem of failed software is often the result of an unsystematic transfer of business requirements to the implementation level [117]. To solve this problem, the MDA approach permits us (software engineers and/or non-informatics experts) to model the software to be developed and automatically generate the executable source code. This section presents in section 3.2.1 the role of models in software engineering, section 3.2.2 presents an overview of MDA, and section 3.2.3 presents the use of MDA to develop software in the healthcare domain.

3.2.1 The role of models in software development

A model is a computerized representation of a problem/domain permitting their description in a way that avoids delving into technological details. It represents a specification of the function, structure and behavior of a problem within a given context, and from a specific point of view [102]. To do this, it provides abstractions of a system that allows various stakeholders to reason about the system from different viewpoints and abstraction levels. One way to build models is to identify the concepts of the domain, the links (relations) between these concepts, and to make a diagram representing these concepts which are in relation. The concept's names are familiar to people working in the domain of the problem, and not to IT experts [117].

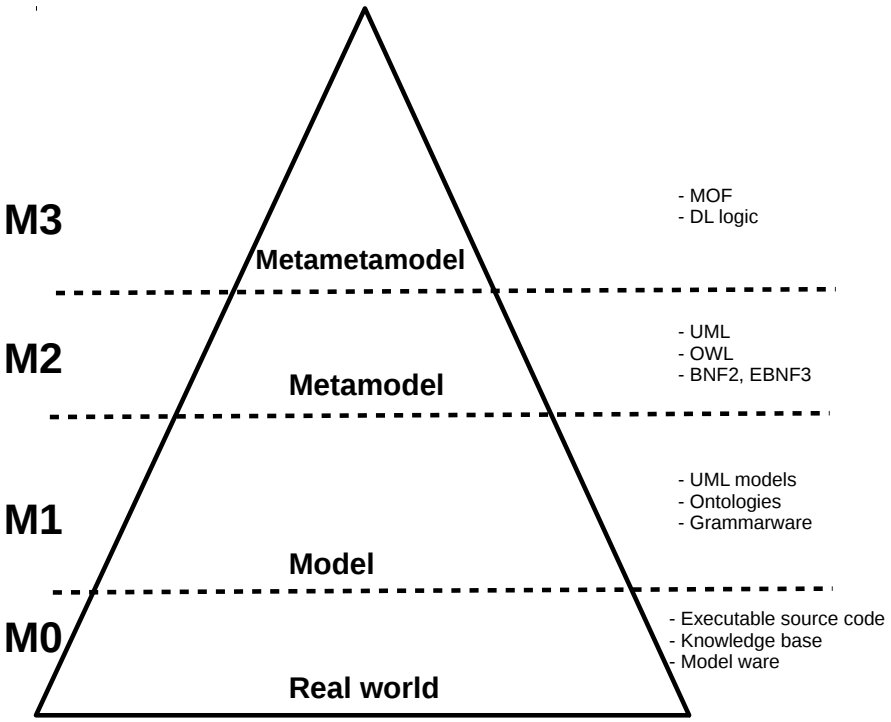


Figure 9: The OMG meta-pyramid of models

The traditional framework for modelling is based on an architecture with four layers, presented by the figure 9 and described as follows [31, 44]:

- **The Real world layer:** At the base of the pyramid, the real world represents the problem/-domain that one wants to model. It is comprised of all the elements that we want to describe;
- **The model layer:** The models (for example, UML class diagram) representing the abstraction of the real world/the problem make up the first level. It is used to answer the questions about the modelled system in exactly the same way that the system would have answered itself. It can replace the system in certain situations;

- **The meta-model layer:** The notion of model refers to the notion of well-defined languages. The language used to expressed the model (for example, the UML language) is called a meta-model ;
- **The meta-meta-model layer:** As for models, the meta-model must be defined from a modelling language. To avoid inordinate increase of various and incompatible meta-models the meta-meta-model layer comprises the language used to define different kinds of meta-models. In theory, one can continue with a language to define the meta-meta-model, etc. However, in the MDA domain, to limit the number of levels of abstraction, the Meta-Object Facility (MOF) is the meta-meta-model that has the property of meta-circularity (that is the ability to describe itself).

Models can be used in many ways, for example, to predict the qualities (e.g. performance) of a system, validate designs against requirements, communicate system characteristics to business analysts, architects and software engineers, and as the blueprint for system implementation [41]. In a broader sense, one can distinguish:

- **Usage modelling:** Usage models such as use cases, use cases scenarios, use case diagrams, and user stories permit us to identify how people work with the system, what they do with the system and how the system supports their usage;
- **Process modelling:** Process models (e.g., data flow diagrams, flow charts, activity diagrams) explore how users work with the system by considering the flow of activities being performed;
- **User interface (UI) modelling:** UI models (e.g., UI prototypes, UI flow modelling) permit us to show using a visual diagram the different user interface artefacts (buttons, text field, text area, etc.) which permit the users interact with the system;
- **Conceptual domain modelling:** Conceptual models such as class diagrams, object diagrams permit us to identify the entities, their properties and their relationships within the problem domain;
- **Architectural modelling:** Architectures (e.g., free-form diagrams, component diagrams, package diagrams, deployment diagrams, network diagrams) are the high-level design of a software;
- **Dynamic design modelling:** Dynamic models such as sequence diagrams, interaction diagrams, communication/collaboration diagrams are used to explore the behavioral aspects of the system for one/many objects involved in the system.

To build models, the concepts and relations between these concepts are represented diagrammatically (visual modelling), or using text. The main advantage in the use of visual notations in the development of software systems is the ease of comprehension of graphical notations by non-technical experts [86]. Then, by using the visual model, non-IT experts are allowed and able to customize these models to their particular context, needs and feel confident that the customization is trustworthy and accurate [86]. In the medical domain for example, many domain experts

use dedicated tools integrating graphical modelling features (e.g., Protégé software) to build and maintain ontologies [112].

3.2.2 An overview of MDA

Model Driven Engineering (MDE) is a software development approach based on the use of models to develop software artefacts. Its goal is to improve the productivity and maintainability of software by raising the level of abstraction from source code, to high-level domain-specific models such that developers can concentrate on application logic rather than implementation details. It is identified by several paradigms such as Model Driven Software Development (MDDS), Language Oriented Programming (LOP), Domain-Specific Modelling (DSM), Framework Specific Modelling (FSM), Model-Driven Architecture (MDA), etc. A great challenge of MDE is to make modelling and the use of models directly benefit the individuals, and not just restricted to a select few. For this to be possible, individuals are provided with straightforward access to models that encode global information relevant to their context [34]. Then, the Object Management Group (OMG)[1] has adopted and promoted the MDA as an architecture and framework for software development based on models.

3.2.2.1 A broader description of the MDA approach

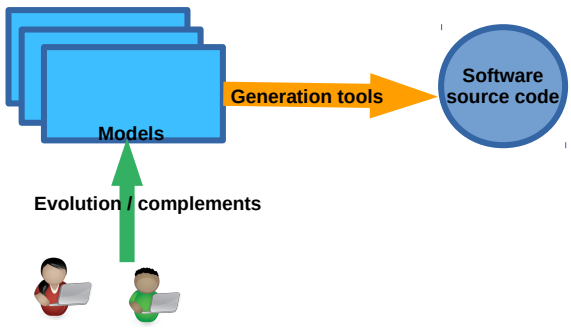


Figure 10: MDA Software development approach

The MDA approach is an approach (Figure 10) to software specification, design, and implementation which provides guidelines for structuring software specifications that are expressed as models. It focuses on forward engineering in which the executable source code is (semi)automatically generated from abstract, human-elaborated modelling diagrams such as a class diagram [8, 9, 106, 117]. A class diagram describes the structure of a domain by identifying the domain classes (e.g., patient), their attributes (e.g., age, sex), their operations (e.g., calculate a body mass index) and the relationships amongst classes (e.g., the relation between a patient and his/her appointments at

[1] https://www.omg.org/

the hospital) [86, 112]. Unlike object-oriented design approaches where the basic concept is the object, the basic concept of MDA is the model. Models are used throughout the development life cycle of an application, that is, from the application specification to code generation. The MDA approach allows developers to focus exclusively on the specification of the business logic and to generate subsequently and automatically the source code corresponding to the specifications made previously. For the generation task, the MDA approach proposes to provide developers with automation tools [20, 54]. When adopting the MDA approach, a natural counterpart is the framework which is the body of code that implements the aspects that are common across an entire domain, and exposes as extension points those elements of the domain that vary between one application and another. This framework permits us to link the business level and the information technology [117, 131].

The main concepts to master when using the MDA approach are:

- **System:** The context of MDA is the software system, either preexisting or under construction;
- **Model:** The main element permitting us to formally specify the function, structure and behaviour of the system is the model;
- **Model driven:** this is the approach to software development whereby models are the main source for documenting, analyzing, designing, constructing, deploying and maintaining a system;
- **Architecture:** This is a specification of the parts, connectors of the system and the rules for the interactions of these parts using the connectors;
- **Platform:** a platform is a set of technologies providing a coherent set of functionalities to solve a problem;
- **Platform independency:** This is a property that models exhibit when they are built independently of the features of a platform;
- **Platform model:** Describes a set of technical concepts representing its constituent elements, the services it provides and constraints on the use of these elements and services by another part of the system;
- **Model transformation:** This is the process consisting of the conversion of one model to another one;
- **Implementation:** These are the specifications providing all the information which are used to construct and put a system into service.

The three primary goals of MDA are:

- **Portability:** To permit the portability, the MDA approach separates the models and the transformations such that height level models do not contain low level platform and technical details. Then, the model can move across different tools and environments. If the underlying

platforms changes or evolves, the upper level models can be used on a new platform directly without any remodelling;

- **Interoperability:** Today, it is rare to develop enterprise applications that do not communicate with others (internal and external of organisation). This communication is done in a heterogeneous and distributed manner. With the MDA approach, the horizontal model mapping and interaction is done to permit heterogeneous systems to interoperate. Then, the development of integration bridges with legacy and/or external systems is greatly facilitated and new infrastructure can be more easily integrated and supported by existing ones;
- **Reusability:** To facilitate the tasks of design and implementation, designers and developers reuse existing models and tools. Best practices in designing applications are the key elements for improving productivity and quality. Then, businesses are able to extract greater value out of their investments in tools.

To develop applications using MDA approach, two approaches are to be consider [86, 117]:

- **The development by a domain experts:** In this approach, domain experts directly create and maintain specifications of their information and workflows. It entails that they master modelling and the use of the modelling tools (training on approach and tools is required). Given that the domain expert is the person who masters the domain and the context, this approach permits us to obtain a good model for the domain and given the context [117].
- **The development by domain experts and IT experts**: This approach involves domain experts and the IT experts during the development. The domain experts elaborate the specifications, the IT experts build the model, generate the software and the domain experts validate the software built. In this approach, domain experts are not necessarily expert modellers [112].

In recent years, MDA has been adopted successfully and implemented in many small and large organizations [35] such as automotive, aerospace, telecommunications, business information systems, health information systems [90, 112], crisis emergency [36], etc. In section 3.4, we will present the use of MDA to improve the development of tuberculosis surveillance systems.

3.2.2.2 MDA Process

The two key concepts of MDA are models and transformations. All the models are well defined by a modelling language, itself defined by syntax and semantics in a metamodel. The transformation from one model (e.g., requirements) to another model (e.g., design) is a systematic process explicitly defined by transformation rules. During the development of applications using the MDA approach, the business models as a result of the discussion between domain experts and modelling experts are (partially) automatically transformed through the three models layers ($CIM \longrightarrow PIM \longrightarrow PSM$). The process of building applications using the MDA approach can be summarized as follows [112, 131]:

- **The construction of the Computational Independent Model (CIM):** The CIM contains all the information about the business model of the application to be developed. Built using the vocabulary that is familiar to the domain experts, the CIM focuses on the context and requirements of the system without consideration for its structure or processing; describes the supported business case in a complete and comprehensive manner; presents exactly what the system is expected to do, but hides all the information technology related specifications to remain independent of how that system will be (or currently is) implemented. For this purpose, an adequate graphical representation is needed (e.g., UML sequence diagrams), which is understandable and intuitive from the perspective of domain experts;

- **The development of the Platform Independent Model (PIM)**: Obtained from the CIM, the PIM is exactly what its name denotes: a model that is free of specific hardware or computing platform requirements and constraints. It addresses the modelling expert, who internalized the business context of the model, the software engineer, who will externalize the technical context of the model; and exhibits a sufficient degree of independence so as to enable its mapping to one or more platforms. An example of a PIM is the class diagram modelling the domain and presenting the entities and the relationships between these entities;

- **The construction of the Platform Specific Model (PSM):** The PIM constructed can be transformed into one or more PSM for different target platforms like Java Enterprise Edition (JEE) or .NET by integrating platform specific details to the PIM. The PSM contains detailed descriptions and elements specific to the targeted implementation platform. It is obtained by combining the specifications in the PIM with the details required to stipulate how a system uses a particular type of platform;

- **The generation of executable source code:** The PSM containing the technology-related aspects of the target platform is derived into software solutions which are application source code, configuration files, component descriptor files, deployment files, build scripts, documentation, etc. Depending on the maturity of the MDA toolset, code generation varies from significant to substantial or, in some cases, all the source code is generated.

During the development of applications using the MDA approach, it is the first and the second steps in the process that involve creativity and manual work; steps three and four are automated by the use of automated tools [106].

3.2.2.3 MDA tools

Tools are an essential part of MDA. In fact, tools support the overall modelling and transformation process [112]. Then, designing and developing the MDA tools environment properly is one of the most crucial steps when one has adopted the MDA approach. It is crucial to establish the right infrastructure for the development environment, along with a dedicated team to support, manage and continually ameliorate this environment to meet the development requirements. The IT team must carefully evaluate and select (or develop if the tool doesn't exist) the right tools with the capabilities to meet the organization's specific requirements.

In the MDA domain, one can distinguish two types of tools. Some are used to develop the tools for the modelling and the generation of software artefacts and the others are used for modelling the domain and generating the executable software. Amongst the tools used to develop the MDA tools, we distinguish:

- **Eclipse software:** The Eclipse software is an open source Integrated Development Environment (IDE) developed in Java and primary used to develop software in several programming languages such as Java, C++, C, Haskell, JavaScript, PHP, Python, R, Ruby, etc. Eclipse is based on a small run-time kernel and can be customized by adding additional modules in the form of plug-ins for dedicated tasks such as programming, testing, modelling, etc. The Eclipse platform contains plug-ins for graphical and textual model development. These modelling plug-ins focus on model-based development are separated in several categories: tools for model development, model transformation, concrete syntax development, abstract syntax development, etc. They implement various modelling standards used in industry such as UML, SysML, OCL, etc.
- **Eclipse Modelling Framework (EMF):** EMF integrates a modelling framework and a code generation tool for developing software using models. It includes tools for model definition (Ecore), a tool (EMF-Edit) for building editors for EMF models and a code generation facility (EMF.Codegen);
- **Graphical Modelling Framework (GMF):** The GMF provides a set of features for the development of graphical editors based on EMF;
- **Xpand:** The goal of Xpand programming language is to generate the source code from EMF models;
- **Xtext:** This is an open source framework used to develop programming languages and Domain Specific Languages (DSL). The grammar of the language written with Xtext describes how an Ecore model is derived from a textual notation;
- **Modelling Workflow Engine (MWE):** MWE is used to support orchestration of different Eclipse modelling to be executed within Eclipse. It is mainly used to configure the Xtext code generator;
- **Atlas Transformation Language (ATL):** This is an open source EMF model-to-model transformation language;
- **Acceleo code generator:** Acceleo is used to generate text given an EMF model (model-to-text transformation). With Acceleo, from an EMF model, the source code in several programming languages such as Java, PHP, Python can be generated.
- etc.

The tools used to generate executable source code such as Acceleo[2], Imogene[3], AndroMDA[4], etc. are composed of three main components [102]:

[2]https://www.eclipse.org/acceleo/
[3]https://github.com/medes-imps/imogene/wiki
[4]https://www.andromda.org/

- **A domain-specific modelling language:** Also called modelling languages, these languages focus on a particular problem and are designed so that they can represent the problem domain which is being addressed. It is composed of the domain concepts and rules and specifies the mapping from the model constructed to specific source code [35];
- **A model editor:** Textual and graphical editors, together with debugger and code generators are used to build models and generate executable source code with little effort;
- **A code generator:** The code generator is the module which permits us to generate the executable source code from the models.

Note that MDA is not appropriate if it is used to develop only one software. In fact, the time spent to develop an advanced tool with appropriate modelling formalism, get the model right and generate the software will be more than developing the same software using general purpose language.

3.2.3 MDA in healthcare

The domain of health informatics is complex. In fact, it is an evolutionary IT ecosystem of information systems organized in a network of applications of different design and developers. This IT ecosystem attempts to provide solutions at numerous levels and in multiple disciplines such as real-time monitoring of intensive care unit patients, clinical decision support, distributed interoperable health records, administrative and financial decisions, etc. [112, 117]. To efficiently master the challenges of the increasing complexity of health information systems and provide solutions to a wide range of problems of these systems, the literature review shows that the MDA approach has been widely adopted [8, 9, 112, 117]. In this section, we will present the adoption of the MDA approach in the health information domain. Thereafter, we will present the tools based on MDE to enhance epidemiological surveillance and at the end, the combination of MDA and agile methodology to enhance the development of health information systems.

3.2.3.1 Adoption of MDA in the health domain

There is a significant number of examples on the adoption of the MDA approach in the health informatics domain [8, 112]. To cite a few:

- Blobel and Pharow [22] presented a work done in Germany consisting of the establishment of a health telematics platform combining card enabled communication with network based interoperability;
- Schlieter et al. [117] attempted to create a specific MDA for the health domain. They show how an MDA approach can be used in order to build a method, fundamental for a systematic creation of an application system. Their use-case is an application for IT based workflow support for an interdisciplinary stroke care project;

- Curcin et al. [33] presented a work on the use of MDA to build tools which allow non-IT experts to model their requirements and to generate data collection tools for medical organizations;
- Jones et al. [70] presented the use of MDA to develop m-health (Mobile-health) application providing an integrated set of personalized health-related services to the user;
- Rayhupathi and Umar [107] proposed the use of MDA to develop a system capable of tracking patient information;

3.2.3.2 MDA tools to enhance epidemiological surveillance systems

In the field of epidemiological surveillance, MDA tools are generally used to develop data collection tools. Some integrate the generation of the main statistics. In this field, we have identified the following tools:

- **Imogene:** It is an open source platform based on MDA, developed by MEDES in France and used for the generation of data collection software. The Imogene Studio provides graphical tools for creating templates and generating tools that helps to generate a set of applications (for different platforms) based on models. The Imogene model editor allows the creation of a model of the entities involved in the application. These entities are the forms used for data collection. The modelling task consists of defining the forms to be generated, the fields they contain and the properties associated with all objects. Once the model is built, it proceeds to the automatic generation of the code. Taking advantage of the MDA approach, Imogene makes it easy to update an already deployed information system. This process requires the update of the model that contains the design form, the rebuild and the redeployment of the application.

  Imogene was used to generate data collection applications during the prevention and follow-up of diabetes in France in 2009[5], in Cameroon and Georgia to generate data collection tools for the epidemiological surveillance of tuberculosis. It is the foundation of the development of the EPICAM platform for epidemiological surveillance of tuberculosis presented in section 4.

- **Magpi:** This is a platform developed by the NGO DATAGYNE. It offers free versions and paid versions when collecting large amounts of data. It is used to generate mobile data collection forms. When using Magpi, the forms are designed on the `www.magpi.org` website and downloaded on a mobile phone as a JAVA ME, Android or ios application for data collection. All collected data is saved on a remote server and can be analyzed later. Magpi has been used in around 170 countries including Kenya, Malawi, India, and Pakistan for epidemiological surveillance.

- **Open Data Kit(ODK):** This is an open source platform developed at the University of Washington. It can generate data collection forms for mobile phones using the Android operating system. The forms are designed using Excel and a data collection tool is generated. The data

[5] http://www.medes.fr/home_en/telemedicine/tele_epidemiology/epidefender.html

is retrieved and saved (using the Internet or SMS if the Internet is not available) on a remote server and can be analyzed [25]. ODK has been deployed in several contexts including paediatric patient management in Tanzania, war crimes in the Central African Republic, school attendance in India.

Although the previous solutions permit us to generate data collection applications used for epidemiological surveillance, in chapter 2, we have shown that the epidemiological surveillance systems are composed of more functionalities than the data collection functionalities.

3.2.3.3 The use of MDA and agile in healthcare domain

In the previous paragraphs, we have presented the use of the MDA approach to model and generate health information systems. However, healthcare models, like all models, may never be complete. In fact, the development of a healthcare information system is a continuous process, since health problems cannot be fixed and be addressed at once. To address them, the tools must be up-to-date. Then, after initial development, the healthcare models must undergo revision and harmonization cycles as effort continues to improve and adjust them. To do this, the MDA approach is combined with the agile approach [112].

A few authors have proposed a combination of the Agile methodology with the MDA to develop health information systems: Atanasovski et al.[8] shows that agile methodology permitted fast deployment and adoption of an integrated health information system composed of EHR (Electronic Health Record), Electronic Prescriptions, Electronic Referrals, Hospital Stay and Surgeries Information System, Laboratory Information System and a Radiology Information System in which all medical and health related information about patients, health workers, facilities, documents, and procedures are stored and processed. However, problems can arise in the future if the current implementation technology becomes outdated or obsolete. Then, they combined the Agile methodologies and the MDA to assure its extensibility, soundness, interoperability and standardization; by combining the agile software development and MDA, Blobel and Pharow [22] have developed a health telematics platform which combines card enabled communication with network based interoperability.

3.3 MDA approach for epidemiological surveillance systems

Addressing the epidemiological surveillance systems problems and challenges requires a fast, flexible, collaborative methodology to develop the health information. The section 3.1 shows that agile is more flexible, collaborative, and user focus; the section 3.2.2 shows that the MDA approach used models to develop solutions that are interoperable, portable, reusable and that evolve easily; and the section 3.2.3 presents some tools based on MDA and used to enhance the development of health information systems. In this section, we present a methodology (resume by the figure 11) based on MDA and Scrum methodology to develop epidemiological surveillance systems. This methodology is composed of the Pre-development step, the Development step and the Post-development step.

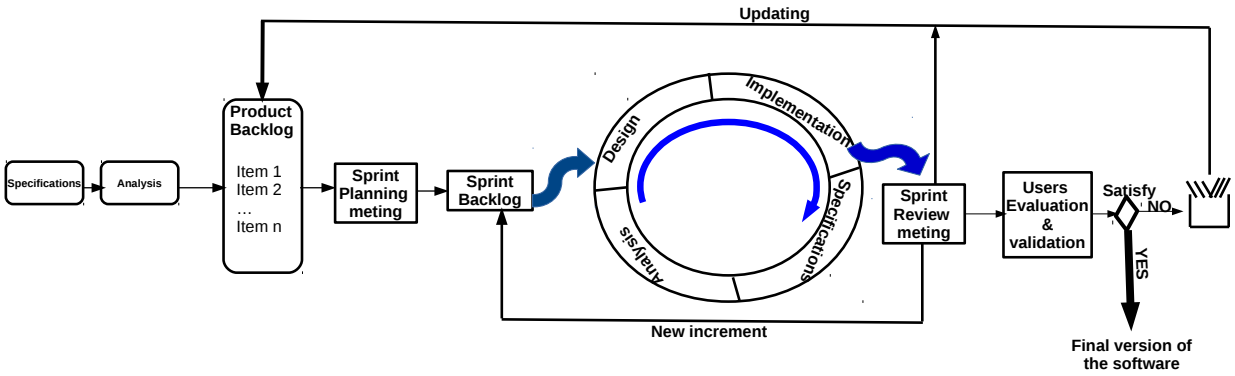


Figure 11: The combination of MDA and Agile for epidemiological surveillance systems development

3.3.1 The Pre-development step

During the Pre-development step, IT experts work closely with domain experts in order to make system specifications and analysis. The specification and analysis are used to define the Product Backlog and the first Spring Backlog. In addition, they contain sufficient information to develop the first working version of the software.

3.3.1.1 System specifications

The system specifications or requirements engineering is the identification of the features and behaviour of the system to develop. It is based on the combination of problems and opportunities that provides the motivations for a new system; describes the reasons why the customer is looking to build the system; and includes a variety of elements that attempts to define the intended functionalities required by the customer to satisfy their users. To identify these functionalities, the IT experts will use all the sources of information such as books, existing tools used by users; will work closely with the domain experts which in our case are epidemiologists, doctors, nurses, medical technicians, hospital administrators, etc. System specifications can be grouped in functional and non-functional specifications:

- **Functional requirements:** They define the services that the system will provide to the users. To define system requirements, what the system must accomplish (calculations, control, data manipulation and processing) must be defined. An example of functional specification in epidemiological surveillance is: "the system must permit us to save patients data".
- **Non-functional requirements:** These are constraints imposed by the customer and the as-

sumptions made by the requirements on the system design or implementation. Non-functional requirements involve performance requirements, security, reliability, availability, serviceability, scalability, maintainability, interoperability, reusability, portability, adaptability, licensing, etc.

At the end of the system specification, a document containing a structured collection of information that embodies the requirements of the system developed is produced and validated by all parties.

3.3.1.2 System analysis

The system analysis is the study of the system, its different modules in order to identify its objectives, its boundaries and make the requirements clear, complete, consistent and unambiguous. It is critical to the success or failure of the software developed. Its main goal is to ensure that the requirements are complete and consistent and all parties agree on the system to develop. The system analysis must be defined to a level of detail sufficient for system design, development and testing. We recommend the use of the UML language to build the system analysis artefacts.

During the analysis of the system, the CIMs are constructed. They include all the elements such as actors, use cases, use cases description, state-transition diagrams, collaboration diagrams, etc. that permit to well understand the system and determine the system boundaries.

- **Actors (or actors models):** The actor specifies a role played by an external entity which can be a human or a system with the system. The actor interacts with the system and during this interaction, it executes one or many use cases;
- **Use cases (use case models):** Use cases capture requirements, describe functionalities provided by the system. A use case is a unit of useful functionality performed by actors to which the use case applies in collaboration with other actor(s) and which yields an observable result.
- **Use cases description:** Each use case description is composed of a set of scenarios that convey how the system should interact with actors to achieve a specific business goal. To make use case description, domain expert language is preferable to technical language. For instance, the use cases models artefacts used to describe the use cases are: the use case diagram, which is a graphical representation showing which actors can operate with the use case; the use case text description, which describes step-by-step interactions and dialogue between the actors and the system.

  The main elements of use case description are: (1) the title which communicates the goal of the use case; (2) the actors who interact with the system by executing a use case; (3) preconditions are the constraints necessary to be verify before the use case can start; (4) the postconditions are the conditions to be verified when the use case execution is finished; (5) the use case description (or story) is the step-by-step action and interaction between the actor and the system; (6) the use case ID which is a unique identifier of the use case; (7)

the resume is a brief description of what the use case does; (8) created by which gives the author(s) of the use case. In some cases, the authors of the use case can be the domain experts and the IT experts; (9) the priority gives the level of importance of the use case given other use cases. The priority is particularly important since we are using an approach in which the main functionalities must be developed and put in service before the development continues.

In a broader sense, the analysis takes as input the system specification and construct the CIMs. During the analysis, the customer must be available to provide clarifications on the requirements and validate the identified boundary of the system.

3.3.1.3 Product Backlog definition and the First Sprint Backlog definition

The system specification and analysis permitted us to identify, describe and prioritized the system use cases. These use cases are organized in a group (Product Backlog List) in order to define the different modules of the software. The effort estimated to develop every group is estimated and these groups are prioritized according to the degree of importance of the use cases which constitute the group.

After the Product Backlog is defined, the Sprint Backlog, containing the group of requirements with higher priority is defined as the first Sprint Backlog.

3.3.2 The Development step

The first stage of the development is the Sprint Planning Meeting which is held to decide upon the goal, the requirements, the time expected for each Sprint. Thereafter, the development follows an iterative and incremental (each increment is called a Sprint) process. An increment takes as input a Sprint Backlog and conducts it in an iterative manner until the Backlog Items contained in the Sprint Backlog is finished. During the iterative development, just small changes must be considered. Important changes must be introduced in the Product Backlog, prioritized and integrated in a Sprint Backlog for further development. In a broader sense, the development step is composed of two main phases: the first one is the "first Sprint" and the second one is composed of one or many Sprint-"the next Sprints".

3.3.2.1 The first Sprint

The first Sprint contains the priority specifications held in the first Sprint Backlog that can be used to develop the first version of the software. It is executed in several iterations in the following order: system conception $\rightarrow$ system implementation $\rightarrow$ system testing $\rightarrow$ system specifications $\rightarrow$ system analysis (see the figure 11). During these iterations, the Scrum Meetings (daily and weekly) will permit us identify the impediments of the project and remove; will permit the team to progress in the comprehension of the customer needs and the developers to be confident in the

development. At its end, a Sprint Review Meeting is held in order to evaluate, validate this first version and decide which direction to take.

3.3.2.2 System design

Unlike the requirement specifications and analysis which focus on what the system has to do, system design focuses on how to accomplish the objective of the system. During the system conception, the specifications and the analysis are used as the basis to identify the various system components and how they communicate, how external systems will communicate with the system, to make choices on software architecture, MDA tools, deployment environment, to build different models and to transform these models into executable source code. At this stage, the software specification starts to become a reality. System conception involve the following elements:

- **Architecture design:** System architecture is the blueprint of the system permitting to convert software specifications and analysis such as flexibility, scalability, feasibility, reusability, and security into a structured solution that meets the technical and the business expectations. It provides a solution that the technical team can create and design for the entire application. It describes the main components of the system and how they interact with each other and with external systems in order to furnish services to users. Further, it involves a set of significant decisions about the technologies that will be used to develop the system. During the software design, a global architecture can be designed, follow by the description of the architecture of each components;
- **Tools choosing:** The system architecture combined with the non-functional specifications will guide the choice of the design, development and deployment tools. Given that we are using a MDA approach, a MDA tool must be chosen for source code generation. To do so, a review of existing tools are made and the tools obtained are compared to select the most adapted given the CIM. At this stage, three situations may occur: (1) there exists a tool that can be used to make models and generate the application source code. Then, it is used; (2) there is a tool that can be used to model and generate a part of the source code. Whether the tool is used to model and generate the part of the source code and the rest of the functionalities is developed and integrated in the source code, or the MDA tool is completed with new functionalities and the tool obtained is used to model and generate the entire source code; (3) there are no tools that correspond to users' needs. Then, the MDA tool is designed, developed and used. Blagoj et al. [8] suggested that to develop/update a MDA tool, one must be sure that this tool will be used in several projects. If not, the time spent to develop the MDA tool and the model will be bigger than working only with code and documentation.
- **User Interfaces Design:** User interfaces design organizes the different front-end elements that will be used to manipulate and control the software. With the MDA approach, the template composed of UI design and components that will be used to generate the user interface are embedded in the MDA tool. Then, the generated user interfaces for all the software are inflexible.
- **Platform Independent Model design:** The CIM obtained during the design phase is combined with system specifications to construct the PIM whose main artefact is the class dia-

gram. Class diagram describes the structure of a domain by identifying the domain classes (e.g., patient), their attributes (e.g., age, sex), their operations (e.g., calculate a body mass index) and the relationships amongst classes (e.g., the relation between a patient and his appointments at the hospital) [9]. To build the class diagram, we advise to make a data dictionary containing in his right size the entities, in his middle size the properties of these entities and in the left size their descriptions. This data dictionary must be validated by the customer after its construction.

Unlike the application development using general purpose language, in our approach, the customer involves in the conception in order to validate all the CIM, the PIM before the development of the solution.

3.3.2.3 System implementation and testing

The implementation involves the construction of the PSM using the PIM and the generation of the executable source code. Depending on the MDA tool, this construction can be automatic. Further, the MDA tool uses the PSM to generate executable source code.

The first version of the software is tested after its generation to ensure that it provides the expected features to users. After each iteration, two types of testing are made:

- **Developers testing:** The generation of the first version of the software is checked by the developers to ensure that it fulfills the requirements without any bugs;
- **User testing:** After the developers testing, real users involved in the development will test the product using a real environment scenario in order to quickly identify any areas of improvement and issues. For example, health workers will enter the real data in the system. The feedback of these users will be used to refine the specifications or enrich the Product Backlog.

3.3.2.4 Refining system specifications and analysis

After the definition of the first version of requirements and analysis, they will fall under changes. In the first Sprint, the Backlog Items contained in the Sprint Backlog are developed iteratively. After each iteration, regular checks are made with stakeholders to ensure that the right functionalities are being developed. The feedback is used to identify any improvement, any issues and then refine the specifications and the analysis before continue to iterate.

3.3.2.5 System validation

The results of the first iteration is the baseline version of the software. After its development, a Sprint Review Meeting takes place. This meeting is essentially the demo of the application devel-

oped. During the sprint review, the project is assessed against the sprint goal determined during the sprint planning meeting. Ideally, the team has completed each Product Backlog Item brought into the Sprint, but it's more important that they achieve the overall goal of the Sprint.

After the Sprint Review Meeting, the end users in a real environment will involve the validation of the base version. This base version will be used before the development of the whole application. Users' feedback will be used to improve the specifications. These feedback can be integrated in another Sprint or used to define a new Sprint if it is important.

3.3.2.6 The next Sprints

After the system validation, the Product Backlog List is reviewed by the Scrum Team and updated if necessary. Additional to system functionalities, Backlog Items will also include, bug fixes, defects, requested enhancements and technology upgrades.

After all the Sprints are completed, the mature version of the software validated by the end users is produced.

3.3.3 The Post-development step

The post-development step consists on two activities:

- **Deploying the system and full users training:** The final version of the system is deployed and the all the users are trained in order to achieve the users needs;
- **Training the users in the use of MDA tool:** Users involved in the development process will be trained on the use of the MDA tool to update the model and generate new versions of their system themselves. This will permit us to reflect changing needs and understanding of the business. In fact, our goal is not to leave the software development to the domain expert, but to facilitate his/her task of the updating of the system to make it more close to his specifications without necessarily needing the help of an IT-expert.

3.4 The EPICAM platform for tuberculosis surveillance

The EPICAM (Epidemiology in Cameroon) project aims to improve Cameroon's health system through a pilot project consisting to set up an electronic tuberculosis surveillance network. It contributed to Cameroon's 2009-2015 strategic plan for strengthening the Health Information System by providing an epidemiological surveillance platform to enhance epidemiological surveillance. This section presents how the EPICAM platform was developed and used to improve the tuberculosis surveillance in Cameroon. Then in the section 3.4.1, we will present the development of the platform and in the section 3.4.2, we will present the main results obtained after its use by the National Tuberculosis Control Program in Cameroon.

3.4.1 EPICAM platform development

In this section, we present how we have applied the methodology presented in section 3.3 to develop the EPICAM platform for epidemiological surveillance of TB. Then, we will present the Pre-development, the Development and the Post-development of this platform.

3.4.1.1 The Pre-Development of the EPICAM platform

The Pre-development step was the first step of the development of the EPICAM platform. It involves the system specification, the system analysis and the Product Backlog definition.

System specifications. To efficiently fight against tuberculosis, the National Tuberculosis Control Program has established a surveillance system through which it collects and shares data with the health professionals in health centers, health districts, health regions, the ministry of health, the general population and partners (Global fund, WHO, Center Pasteur of Cameroon, etc.) However, this system is manually managed causing problems (we presented some in chapter 3). To overcome these limits, the NTCP expresses the need of an electronic system. To develop this system, existing systems were studied on several levels of the health system. First of all, the tools generally used for data collection (paper forms, patients cards, register), and data analysis (excel documents) were collected at the central, regional and peripheral levels. Thereafter, a survey on the main problems of existing systems, the specific needs of end users and their knowledge on the use of computers were made at all levels of the organization. This survey, involving 12 health professionals was done in three health centers (two in Yaounde and one out of Yaounde), one region and at the NTCP. This survey permitted to identified additional problems that the health workers have in the field: difficulties of the collection and the transmission of data using paper for the collection and the road for the transmission; difficulties to share patient information with colleagues after a transfer/referral; they are generally faced with the problem of stock-outs of drugs; difficulties to guide the patients to the health center near to his home; difficulties to recall patients who have not come to their appointments; difficulties to update data collection form which generally change each years; difficulties to make SMS awareness; difficulties in using the computer system; power/Internet outage. At the end of this survey, the users confirmed that the electronic system will enhance their work as long as they are trained on the use of computers and the problem of power/Internet outage are considered.

The study of existing systems permitted to identify functional and non functional specifications.

Functional specifications. In the broader sense, the electronic system must permit to:

- Collect, verify, synthesize data and make reports (weekly, monthly, quarterly, yearly, etc.) accessible at the district level, the regional level, and the central level;
- Follow patients and make SMS recall for those who did not come to an appointment;
- Manage anti-tuberculosis drugs so as to prevent stock-outs;
- Locate the closest hospitals with respect to the patient's home;

- Sensitize the population by SMS.

Non-functional specifications. Given the context, some additional constraints must be considered when developing the solution:

- The discussion was about the use of mobile phones/computers for data registration. The health personnel return that the mobile phones may be more difficult to use than the computer given the volume of data they manage each day;
- Support the annual update of the data collection and reports tools: According to their needs in data, each year, the NTCP update the tools used for data collection by adding or removing some fields;
- Remote deployment of updates: After the supports update, all the users must have access to the new version when they connect to the server;
- Permitting to user to work offline and update their data when they connect to the Internet: The system must permit to the users to work offline (when there is power/Internet outage) and send the data to the server when all the conditions are fulfilled;
- The tool must be Open Source: In fact, the system will be used to develop other epidemiological surveillance systems for other diseases, permitting to these systems to interoperate;
- The deployment platform must be robust: The operating system to use must permitted to avoid viruses;
- The software must be delivered with relevant documentation: The documentation must permitted to the health personnel to be autonomous when using/updating the system;

System analysis. The analysis consisted of determining the actors, the use cases of the system and to describe most important use cases.

The main actors of the system. All the staff of the NTCP and their partners interact with the system. They are organised in different roles:

- At the central level, the actors identified were the administrator, responsible for configuring the system and managing the users; the program manager, responsible for coordinating the entire system; the person in charge of the follow-up evaluation and epidemiological surveillance who compile the data from the field and build the statistics needed; the person in charge of the control of data quality; the responsible for drugs management; the person in charge of training; the laboratory data manager in charge of the production of laboratory statistics; the communication manager in charge of the organisation of the communication in the system and population awareness.
- At the regional level, the regional delegate and her/his staff monitors the districts and the health centers in the region. He/she uses the statistics for decision making.

- At the district level, the health district and her/his staff monitors the activities in the health centers. He/she uses the statistics in the district for decision making.
- At the level of health center, the nurse or the doctor uses the system to follow patients treatment:
  - The pharmacist manages the stock of drugs and gives drugs to the patients,
  - The laboratory staff records data on the laboratory tests and results,
  - The physician enters patient data and follows her/his treatment.

Use cases. The main use cases identified were:

1. The collection, verification, synthesization of data from hospitals and laboratories;
2. Making reports accessible to the District unit, Regional unit, NTCP and partners;
3. The management of patients treatments at the treatment centers and laboratories. This involves the registration of patient information, examinations, laboratory tests, transfer/referral, the automatic notification of patients by SMS on the availability of laboratory tests results, the displaying of the patients tuberculous who did not come for their laboratory tests results, the displaying of patients who missed their appointments, etc.
4. The management of the NTCP, the health regions, the heath districts and the health centers;
5. The geolocation of the health centers and the laboratories;
6. The management of NTCP staff (arrival, departure, etc.);
7. The management of health personnel training sessions and making the training supports (text/videos) available for further reuse;
8. The management of drugs in order to prevent a stock-out;
9. The sensitization of the population using SMS;
10. The searching of all elements recorded;
11. Exporting data for further analysis.

Product Backlog definition. Once the use cases were identified, we have defined the product backlog in accordance with the NTCP. These were divided into the following items:

- Platform design,
- The development of user management module,
- The development of data management module,
- The development of drug management module,

- The development of the reporting module,
- The development of the mapping module,
- The development of the awareness module.

The first Sprint Review Meeting permitted us to adopt the platform design. The development of the user management module and the data collection module was identified as our first Sprint.

3.4.1.2 The Development of the EPICAM platform

The Pre-development permitted us to identify the user's needs and its analysis, the product backlog and different Springs Backlogs. The development consisted in five Sprints.

Sprint 1: the development of a prototype. The first Sprint consisted in the design of the platform and the development and the testing of the first version of the prototype consisting of the user management module and the data collection module.

System design. Given the system specification and analysis, the architecture, the user interfaces, the Platform-Independent model were made.

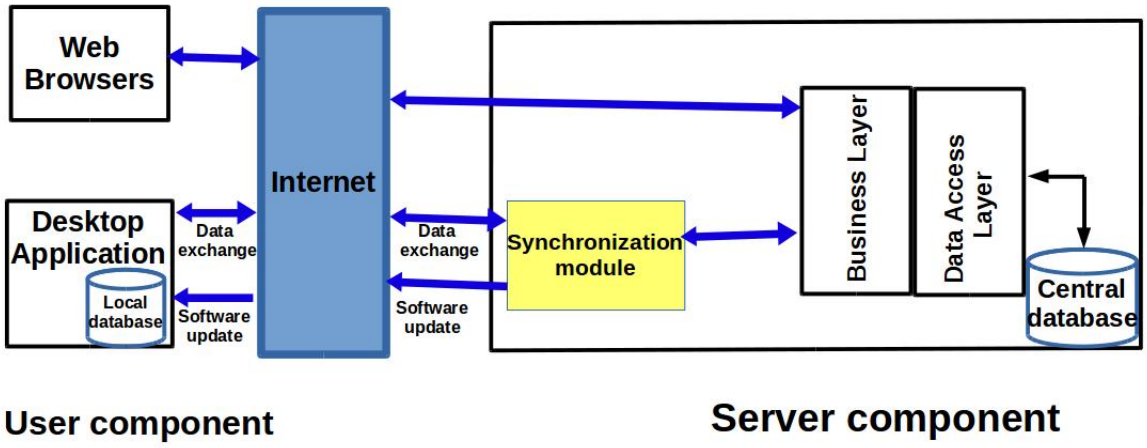


Figure 12: System architecture

1. **Architecture design.** The architecture presented by the figure 12 is the architecture of the system which suits an environment like Cameroon where access to the Internet is not always available. This architecture is composed of a user component and a server component. The user component may be a web browser connected to the server via the Internet. In this case, the client shares data with the server in a synchronized manner. In the case of the offline use of the software, the user component will be a desktop/mobile application with a local database. Data is saved in a local database and synchronized with the server database. This

system permits users to continue their work even if there is no light or Internet and update the server when all the conditions are fulfilled. The synchronization module works as a mediator, which permits the server get new data from the desktop application local database and the desktop application get new updates from the server.

2. **Choice of MDA tool:** The survey made in section 3.2.2 presents Imogene as a platform based on MDA for the development of data collection applications. It permits users to use a graphical editor to model the application and generate the executable source code. With the application generated, it is possible to save data in a local database and update to the server periodically. As presented in section 3.2.2, other MDA tools used in the medical domain such as ODK and Magpi can be used to generate data collection forms, but only for mobiles phones and not for computers. Then, we adopted Imogene as our MDA tool. A deep analysis of Imogene shows that it permits to model and generate only two modules of the EPICAM software: the data collection module and the user management module.

3. **User Interfaces Design:** During the user interface design, a deep discussion with the personnel of the NTCP permitted us to decide how the elements on the user UI will be organized.

4. **Platform Independent Model design:** To construct the PIM, we have used the specification and the analysis to build a data dictionary. This data dictionary contains the entities, their attributes, their relationships and a clear definition of each entity given by domain experts. The definition given by the experts will permit us to present these entities on the user interfaces. From the data dictionary, a class diagram (PIM) was built (see figure 13). This class diagram was used to represent the entities, their properties and relations in a graphical manner.

System implementation.

Imogene was used to build a Platform Specific Model (figure 14), which was used to generate the first prototype. This first prototype consists of:

- **An administration application:** Used to manage health workers information, their roles, and their access rights on data;
- **A Web application:** Used for the collection of patient data, managing anti-TB drugs and follow-up appointments of patients. This application works in a synchronized manner i.e. when the health personnel using the application is connected to Internet;
- **Desktop application:** The desktop application has the same functionalities as the Web application, but is used in an asynchronous mode i.e. when Internet connection is not available, the system uses the local database as storage system and the local database is synchronized with the server when the Internet becomes available;
- **Synchronization application:** Used to synchronize the client with the server (updating data or updating client applications).

The figures 15 and 16 present the entry point of the EPICAM platform and the patient registration form.

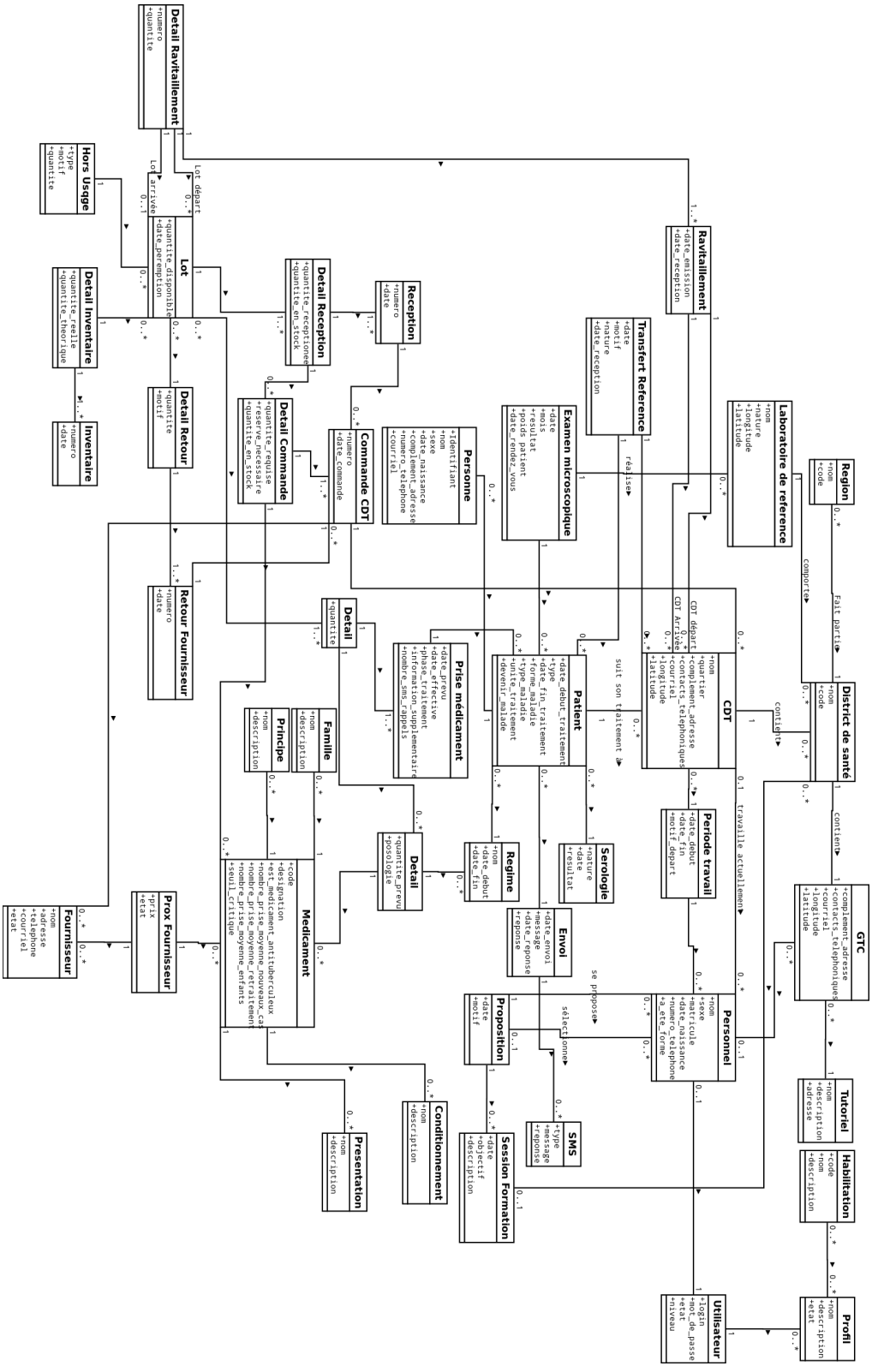


Figure 13: The Platform Independent Model of EPICAM

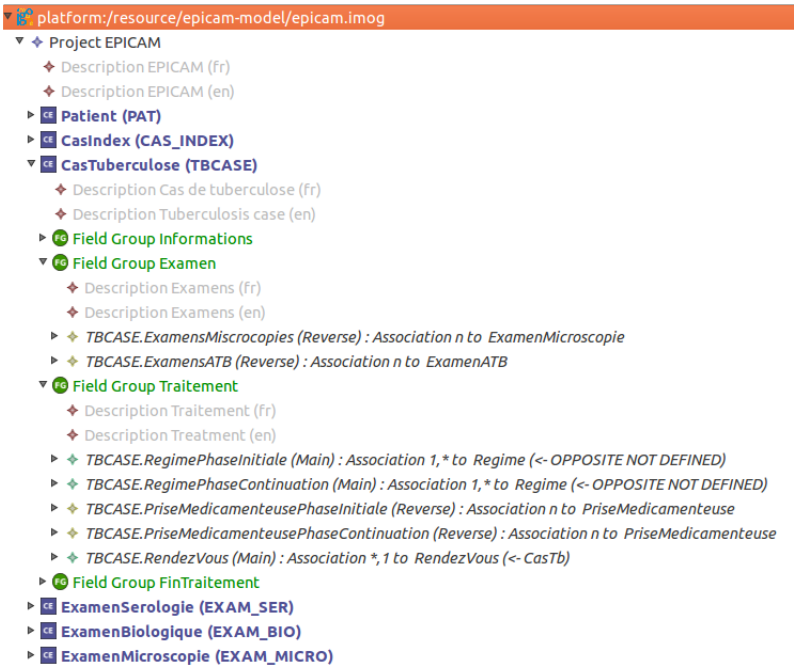


Figure 14: The Platform Specific Model of EPICAM constructed using the PIM

Figure 15: The entry point of the EPICAM platform

Testing and updating the Product Backlog. The Scrum Meetings (some were done with the health personnel at the NTCP) permitted the evaluation of the user management and the data collection modules. To do this, real data was entered in the system, the missing field and some fields names errors were noted. It was also noted that some users from the central level can access personal patient information. The responsible at the NTCP tell us that only the health workers in charge of the patient must have access to personal information of patients. Users' remarks permitted us to integrate the specification, analysis and design of user rights to the data in the Product Backlog.

Second and next iterations. After the development of the data collection module, many iterations were conducted. These iterations permitted us to integrate all the information that the data collection forms must have and correct some typology errors. It consists of the design, the development, and the testing. The design consisted of the PIM update, the development to the PSM update

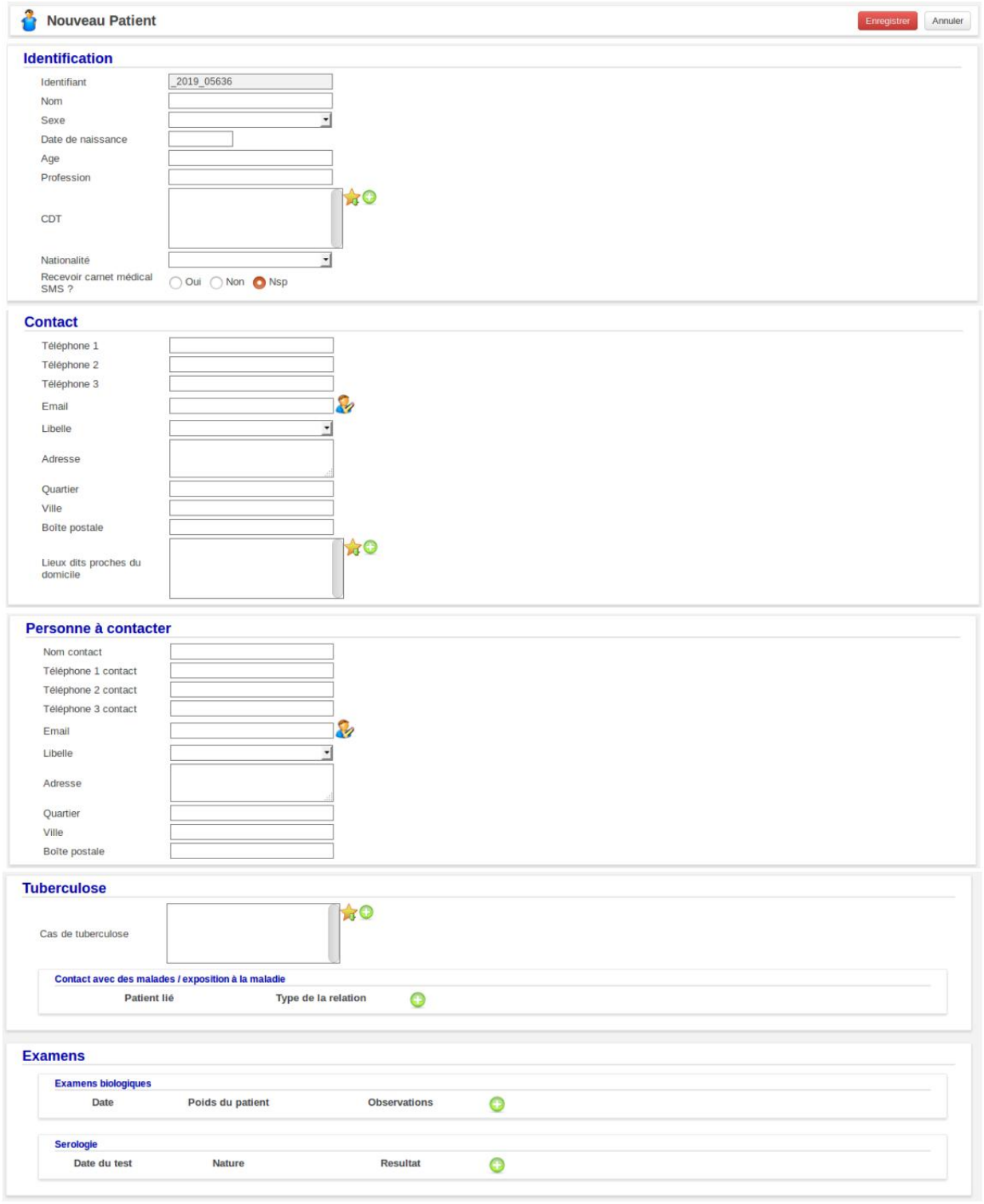


Figure 16: The patient registration form

and a new version of the application were generated.

Sprint 2: The development and integration of user management module. Given that med-

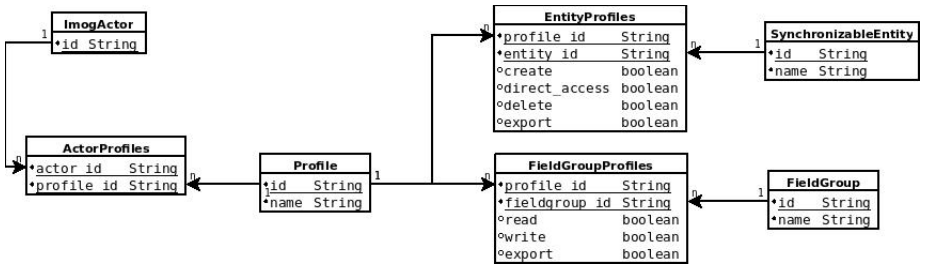


Figure 17: The class diagram of users management in the EPICAM platform

ical data is very sensitive, in the second Sprint, we prioritized the development of the user management module. It was noted during the specifications and analysis that users have different roles depending on their service. In some hospitals where there is not enough staff, a user may have multiple roles. For example, a user may be the person who consults the patient and gives it medication. During the design, we constructed the class diagram modelling the user management (figure 17). Given this class diagram, the user management module was developed (with many iterations) using Java language, Google Web Toolkit, Spring security framework, maven, xtext, xpand, OAW. After the development, this module was integrated in the source code generator.

With the user management module, when the administrator creates a user, he/her allows him/her to read/modify the data. When a user login, he/she can access only the data he/she is entitled to. The figure 17 presents a screenshot of the interface permitting the attribution of rights to users.

The second Sprint permitted us to obtain a first version of the system that the end users in the field can use in a real environment. This version was deployed in six hospitals in Douala and Yaounde (which are the largest cities in Cameroon). User feedback allowed us to complete the specifications (e.g., integrate clinical radiology exams) for the next Sprints.

Sprint 3: Integration of end users feedback. After the deployment in the testing centers, the users tell us that there are two scenarios to be considered in the hospital during TB patients treatment. The first scenario is the scenario in which the patient is registered when the medical exams have confirmed that it is a TB patient. Then, the physician can start the registration of the TB patient. In the other cases, the patients follow the patient circuit in the hospital. Then, in the system, we have added other forms permitting the registration of the patients before their medical exams. The application updates have been validated by the users in the hospitals.

Sprint 4: The development of the reporting and the mapping modules. As the previous Sprints, the reporting and the mapping modules were developed in many iterations. The reporting module was developed using Business Intelligence and Reporting Tool (BIRT). The mapping module was developed using Open Street Map (OSM). During these iterations, the users have validated the reports that they are expected to the platform, and have tested the use of the map to geolocate the hospital near to a given patient. Note that given the time limits to produce the platform, we have integrated the reporting and the mapping source code in hard in the source code generator.

Sprint 5: the development of the SMS module The SMS module was specified by the users at the central level and developed using the Spring Rest API. In fact, to send SMS using the

platform, the system must use a service provided by an SMS provider. The SMS module were tested, validated and integrated in the source code generator.

3.4.1.3 Summary

During the EPICAM project, we have developed a new MDA-based platform based on Imogene for the generation of epidemiological surveillance platforms. In addition to the data collection module generated by Imogene, this tool integrates:

1. **The reporting module:** It is the module for epidemiological report generation (in pdf format) using BIRT;
2. **The geolocalization module:** Using OpenLayer, a library for creating interactive map on the Web, this module permit us geolocate hospitals near to the patient' home,
3. **SMS module:** For the sensitization and patient recall;
4. **Managing drugs module:** The goal of this module is to manage TB drugs in order to prevent stockouts;

EPICAM was developed and released under the LGPL license[6]. Its goal being to provide full software support for teams desiring to implement epidemiological surveillance systems in their environment. EPICAM was used to generate the last version of the epidemiological surveillance of TB in Cameroon.

After the development and the deployment of the TB surveillance platform, the user's feedback (e.g. update a field) permits us to complete, generate and deploy new versions of the applications. For users, everything is transparent because the synchronization module updates the remote application automatically.

3.4.2 Main results obtained during EPICAM use

The system was deployed in twenty five pilot hospitals. The figure 18 presents the distribution of the deployment in the country where the patients are treated for tuberculosis. We supervised the system in the course of the year 2015. Around 3900 patients which represents 15.6% of the annual number of TB cases in Cameroon were registered and followed.

The figures 19, presents a resume of TB register in Cameroon and the figures 20, 21 present the screening and the treatment reports in the first trimester of the year 2015.

[6] `https://github.com/\\ummiscolirima/EPICAM`

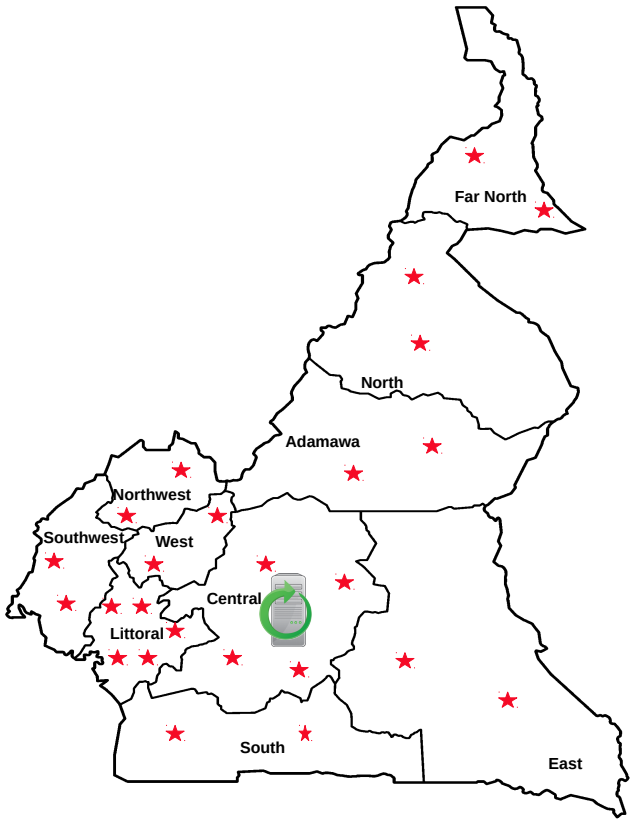


Figure 18: Distribution of the deployment of the EPICAM platform

3.4.2.1 Users' feedback

According to users' feedback, the system allowed:

- At the central and regional level, a better visibility of the health problems and an early detection of issues. For example, by using the system, an epidemiologist found that low-weight patients died the most.
- At the peripheral level, the system permits health workers to improve the patients' management and follow-up. A testimony of a health worker: "one day, a patient came for consultation and during his registration in the system, I found that he had stopped his treatment before the 6 months recommended and the disease resumed. An interview with him revealed that he had started drinking alcohol. The examination revealed that the patient was now a TB-MDR patient."

  The searching functionality permits health workers to access easily to patient information. In addition, losing the patient follow-up card will no longer be a problem because all the patients' information is saved in the database. However, during the transfer/referral of a patient in a hospital where the system is not deployed, it is not easy for the health worker to continue the follow-up if the patient does not have his follow-up card.

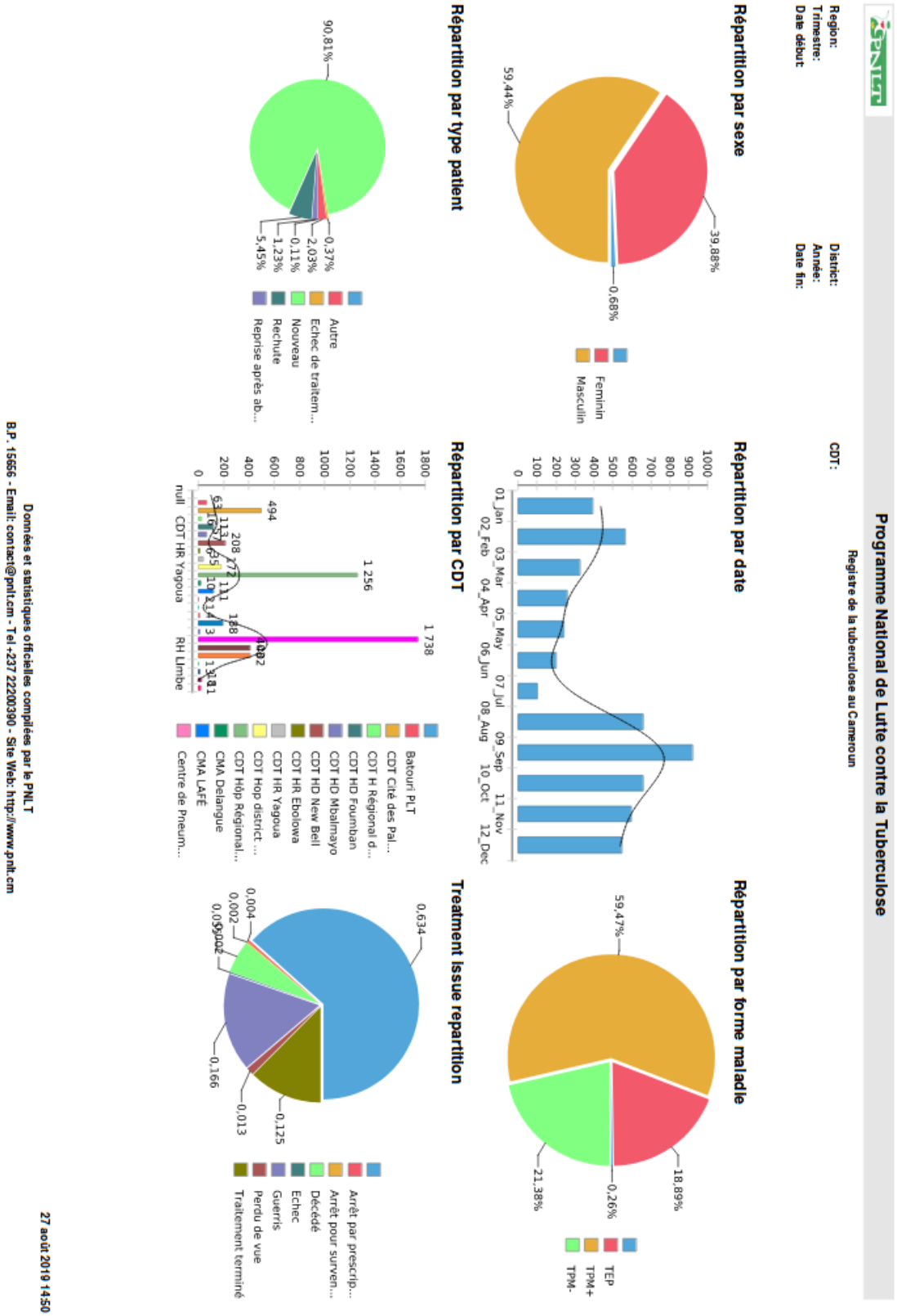


Figure 19: A summary of TB situation in the pilot centers in 2015

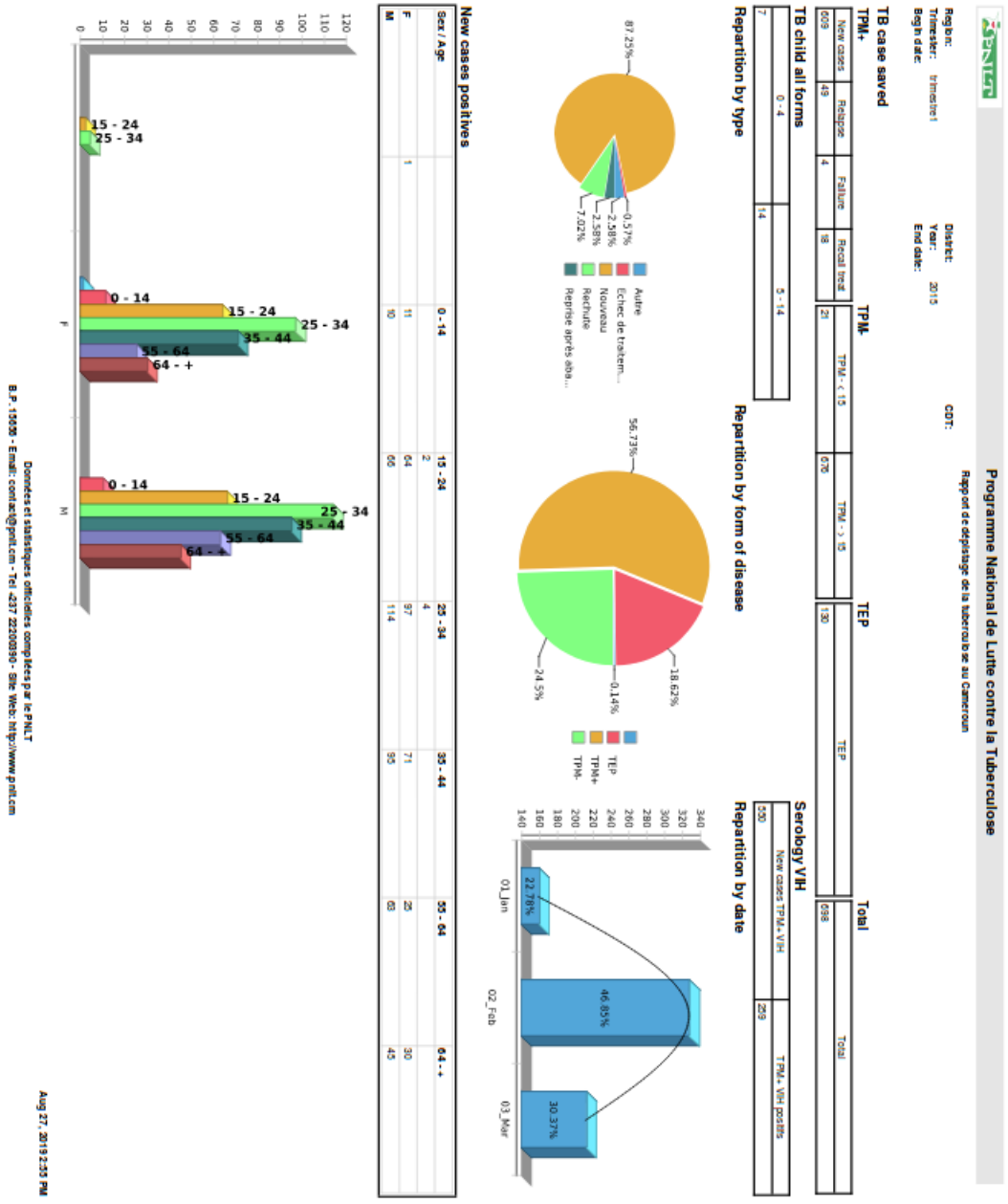


Figure 20: Screening report in the first trimester of 2015

Given the success of this pilot phase, the NTCP have adopted the softwares as its electronic epidemiological surveillance softwares and extended it in twenty new health centers during the years 2016 and 2017.

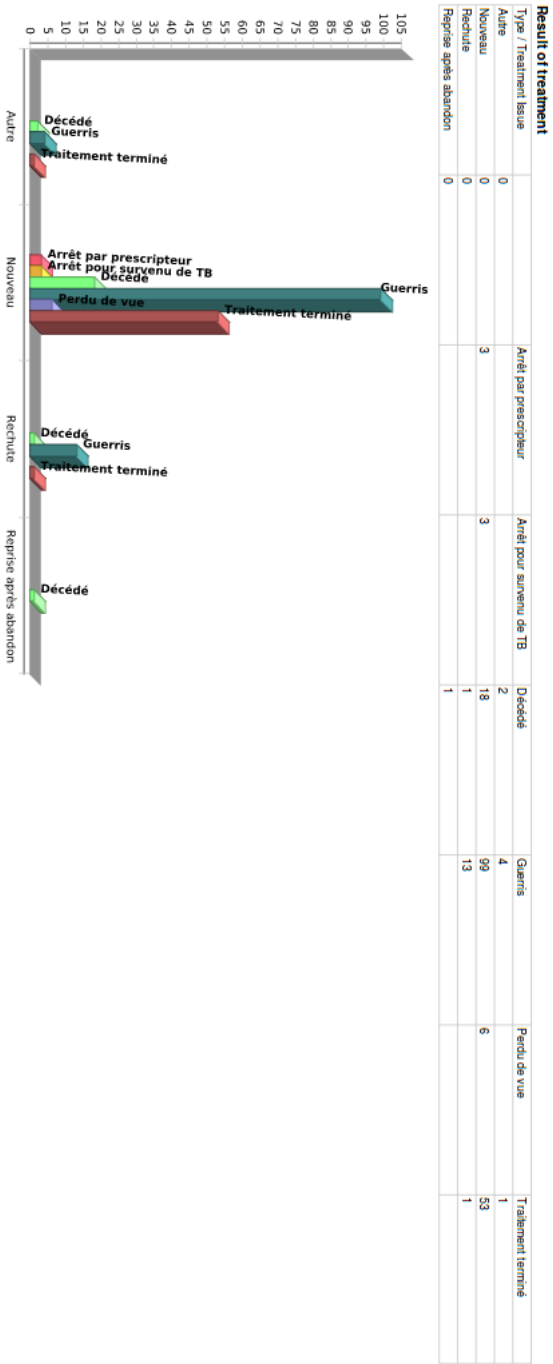


Figure 21: Treatment report in the first trimester of 2015

The surveillance system was deployed at the server side using the PostgresQL database and the client side using the H2 database. Searching information in these databases consists of making a SQL query. However, reasoning cannot be made using this kind of data structure to deduce new information (e.g., automatic detection of TB-MDR patients). In chapter 6, we propose a solution based on a domain ontology for the annotation of epidemiological data.

3.5 Conclusion

In this chapter, we have seen that the software developers of today face software development issues such as the demand of richer functionalities in shorter timescales, the rapid evolution of systems deployed. To solve these problems, software development methods, methodologies and tools are proposed. To support early and quick production of working code, agile methodologies propose to adapt the development to the changes. To this end, the software is developed in an iterative and incremental manner. On the other hand, the Model Driven Architecture approach improves the productivity and maintainability of software by allowing to model the software and automatically generate the executable source code. To this end, MDA tools are developed to support the overall modelling and transformation process. For instance, Imogene is a MDA tool used for the modeling and the generation of data collection software. Addressing the epidemiological surveillance problems and challenges requires a fast, flexible, collaborative methodology to develop epidemiological surveillance systems. Then, we proposed in this chapter an approach based on the combination of agile methodology (particularly Scrum) with MDA, and a tool for modeling and generating epidemiological surveillance systems. The tool, named EPICAM is an extension of the Imogene platform. The methodology and the tool proposed were used to develop a platform for epidemiological surveillance of tuberculosis in Cameroon. During the periods of 2014-2015, the system was deployed in twenty five pilot hospitals. It allows to follow around 3900 patients which represents 15.6% of the annual number of TB cases in Cameroon. Given the success of this pilot phase, the NTCP have adopted the software as its electronic epidemiological surveillance software and extended it in twenty new health centers during the years 2016 and 2017.

Note that the surveillance system developed for NTCP was deployed at the server side using the PostgresQL database and the client side using the H2 database. Searching information in these databases consists of making a SQL query. However, reasoning to deduce new information cannot be made with this kind of data structure. In the next chapters, we propose a solution based on a domain ontology.

4 Ontology engineering

Ontologies are knowledge representation languages borrowed by computer science from philosophy and commonly used in the Semantic Web. In philosophy, ontologies help to model reality so as to distinguish what is real and its categories from what is not real [58]. In computer science and information science, ontologies help to model a domain/problem. The main goal of this chapter is to present how ontologies are constructed. It is divided in three main sections: section 4.1 presents ontologies, section 4.2 presents how ontologies are build, section 4.3 presents ontology learning and section 4.4 presents related works on ontology learning from source code.

4.1 Ontologies

The goal of the first section of this chapter is to define the notion of ontology and how it is modelled. Section 4.1.1 will present the notion of knowledge, section 4.1.2 will present the ontologies and section 4.1.3 will present how ontologies are modelled.

4.1.1 The notion of knowledge

Knowledge are facts, information and skills acquired through experience or education for understanding of a subject area. In this area, it describes concepts and facts, relations among them and mechanisms to combine them in order to solve problems in a domain [54]. To be useful, knowledge must be acquired from domain experts/resources and represented by a formal model such as semantic networks, system architecture, Frames, rules, ontologies, and logic.

The theory behind knowledge representation is cognitive science which studies human thinking in terms of representational structures in the mind and computational procedures that operate in those structures. It is assumed that the human mind has mental representations analogous to computer data structures and that the computational procedures of the mind are similar to computational algorithms. In this field, the different mental representations of the human mind are cited as

follows: logical propositions, rules, concepts, images, and analogies. These constitute the basis of the different knowledge representation techniques of the human knowledge such as rules, frames, and logic. Every representation provides some guidance about how knowledge can be organized for efficient computation (e.g. frames are suitable for taxonomic reasoning) [54]. The domain of cognitive science distinguishes the different types of knowledge that humans commonly use [54]:

- **Procedural knowledge:** Describes how things can be done. Example of such knowledge includes rules, problem-solving strategies, agendas, and procedure manuals;
- **Declarative knowledge:** It is about what is known about a topic or a problem. For example, facts that are either true or false;
- **Metaknowledge:** Describes the knowledge behind knowledge;
- **Heuristic knowledge:** Express as a simple heuristic which help to guide the problem solving process and moving through the solution space;
- **Structural knowledge:** Describes the relationship between the different pieces of knowledge from other categories;
- **Inexact and uncertain knowledge:** Described a prior, a posterior, and conditional probabilities of events.
- **Commonsense knowledge:** Denotes a vast amount of human knowledge about the world which cannot be put easily in the form of precise theories.
- **Ontological knowledge:** Describes a category of things; a domain and the terms that people use to talk about them; the relations between categories, and the axioms and constraints in the domain. Its main components are concepts, properties of concepts, axioms and rules. In section 4.1.2, we shall be talking about ontologies.

After the knowledge of a particular domain is gathered, it is organized and stored in a knowledge base. These knowledge can be retrieved when need be. This is called knowledge retrieval. This is done through reasoning to obtain conclusions, inferences, and explanation. In order to develop practical knowledge bases, knowledge engineers have to execute a process consisting of [54]:

- Understanding knowledge properly, transforming it making it suitable for the application of the various knowledge representation formalism;
- Encoding knowledge in a knowledge base using appropriate representation techniques, languages, and tools;
- Verifying and validating knowledge by running the practical intelligent system that relies on it;
- Maintaining knowledge in the course of time.

Knowledge helps to add semantics to the Web. However, in the next section, we shall be talking about a particular type of knowledge that is ontologies.

4.1.2 Ontologies

This section defines the notion of ontology. Then, it presents its main components and typology.

4.1.2.1 Definitions

In literature review, several definitions of ontologies were noted. Gruber defines ontology as an explicit specification of a conceptualisation. Based on Gruber's definition, many other definitions were proposed: Borst defines ontology as a formal specification of a shared conceptualization; Studer merged Gruber and Borst definitions and defines ontology as an explicit, formal specification of a shared conceptualization; Guarino gives a definition using the modelling tool that is logic. According to him, an ontology is a logical theory of a conceptualization [58]. These definitions present the key elements of an ontology:

- **Conceptualization:** Refers to an abstract model of some phenomenon in the world. This model is made up of relevant domain concepts, relations, and how concepts relate to each other.
- **Explicit:** Refers to the fact that the meaning of all concepts and the constraints on their use must be explicitly defined. All concepts must be correctly interpreted by machines.
- **Formal:** Refers to the fact that the ontology should be machine-readable (understandable and interpretable).
- **Shared:** Here, the ontology must capture a consensual knowledge (accepted by a group of domain experts) and must be shared to facilitate communication.

4.1.2.2 Basic ontological components

An ontology is composed of these basic components [58]:

- **Concept**, also called $Class$, represents a category of objects. For instance "$Health\_facility$" is the concept of all health facilities including health centers and clinics;
- **Individual** is an instance of a concept and corresponds to a concrete object. For example, from the concept "$Person$", "Bob" is an individual;
- **Property** is used to describe the characteristics of individuals of a concept. They are composed of DataProperties and ObjectProperties. DataProperties are properties whose values are data types. For instance, "age" of type "$Integer$" can be a property of an instance of the concept "$Person$". ObjectProperties are special attributes whose values are individuals of concepts. For instance, "$examined\_in$" defines a relationship between the concept "$Person$" and the concept "$Health\_facility$" ("A person is examined in a health facility");

- **Class/Property hierarchy** is one of the most important relation used to organize concepts and properties in the ontology. It is used to organize concepts/properties through which inheritance mechanisms can be applied. For instance, "$Patient$" is subClassOf "$Person$" is a hierarchical relation between these two classes. Class/Property taxonomies are generally used to construct the so called lightweight ontologies or taxonomies;

- **Axiom** is used to model statements that are always true. Heavyweight ontologies add axioms and constraints to lightweight ontologies. Axioms and constraints clarify the intended meaning of the terms in the ontology. For example, the assertion "the concepts "Men" and "$Women$" are disjoint" is an axiom;

- **Rule** is a statement in the form $\frac{P_1,...,P_n}{P}$, this means that if the statement P is true, then, the statements $P_1, ..., P_n$ are true. Rules are used for knowledge inference purposes.

4.1.2.3 Types of ontologies

Several authors have classified ontologies [58]. However, in this thesis, we will be working on domain ontologies and application ontologies. Thus, in the following paragraphs, we will present the classification of ontologies according to Guarino [58]. This classification is done according to the level of dependence of the ontology on a particular task:

- **Top-level/Upper-level ontologies:** They are also called cross-domain ontologies. They describe general concepts and provide general notions under which all root terms in existing ontologies should be linked. They use general concepts like time, space, event and can be shared and transferred from one context to another. Top-level ontologies are absolutely independent from a specific domain or from a specific problem. Examples of top-level ontologies are two ontologies build by Guarino and al. [58] One is universals (a universal is a concept like "car") and the other is particular (a particular is an individual like "your car").

- **Domain ontologies:** They provide vocabularies about concepts within a domain and their relationships; about the activities taking place in that domain; and about the theories and elementary principles governing that domain. They are limited to the representation of concepts in a given domain and in some cases, they are a specialization of an upper-level ontology. For example, the term "$City$" in a domain ontology is a specialization of a more generic concept "$Location$" which is a specialization of the term "$SpatialPoint$" that may be defined in an upper-level ontology.

- **Task ontologies:** Task ontologies describe the vocabulary related to a generic task or activity (diagnosing, scheduling) by specializing the terms in the top-level ontologies. They are used to model tasks or processes and how these tasks are related. They provide a systematic vocabulary of the terms used to solve problems associated with tasks that may or may not belong to the same domain. For example, an ontology can be constructed that describes the task of a health professional in a Hospital.

- **Application ontologies:** Application ontologies are ontologies that are application-dependent. They combine domain ontology and task ontology. They contain all the definitions needed

to model the knowledge required for a particular application. These ontologies often extend and specialize the vocabulary of a domain and task ontologies.

Note however that these ontologies can be heavyweight ontologies or lightweight ontologies depending on the conceptualization used. They are called lightweight ontologies when they define only hierarchies of types and/or properties and heavyweight ontologies when they are more expressive, using restrictions, inferences, and class construction.

4.1.3 Knowledge modelling

Knowledge can be modelled using many modelling techniques such as semantic networks, system architecture, Frames, rules, logic [54]. Ontological knowledge particularly can be classified according to the formalism used for their modelling. They can be modelled using rules or software engineering techniques (lightweight ontological knowledge); or logical techniques (heavyweight ontological knowledge) [58, 63]. In this section, we are going to present the modelling of heavyweight ontological knowledge and lightweight ontological knowledge.

4.1.3.1 Modelling heavyweight ontological knowledge

Logic can be defined by $L = (S, \models)$ where S is a set of statements and $\models$ is an entailment relation. It is used to make formal deductions and inferences, study correct and incorrect reasoning and modelled knowledge [113].

Logical languages. A logical language is defined by a syntax and a semantic [58, 63]:

- **The Syntax** is composed of a collection of symbols and rules which are combined as formula;
- **The Semantic** gives meaning (interpretation) to symbols and formulae. With the semantic, one can use facts to make deductions, to reason by building demonstrations (e.g., to demonstrate that a patient has tuberculosis by demonstrating that he/she has been tested positive for Koch's Bacillus (KB)).

Logical deduction helps to derive formulae (provable formulae or theorems) from starting formulae (axioms) or rules (inference rules) [63, 113]. There are two ways to show that a formula is a logical consequence of another formulae: The resolution method and the tableau algorithm. With these methods, it can be demonstrated for example, in the case of epidemiological surveillance that any patient who takes tuberculosis drugs has positive sputum results [113]. Several logical formalisms can be used to model knowledge. In the following, we are going to present the propositional logic, the first order logic and the description logic.

The Propositional Logic (PL). The Propositional logic or the calculation of the propositions defines the rules of deduction which connect the propositions to each other without examining

the contents. A proposition is a fact, a theorem, an utterance that is either true or false. It can be demonstrated or refuted. The formulae are the propositions that can be formed by combining the atomic propositions [113]. For example, the statement "a TB case is a person" can be modelled using a complex formula consisting of two propositions (t, h) and a binary operator: $t \longrightarrow h$ (t, represents a case of tuberculosis and h a person). To model knowledge with Propositional Logic, one must specify the syntax and the semantic [113]:

- The syntax defines allowable facts. Atomic facts consist of a single proposition that can be true or false. Complex statements are constructed from simpler ones using parentheses and logical connectives. There are five connectives generally used: $\neg$ called a negation, $\cap$ called a conjunction, $\cup$ called a disjunction, $\Rightarrow$ called an implication also known as "if-then" statements, $\Longleftrightarrow$ called if and only if. All sentences are constructed from atomic sentences and the five connectives.
- The semantic specifies how to compute the true value of any sentence given a model. It specifies how to compute the truth of atomic sentences and how to compute the truth of sentences formed with each of the five connectives. The rules can be expressed with truth tables that specify the truth values of a complex sentence for each possible assignment of truth values to its components. It is to map all atomic propositions to $\{t, f\}$. If F is a formula and I an interpretation, then, $I(F)$ is a truth value computed from F and I via a truth table.

A knowledge base built using PL can be validated using Resolution or analytic tableaux method by demonstrating that it is satisfiable.

Note that with PL, one can only make statements and assertions about single objects. It is impossible to summarize objects into a set or a class (ontological concepts) and to make statements about a set of things; to make relationship among propositions (data/object properties); to make arguments on a set of objects without explicitly naming them [113]. For example, it is not possible to model the statement "anyone with positive sputum exams is a case of tuberculosis".

First Order Logic (FOL). FOL is define by (V, C, F, P) where V is a set of variables which is countably infinite, C a set of constant symbols, F a set of function symbols (each function comes with an arity), P a set of predicate or relation symbols (each P comes with non negative integer as its arity). It is assumed that the world consists of objects with certain relations among them that do or don't hold. Thus, it is used to express facts about some or all the objects in the universe [113]. Contrary to PL, FOL can be used to represent knowledge of complex environments in a concise way because it is sufficiently expressive to represent a good deal of commonsense knowledge. The language of FOL is built around its syntax and its semantic:

- To represent the syntax, the model of FOL defines the formal structures that constitute the possible world under consideration. The basic syntax elements of FOL are the symbols that stand for objects, relations, and functions.
  - Objects are constant symbols. The domain of a model is the set of object or domain elements it contains. The domain is required to be nonempty. Every possible world must contain at least one object.

  - Relations are predicates symbols consisting of a set of tuples of objects that are related. Each predicate symbol comes with an arity that fixes the number of arguments;
  - The model can contain some relations considered as functions. Every function has an arity that fixes the number of arguments.

- To define the semantic, a model consists of a set of objects and an interpretation that maps constant symbols to objects, predicates symbols to relations on those objects, and function symbols to functions on those objects:

  1. A term is a logical expression that refers to an object;
  2. Predicate symbols refer to relations among terms;
  3. Atomic sentence (or atom) is formed from a predicate symbol optionally followed by a parenthesized list of terms;
  4. Complex sentences can be constructed using logical connectives;
  5. Quantifiers (universal quantification - $\forall$ and existential quantification $\exists$) are used to express properties of entire collections of objects, instead of enumerating objects by name;
  6. Nested quantifiers are used to express more complex sentences using multiple quantifiers;
  7. $\forall$ and $\exists$ are intimately connected with each other through negation ($\forall x \neg P(x, Q)$ is equivalent to $\neg \exists x P(x, Q)$);
  8. Equality symbol ($=$) is used to signify that two terms refer to the same object. It can be used to state facts about a given function with negation to state that two terms are not the same.

In a broader sense, with FOL, one can:

- Make reasoning on a set of object: e.g., $\forall\, TBCases \longrightarrow \exists\, Hospital \cap treatedHospital.TBCases$ to say that "TB cases are treated in Hospital",
- Deduce formulae: e.g., "people do TB tests. If a person is tested as positive to KB then he is a TB case. A TB case follows treatment for at least 6 months with Rifampicin. Bob has been on Rifampicin for 6 months. Can we assume that Bob was a TB case?"

FOL is suitable for modelling ontologies but, it is difficult to achieve a consensus in modelling; cumbersome for modelling; complex if one have to make calculations because it is not decidable; the same knowledge can have several possible interpretations; it is complex to prove the accuracy and the completeness of the statements [130]. In the following paragraphs, we are going to present another knowledge representation formalism consisting of a family of logic called Description Logics.

Description Logics (DLs). This is the name of a family of knowledge representation formalism where the majority are a decidable fragment of FOL. DLs provide concepts (classes), roles

(properties), operations (and, or, not, some, all, atleast, atmost, any, ...) on the primitive elements of language, a classification mechanism based on the subsumption relation between concepts or roles. They help to represent the terminological knowledge and assertional knowledge of a domain in a formal and structured way and to reason effectively minimizing the response time on this knowledge [63]. The applications of DLs are numerous: Semantic Web, medicine, bioinformatics, knowledge engineering, software engineering, etc. Let's define its syntax and semantic as we have done with PL and FOL:

- The syntax of DLs defines concepts, roles, individuals, and operators.
  - Concepts are unary predicates that represent entities and classes e.g., Location:{x | Location(x)};
  - Roles (also called properties) are binary predicates that represent relations between one concept and another e.g. treatedAt: {(x, y) | treatedAt(x, y)} connects two individuals that belongs to different classes.
  - Individuals (or concept assertion) are constants that are the instantiation of a class. e.g. we can have the assertion $Person(Bob)$ to say that Bob is a person.
  - Operators (or constructors) are used to construct complex representations of concepts or roles. To guarantee the decidability and low complexity of DLs, the expressivity of all the operators are limited. Fundamental operators to define class and properties are: logical conjunction: ($\cap$), logical disjunction: ($\cup$), negation: ($\neg$), restricted form of quantification : ($\exists, \forall$).
- The semantic of DLs is given by an interpretation consisting of the interpretation domain (D) and an interpretation function (I). The interpretation function interprets all atomic concepts and all atomic roles. Atomic concepts will be mapped to subsets of domain of discoursure and the role will be mapped to a subset of a cartesian product of the domain of discoursure.

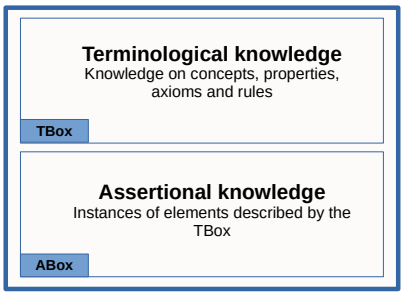


Figure 22: Knowledge Base in DL

DLs divide knowledge into 2 parts as presented by the figure 22:

- **TBox** (Terminological Box): It contains the terminological knowledge and describes the general knowledge of the domain. It is composed of **Classes** (describing the concepts of the domain), **Roles** (defining relationships between concepts) and **Axioms** (additional constraints on classes and roles).

- **ABox** (Assertional Box): It represents a configuration, a situation or a specific data of the system. It describes individuals by naming them and specifying them in terms of concepts and roles, assertions that relate to these individuals. Several ABox can be associated with the same Tbox [63].

There are several families of descriptive logic: AL, ALN, ALC, SH, ALCN, ALCQ, ALCF, SHOIN, SHIQ, SHIF. The difference between one family and another is mainly expressed in terms of expressivity.

With DLs, deductions help to derive new formulae from the starting formulae by means of the rules (inference rules). To make deduction, DLs distinguish Closed World Assumption (CWA) in which all knowledge is specified without giving the possibility to extend the model. Any assertion that cannot be proven true is false; and the Open World Assumption (OWA) in which the model is specified by giving other people the ability to extend it. If a request is made on the KB without answer, the KB will return "don't know" (In the real world there is not enough information). As DLs use OWA, this can lead to a problem of undecidability of the knowledge base (because in the case where it is false, the algorithm can run indefinitely). To show that a knowledge base is consistent, one can use either Resolution or analytic tableaux method. In DLs these methods are extended to stop in a finite time by adding the stopping conditions.

4.1.3.2 Modelling lightweight ontological knowledge

Lightweight ontological knowledge are knowledge with restricted expressivity in which concepts are connected with other concepts using untyped association. Lightweight ontologies include concepts, concept taxonomies, relationships between concepts, and properties that describe concepts. Software engineering and rules techniques are used to model this kind of knowledge because they impose a structure to the domain knowledge and constrain the interpretations of terms [58].

Rules. Rules define constraints and always resolve to either true or false. They are composed of the dependent clause expressing the condition and the main clause expressing the consequence. Placed at the top of the Semantic Web stack, they can be seen as an extension of FOL that describe knowledge that often depends on the context and cannot be easily modelled using DLs [52]. Rules can be used to assert control or influence the behaviour of a system. One common use is to standardize access control to applications e.g. "Provide statistics of the hospital in which a decision maker is located". One can distinguish general Inference rules (Premise $\rightarrow$ Conclusion), assumptions rules (Cause $\rightarrow$ Effect), production rules (Condition $\rightarrow$ Action). Rules are often combined with DLs to model knowledge. But, combining DLs and rules can give rise to undecidability. The decidable fragment called DATALOG is generally used [52].

The Use of software engineering techniques. Many software engineering techniques may be used to model lightweight ontological knowledge.

1. **UML technique:** UML might be used for knowledge modelling because it has a graphical language easy to understand by people outside the computer science domain. For instance,

	Expressivity	Ontology type	Decidability	Make calculation
PL	Weakly expressive	lightweight	Decidable	Yes
FOL	Strongly expressive	Heavyweight	Non Decidable	Yes
DL	Strongly expressive	Heavyweight	Decidable	Yes
Rules	Weakly expressive	lightweight	Decidable	Yes
UML	Weakly expressive	lightweight	Decidable	No
Database	Weakly expressive	lightweight	Decidable	No
MDA	Weakly expressive	lightweight	Decidable	No

Table 4.1: The summary of knowledge modelling approaches

class diagram notations helps to model concepts using classes, taxonomies using the generalization relation between classes, attributes using class attributes, and formal axioms using Object Constraint Language (OCL) [54, 113]. The main limit is that OCL cannot be used to represent all axioms [58, 113].

2. **MDA techniques:** With the MDA technique, the system is modelled using a meta-model and the resulting model is used to generate the application source code [20, 54]. Knowledge can be extracted from meta-models [54] by using the correspondence between MDA models components and knowledge components. The figure 14 presents the meta-model of EPICAM, very close to the representation of the ontologies using the Protege editor. When the meta-model is designed with Ecore, then, as for UML, the constraints can be added using the OCL language.

3. **Database techniques:** A database is an organized collection of data for a rapid search and retrieval of information. It can be modelled using the Entity-Relation (ER) model. Ontological knowledge can be modelled using database design techniques by matching the ER model with knowledge components [58]. The ER notations allows modelling classes through ER-entities, taxonomies using the generalization relationship between ER-entities; attributes through ER-attributes; and formal axioms with integrity constraints. As with UML, the main limit is the difficulty to model and to evaluate all axioms.

In this section, we presented the different ontological knowledge modelling formalisms with their advantages and disadvantages. The table 4.1 summarizes these knowledge modelling approaches and their limits. In this thesis, we are particularly interested in ontological knowledge. As presented by many authors [58, 63] and the table 4.1, DLs are good candidates for ontological knowledge modelling.

4.2 Ontology engineering

Ontologie engineering consists of a set of activities that concerns ontology development process, the ontology life cycle, the methodologies, tools and languages for building ontologies. The goal

of this section is to present the different artefacts used to build ontologies. In section 4.2.1, we will present the ontology development process, methods and methodologies. In section 4.2.2 we will present the knowledge representation languages and query languages. In section 4.2.3 we will present the tools used to build and manage ontologies.

4.2.1 Ontology construction process, methods and methodologies

4.2.1.1 Ontology development process

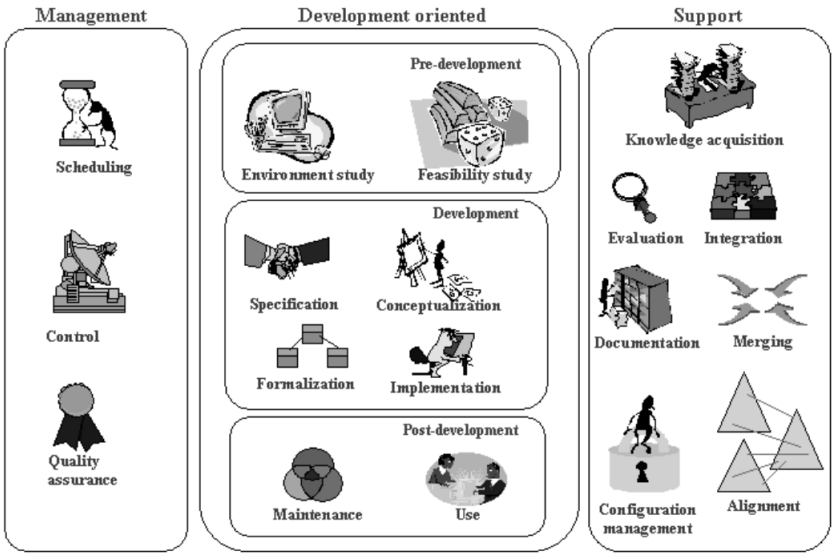


Figure 23: Ontology development process [58]

The ontology development process (see figure 23) refers to activities that are being performed when building ontologies without identifying the order in which these activities should be performed. These activities are very important especially in the case where the ontology is being built by geographically distant cooperative teams. The ontology life cycle describes the different phases involved in the ontology development. When this life cycle is well defined, error detection is done much earlier and it is possible to control the quality, the delays and the costs of the development. The control will help for example to know if the ontology was well built and the validation will help to know if the ontology responded to the needs of the domain experts. The ontology development activities can be organized in 3 categories [58]:

1. **Ontology management activities:** It consists of scheduling activity, control activity and quality assurance activity.
    - **The scheduling activity:** Consists of identifying the problem to be solved, the tasks to be performed, their scheduling, the time needed and the resources for their realization.
    - **The control activity:** Here, all the steps must be checked in order to ensure that all scheduled tasks are completed as intended.

- **The quality assurance activity:** The quality assurance activity ensures that all produced resources (ontology, documentation) during the development process are of good quality.

2. **Ontology development activities:** It involves the Pre-development, the Development and the Post-development activities.
   - **The pre-development activity:** During this activity, a situational analysis (environmental study) and a feasibility study are carried out in order to identify the platforms where the ontology will be integrated.
   - **The development activity:** The development activity is made up of the specification, the conceptualization, the formalization and the implementation. The specification permits to respond to the following questions: Why is the ontology being built? What its intended uses are and who the end-users are? The conceptualisation involves structuring the domain knowledge into a conceptual model. The formalisation is the transformation of the conceptual model into a formal or semi-computable model. The implementation is the serialization of the computable model into an ontology representation language.
   - **The post-development activity:** The post-development activity is made up of the maintenance activity and the reusing activity. The maintenance activity involves the updates and the corrections of the ontology if need be. The reusing activity consists of reuse the ontology constructed by other ontologies or applications.
3. **Ontology support activities.** These include a series of activities performed at the same time as the development activities, without which the ontology could not be built. These activities are:
   - Knowledge acquisition activity which consists of gathering knowledge from identified sources (human, text, databases, meta-models, source code, etc.).
   - Evaluation activity consists of the validation of the ontology and associated resources (documentation, software environment) by verifying if it is really the shared conceptualisation of the modelled domain.
   - Integration, merging, and alignment activities consisting of the construction of a new ontology from already existing ones.
   - Documentation activity. This gives every detail on all the completed stages and products that are being produced.
   - Configuration management activity consists of managing all versions of the ontology and documentation so that when there are errors, knowledge engineers can go back to rectify or make corrections on the previous versions.

4.2.1.2 Ontology construction methods

To build ontologies, knowledge engineers may choose amongst the existing methods. For this purpose, they must ask themselves some questions. For example, is it necessary to use the top-

down, bottom-up, or middle-out approach? Is it necessary to build it manually, automatically or semi-automatically? Or will the ontology be built from existing ones?

Top-down approach: From general to particular. In the top-down approach, one starts from general concepts and evolves towards major specializations. Firstly, one identifies the most general concepts and creates categories at the most general level as possible. The main advantage of using a top-down approach is a better control of the level of details. However, starting at the top can equally result in choosing and imposing arbitrary and possibly not needed high level categories [58].

Bottom-up approach: From particular to general. In this approach, the ontology is constructed from the most specific concepts, which are then grouped into categories. The main result here is a very high level of detail of terms obtained. However, this approach increases the overall development effort and makes it difficult to spot commonality between related concepts and equally increases the risk of inconsistencies [58].

Middle-Out approach: Starting in the middle. In this approach, an intermediary layer of concepts serves as a starting point. The development can go in both directions. It is recommended to identify firstly the core of basic terms, then, specify and generalize as required. This approach strikes a balance in terms of the level of details. Details only arise as necessary by specializing the basic concepts, so that some wasted efforts are avoided [58].

(Semi)automatic construction of ontologies. Ontologies can be built manually, automatically or semi-automatically. During manual construction, the various resources containing knowledge are collected, terms are identified and the ontology is constructed. Automatic construction (also called ontology learning) implements the generation of terms automatically from resources [58]. Ontology learning is detailed in section 4.3.

Building ontologies by reusing existing ontologies. Before the development of a new ontology, one can consider the reuse of existing ones. There are several ways to reuse existing ontologies [58]:

- **Ontology re-engineering:** This is used when domain experts do not agree on the content of ontologies or the conceptual model of the ontology is absent. Thus, the re-engineering process will consist of recovering the model of an ontology and transforming this model into another ontology more appropriate.
- **Ontology enrichment:** This is the process of adding new knowledge in an ontology to have a more complete one. This will help to ensure their growth and to continue to meet the needs of users [108].
- **Ontology Fusion.** This consists of creating a new ontology by merging existing ones. There are several methods of ontology fusion: ONIONS allows the creation of an ontology library from multiple sources, FCA-Merge merges two ontologies into one set of domain documents, PROMPT takes as an input two ontologies and creates a list of matches [58].
- **Ontology alignment.** This consists of creating links between several ontologies without modifying them, hence preserving the original ontologies. It is often used for complementary

domains.

- **Cooperative Construction of Ontology (Co4).** This is a protocol developed at INRIA for the collaborative construction of KBs. Its goal is to allow the discussion of people and knowledge commitment in the KBs of a system.

4.2.1.3 Ontology construction methodologies

The first methodologies for ontology construction were inspired by the experience of domain experts and knowledge engineers. However, a series of methodologies have been reported: Cyc methodology, TOVE methodology, METHONTOLOGY, SENSUS methodology, On-To-Knowledge methodology, TERMINAE, Ontology Development 101 and NeOn methodology [58]. With time, other proposed methodologies, based on software engineering (Unified Process Methodology for ontology design), software architectures (ontology design pattern methodology) and agile methodologies appeared [1]. This section details the NeOn methodology used in this thesis.

NeOn Methodology for Building Ontology Networks. An ontology network is composed of ontologies related together via meta-relationships such as mapping, modularization, version, and dependency relationships. Suárez-Figueroa et al.[127] proposed a methodology that they named NeOn methodology to build such ontology. This methodology takes into account the existence of multiple ontologies in ontology networks, the collaborative ontology development, the dynamic dimension, and the reuse and re-engineering of knowledge resources. It is composed of a set of scenarios that the knowledge engineer can combine in different ways, and any combination should include Scenario 1.

- **Scenario 1: From specification to implementation**. The first scenario is composed of: (1) Knowledge acquisition activity, carried out during the whole development. During this activity, knowledge engineers simultaneously acquire knowledge and make the specification that the ontology should fulfill. This gave rise to the ontology requirements specification document (ORSD). Then, a quick search for knowledge resources using the terms in the ORSD permits us to know which types of resources are available for a possible reuse. (2) The scheduling activity can start. It uses ORSD and knowledge resources (it exists) to carry out the rest of the activities (i.e., conceptualization, formalization, and implementation) using existing methodology such as On-To-Knowledge.

- **Scenario 2: Reusing and re-engineering non-ontological resources (NORs)**. Knowledge engineers decide which NORs to be used. Then, from these resources the terms are extracted and the ontology is built.

- **Scenario 3: Reusing ontological resources**. In this scenario, the knowledge engineers use existing ontological resources to build the ontology. From each ontology selected, a part or a whole can be reused. Knowledge engineers can also perform a re-engineering of ontological resources (following Scenario 4) or the merging of ontologies of the same domain to obtain a new ontology (following scenarios 5 and 6).

- **Scenario 4: Reusing and re-engineering ontological resources**. In this activity, knowledge engineers re-engineer ontological resources before their integration in the ontology (corresponding activity of Scenario 1).
- **Scenario 5: Reusing and merging ontological resources**. This scenario consists to merge/align ontological resources of the same domain. The merging will permit to obtain a new ontology. The alignment will permit to establish links among the selected resources in order to create a network.
- **Scenario 6: Reusing, merging and re-engineering ontological resources**. In this scenario, knowledge engineers decide not to use the set of merged ontological resources such as it is, but to re-engineer it. The set of merged ontologies re-engineers is integrated in the corresponding activity of Scenario 1.
- **Scenario 7: Reusing ontology design patterns (ODPs)**. In this scenario, knowledge engineers access ODPs repositories in order to reuse ODPs. ODPs can be used to reduce modelling difficulties, speed up the modelling process or check the adequacy of modelling decisions.
- **Scenario 8: Restructuring ontological resources**. This scenario can performed as followed: (1) modularizing the ontology in different ontology modules; (2) pruning the branches of the taxonomy not considered necessary; (3) extending the ontology including new concepts and relations; and (4) specializing those branches that require more granularity and including more specialized domain concepts and relations.
- **Scenario 9: Localizing ontological resources**. This is to adapt an existing ontology to one or various languages and culture communities to obtain a multilingual ontology. For example, the translation of all ontology labels into one or several natural languages.

4.2.2 Knowledge representation languages and queries languages

Once the ontology is modelled, it will be put in a form understandable by the machine and queries will be made to retrieve information. At this level, a very important decision is to choose the language(s) to model the knowledge and to make queries. Among the multitude of knowledge representation languages, the selection criteria is based on the knowledge base that one wants to build and the inference mechanisms needed by applications that will use the ontology. In the following paragraphs, we will present some knowledge representation languages and queries languages.

4.2.2.1 Knowledge representation languages

In early 1990, a set of knowledge representation languages were invented. These languages were based mostly on FOL (Cycl, KIF, Ontolingua, OCML, and Flogic) and others were based on DLs and production rules (LOOM). The Internet growth has led to the creation of languages to exploit the characteristics of the Web (RDF(S), OWL). These languages are called Web-based ontology languages or Ontology Markup Languages. Knowledge representation Languages can be classified

according to the goal to which they aim at. One can distinguish languages to improve the process of ontologies building (OCMl), languages that helps to make inferences (OIL, OWL), languages that permits the design of ontologies (Ontolingua), languages that permits exchange on the Web (RDF(S), OWL, SHOE, OIL) [58]. We will be focusing on the languages that permit the exchange of data on the Web in general, and especially on RDF(S) and OWL languages. We will present these languages according to the following two main dimensions: Knowledge Representation (description of how the components in the ontology can be modelled) and Reasoning Mechanisms (used to create an inference engine with the corresponding deductive mechanisms).

1. **RDF(S)**. The acronym RDF stands for Resource Description Framework where **Resource** is everything that can be uniquely identified and referenced simply by using an Uniform Resource Identifier (URI). **Description** means that all the resources are described (e.g., by using properties and relationships between resources). **Framework** means that they are based on a formal template defining all possible relationships between resources. The RDF graph can be represented by a set of triples (Subject, Predicate, Object). The triple is the smallest description structure in RDF and each one represents a declaration. All statements follow the same pattern. The subject is the resource to be described, the predicate (property value) refers to a property type applicable to that resource and the object represents data (literal) or other resources. An RDF document is a tagged multigraph in which each triple corresponds to an oriented arc whose label is the predicate, the source node is the subject and the target node is the object [52, 58, 63].

   RDF(S) is an extension of the RDF language which provides RDF documents with a structure. Its purpose is to provide an encoding and interpretation mechanism to represent resources for software, and to describe and link all Web resources [52, 58, 63].

   - **RDF(S) Knowledge representation:** RDF(S) provides the most basic primitives for ontology modelling achieving a balance between expressivity and reasoning. In RDFS, concepts are known as classes. Classes are referenced either by their name or by a URL to a Web resource and can include their documentation and their super-classes. Attributes of classes are defined as properties. The domain of a property is the class to which the attribute belongs, and the range is the type of the attribute value. No cardinality constraints nor default values can be defined for attributes. Class attributes can be represented by defining the domain of the property, and including the property value in the class definition. Concept taxonomies are built in RDF(S) by defining a class as a subclass of one or more classes. However, neither disjoint nor exhaustive knowledge in concept taxonomies can be expressed. Binary relations between classes are defined in RDF(S) as properties. However, relations of higher arity cannot be represented directly. Assertions made by instances can be represented using reification (transforming the value of a property into a statement).
   - Reasoning mechanisms: Most of the inference systems for RDF(S) are devoted to querying information about RDF ontologies.

   RDFS is used to represent the lightweight ontologies that can be serialized using RDF/XML, N-Triples, Turtle and the triples serialized can be saved in files, in a database or in a triple-store.

2. **OWL (Ontology Web Language)**. The OWL language is an XML language used to represent ontologies modelled using DLs and for publishing and sharing knowledge on the Web. It helps to represent a very rich knowledge, to reason on the data and satisfies the following conditions: expressivity, clarity, readability, unambiguity, extensibility [58, 63]. It is divided into three layers: OWL Lite, OWL DL, OWL Full.

   - **OWL-Lite:** It corresponds to the SHIF(D) family of DLs and is characterized by its simplicity, the ease of programming, and its quick reasoning. It has been designed to express simple constraints for which inference algorithms are decidable.
   - **OWL-DL:** It is the SHOIN(D) family of DLs. It has a higher expressivity than OWL-Lite. With OWL-DL, real world elements are represented by concepts, roles and individuals. Concepts and roles have a structured description to which semantics are associated and any manipulation of semantics must be consistent with that semantics. It is composed of :
     - **OWL EL (Existential Language):** A family of description logic that only provides the existential quantification of variables.
     - **OWL QL (Query Languages):** It permits answer to queries that can be rewritten in a relational query language.
     - **OWL RL (Rule Language):** It indicates reasoning profiles that can be implemented using a rule-based system.

   - **OWL-Full:** This is a language that has been designed to ensure compatibility with RDFS without providing decidability of inference algorithms. It is characterized by maximum expressivity, full compatibility with RDF/RDFS and very complex reasoning. However, it is slow and undecidable.

   Concerning knowledge representation and inference mechanism in OWL:

   - **Knowledge representation:** Different OWL languages have different knowledge representation. In a broader sense, with OWL, concepts are known as classes. A class may contain its documentation and a list of primitives that defined it. These are the superclasses, equivalent classes, disjoint classes, conjunction, disjunction, negation of other classes, the enumeration of all classes instances, collection of individuals of the classes, property restriction (existential restriction, role filter, number restriction) that contains a reference to the property to which the restriction is applied and an element for expressing the restriction. Class attributes must be defined as properties in the ontology. There are two types of properties: ObjectProperty (whose range is a class) and DatatypeProperty (whose range is a datatype). To define a property, one may explain its domain and range. Besides, in OWL, one can define properties hierarchies, properties equivalences, inverse properties, transitive properties, symmetric properties, global cardinality restrictions on all kinds of properties, functional properties and inverse functional properties. Higher arity relations must be defined as concepts. Instances are defined using RDF vocabulary. With instances, one can assert that two instances are equivalent or different, and a set of individuals are different from each other.
   - **Reasoning mechanisms:** Different ontology languages have different expressivity and inference mechanisms. In the broader sense, the semantic of OWL is described in two

different ways: Firstly, as an extension of the RDF(S) model and secondly, as a direct model-theoretic semantics of OWL. OWL allows the inclusion of additional statements in its ontologies apart from those explicitly defined in the language. Multiple inheritance is allowed in OWL ontologies. Constraint checking can be performed on the values of properties and their cardinalities. OWL assumes monotonic reasoning, even if class definition or property definitions are split up in different Web resources. This means that facts and entailments declared explicitly or obtained with inference engines can only be added, never deleted, and that new information cannot negate previous. RDF(S) query engines, storage systems, and parsers can be employed to manage OWL ontologies since they can be serialized in RDF(S).

In the above paragraphs, we presented two Semantic Web languages for sharing and exchanging data on the Web. We have seen previously that the RDF language helps to present facts without necessarily bringing semantics to the types of data. The RDFS language complements the RDF language by making it possible to create the data types and by creating the class hierarchy, and the OWL language comes with more semantics, permitting the representation of classes, properties with restrictions on these classes and these properties. Note that, OWL-DL does not represent the relationship between the roles. So, all the knowledge cannot be represented using OWL-DL.

4.2.2.2 Rule Languages

Rules can be seen as an alternative or complementary stack to represent semantics and inferences over the web of data. A rule system is a specific implementation of syntax and semantics, which can extend to include existential quantification, universal quantification, logical disjunction, logical conjunction, and so on. The semantics of RDFS and some subset of OWL (OWL2 RL) can be axiomatized as first-order logic that can be used as a foundation for a rule-based implementation. Thus, the rules can be seen as part of an ontology supplementing RDF or OWL declarations [52, 58, 64]. In the following paragraphs, we will present three rules languages: RIF, DATALOG and SWRL.

1. **Rule Interchange Format (RIF).** This is designed to exchange rules on the Web in general and the Semantic Web in particular. RIF has become a W3C (World Wide Web Consortium) recommendation since 2013. It is an extensible set of rule dialects.

   RIF rule includes 3 dialects: a basic dialect, a basic logical dialect, and the production rule dialect [52].

   - **RIF-CORE:** It is the core of all the primitives common to RIF dialects. It corresponds to the HORN logic without symbol function.
   - **RIF-BLD (Basic Logic Dialiect):** It consists of a set of well-formed formulae from terms built on one alphabet. It helps to represent logic programs on the positive facts and corresponds to the HORN logic without the symbol of equality. The reasoning is based on the deduction of new facts by the instantiation of the universal rules (application of the rule of modus ponens and evaluation of conjunctive or disjunctive formulae).

- **RIF-PRD (Production Rule Dialect):** This is used to represent the production rules whose application triggers actions to add, modify, delete facts in a class.

2. **DATALOG.** DATALOG has been developed at the beginning for deductive databases. The idea was to couple a database to a set of logical rules, allowing the deduction of information [52]. DATALOG rules are used to mix classes and relations. A knowledge base (DATALOG programs) are a set of HORN clauses without function symbols. OWL DL does not allow the mixing of classes and properties but, DATALOG permits it. By combining OWL-DL with DATALOG permits to make the ontology more expressive.

3. **Semantic Web Rule Language(SWRL).** SWRL is the semantic Web rules language proposed as a W3C recommendation in 2004. It is based on the combination of OWL and RuleML / DATALOG. The idea being to use DATALOG rules on OWL ontologies to model more knowledge [94]. To do this, the symbols in the rules can be OWL identifiers.

In the above paragraphs, we presented how ontologies are represented in a machine readable form. We also presented OWL's limitations and showed that OWL-DL can be combined with the rules for more expressivity. Note that modelling knowledge is not enough. Some mechanisms must be put in place to obtain information needed.

4.2.2.3 Query Languages

A query language is a language used to request and to retrieve information in a data source. There are several types of Semantic Web query languages. Some permit the extraction of information from the knowledge base and others can, in addition make an inference on the data as seen below.

1. **SparQL Protocol and RDF Query Language (SparQL).** Its purpose is to provide service-level interoperability and structured data across the Internet to easy access to all data on the Web. It permits the extraction of all types of data contained in RDF; The exploration of the data; the transformation of RDF(S) from one vocabulary to another; The building of new graphs from RDF query graphs; The updating of RDF graphs; The discovering of hidden information using the SparQL service and the federation of data from multiple SparQL queries.

2. **Semantic Query-Enhanced Web Rule Language (SQWRL).** It is a query language for extracting information from OWL ontologies [94]. It offers two types of queries operators. **The Basic Operators** use the SWRL rules as a pattern by replacing the elements with the SQWRL selection operations. **Collection Operators** provide a set of operators for grouping, aggregation, disjunction, etc.

4.2.3 Ontology development tools

Building ontologies is complex and time consuming. Ontology development tools provide interfaces that help users carry out some of the main activities of the ontology development process

such as conceptualization, implementation, consistency checking and documentation. Ontology tools can be classified by categories:

- **Ontology development tools:** These tools are used to build a new ontology from scratch. They also give support to ontology documentation, ontology export and import, ontology graphical edition, etc.
- **Ontology merging and alignment tools:** These tools are used for merging many ontologies or for aligning different ontologies.
- **Ontology learning tools:** These can be used to derive ontologies (semi)automatically from data sources.
- **Ontology querying tools and inference engines:** These allow querying ontologies easily and performing inferences with them.
- **Ontology evaluation tools:** These tools are used to evaluate the content of ontologies and their related technologies. It tries to reduce problems when one needs to integrate and use ontologies in other ontologies or in information systems.

Tools allowing the manipulation of the ontology can be classified in several categories: edition, integration, visualization, validation, interrogation, extraction, exportation, storage, inference engines, and development tools. A tool can belong to several categories depending on the features it offers. According to this classification, we can distinguish:

- **Protege, Ontolingua Server, WebOnto and OntoSaurus:** These are tools used to build ontologies completely by hand, automatically or by integrating existing ontologies.
- **Virtuoso and R2RML:** These set of tools allow to build ontologies automatically by extracting it from the database and storing it in a triple store.
- **Protege, VOWL, Ontolingua Server, WebOnto and ODE:** These allow the visualization of the ontology in the form of a graph.
- **Facct ++, Hermit, Pellet, Drool and Racer:** These make inferences about knowledge bases. They will also validate if the knowledge base is consistent.
- **Virtuoso and R2RQ:** These allow access to data from relational databases as an RDF graph.
- **Sesame, Jena and Virtuoso:** These tools offer storage tools for RDF triples.
- **Sesame, Jena and SWRL API:** These tools provide APIs for developing Semantic Web applications.

In this thesis, we will particularly use Protege and SWRL:

- Protege is an Open-source software available in desktop and web version and developed by Stanford University [92]. Its plug-in system makes it expandable. It includes many features: an ontology editor with which one can define the hierarchy of classes and properties, annotate resources and make restrictions on classes and properties; an interface allowing the integration of ontologies (fusion, mapping, alignment); an interface to define SWRL rules and SQWRL requests; an inference engine to validate the ontology; a rules engine to validate rules written in SWRL and execute queries written in SQWRL; a SparQL query interface; an ontology visualization interface; an interface to export data in different formats (RDF/XML, RDF/Turltle, OWL/XML, Json). The Web version can allow multiple users or groups of users to build distributed ontologies.
- SWRL API is a Java API for programming Semantic Web applications by applying inference rules on ontologies encoded in RDFS or in OWL [94]. It consists of a library collection that allows developers to work with SWRL rules and SQWRL queries in their applications; to model, reason and query knowledge bases; to manage OWL reasoners; to use SWRL rules engines; to execute queries written in SQWRL. Drools is an implementation of SWRL API rules to execute SWRL and SQWRL rules.

4.2.4 Ontology evaluation

Before reusing existing ontologies, their content should be evaluated. The purpose of ontology evaluation is to determine what the ontology defines correctly, what it does not, and what it does incorrectly. It includes ontology verification, ontology validation and ontology assessment [58].

- Ontology verification consists of ensuring that it implements correctly the ontology requirements and the competencies questions.
- Ontology validation verifies if the ontology definitions really model the real world for which the ontology was created.
- Ontology assessment is focused on judging the ontology content from the user's point of view. Different types of users and applications require different means of assessing ontology. Gómez-Pérez and al. [58] proposed the evaluation of ontologies using some evaluation criteria:
  - **Technical evaluation:** This is done by the developers. It permits to know if the ontology is well built, and if it is consistent.
  - **User evaluation:** This is performed by users. It permits to check if the ontology meets their needs.
  - **Consistency evaluation:** This is performed to check whether it is possible to obtain contradictory conclusions from valid input definitions. A given definition is consistent if and only if the individual definitions are consistent and no contradictory knowledge can be inferred from other definitions and axioms.

- **Completeness evaluation:** It is difficult to prove the incompleteness of an ontology. But, if a concept or a definition is missing, the ontology can be said to be incomplete. In a broader sense, an ontology is complete if and only if:
  - Everything that is supposed to be in the ontology is explicitly stated or can be deduced;
  - Any definition is complete. This can be determined by: (a) precise knowledge of the definition (does it define the world?) (b) all knowledge that is necessary, but not explicit (it should be noted that definitions can be inferred from other definitions and axioms).
- **Conciseness evaluation:** An ontology is concise if (a) it does not store useless definitions, (b) there are no redundancies between definitions of terms, (c) redundancies definitions can not be deduced from explicit definitions.

4.3 Ontology learning

Acquiring knowledge for building an ontology from scratch, or for refining an existing ontology is costly in time and resources. Ontology learning techniques are used to reduce this cost during the knowledge acquisition process. Ontology learning refers to the extraction of ontological knowledge from unstructured, semi-structured or fully structured knowledge sources in order to build an ontology from them with little human intervention [7, 75, 121, 146]. In this section, we will present knowledge sources generally used for ontology learning (section 4.3.1), some ontology learning techniques (section 4.3.2) and ontology learning evaluation (section 4.3.3).

4.3.1 Knowledge sources for ontology learning

The process of developing an ontology requires knowledge acquisition from any relevant sources. There are several possible sources of knowledge: domain experts or unstructured, semi-structured, and structured sources [127].

4.3.1.1 Domain experts

A domain expert is a person knowledgeable of a domain. To get knowledge from domain experts, a knowledge engineer conducts interviews. This process might lead to knowledge loss or even worse, introduce errors because misunderstandings arises frequently in human communication.

4.3.1.2 Unstructured knowledge sources

Unstructured knowledge sources contain knowledge that do not have a pre-defined organization. These are all kinds of textual resources (Web pages, manuals, discussion forum postings, spec-

ifications, analysis and conception documents, source code comments) and multimedia contents (videos, photos, audio files) [5, 7, 23, 75, 30, 55, 121]. Unstructured sources are the most recurrent and can permit us to extract a more complete knowledge. However, the unstructured sources are easily accessible to human information processing only. For example, extracting formal specifications from arbitrary texts is still considered a hard problem because sentences might be ambiguous and, in some cases, no unique correct syntactic analysis is possible [64].

4.3.1.3 Structured knowledge sources

Structured knowledge sources contain knowledge described by a schema. It is advantageous to use these knowledge sources because they contain directly accessible knowledge [64]. Some structured knowledge sources include:

- Ontologies: Before constructing an ontology from scratch, one may look at other ontologies that could be reused [104, 124, 127];
- Knowledge bases: In knowledge bases, one can generate discovered rules as input to develop a domain ontology [7, 72];
- Database : Terms to be used to build an ontology can be extracted from a database schema [7, 29, 32, 66, 144].

4.3.1.4 Semi-structured knowledge sources

Semi-structured knowledge sources contain knowledge having a structure that already reflects part of the semantic interdependencies. This structure facilitates the extraction of a schema [64]. Some examples of semi-structured knowledge sources are:

- Folksonomies/thesaurus: It is advantageous to extract knowledge from folksonomies or/and thesaurus to build an ontology because they reflect the vocabulary of their users [53, 136];
- XML (Extensible Markup Language): The aim of XML data conversion to ontologies is the indexing, integration and enrichment of existing ontologies with knowledge acquired from XML documents [59];
- UML/meta-model: To learn an ontology from UML or/and meta-model, one approach is to extract OWL classes and properties from diagrams or to use Ontology UML Profile (OUP) which, together with Ontology Definition Meta-model (ODM), enable the usage of Model Driven Architecture (MDA) standards in ontological engineering [54];
- Entity-relation diagram: They can be used to learn ontologies because they are used to describe the information managed by the databases [45];
- Source code [13, 14, 23, 51, 144]: Generally, in source code, the names of data structures, variables, functions are close to the terms of the domain.

A lot of work has been done on the extraction of ontological knowledge from texts, databases, XML files, vocabularies, and the use of ontologies to build or enrich other ontologies. This has resulted in a wide range of models, techniques and tools for the generation of knowledge structure that can be considered as an intermediate process when constructing ontologies. It should be noted that few works go beyond extracting concepts and properties from source code whereas axioms and rules are also key elements of ontologies.

4.3.2 Ontology learning techniques

To extract knowledge from knowledge sources, many techniques are used [7, 61, 75, 121]. Shamsfard and Barforoush [121] proposed a classification of these techniques by considering symbolics, statistics and multi-strategies.

4.3.2.1 Symbolic techniques

In symbolic techniques, the extraction process consists of examining text fragments that match some predefined rules, looking for lexico-syntactic patterns corresponding for instance to taxonomic relations or scanning for various types of templates related to ontological knowledge. A symbolic method can be rule-based, linguistic-based or pattern-based.

1. Rule-based models are represented as a set of rules where each rule consists of a condition and an action [146].
    - *Logical rules* may be used to discover new knowledge by deduction (deduce new knowledge from existing ones) or induction (synthesize new knowledge from experience). For example, inductive logic programming can be used to learn new concepts from knowledge sources [7, 30, 83, 121];
    - *Association rules* aim at finding correlations between items in a dataset. This technique is generally used to learn relations between concepts [5, 7, 30, 121] and can be used to recognize a taxonomy of relations [7] or to discover gaps in conceptual definitions [30, 121, 139].
2. *Linguistic* approaches (syntactic analysis, morpho-syntactic analysis, lexico-syntactic pattern parsing, semantic processing and text understanding) are used to derive knowledge from text corpus [7, 121]. This technique can be used to derive an intentional description of concepts in the form of natural language description [139].
3. *Pattern/Template-driven* approach allows searching for predefined keywords, templates or patterns. Indeed, a large class of entity extraction tasks can be accomplished by the use of carefully constructed regular expressions [81].

Although very powerful for particular domains, symbolic techniques are inflexible because of their strong dependency on the structure of the data. Symbolic techniques are precise and robust, but can be complex to implement, and difficult to generalize [121].

4.3.2.2 Statistic-based techniques

Statistic analysis for ontology learning is performed from input data to build a statistical model [7, 75, 121, 146]. Several statistical methods for extracting ontological knowledge have been identified in the literature:

1. *Co-occurrence or collocation detection* identifies the occurrence of some words in the same sentence, paragraph or document. Such occurrences hint a potential direct relation between words [73]. These techniques can be used to discover terms that are siblings to each other [26].
2. *Clustering* can be used to create groups of similar words (clusters) which can be regarded as representing concepts, and further hierarchically organize these as clusters. This technique is generally used for learning concepts by considering clusters of related terms as concepts and learning taxonomies by organizing these groups hierarchically [30]. Ontology alignment can use agglomerative clustering to find candidate groups of similar entities in ontologies [139].
3. Hidden Markov Models (HMMs) define a generative statistical models that are able to generate data sequences according to rather complex probability distributions and that can be used for classifying sequential patterns [47, 113, 120]. Zhou and Su [145] have used HMM for Named Entity Recognition; Maedche and Staab [5] have used the n-gram models based on HMMs to process documents at the morphological level before supplying them to term extraction tools. Labsky et al. [77] present the use of HMMs to extract information on products offered by companies from HTML files.

4.3.2.3 Multi-Strategy learning

Multi-Strategy learning techniques leverage the strengths of the above techniques to extract a wide range of ontological knowledge from different types of knowledge sources [7, 121, 146]. For example, Maeche and Staab [5] present the use of clustering for concept learning and association rules to learn relations between these concepts.

4.3.3 Ontology learning evaluation

After the extraction process, the evaluation phase permits to know whether the knowledge extracted is accurate and to conclude on the quality of the knowledge source. The evaluation of ontological knowledge is coined by several authors in the literature [6, 38]. Dellschaft and Staab [38] have proposed two ways to evaluate ontological knowledge: (1) In manual evaluation by human experts, the knowledge is presented to one or more domain experts who have to judge to what extent it is correct; (2) The comparison of the knowledge to existing reference vocabularies/ontologies to ensure that it covers the studied domain.

4.4 Related works on ontology learning from source code

Despite the large amount of available source codes and the fact that they may contain relevant knowledge of the domain [13, 14, 23, 144] addressed by the software, the number of existing work on knowledge extraction from these sources is quite low. Parser-based approach and machine learning techniques are commonly used in knowledge extraction from source code.

4.4.1 Parser-based approach

A straightforward solution to extract knowledge from source code is to use a parser. There are works in this direction for generating knowledge base (RDF triples) or extracting ontological knowledge (concepts and properties) from source codes using parsers. For instance, CodeOntology [10, 11] parser is able to analyze Java source code and serialize it into RDF triples. From these triples, highly expressive queries using SPARQL (SPARQL Protocol and RDF Query Language) can be executed for different software engineering purposes including the searching of specific software component for reuse. Ganapathy and Sagayaraj [51] used QDox[1] generator to generate an ontology that will further enable the developers to reuse source code efficiently. QDox generator is a parser that can be used for extracting classes, attributes, interfaces and method definition from Java source code. In the approach proposed by [144], the authors defined the components parts of the source code and break down the source code into these components. The source code is browsed and the different components are analyzed in order to take an appropriate action which is the extraction of knowledge sought. This knowledge can be used in supplementing and assisting ontology development from database schemas.

Beyond RDF triples, terms, concepts and properties extraction, existing parsers do not provide services for axioms and rules extraction. To overcome these limits, they need to be improved. However, building and/or updating parsers for programming languages is a non-trivial, laborious and time-consuming task [46, 91].

4.4.2 Machine learning-based approach

Machine learning approaches are also proposed to extract knowledge from source code.

Kalina Bontcheva and Marta Sabou [23] have presented an approach for ontology learning from software artifacts such as software documentation, discussion forums and source code by using the language processing facilities provided by GATE 2 platform[2]. GATE 2 is an Open source software developed in Java for building and deploying Human Language Technology application such as parsers, morphology, tagging, Information Retrieval tools, Information Extraction components, etc. To extract concepts from source code, Kalina Bontcheva and Marta Sabou used the GATE key phrase extractor, which is based on TF.IDF (term frequency/inverted document frequency).

[1] https://github.com/paul-hammant/qdox
[2] https://gate.ac.uk/

The TD.IDF approach is an unsupervised machine learning technique which consists of finding words/phrases that are characteristic of the given text, while ignoring phrases that occur frequently in the text simply because they are common in the language as a whole. When using TF.IDF on the source code, high frequency terms specific to the programming language can be eliminated and only terms specific to the given software project would be selected as relevant to the domain (ontology concept). This approach is used to extract concept. However, ontological knowledge is also made up of properties, axioms and rules.

Labsky et al. [77] presented an approach for information extraction on product offered by companies from their websites. To extract information from HTML documents, they used Hidden Markov Models to annotate these documents. Tokens modelled by this HMM include words, formatting tags and images. The HMM is modelled using four states: the target state (T) which is the slot to extract, the prefix and the suffix state (P, S) which constitute the slot's context, and the irrelevant tokens modelled by a single background state (B). This approach permitted the extraction of slots and the relation between nearby slots. For example product image often follows its name. Unlike the authors approach which consists of terms extraction, our approach uses meta-data extracted from source code in order to identify to which ontological component every term/group of terms corresponds to.

4.5 Conclusion

This chapter presented ontologies engineering. In effect, ontologies are knowledge representation languages used to model a domain/problem. Then, we presented in detail different types of ontologies and the methodologies, methods and tools involved in their development. Ontologies can be classified by lightweight ontologies and heavyweight ontologies. Lightweight ontologies are modelled using rules or software engineering techniques. Heavyweight ontologies are modeled using logical techniques. To develop them, knowledge must be acquired from different sources such as domain experts, unstructured, semi-structured, and structured sources. Semi-structured knowledge sources contain knowledge facilitating the extraction of a schema. For instance, in the source code, the names of data structures, variables, functions are close to the terms of the domain and surrounded by a set of keywords. Several methods are proposed for ontologies development. The top-down method consists of starting from general concepts and evolves towards major specializations. The bottom-up method involves the construction of the ontology from the most specific concepts, which are then grouped into categories. The middle-Out method consists of an intermediary layer of concepts that serves as a starting point. With the manual construction method, the various resources containing knowledge are collected, terms are identified and the ontology is constructed. Automatic or semi-automatic approaches, also called ontology learning implements the generation of terms automatically from knowledge sources. The methods proposed for this task can be classified in symbolic techniques, statistical techniques and multi-strategy techniques. Amongst ontologies development methodologies proposed in the literature, we used the NeOn methodology in this thesis. It is composed of a set of scenarios that the knowledge engineer can combine in different ways. Once the knowledge is acquired from knowledge sources, knowledge representation languages such as RDFS, OWL allow to put them in a form understandable by the machine and Queries languages are used to retrieve information. Given that building ontologies is a tedious

task, ontologies development tools allow us to carry out some of the main activities of the ontology development process. Then, tools are used to build ontologies from scratch, by merging/aligning many ontologies, by semi-automatically extracting knowledge from knowledge sources, etc.

We have seen in this chapter that despite the large amount of source code available and the fact that they contain relevant domain knowledge, they are rarely used for ontology building. To use source code to construct an ontology, knowledge must be extracted. In chapter 6, we proposed a method for ontology learning from source code.

5 Ontology learning from source code using Hidden Markov Models

Source code contains well-defined words in a language that everyone understands (for example the elements generally found on the user interface), some statements with a particular lexicon specific to the programming language and to the programmer. For example, in Java programming language, the term "class" is used to define a class, the terms "if", "else", "switch", "case" are used to define the business rules (candidate to become rules). Other terms defined by the programmer such as "PatientTuberculeux" are used to represent the names of classes (candidate to be concept); the term "examenATB" is used to define the relation (ObjectProperty) with cardinality (candidate to become axiom) between the classes "PatientTuberculeux" and "Examen"; and the group of terms "int agePatient" is used to define a property (DataProperty) of the class "PatientTuberculeux". This chapter presents how ontological knowledge can be extracted from Java source code to build an ontology. Given that the approach used to extract the knowledge is based on Hidden Markov Models, which is a probabilistic model, this section will present the probabilistic models before the presentation and the use of the approach. Then, the section 5.1 presents the probabilistic models in general, section 5.2 presents the Hidden Markov Models, section 5.3 presents the source code and section 5.4 presents the approach.

5.1 Probabilistic models

Probabilities are used to build models in order to represent random phenomena. Temporal probability models particularly are used to model phenomena that can be represented as a set of events evolving in time. A probability model for a particular experiment is a probability distribution that predicts the relative frequency of each outcome if the experiment is performed a large number of times. For example, in a model of "weather" tomorrow, the outcomes might be $sunny$, $cloudy$, $rainy$, and $snowy$. A subset of these outcomes constitutes an event. For example, the event of precipitation is the subset consisting of $\{rainy, sunny\}$. This section presents computations with probabilities and Probabilistic models.

5.1.1 Computations with Probabilities

Probability theory is the mathematical study of random phenomena. As a mathematical foundation for statistics, probability theory is essential to many human activities that involve quantitative analysis of data. Methods of probability theory also apply to description of complex systems giving only partial knowledge of their state [47, 113]. For instance, the insurance industry and markets use actuarial science to determine pricing and to make trading decisions [47]. In this section, some important definitions in the field of probability theory and mathematical statistics will be presented that are relevant for the further presentation of HMMs.

Definition 5.1 (Probability)**.** *Probability is quantified as a number between 0 and 1, where 0 indicates impossibility and 1 indicates certainty (formula 5.1). The higher the probability of an event, the more likely it is that the event will occur [47, 113].*

$$P : \begin{cases} \Omega \longrightarrow [0,1] \\ \omega \longrightarrow P(\omega) \\ 0 \leq P(\omega) \leq 1 \end{cases} \tag{5.1}$$

Definition 5.2 (Random experiment)**.** *A random experiment describes a procedure which can be repeated arbitrary often and produces a random result from a well defined set of possible outcomes [47, 113].*

Definition 5.3 (Random event)**.** *A random event (also called elementary event) is a single result or a set of potential results of a random experiment [47, 113].*

Definition 5.4 (Sample space)**.** *A sample space (also called universe) is the complete set of all possible results of a random experiment.*

Sample space is noted by Ω and ω refers to the elements of the space. The probability of the entire sample space is 1, and the probability of the null event is 0. The usual set of operations (conjunction, disjunction and complement) are generally applied to events with respect to the sample space.

Definition 5.5 (Relative frequency)**.** *A relative frequency ($f(A)$) of an event A that occurred n times during a N fold repetition of a random experiment is obtained as the quotient of its absolute frequency $c(A) = n$ and the total number of trials N (formula 5.2).*

$$f(A) = \frac{c(A)}{N} = \frac{n}{N} \tag{5.2}$$

A pragmatic derivation of the notion of probability is directly based on the relative frequency of an event. Then, the occurrence of the event A is defined as its probability, $P(A)$ as its relative frequency. This probability satisfies three axioms:

- The measure of each event is between 0 and 1, written as $0 \leq P(A = a_i) \leq 1$. Where A is a random variable representing an event and a_i are its possible values.

- The measure of the whole set is 1: $\sum_{i=1}^{n} P(A = a_i) = 1$.
- The probability of a union of disjoint events is the sum of the probabilities of the individual event: $P(A = a_1 \cup A = a_2) = P(A = a_1) + P(A = a_2)$, where a_1 and a_2 are disjoints.

Definition 5.6 (Joint probability)**.** *Joint probability (also called intersection) of two event A and B is the event occurring on a single performance of an experiment and is denoted by $P(A \cap B)$. Full joint probability distribution is the joint probability for all the random variables.*

Definition 5.7 (Independence)**.** *The events A and B are statistically independent if the observation of B don't provide information about the occurrence of A and vice versa.*

When events are independent, the joint probability $P(A, B)$ is defined by the formula 5.3.

$$\begin{cases} P(A|B) = P(A), \\ P(B|A) = P(B), \\ P(A, B) = P(A)P(B)(B), \\ P(A \cap B) = \emptyset. \end{cases} \tag{5.3}$$

Independent assertions are usually based on knowledge of the domain and help in reducing the size of the domain representation and the complexity of the inference problem.

In ideal cases, exploiting conditional independence reduces the complexity of representing the exponential joint distribution in linear.

Definition 5.8 (Union)**.** *Union probability of two event A and B (denoted as $P(A \cup B)$) where A and B occurs on a single performance of an experiment is define by $P(A \cup B) = P(A) + P(B) - P(A \cap B)$*

Definition 5.9 (Unconditional or prior probabilities)**.** *Unconditional or prior probabilities (or just "priors") is the degree of belief in propositions in the absence of any other information.*

In fact, most of the time, when calculating probability, some information called evidence has already been revealed.

Definition 5.10 (Conditional probability or posterior probability)**.** *The conditional probability for the occurrence of an event A under the condition of the occurrence of the event B having occurred before is derived from the probability of A and B occurring jointly and the unconditional probability of B (formula 5.4).*

$$P(A|B) = \frac{P(A, B)}{P(B)}, P(B) \neq 0 \tag{5.4}$$

A and B are conditionally independent if $P(B|A) = P(B)$ (or equivalently, $P(A|B) = P(A)$).

Definition 5.11 (Marginalization)**.** *Marginal probability is the probability on a subset of the random variables.*

Marginalization and conditioning turn out to be useful rules for all kinds of derivations involving probability expressions.

Definition 5.12 (Random variables)**.** *Random variables can be seen as a function that takes an elementary event and returns a value. A random variable X on a sample space Ω is a rule that assigns a numerical value to each outcome of Ω. In other words, a function from the set Ω into the set $\mathbf{R}$ real of numbers (see formula 5.5)*

$$X : \Omega \longrightarrow \mathbf{R} \tag{5.5}$$

They are characterized by means of their distribution function:

- *A discrete random variable (e.g., X) is a random variable that takes its value on a countable infinite number of values (e.g., $x_1, x_2, ..., x_N$).*
- *A continuous random variable (e.g., X) is a random variable that takes arbitrary values (e.g., $x \in R$).*

Definition 5.13 (Probabilistic inference)**.** *Probabilistic inference is the computation of posterior probabilities for query propositions giving observed evidence.*

Theorem 5.1 (Chaining rule)**.** *Chaining rule allows the link between a marginal probability and conditional probability. For n random variable, it is given by* $P(X_1, ..., X_n) = \prod_{1}^{n} P(X_{i+1}|X_i, ..., X_1)$

Theorem 5.2 (Bayes' Rule)**.** *Bayes rule (or Bayes' theorem), is derived from chain rule by the equation 5.6.*

$$\begin{cases} P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}, \\ P(X) \neq 0 \end{cases} \tag{5.6}$$

Bayes' theorem allows to compute the posterior probability $P(B|A)$ of event B from the conditional probability $P(A|B)$ by taking into account model knowledge about the events A and B in the form of the associated prior probabilities. Bayes' rule is very useful in practice because there are many cases where one do not have a good probability estimation for $P(X|Y)$, (PY) and $P(X)$ and need to compute $P(Y|X)$.

5.1.2 Probabilistic models

This section presents two main groups of probabilities models (Bayes networks, Dynamic Bayes Networks), how to estimate their parameters, and how they are used.

5.1.2.1 Bayes networks (BNs)

A Bayesian network is a directed acyclic oriented graph in which each node is annotated with quantitative probability information (see figure 24). Its topology is defined by the set of nodes and links-specifying the conditional independence relationships that hold in the domain, in such a way that is made precise shortly. The intuitive meaning of an arrow is typically that X has a direct influence on Y, which suggests that causes should be parents of effects. It is usually easy for a domain expert to decide what direct influence exists in the domain than specifying the probabilities themselves.

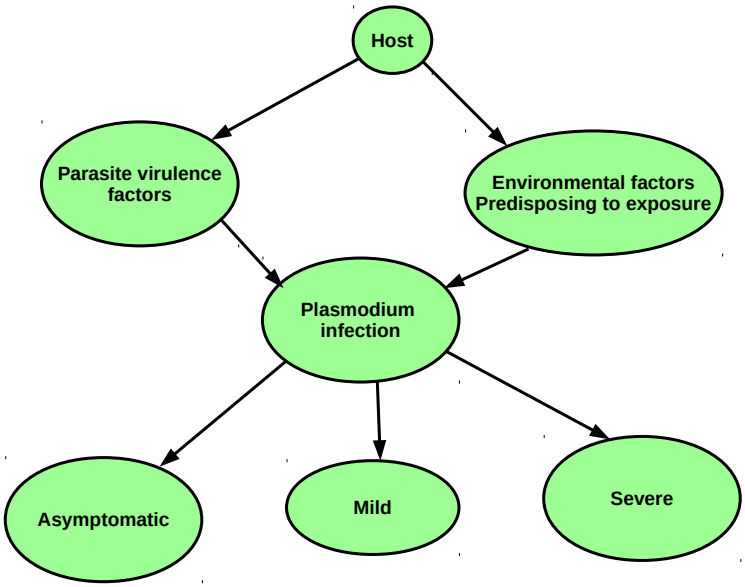


Figure 24: Bayes Network example

Bayes networks are used to represent the probabilistic knowledge of a given application. For example, a clinician's clinical knowledge of causal relationships between diseases and symptoms. They are useful for modelling the knowledge of an expert system or a decision support system, in a situation where causality plays an important role. Pathfinder application is an example of an application based on Bayes Network developed by extracting the expertise of different systems and modelled in a Bayesian network. The predictions were as good as those of the doctors' expertise [113].

One way to define what the network means (its semantics) is to define the way in which it represents a specific joint distribution over all the variables. This is done by defining parameters associated with each node corresponding to conditional probabilities $P(X_i|Parents(X_i))$. The resume of the specification of the Bayesian networks is given by:

- Each node corresponds to a random variable, which may be discrete or continuous;
- A set of directed links or arrows connects pairs of nodes. If there is an arrow from node X to node Y, X is said to be a parent of Y;

- The graph has no directed cycles. It is a directed acyclic graph;
- Each node X_i has a conditional probability distribution $P(X_i|Parents(X_i))$ that quantifies the effect of the parents of the node.

5.1.2.2 Dynamic Bayes Networks(DBNs)

Dynamic Bayes Networks (DBNs) is a Bayes network considering that the situation to be modelled is dynamic (the world can change with time). In this model, the world can be seen as a series of snapshots, or time slices where each of which contains a set of random variables, some observable and others not. The interval between time slices depends on the problem. DBNs are used to model dynamic situations, e.g., situations in which information is collected as time passes. In this case, the random variables are indexed by time and [113]:

- Dynamic changes are seen as a sequence of states in which each state represents a situation at a given time t;
- The random variable is indexed by time where:
  - X_t represents the set of unobservable (hidden) variables describing the state of the modelled environment at time t,
  - E_t represents all the variables observed (evidence) at time t.

In DBNs, dynamic changes are caused by a so-called Markovian process. In fact, in these networks, the current state depends only on a finite number of previous states. For example, in a first-order Markov process, the hidden state at time $t-1$ determines the hidden state at time t. The hidden state at time t is independent of other hidden states.

By resuming, DBNs is a Bayesian network in which the system is considered by considering:

- $P(X_0)$ which specifies how everything get started (the prior probability distribution at time 0);
- $P(X_{t+1}|X_t)$ called the transition model, which specifies the conditional distribution $P(X_t|X_{t-1}, X_{t-1}, ...)$. In other words, a state provides enough information to make the future conditionally independent of the past, that is $P(X_t|X_{0:t-1} = P(X_t|X_{t-1})$.
- $P(E_t|X_t)$ called the sensor model or the observation model, which specifies the evidence variables E_t which could depend on previous variables as well as the current state variables.

There exist specific types of DBNs:

- Markov Chain is a special type of DBN having a finite set of states, and only the current state influences where it goes next [47]. For example, in the source code, each source file can be modelled by a sequence of words.

- Hidden Markov Models are special types of DBN in which the state of the process is described by a single discrete random variable (which is considered to be hidden) and the possible value of the variable are the possible states of the world (observations). HMM is detailed in section 5.2.

5.1.2.3 Inferences in temporal models

The basic task for any probabilistic inference system is to compute the posterior probability distribution for a set of query variables, given some observed events. In the context of DBNs, from the belief state and a transition model, the task is to predict how the world might evolve in the next time step and to update the belief state. In the next paragraphs, we will present the main inference task that must be solved in DBNs.

- **Filtering (state estimation):** The purpose of filtering is to calculate the belief state, that is, the posterior distribution of the most recent hidden variable ($P(Xt|e1 : t)$). For example, what is the probability that the word the programmer will enter now is "$public$".
- **Prediction:** The purpose of prediction is to calculate the posterior distribution on a future state given all evidence to date ($P(X_{t+k}|e_{1:t})$where $k > 0$). For example, what is the probability that the tenth word that will be entered by the programmer is "int".
- **Smoothing:** Smoothing consists of calculating the posterior distribution on a past state given evidence up to the present, that is $P(X_k|e_{1:t})$where$0 \leq k \leq t$. For example, what is the probability that the word "$final$" has been entered by the programmer.
- **Most likely explanation:** The purpose of the most likely explanation is to find the sequence of states that best explains the observations ($argmax_{1:t}P(X_{1:t} \leq e_{1:t})$). For example, what is the set of keywords in a program.

5.1.2.4 Learning temporal models

Learning probabilistic models is to estimate the model parameters by taking into account the specified model structure. In the particular case of temporal models, the task of learning is to determine the transition and sensor models. This task can be done by using data or by using data and a specialized algorithm [113].

- Learning on data: Parameters required can be learned from sample data. Statistical parameter estimation methods provide reliable results with sufficiently many training samples data;
- Using a specialized algorithm such as Baum-Welch algorithm, the Baldi-Chauvin algorithm, Segmental k-Means Algorithm, Viterbi training algorithm [47, 113].

5.2 Hidden Markov Models (HMMs)

Hidden Markov Models are particular types of Markov Chain composed of a finite state automaton with edges between any pair of states that are labeled with transition probabilities. It also describes a 2-stage statistical process in which the behavior of the process at a given time t is only dependent on the immediate predecessor state. It is characterized by the probability between states $P(q_t|q_1, q_2, ..., q_{t-1}) = P(q_t|q_{t-1})$ and for every state at time t an output or observation o_t is generated. The associated probability distribution is only dependent on the current state q_t and not on any previous states or observations: $P(o_t|o_1, ..., o_{t-1}, q_1, ..., q_t) = P(o_t|q_t)$ [43, 47, 49, 76, 120].

A first order HMM perfectly describes the source code because it can be seen as a string sequence typed by a programmer in which the current word (corresponding to an assigned hidden state) depends on the previous word. In this HMM, the observed symbol depends only on the current state [47, 113, 120]. Equation 5.7 presents the joint probability of a series of observations $O_{1:T}$ given a series of hidden states $Q_{1:T}$. The HMM of figure 30 shows how the source code can be modeled using a HMM. In this figure, the observations are the words ("public", "class", "Patient", etc.) typed by the programmers and each of these words are labeled by the hidden states "PRE", "TARGET", "POST", and "OTHER".

$$P(O_{1:T}, Q_{1:T}) = P(q_1)P(o_1|q_1)\prod_{t=2} P(q_t|q_{t-1})P(o_t|q_t) \tag{5.7}$$

Filtering, smoothing, prediction, and the most likely explanation are four uses of HMMs. The probability that a string O is emitted by a HMM M is calculated as the sum of all possible paths by the equation 5.8.

$$P(O \mid M) = \sum_{q_1,...,q_l} \prod_{k=1}^{l+1} P(q_{k-1} \to q_k)P(q_k \uparrow o_k) \tag{5.8}$$

Where q_0 and q_{l+1} are limited to q_I and q_N respectively and o_{l+1} is an end of word. The observable output of the system is the sequence of symbols emitted by the states, but the underlying state sequence itself is hidden.

In the most likely explanation, the goal is to find the sequence of hidden states $V(O \mid M)$ that best explains the sequence of observations (equation 5.9) [47, 113, 120]. To this end, the sequence of states $V(O \mid M)$ which has the greatest probability to produce an observation sequence is searched.

For example, in automatic translation, one may want the most probable string sequence that corresponds to the string to be translated. In this case, instead of taking the sum of the probabilities, the maximum must be chosen (equation 5.9).

$$P(O \mid M) = max_{\ q_1 \ldots q_l \in Q^l} \prod_{k=1}^{l+1} P(q_{k-1} \rightarrow q_k) P(q_k \uparrow o_k) \tag{5.9}$$

Before using the model, its parameters (transition probabilities, emission probabilities and initial probabilities) must be calculated using statistical learning or specialized algorithms [47].

5.2.1 HMMs structures

In the main application areas of HMM-based modeling, the input data to be processed have a chronological or sequential structure. One assumes that the models are run through in causal chronological sequence and, therefore, the model states can be arranged sequentially. Transition probabilities to states that describe data segments lying backwards in time are constantly set to zero. In graphical representations of HMMs (see figures 25, 26, 27, 28), such edges which are excluded from possible state sequences are omitted for the purpose of simplification. The diagram of figure 25 shows the general architecture of an instantiated HMM. Each oval shape represents a random variable that can adopt any of a number of values. The random variable $x(t)$ is the hidden state at time t (with the model from the above diagram, $x(t) \in \{x1, x2, x3\}$). The random variable $y(t)$ is the observation at time t (with $y(t) \in \{y1, y2, y3, y4\}$). The arrows in the diagram denote conditional dependencies. For HMMs to be applied for the analysis of data that is already available, one must first assume that the data to be analysed was generated by a natural process which obeys similar statistical regularities. Then one tries to reproduce this process with the capabilities of HMMs as closely as possible. If this attempt is successful, inferences about the real process can be drawn on the basis of the artificial model.

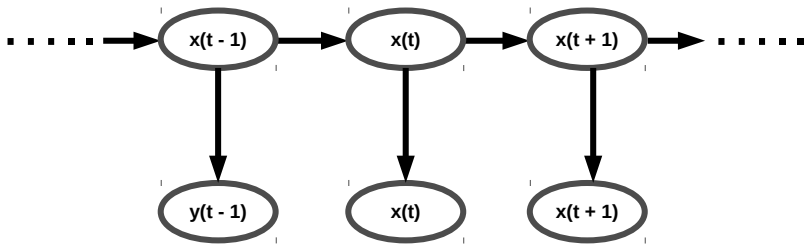


Figure 25: General architecture of HMMs

There are many types of HMMs [47]:

- **Linear HMMs (figure 26)** are the most simple models in which only transitions to the respective next state and to the current state itself are possible with some positive probability.

This model is used to capture variations in the temporal extension of the patterns described with the help of the self-transitions.

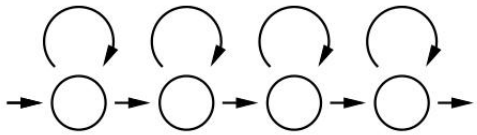


Figure 26: Linear HMM

- **Bakis models (figure 27)** are models in which the modeling of duration is achieved if the skipping of individual states within a sequence is possible. This model is widely used in the field of automatic speech and handwriting recognition.

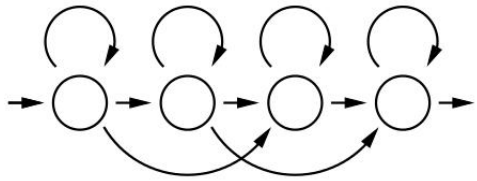


Figure 27: Bakis HMM

- **Left-to-right models (figure 28)** are used to model larger variations in the temporal structure of the data.

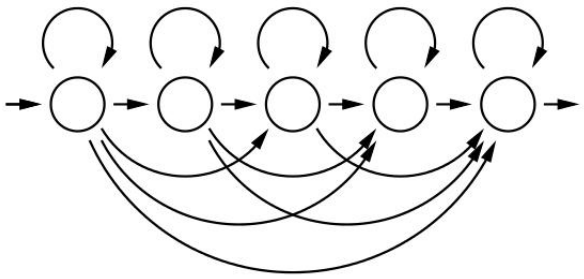


Figure 28: Left-to-right HMM

- **Ergodic model (figure 29)** are models having a completely connected structure.

Several inference problems are associated with hidden Markov models: filtering, prediction, smoothing, and the most likely explanation [47, 113].

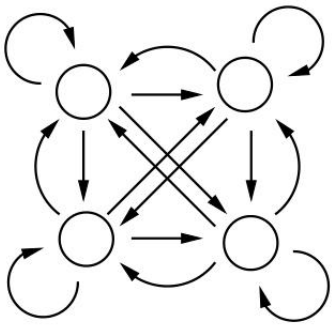


Figure 29: Ergodic HMM

5.2.2 Parameters estimations

Parameter estimations consist of finding, given an output sequence (or a set of such sequences), the best set of state transition and emission probabilities. The task is usually to derive the maximum likelihood estimation of the parameters of the HMMs given the set of output sequences. This task can be done by training the model on a dataset using statistical learning or using specialized algorithms such as Baum-Welch algorithm, the Baldi-Chauvin algorithm, Segmental k-Means Algorithm, EM algorithm, Viterbi training algorithm [47, 113].

Statistical parameter estimation methods provide reliable results with sufficiently many training samples data. Powerful HMMs can thus be created only if sample sets of considerable size are available for the parameter training. Moreover, only the parameters of the models and not their configuration (e.g., the structure and the number of free parameters) can be determined automatically by the training algorithms. Considered intuitively, the parameter estimation methods for HMMs are based on the idea to "observe" the actions of the model during the generation of an observation sequence. The original state transition and output probabilities are then simply replaced by the relative frequencies of the respective events.

5.2.3 HMMs usage

Hidden Markov Models are applied in many fields where the task is the modelling and analyzing of chronological organized data as, for example, genetic sequences, handwriting texts, automatic speech recognition. In the following paragraphs, we are going to present some successful applications for music genre classification, speech and handwriting recognition.

5.2.3.1 Music genre classification

Musical genres are labels used to distinguish between different types or categories of musical style. The growing amount of music available creates a need for automated classification. This task can be done by assigning a genre according to the listening impression. Music can be considered as a

high-dimensional digital time-variant signal, and music databases can be very large. As music is a time-varying signal, several segments can be employed to extract features in order to produce a set of feature vectors that characterizes a decomposition of the original signal according to the time dimension. Then, HMMs can be used for music genre classification [67]. Iloga and al. proposed an approach based on HMMs that represent each genre with a state, the model statistically captures the transitions between genres. This approach is used to classify music genres by modelling each genre with one HMM [67].

5.2.3.2 Speech recognition

In automatic speech recognition, the output of the models corresponds to a parametric feature representation extracted from the acoustic signal. In contrast, the model states define elementary acoustic events (speech sounds of a certain language). Sequences of states then correspond to words and complete spoken utterances. If one is able to reconstruct the expected internal state sequence for a given speech signal, then hopefully the correct sequence of words spoken can be associated with it and the segmentation and classification problem can be solved in an integrated manner. The possibility to treat segmentation and classification within an integrated formalism constitutes the predominant strength of HMMs. When decomposing models for spoken or written words into a sequence of sub-word units, we implicitly assumed that more complex models can be created from existing partial HMMs by concatenation. Such construction principles are either explicitly or implicitly applied in order to define compound models for different recognition tasks. There are many speech recognition systems based on HMMs [47]. For instance the speech recognition system of RWTH Aachen University [47]; ESMERALDA development environment for pattern recognition [47].

5.2.3.3 Handwriting recognition

In the field of handwriting recognition, the signal data considered can be represented as a linear sequence of words written. The temporal progress of the writing process itself defines a chronological order of the position measurements which are provided by the respective sensors. The time-line of the signal thus virtually runs along the trajectory of the pen. The classical application of automatic processing of writing is called Optical Character Recognition (OCR). The goal is to automatically transcribe the image of the writing into a computer internal symbolic representation of the text. Many systems based on HMMs are developed to address the problem of hand writing recognition [47]: Ratheon BBN Technologies [47]; RWTH Aachen handwriting recognition system [47]; ESMERALDA Offline HWR Recognition System [47].

5.3 Source code

During software development, it is recommended to write the source code according to good programming practices, including naming conventions [19]. These practices inform programmers on

how to name variables, organize and present the source code. This organization can be used to model source code using HMMs. For example, from Java source code, we can say that at a time t, the programmer enters a word (e.g. "public" at the beginning of a Java source file). Thus, the keyword "public" at time t conditions the next word at time $t+1$ which in this case can be "class", "int", etc. We can say that PRE and $TARGET$ are the hidden states and "public" and "class" are respectively their observations.

5.3.1 Source code description

Source code contains several types of files: files describing data, files processing data, user interface files and configuration files.

5.3.1.1 Files describing data

These files describe the data to be manipulated and equally, some constraints on this data (e.g., data types). In Java EE for example, there are entities whose names are close to the terms of the domain that will be transformed into tables in the database. These files often contain certain rules to verify the reliability of the data. Thus, from these files, we can retrieve concepts, properties, axioms and rules.

5.3.1.2 Files containing data processing

Located between user interface files and data description files is the data processing files of the source code consisting of:

- **Control:** For example, restricting certain data from certain users (e.g., only the attending physician has the right to access the data), checking the validity of a field (checking whether the data entered in an "age" field is of type integer);
- **Calculation:** For example, converting a date of birth into an age, determining the date of the next appointment of a patient, calculating the body mass index of a patient based on his/her weight and height.

These are the algorithms implementing the business rules to be applied to the data. They are thus good candidates for axioms and rules extraction.

5.3.1.3 User interfaces files

The User interfaces are composed of files which describe the information that will be presented to users for data viewing or recording. Unlike the first two file types, these files contain the words of a human-readable vocabulary that can be found in a dictionary. User interfaces usually provide:

- Translations allowing navigation from one language to another, control for users to enter the correct data;
- An aid allowing users to know for example, the role of a data entry field.

User Interfaces are therefore good candidates for concepts and their definitions, properties, axioms and rules extraction.

5.3.1.4 Configuration files

These files allow developers to specify certain information such as the type and path of a data source, different languages used by users, etc. For instance, from these files, the languages labels (e.g. English, French, Spanish) for terms can be extracted.

The files we just presented generally contain comments that can be useful for knowledge extraction or ontology documentation. Knowledge extraction from user interfaces/web interfaces has already been addressed in [26, 144], knowledge extraction from text has been presented in [4, 5, 23, 30]. In this chapter, we will focus on knowledge extraction from files describing data and their processing.

5.3.2 Modelling source code using HMMs

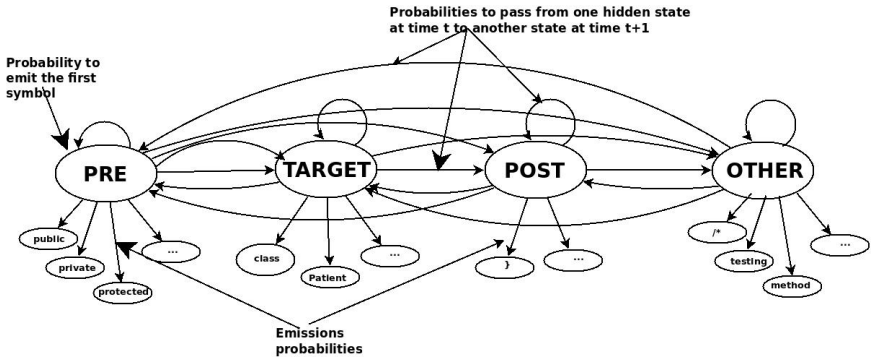


Figure 30: An example of HMM modeling the Java source code

A first order HMM perfectly describes the source code because it can be seen as a sequence typed by a programmer in which the current word (corresponding to an assigned hidden state) depends on the previous word. In this HMM, the observed symbol depends only on the current state [47, 113, 120]. Formula 5.7 presents the joint probability of a series of observations $O_{1:T}$ given a series of hidden states $Q_{1:T}$. The HMM of figure 30 shows how the source code can be modelled using a HMM. In this figure, the observation states are the words ("public", "class", "Patient", etc.) typed by the programmers and each of these words are labeled by the hidden states "PRE", "TARGET", "POST", and "OTHER".

The probability that a word X is emitted by a HMM M is calculated as the sum of all possible paths by the formula 5.8.

Where q_0 and q_{l+1} are limited to q_I and q_N respectively and x_{l+1} is an end of word. The observable output of the system is the sequence of words in the source code files and emitted by the states, but the underlying state sequence itself is hidden.

In the most likely explanation, the goal is to find the sequence of hidden states (or source code labels) $V(X \mid M)$ that best explains the sequence of observations (words in the source code-formula 5.9) [47, 113, 120]. To do this, the sequence of states $V(X \mid M)$ which has the greatest probability to produce a sequence of source code is searched [47, 113, 120].

5.4 Ontology Learning from Source Code

The previous sections present how the source code can be modelled using HMMs. This section presents how the HMMs can be modelled, trained and used to extract knowledge from Java source code. The experimentation is made on EPICAMTB, the epidemiological surveillance platform presented in chapter 3 and all the source code used are available on github[1] Then, sections 5.4.1, 5.4.2, 5.4.3, 5.4.4 will present the approach we proposed for knowledge extraction from source code, the definition and training of HMMs for knowledge extraction from Java source code, the extraction of knowledge from EPICAMTB source code and the evaluation of knowledge extracted respectively.

5.4.1 An approach based on HMMs for ontology learning from source code

To extract knowledge from Java source code, we designed a method divided into five main steps: data collection, data preprocessing, entity labeling, formal language translation, and knowledge validation.

5.4.1.1 Data collection and preprocessing

This section presents the first and the second step of the approach which are data collection and data preprocessing.

Data collection. The data collection step consists of the extraction of a dataset necessary for the next steps. In Java files, statements for importing third-party libraries and comments are deleted. We proposed the definition of a regular expression that allows them to be identified.

Data preprocessing. The purpose of data preprocessing is to put data in a form compatible with the tools to be used in the next steps. During this phase, potentially relevant knowledge will be

[1] `https://github.com/jiofidelus/source2onto/`

identified and retrieved, and some entities will be re-coded. The problem of extracting knowledge from the source code has been reduced to the problem of syntactic labeling. This is to determine the syntactic label of the words of a text [113]. In our case, it will be a matter of assigning a label to all the words of the source code and extracting the words marked as target words. This problem can be solved using HMMs [113, 120]. In the following paragraphs, we will first present the HMM structure for source code modelling. Then, we will show how this HMM is trained and finally, how it is used to extract the knowledge from Java source code.

HMMs structure definition. To define the structure of the HMMs, we manually studied the organization of the source code of Java language. Generally, data structures, attributes, and conditions are surrounded by one or more specific words. Some of these words are predefined in advance in the programming language. To label the source code, we have defined four labels, corresponding to four hidden states of the HMM:

- **PRE:** Corresponding to the preamble of the knowledge. This preamble is usually defined in advance;
- **TARGET:** The target, (i.e. the knowledge sought) may be preceded by one or more words belonging to the PRE set. The knowledge we are looking for are the names of classes, attributes, methods, and the relationships between classes. They are usually preceded by a meta-knowledge which describes them. For example, the meta-knowledge "class" allows for concept identification;
- **POST:** Any information that follows the knowledge sought. In some cases, POST is a punctuation character or braces;
- **OTHER:** Any other word in the source code that neither precedes nor follows the knowledge sought.

An example of HMM annotated with labels is given by Fig. 30. Concepts, properties, axioms, and rules are usually arranged differently in the source code. We propose the definition of two HMMs which permit them to be identified: one to identify concepts, properties, axioms and the other one to identify rules.

Learning Model Parameters. There are several techniques to determine the parameters of a HMM: Statistical learning on data, specialized algorithms such as Baum-Welch or Viterbi training [47, 113]. In this paper, we have chosen statistical learning on data to train the HMMs modelled in the previous paragraphs. Thus, we assumed that we have access to T source code files labeled f_t knowing that f_t is not just a sequence of words, but a sequence of words pairs with the word and its label (see figure 30) modelled by the equation 5.10. To train the model, we assume that we can define the order in which the different words are entered by the programmer. We assume that before entering the first word, the programmer reflects on the label of that word and as a function of it, defines the label of the next word and so on. For example, before entering the word $public$, the programmer knows that its label is PRE and that the label of the next word is $TARGET$. Thus, the current word depends only on the current label, the following label depends on the previous label, and so on. The process continues until the end of the file.

$$\begin{aligned} f_t &= [(w_1^t, e_1^t), ..., (w_d^t, e_d^t)], \\ words(f_t) &= [w_t^t, ..., w_d^t], \\ labels(f_t) &= [e_1^t, ..., e_d^t]. \end{aligned} \tag{5.10}$$

In the equation 5.10, w_i and e_i are words and labels of f_i files respectively. In practice, w_i are words contained in the source code (observations) and e_i are the labels of w_i used as hidden states.

From the training data, we can extract statistics on:

- The first label $P(q_1)$ (equation 5.11). A priori probability that the first label is equal to the word $'a'$ is the number of times the first label in each file of the source code is the word $'a'$ divided by the number of source code files.

$$P(Q_1 = a) = \frac{\sum_t freq(e_1^t = a, f_t)}{T} \tag{5.11}$$

- The relation between a word and its label $P(O_k \mid q_k)$ (equation 5.12). The conditional probability that the k^{th} word is $'w'$, knowing that the label is $'b'$ corresponds to the number of times the word $'w'$ associated with the label $'b'$ in the source code file f_t normalized with the fact that the label $'b'$ is associated with any other word in f_t source code. For example, "Patient" can be a concept, an attribute, but cannot be a rule.

$$P(O_k = w \mid q_k = b) = \frac{\alpha + \sum_t freq((w, b), f_t)}{\beta + \sum_t freq(('*, b), f_t)} \tag{5.12}$$

  To avoid zero probabilities for observations that do not occur in the training data, we added smoothing terms (α and β).

- The relation between the adjacent syntactic labels is $P(q_k \mid q_{k+1})$ (equation 5.13). The probability that q_{k+1} is equal to label $'a'$ knowing that q_k is equal to label $'b'$ (previous hidden state) is the number of times $'a'$ follows $'b'$ in the source code of the training data divided by the number of times that $'b'$ is followed by any other label.

$$P(q_{k+1} = a \mid q_k = b) =$$

$$\frac{\alpha + \sum_t freq(b, a), label(f_t)}{\beta + \sum_t freq(b, *'), label(f_t)} \tag{5.13}$$

To avoid zero probabilities for transitions that do not occur in the training data, we added smoothing terms (α and β).

Let us consider the HMM in Fig. 30. Then, training data to identify concepts and attributes would be: [("public", PRE), ("class", TARGET), ("Patient", TARGET), ("extends", TARGET),

Table 5.1: The initial vector - probability to have a state as the first label

f(PRE)	f(TARGET)	f(POST)	f(OTHER)

Table 5.2: An example of a transition table

States	PRE	TARGET	POST	OTHER
PRE	f(PRE,PRE)	f(PRE,TARGET)	f(PRE,POST)	f(PRE,OTHER)
TARGET	f(TARGET,PRE)	f(TARGET,TARGET)	f(TARGET,POST)	f(TARGET,OTHER)
POST	f(POST,PRE)	f(POST,TARGET)	f(POST,POST)	f(POST,OTHER)
OTHER	f(OTHER,PRE)	f(OTHER,TARGET)	f(OTHER,POST)	f(OTHER,OTHER)

("ImogEntityImpl", TARGET), ("{", OTHER), (...), ("int", TARGET), ("age", TARGET), ...]. Tab. 5.1 presents the initial vector, which is the probability that the first label is PRE, TARGET, POST, or OTHER; Tab. 5.2 presents the transition vector containing the frequencies that a state follows another state; and Tab. 5.3 presents the emission vector containing the frequencies that a state emits an observation.

Knowledge extraction. The model previously defined and trained can be applied to any Java source code in order to identify $TARGET$ elements. It will be necessary to find from the files f_1, ..., f_n, a sequence of states q_1, ..., q_n that is plausible. For this, equation 5.9 will be used to determine the most plausible string sequence. From this string, the hidden states will be identified and the targets (words that are labeled $TARGET$) will be extracted. In our approach, we used the Viterbi algorithm which provides an efficient way of finding the most plausible string sequence of hidden states [48, 135]. The algorithm 1 gives an overview of the Viterbi Algorithm. More details can be found in [47].

Any source code can then be submitted to the HMM trained and a table similar to Tab. 5.10 containing the probability for the hidden states to emit a word from the source code is built.

Re-coding variables. Programmers usually use expressions made up of words from a specific lexicon, sometimes encoded with "ad hoc" expressions, requiring specific processing to assign a new name or a label understandable by humans before using. These words are generally divided into words or groups of words according to the naming conventions of the programming language.

Table 5.3: An example of an observation table

	package	pac	;	public
PRE	f(PRE,package)	f(PRE, pac)	f(PRE,;)	f(PRE,public)
TARGET	f(TARGET,package)	f(TARGET, pac)	f(TARGET,;)	
POST	f(POST,package)	f(POST, pac)	f(POST,;)	f(POST,public)
OTHER	f(OTHER,package)	f(OTHER, pac)	f(OTHER,;)	f(OTHER,public)

	class	patient	...
PRE	f(PRE,class)	f(PRE,patient)	...
TARGET	f(TARGET,class)	f(TARGET,patient)	...
POST	f(POST,class)	f(POST,patient)	...
OTHER	f(OTHER,class)	f(OTHER,patient)	...

Algorithm 1: The Viterbi algorithm [47, 135]

1 $Let\ M = (\pi, A,\ B)\ our\ HMM$
2 $With\ \pi$ the vector of start probabilities, A the matrix of state-transition probabilities, and B the matrix of observation probabilities
3 $Let\ \delta_t(i) = max_{q_1,...,q_{t-1}} P(O_1, ..., O_t, q_1, ...q_{t-1}, q_t = i|M)$
4 **1. Initialization**
5 $\delta_1(i) := \pi_i b_i(O_1) \quad \psi_1(i) := 0$
6 **2. Recursion**
7 For all times $t, t_1, ..., T-1$:
8 $\delta_{t+1}(j) := max_i\{\delta_t(i)a_{ij}\}b_j(O_{t+1})$
9 $\psi_{t+1}(j) := argmax_i\{\delta_t(i)a_{ij}\}$
10 **3. Termination**
11 $P^*(O|M) = P(O, q^*|M) = max_i\delta_T(i)$
12 $q_T^* := argmax_j\delta_T(j)$
13 **4. Back-Tracking of the Optimal Path**
14 for all times $t, t = T-1, ..., 1$:
15 $q_t^* = \psi_{t+1}(q_{t+1}^*)$
16

For example, we can have "PatientTuberculeux" $\rightarrow$ "Patient tuberculeux", "agePatient" $\rightarrow$ "Age Patient", "listeExamens" $\rightarrow$ "liste Examens", etc. Therefore, during the re-coding, these names are separated in order to find their real meaning in human understandable language.

5.4.1.2 Entities labeling and translation into a formal language

After the extraction of knowledge, the two last steps consist of giving labels to all the terms extracted and given these labels, translating the knowledge extracted into a formal language.

Entities labeling. The extraction of relevant terms has yielded knowledge and meta-knowledge. This knowledge and meta-knowledge will permit us to identify to which ontological components they may belong to. For example, the code: "class Patient extends Person int age", submitted to a trained HMM to identify concepts and relations will identify three meta-knowledge ("class", "extends" and "int") that will be used to identify two concepts (Patient and Person), one attribute of type integer and a hierarchical relation between "Patient" and "Person". From the extracted knowledge, two candidates to be concepts are related if one is declared in the structure of the other. One may identify three types of relations:

- ObjectProperty: If two classes 'A' and 'B' are candidates to be concepts and 'b' of type B is declared as attribute of class 'A', then classes 'A' and 'B' are related. The attribute 'b' is an ObjectProperty having 'A' as domain and 'B' as range.
- DatatypeProperty: If a class 'A' is a candidate to be a concept and contains the attributes 'a' and 'b' of basic data types (integers, string, boolean, etc.), then, 'a' and 'b' are DatatypeProperty having the class 'A' as domain;

- Taxonomy (subClassOf): If two classes 'A' and 'B' are candidates to be concepts and the class 'B' extends the class 'A' (in Java, the keyword "extends" is used), then, one can define a taxonomic relation between the classes 'B' and 'A'.

Translation in a formal language. Once all relevant knowledge are identified in the previous phase, they are automatically translated to a machine readable language. We use OWL for concepts, properties and axioms, and SWRL for rules.

5.4.1.3 Knowledge evaluation

After the extraction process, the evaluation phase permits us to know if this knowledge is relevant to the related domain and to conclude on the relevance in using source code as a knowledge source. Given that the knowledge extracted is ontological knowledge, two evaluation techniques will be used: (1) Manual evaluation by human experts in which the knowledge extracted is presented to one or more domain experts who have to judge to what extent these knowledge are correct; (2) The comparison of the knowledge extracted (alignment) to gold standards which will be existing ontologies.

5.4.2 HMMs definition, training and use

To extract knowledge from Java source code, two HMMs have to be defined and trained: a HMM for concepts, properties, and axioms identification, and a HMM for rules identification. All the algorithms for HMMs training and usage have been coded in Java[2].

5.4.2.1 HMM structure for concepts, properties and axioms

The HMM used to identify concepts, properties and axioms is defined by:

1. $PRE = \{public, private, protected, static, final\}$, the set of words that precedes TARGET;
2. $TARGET = \{package, class, interface, extends, implements, abstract, enum, w_i\}$, $\forall i$, $w_{i-1} \in PRE \mid\mid w_{i-2} \in PRE \wedge w_{i-1} \in PRE$, the set of all words that we are seeking;
3. $POST = \{\{, ;, \}\}$, the set of words that follow TARGET;
4. $OTHER = \{w_i\}, w_i \notin PRE, \wedge w_i \notin TARGET, \wedge w_i \notin POST$, the set of all other words.

Each HMM state emitted a term corresponding to a word from the source code. We have seen that the observation emitted by the PRE set can be enumerated. However, the observation of

[2]https://github.com/jiofidelus/source2onto

Table 5.4: The initial vector of the HMM for concepts, properties and axioms extraction

PRE	TARGET	POST	OTHER
0.0	1.0	0.0	0.0

$TARGET$ and $OTHER$ sets cannot be enumerated because they depend on the programmer. Then, we considered $data$ to be all the observations emitted by $TARGET$ and $other$ to be all the observations emitted by $OTHER$. We obtained the HMM presented by an initial vector (e.g., Tab. 5.4) a transition vector (e.g., Tab. 5.5), and an observation vector (e.g., Tab. 5.6).

5.4.2.2 HMM structure for rules

Rules can be contained in conditions. Then, we will exploit the structure of source code to extract the rules. For example, the portion of code (if (agePatient$>$ 21) {Patient = Adult}) is a rule determining whether a patient is an adult or not. It must therefore be extracted.

The HMM to identify the rules is composed of:

1. $PRE = \{"\}", ";", "\{"\}$, the set of words that precede one or more TARGET;
2. $TARGET = \{if, else, switch, w_i\} \mid \exists k, r \in N \mid w_{i-k} \in PRE \wedge wi + r \in POST$: the set of all words that follow PRE and precede $POST$;
3. $POST = \{"\}"\}$, the end of the condition;
4. $OTHER = \{w_i\} \mid w_i \notin PRE, TARGET, POST$: the set of all other words.

We can identify the beginning and the end of a condition represented here by the sets PRE and $POST$ respectively. Note that all the observations emitted by $TARGET$ and $OTHER$ sets cannot be fully enumerated. Therefore, we have considered *data* to be all the observations emitted by $TARGET$, and $other$ to be all the observations emitted by $OTHER$.

5.4.2.3 Statistical learning of the HMMs

Learn Java source code (composed of 59 files and 2663 statements) was downloaded from github[3] and from this source code, we used statistical learning on data to calculate the values of the HMMs parameters[4]. Tabs 5.4, 5.5, 5.6, 5.7, 5.8, 5.9 present the initialization, transition and observation vectors respectively obtained after the training step.

[3]https://github.com/mafudge/LearnJava

[4]https://github.com/jiofidelus/source2onto/blob/master/code2onto-model/src/main/java/cm/uy1/training/HMMTrainingData.java

Table 5.5: Transition vector of the HMM for concepts, properties and axioms extraction

	PRE	TARGET	POST	OTHER
PRE	0.1686	0.8260	0.0027	0.0027
TARGET	0.0008	0.7523	0.2461	0.0008
POST	0.0603	0.0033	0.0234	0.9130
OTHER	0.7364	0.1133	0.0025	0.1478

Table 5.6: Observation vector of the HMM for concepts, properties and axioms extraction

	public	private	protected	static	final	data	{
PRE	0.6417	0.1684	0.0053	0.1124	0.0722	0.0	0.0
TARGET	0.0	0.0	0.0	0.0	0.0	1.0	0.0
POST	0.0	0.0	0.0	0.0	0.0	0.0	0.6678
OTHER	0.0	0.0	0.0	0.0	0.0	0.0	0.0

	;	}	other
PRE	0.0	0.0	0.0
TARGET	0.0	0.0	
POST	0.3256	0.0066	0.0
OTHER	0.0	0.0	1.0

Table 5.7: The initial vector of the HMM for rules extraction

PRE	TARGET	POST	OTHER
0.0	0.0	0.0	1.0

Table 5.8: Transition vector of the HMM for rules extraction

	PRE	TARGET	POST	OTHER
PRE	0.0667	0.7999	0.0667	0.0667
TARGET	0.0010	0.9321	0.0659	0.0010
POST	0.0172	0.0172	0.0172	0.9484
OTHER	0.0072	0.0001	0.0001	0.9926

Table 5.9: Observation vector of the HMM for rules extraction

	{	}	;	if	else
PRE	0.8462	0.0769	0.0769	0.0	0.0
TARGET	0.0	0.0	0.0	0.0185	0.0031
POST	0.0	1.0	0.0	0.0	0.0
OTHER	0.0	0.0	0.0	0.0	0.0

	switch	data	other
PRE	0.0	0.0	0.0
TARGET	0.0010	0.9774	0.0
POST	0.0	0.0	0.0
OTHER	0.0	0.0	1.0

5.4.2.4 Knowledge extraction

Once the HMMs are built, we can apply them to the source code of any Java applications in order to extract the knowledge. To do this, the most likely state sequence (equation 5.9) that produced this source code is calculated. To calculate the most likely state sequence, we have implemented the

Figure 31: An overview of the Java source code of the EPICAM project

Viterbi algorithm [47, 48, 135] in Java[5]. In fact, we have exploited the structure of the HMM in the context of dynamic programming. It consists of breaking down the calculations into intermediate calculations which are structured in a table. An example of the Viterbi table is given by the Tab. 5.10. Every element of the table is being calculated using the previous ones. From this table, the Viterbi path is retrieved by getting the frame with the highest probability in the last column and given this frame, to search all the frames that were used to build it. All the elements whose labels are $TARGET$ are extracted as candidates.

5.4.3 Knowledge extraction from the EPICAM source code

This section presents the experimentation of the approach described in section 5.4.1. This experimentation consists in extracting ontological knowledge from EPICAM source code composed of 1254 Java files and 271782 instructions. Fig. 31 presents a screenshot of some concepts from the EPICAM source code.

5.4.3.1 Knowledge extraction from EPICAM

To extract ontological knowledge from EPICAM source code, we proceeded step by step using the method presented in section 5.4.1.

Data collection The source files of EPICAM platform are composed of statements, imported libraries and comments. Data collection involves removing the imported libraries and comments.

[5]https://github.com/jiofidelus/source2onto/blob/master/code2onto-model/src/main/java/cm/uy1/modelUse/KnowledgeExtractionHMM.java

To this end, we defined the regular expression
$import[\backslash\backslash u0000\backslash-\backslash\backslash ufffff]*?;|\backslash\backslash(.)*\backslash n|(/\backslash\backslash* [\backslash\backslash u0000\backslash-\backslash\backslash ufffff]*?\backslash\backslash* /)$ to identify them. Once identified, we wrote a Java program to delete them.

Data preprocessing Data preprocessing consists in extracting the elements likely to be relevant from the source code and re-coding them if necessary. We have used the HMMs defined and trained in section 5.4.2. These HMMs were applied to the source code of EPICAM by calculating the values of the Viterbi table (see Tab. 5.10). Once the table is built, we searched the Viterbi path by getting the frames with the highest probability in the last column and using this frame, we search all the frames that were used to build it. Once the Viterbi path is identified, all the elements labeled *TARGET* are extracted.

	package	**org.epicam**	**;**	**public**	**...**
PRE	0	$\alpha(PRE,2)$	$\alpha(PRE,3)$	$\alpha(PRE,4)$	...
TARGET	1	$\alpha(TARGET,2)$	$\alpha(TARGET,3)$	$\alpha(TARGET,4)$	...
OTHER	0	$\alpha(OTHER,2)$	$\alpha(OTHER,3)$	$\alpha(OTHER,4)$	...

	}
PRE	0
TARGET	1
OTHER	0

Table 5.10: The Viterbi table (α table) built using EPICAM source code

Fig. 32 presents the set of candidates for concepts, properties, and axioms identified and Fig. 33 presents the set of candidates for rules identified.

Re-coding terms and rules To re-code the candidates extracted, we used Java naming conventions. All the candidates were browsed and for the candidates containing the keywords of the programming language, these keywords were removed. For example, consider the term *CasTuberculoseEditorWorkflow* that was extracted from the source code; the terms *Editor* and *Workflow* are keywords of Google Web Toolkit, the technology used to build the EPICAM platform. Then, the terms *Editor* and *Workflow* are removed and the term *CasTuberculose* is retained as a candidate.

After the re-coding, we moved to the next step which is the translation into formal language.

Entities identification and translation into OWL Data preprocessing phase produced a file containing only the meta-knowledge (e.g "package", "class", "extends", "if", "switch") and the knowledge (e.g "patientManagement.Patient", "Patient" or "serology"). We wrote a Java program to browse these files in order to identify relevant knowledge. Meta-knowledge allows the identification of the candidates as concepts, properties and axioms. For example, if the string "package minHealth.Region.District.hospitals.patientRecord ... class Patient extends Person ... int age ... List<Exam> listExam" is extracted, then, the following ontological knowledge is identified:

- **"package minHealth.Region.District.hospitals. patientRecord:"** This is used to identify the class hierarchy;
- **"class Patient extends Person":** This expression means that "Patient" and "Person" are

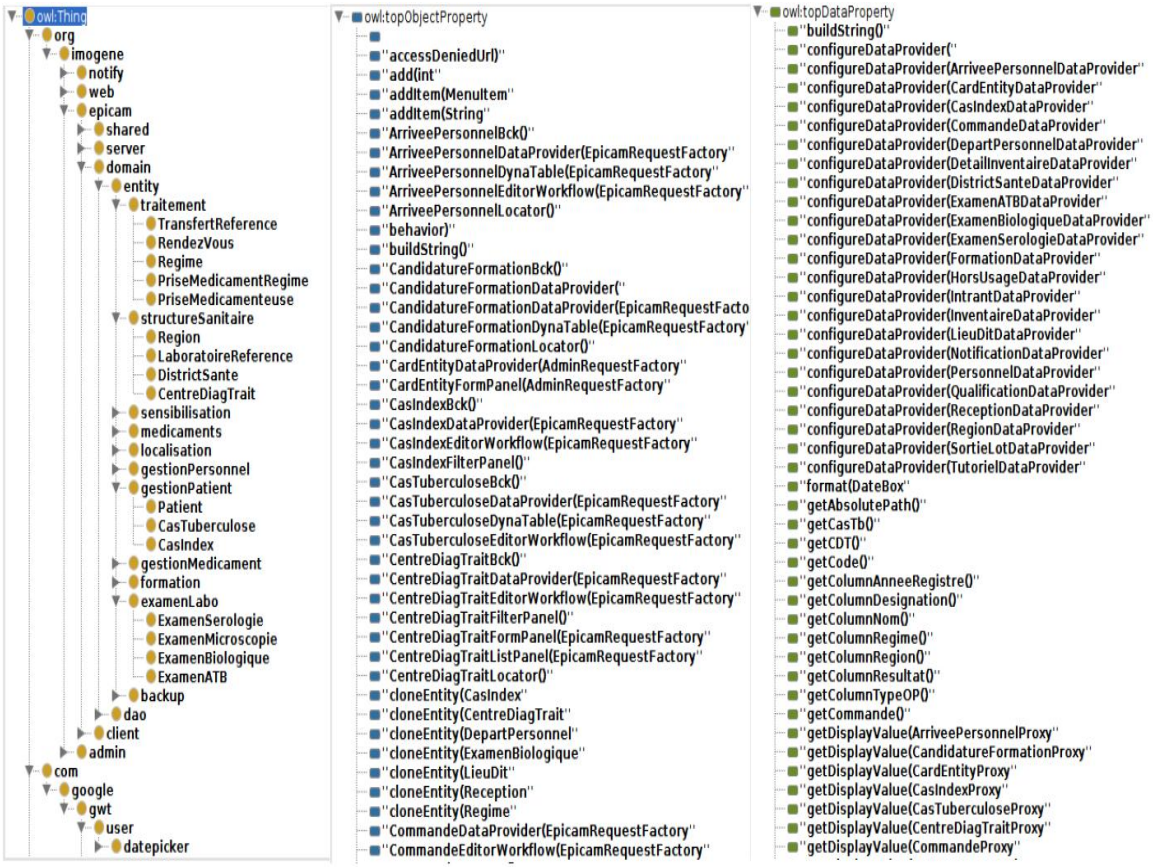


Figure 32: An excerpt of candidates extracted for concepts, properties and axioms

```
if (AccessManager.canDirectAccessPatient()&&AccessManager.canReadPatient()){
Commandcommand=newCommand(){publicvoidexecute(){LocalSession.get().setSearchCriterions
(null,null);History.newItem(TokenHelper.TK_LIST+"/patient/",true);

If(AccessManager.canDirectAccessCasTuberculose()&&AccessManager.canReadCasTuberculose()){
Commandcommand=newCommand(){publicvoidexecute(){LocalSession.get().setSearchCriterions
(null,null);History.newItem(TokenHelper.TK_LIST+"/castuberculose/",true);

if(AccessManager.canDirectAccessExamenATB()&&AccessManager.canReadExamenATB()){
Commandcommand=newCommand(){publicvoidexecute(){LocalSession.get().setSearchCriterions(null,null);
History.newItem(TokenHelper.TK_LIST+"/examenatb/",true);

if(AccessManager.canCreatePatient()&&AccessManager.canEditPatient())patient=
newImogMultiRelationBox<PatientProxy>(patientDataProvider,EpicamRenderer.get(),null);
else patient = newImogMultiRelationBox<PatientProxy>(false,patientDataProvider,EpicamRenderer.get(),null);

if(poidsMin.getValueWithoutParseException()==null&&poidsMin.isValid())delegate.recordError
(BaseNLS.messages().error_required(),null,"poidsMin");//poidsMinshallbesuperiororequalto'0'

switch(typeCAs){case0:nouveauCas++;LOGGER.debug("xxxxxxxxxNombredenouveauxcas:
"+nouveauCas);break;case1:repriseTrait++;LOGGER.debug("xxxxxxxxxNombrederetraitementcas:
"+repriseTrait);break;case2:echecs++;break;case3:rechuttes++;break;default:break;
```

Figure 33: An excerpt of candidates extracted for rules identification

candidates that will become concepts and there is a hierarchical relation between concepts "Patient" and "Person";

- **"int age; List <Exam> listExam":** This expression means that "age" and "listExam" are

properties of the concept "Patient"; the following axiom is also defined: a patient has only a single age (i.e. age is a functional property).

After the identification of entities, we proposed a second Java program[6] to automatically translate them into an OWL ontology[7].

In the same way, rules were also extracted and translated into Semantic Web Rule Language[8]. An example of a rule specifying the rights of a doctor on patient data is given by:
doctorsRule = "Personnel (?pers) ∧ personnel_login (?pers, login) ∧ personnel_passwd (?pers, passwd) ∧ Patient (?p) ∧ RendezVous (?rdv) ∧ hasRDV (?rdv, ?p) ∧ patient_nom (?p, ?nom) ∧ patient_age (?p, ?age) ∧ patient_sexe (?p, ?sexe) ∧ patient_telephoneUn (?p, ?telephone) ∧ rendezVous_dat eRendezVous (?rdv, ?datardv) ∧ rendezVous_honore (?rdv, ?honore) ∧ rendezVous_honore (?rdv, Non) → sqwrl:select (?nom, ?age, ?sexe, ?telephone, ?datardv, ?honore)";

5.4.3.2 Analysis of the elements extracted

The extraction process produced a set of candidates (Figs 32 and 33), but also false positives (Tab. 5.11 presents the statistics). The false positives consist of the set of candidates that belong to the PRE, $POST$ or $OTHER$ sets that normally should not be extracted as observations of $TARGET$. We wrote a Java program to identify and delete them.

Tab. 5.11 presents the statistics of candidates/group of candidates that were extracted. After the extraction process, we obtained different types of candidates/group of candidates:

- **Irrelevant candidates/group of candidates:** These are utility classes and temporary variables. Utility classes are classes that the programmer defines to perform certain operations. These classes usually contain constants and methods. The names of these classes are usually not related to the domain. Temporary variables (e.g., the variables used in a loop) are used temporarily in the source code and are not related to the domain.
- **Relevant candidates/group of candidates:** These are knowledge found. These candidates are composed of synonyms (candidates of identical meaning) and redundancies (candidates that come up several times). We wrote a Java program to identify and remove redundancies candidates automatically.

We also extracted candidates' conditions to be rules. As we did with the candidates to be concepts, properties and axioms, false positives were identified and deleted. From the rules extracted, we found:

[6]https://github.com/jiofidelus/source2onto/blob/master/code2onto-model/src/main/java/cm/uy1/helper/OWLHelper.java

[7]https://github.com/jiofidelus/ontologies/blob/master/epicam/epicam.owl

[8]https://github.com/jiofidelus/ontologies/blob/master/epicam/epicamrules.owl

Candidates	Relevant	Irrelevant
Concepts	1840 (72.87%)	685 (27.13%)
Properties	38355 (81.42%)	8755 (18.58%)
Axioms	3397 (83.22%)	685 (16.78%)
Rules	1484 (07.89%)	17332 (92.11%)

Table 5.11: Statistics on candidates extracted

- **Irrelevant conditions:** These are conditions that are not really important. For example, testing whether a temporary variable is positive or is equal to a certain value. These conditions were the most numerous;
- **Relevant conditions:** Conditions corresponding to a business rule (e.g., testing if a user has access right to certain data).

5.4.4 Knowledge evaluation

The concepts, properties and axioms extracted were translated into an OWL ontology. The extracted rules are represented in SWRL. We used the Protege editor to provide a graphical visualization of the ontology and rules to human experts for their evaluation. Fig. 34 presents an overview of the ontology obtained.

Three experts from the tuberculosis surveillance domain involved in the EPICAM project were invited to evaluate the knowledge extracted. They are from three different organizations in Cameroon (Centre Pasteur of Cameroon, National Tuberculosis Control Program and a hospital in Yaounde). The domain experts were asked to check first if the terms extracted are relevant to the tuberculosis clinical or epidemiological perspectives. Second, they analyzed the axioms and rules. First of all, they found that the terminology was relevant to the tuberculosis. However, they suggested correcting some typos caused by the names of the classes and attributes given by programmers. Axioms and rules were generally correct. Some rules were suggested to be updated as the business rules have evolved (e.g. user access to patient data has been improved taking into account their post such as epidemiologist, physician, nurse or administrative staff).

In line with the experts validation, we evaluated the coverage of the ontology terms by taking reference on other ontologies in the biomedical domain. We used BioPortal [137] as a biomedical ontology repository. BioPortal contains more than 300 ontologies including a large number of medical terminologies such as SNOMED (Systematized Nomenclature of Medicine) [123]. BioPortal has an Ontology Recommender module that is used to find the best ontologies for a biomedical text or a set of keywords [111]. This task is done according to four criteria: (1) the extent to which the ontology covers the input data; (2) the acceptance of the ontology in the biomedical community; (3) the level of detail of the ontology classes that cover the input data; (4) and the specialization of the ontology to the domain of the input data. We gave as input keywords to the Recommender the set of terms (concepts and properties) of the ontology extracted by our HMM. Fig. 35 shows that the ontology terms are covered by many biomedical ontologies. In the first line of the recommended ontologies, we could see that NCIT, SNOWMEDCT, ONTOPARON (accepted by the

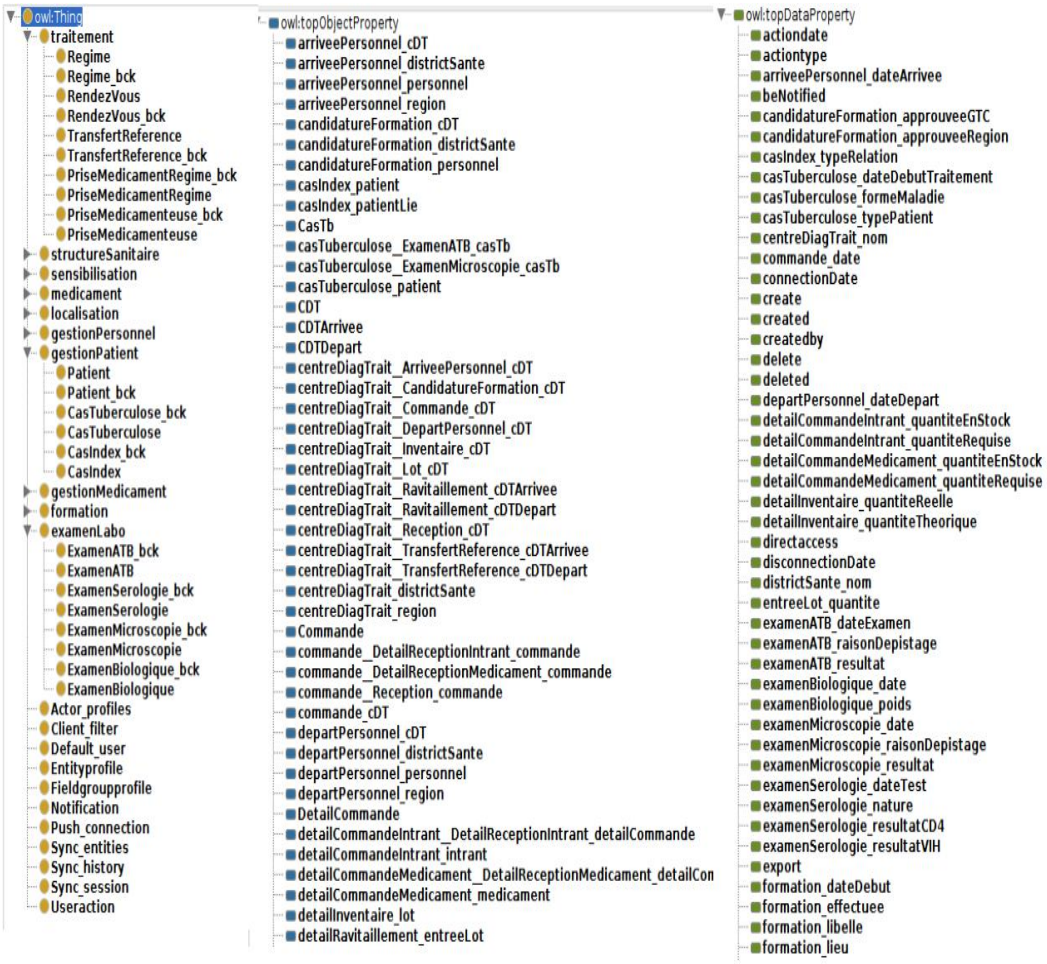


Figure 34: An overview of the generated OWL ontology

community with a score of 75.6%) cover the terms from our ontology with a score of 82.9%, have a level of details of 64% and the level of specialization of 40%. We came to the conclusion that terms extracted by our HMM are relevant to the biomedical domain.

5.5 Conclusion

We proposed in this chapter an approach for knowledge extraction from Java source code using Hidden Markov Models (HMMs). This approach consists of the definition of a Hidden Markov Model, its training and use for knowledge extraction from source code. The HMMs are defined by labeling the source code with the labels PRE, $POST$, $TARGET$ and $OTHER$. Thereafter, they are trained using existing source code and used to extract knowledge. We experimented this approach by extracting ontological knowledge from EPICAM, a tuberculosis epidemiological surveillance platform developed in Java. Evaluation by domain experts (clinicians and epidemiologists) permitted us to show the relevance of the knowledge extracted. In line with the experts' validation, we evaluated the coverage of terms extracted by reference ontologies in the biomedical domain. We used Ontology Recommender from BioPortal repository. The results of the evaluation

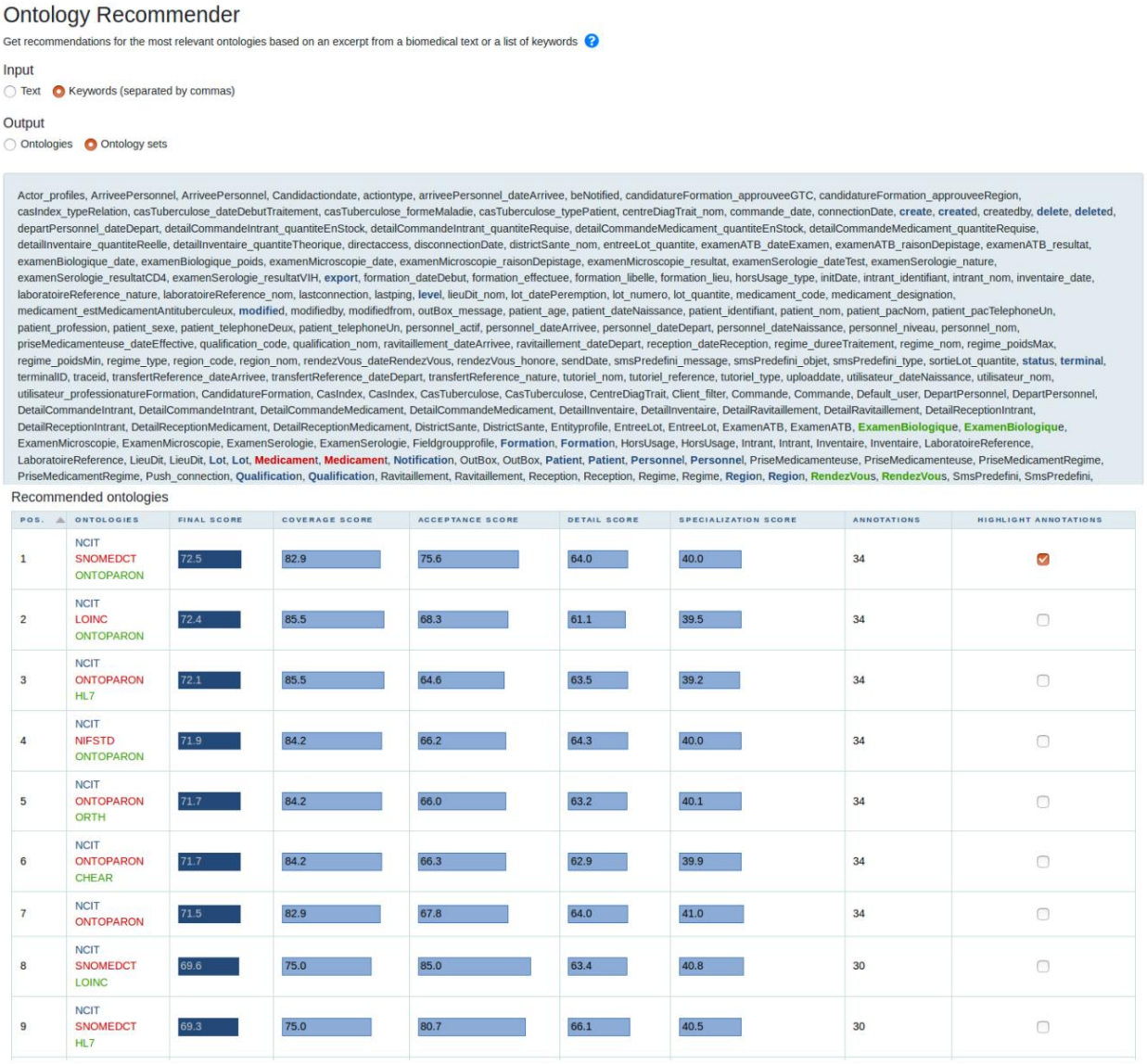


Figure 35: The Ontology Recommender output from the extracted ontology terms

shows that the terms are well covered by many biomedical ontologies (e.g., NCIT, SNOWMEDCT, ONTOPARON). In the chapter 6, we will show how the knowledge extracted in this chapter was used to build an ontology for tuberculosis surveillance.

6 An ontology for Tuberculosis Surveillance System (O4TBSS)

Effective management of tuberculosis requires to put in place a system which provides all needed information to stakeholders. In chapter 3, we have presented the EPICAM platform used for epidemiological surveillance of tuberculosis in Cameroon. This platform permitted the National Tuberculosis Control Program to collect data and obtain the statistics they generally use. The EPICAM platform uses PostgesQL to store data and the SQL language to get information and build statistics tables and graphics. However, a lack of logical and machine-readable relations among PostgresQL tables prevent computer-assisted automated reasoning. The ability to reason, that is to draw inferences from the existing knowledge to derive new knowledge is an important element for modern medical applications [60] such as epidemiological surveillance systems. To support automated reasoning, ontological terms are often expressed in formal logic [60, 82]. In this chapter, we report the development of an Ontology for Tuberculosis Surveillance System (O4TBSS) that will permit users of TB surveillance, by using reasoning mechanisms to derive new knowledge using existing ones. The rest of the chapter is organized as follow: section 6.1 presents the methodology used to construct the ontology, section 6.2 presents the development of the ontology and section 6.3 presents the use cases.

6.1 Ontology development methodology

During the development of the Ontology for Tuberculosis Surveillance System (O4TBSS), we have followed a methodology made up of a set of principles, design activities and phases based on an agile software development methodology [3, 40] presented in chapter 3 and NeOn methodology [126] presented in chapter 4. Our methodology is composed of the Pre-development step presented in section 6.1.1, the Development and the Post-development step presented in section 6.1.2.

6.1.1 The Pre-development step

The Pre-development step involves the specification, the analysis and the design of the application in which the ontology will be integrated. To make the system specifications, the Scrum Team, conducted by the Scrum Master makes an Application Specification Document (ASD). This document contains the users' needs and all the features of the software to develop. The analysis activity uses the ASD to understand the system in order to delineate and identify its features. To do this, we recommend the use of Unified Modelling Language (UML) in order to identify the actors of the system and the use cases that will be executed by these actors. Recall that an actor is any user outside the system who can be a person or another system. He/she uses the system and runs use cases. A use case determines a system functionality and meets a need.

Based on software specifications and analysis, software design specifies how to represent and build the solution. During the design of the software architecture, the different modules and the relations among these modules are defined. If the ontology is necessary, it will be specified in the software architecture and its role will be clearly defined. This step corresponds to scenario 1 of the NeOn methodology in which knowledge engineers make the Ontology Requirements Specification Document (ORSD).

At the end of the Pre-development step, the first version of the application specification, analysis and design is produced. The Product Backlog of the ontology to be built is also produced and a Scrum Meeting will permit us to define the list of tasks to be executed to build the ontology.

6.1.2 The Development and Post-development steps

The goal of the development step is to develop the ontology through repeated cycles (iteratively) and in modules (incrementally), allowing the Scrum Team to take advantage of what was learned during development of earlier versions. The tasks contained in the Product Backlog are organized in many Sprint Backlogs and executed. At each Scrum Meeting, a Scrum Review is made to evaluate the evolution of the development. This step is composed of two main phases: the development of the first version of the ontology and the development of the next versions.

First version The first phase consists of the development of the first version of the ontology given the specifications, the analysis and the design provided by the Pre-development step. It is composed of three activities and proceeds as follows:

1. **Identification of knowledge sources:** During this activity, an inventory of existing knowledge sources (human experts, domain resources, existing ontologies) is made. Firstly, existing ontologies are listed and analyzed. If one of them matches the needs, it is adopted. If not, the resources identified previously must be used to build the ontology. For each resource, determine the method to be used for knowledge acquisition. The method chosen will guide the choice of tool. For example, if existing ontologies are identified as relevant resources, Protege software [93] can be used to build the ontology by importing/merging them.

2. **Knowledge acquisition:** The second activity in the development step is the most critical. Four aspects are to be considered:

   (a) **Acquiring knowledge from domain experts:** Ideally, knowledge must be obtained from domain experts. However, domain experts are not always available for interviews;

   (b) **Acquiring knowledge from existing ontologies:** for each ontology selected during the identification of knowledge sources, a part or the whole ontology can be used. For this task, existing ontologies can be re-engineered or relevant terms can be manually or (semi)automatically extracted. The knowledge obtained can be used to build the ontology by merging/aligning the knowledge extracted;

   (c) **Using domain resources:** When using domain knowledge sources for ontology building, knowledge is manually/automatically extracted from these resources. This is called ontology learning [7, 121];

   (d) **The mixed approach:** The mixed approach consists of the use of existing ontologies, domain resources to acquire the relevant knowledge and build the ontology.

3. **Knowledge representation:** During the knowledge representation activity, the knowledge extracted previously is serialized in a machine readable form. This activity can be composed of: the construction of the ontology which consists of converting the concepts, properties, axioms and rules in a knowledge representation language; the adaptation of the ontology to one or more various languages and cultural communities; and the population of the ontology obtained with instances. After the knowledge representation activity, one obtains a knowledge base which can use the automated reasoning to reason about the knowledge, make inference and infer new knowledge.

After the development of the first version of the ontology, the evaluation is performed. The feedback of the evaluation, presented during the Scrum Meetings will permit us to define the next steps of the ontology development.

The next versions. The second phase is an iterative and incremental phase in which each increment consists of exploiting the evaluation feedback in order to complete specifications, analysis, design and to develop the new versions of the ontology. Each increment involves the Sprint planning meeting which will result in a the set of features that the ontology must meet; knowledge identification consisting of identifying relevant knowledge to be used to complete the ontology constructed during the previous Sprints; knowledge acquisition, which is based on the resources identified and involve the identification of methods and tools for knowledge acquisition; and knowledge representation. At the end of each Sprint, a Sprint Review Meeting permits us to evaluate the ontology given the specifications, analysis and design. Note that at each review, a reasoner is used to check the ontology consistency.

Post-development step. The Post-development step involves the integration of the developed ontology in the related software. For example, a query interface can be developed to allow users to access knowledge.

6.2 Ontology building

The dramatic increase in the use of knowledge discovery applications requires end users to write complex database queries to retrieve information. Such users are not only expected to grasp the structural complexity of complex databases but also the semantic relationships between data stored in these databases. In order to overcome such difficulties, researchers have been focusing on knowledge representation and interactive query generation through ontologies [60, 89, 122]. In clinical practice particularly, Hauer et al. [60] have proved the relevance of the use of ontologies for knowledge discovery. In this thesis, we propose the use of an ontology named Ontology for Tuberculosis Surveillance System (O4TBSS) for knowledge discovery during epidemiological surveillance of tuberculosis. Then, in this section, we will show how the methodology presented in section 6.1 has been followed to develop O4TBSS. Section 6.2.1 will present the Pre-development step and section 6.2.2 will present the development step.

6.2.1 Pre-development

During the Pre-development, the specifications, analysis and design of the application which will integrate the ontology will permit us to determine the need and role of an ontology.

6.2.1.1 Software specifications

To fight against TB, the government of Cameroon has recognized the National Tuberculosis Control Program (NTCP) as a priority program of the Ministry of Health in 2012. The goal of the NTCP is to detect and treat patients with TB and prevent it. To this end, all the stakeholders at the NTCP must have all needed information for decision making. Firstly, we have developed a platform named EPICAM used for the epidemiological surveillance of TB [69].

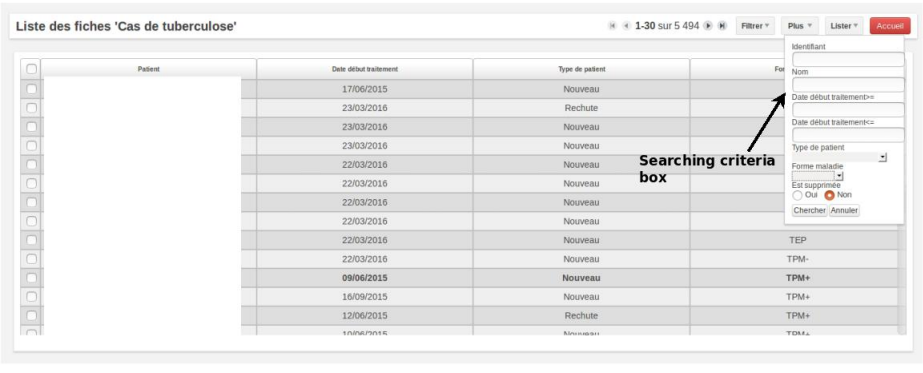


Figure 36: Searching for patients using criteria defined by the NTCP

The EPICAM platform permits the NTCP to obtain data for tuberculosis management. This platform integrates interfaces which allow users to request information. The figure 36 presents

an example of patient search using multiple searching criteria. The EPICAM platform uses PostgreSQL to store data and get information given the search criteria provided by users. However, a lack of logical and machine-readable relations among PostgreSQL tables prevents computer-assisted automated reasoning and useful information may be lost. Then, a new module of the EPICAM platform which enables users access all needed information is required. The main functionalities of this module are:

- Provide all needed information to stakeholders;
- Facilitate the integration of other data sources such as climate and demographic data, in order to establish risk factors;
- Discovering new knowledge from existing. For example, to get the correct answers to the queries like "does patient x be at risk to become TB-MDR," the system must have access to patients' knowledge (e.g, patient characteristics and treatment behaviour) and be able to reason based on this knowledge.

6.2.1.2 Analysis

The new module of the EPICAM software must permit doctors, epidemiologists and decision makers to get access to all the relevant knowledge. The use case these actors will execute is given by the figure 37.

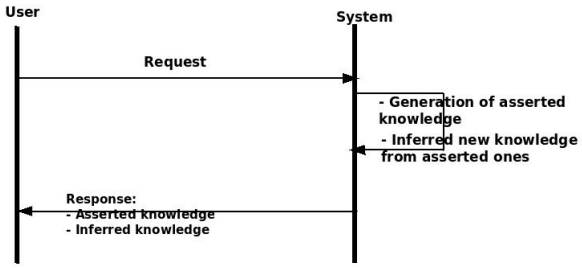


Figure 37: The general use case executed by all users

6.2.1.3 System design

To permit users to have access to all knowledge, the data must be stored using a data structure supporting inferences. As many researchers have proved that ontologies are the best choice for knowledge modelling [85, 122], we have chosen to use an ontology.

The architecture of figure 38 shows how the ontology can be integrated in the existing system. This architecture is composed of two main modules: the EPICAM module [69], which permits stakeholders to obtain tuberculosis data and the OEPICAM module which helps users access information. Note that the EPICAM module is in use. The OEPICAM module is composed of an

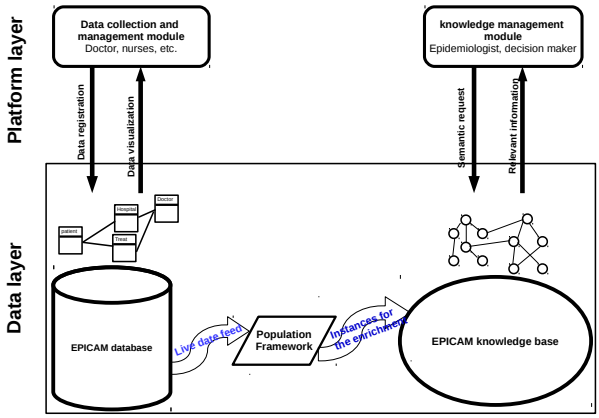


Figure 38: The general architecture presenting the integration of an ontology in the EPICAM platform.

ontology populated with the data extracted from the EPICAM database, an inference system which will be used to infer new knowledge and a user interface which will be used by the users to access information.

6.2.1.4 Product Backlog definition

The product backlog comprises the list of tasks to be executed in order to develop the ontology. They are:

- Identification and evaluation of existing ontologies. This task consists of finding existing ontologies that can be used in the system;
- Identification of domain resources. During the identification of domain resources, existing knowledge sources will be identified;
- Knowledge acquisition from ontological knowledge sources. This task consists of using existing methodologies to acquire knowledge from existing ontologies and domain sources;
- Knowledge representation. After the knowledge is obtained, it is serialized in a machine readable form;
- Ontology population. The ontology obtained after its serialization is populated with instances.

The identification of ontological resources, knowledge acquisition, knowledge representation is based on the NeOn methodology and is done iteratively (in many Sprints) and incrementally (until the ontology fulfills the needs). After the Pre-development step and each Sprint, the Scrum Master organized Scrum Meetings with the Scrum Team composed of the knowledge engineer and the epidemiologist. During these meetings the ontology is evaluated and the Sprint Backlog

containing what to do in the next Sprint is defined. Each evaluation permitted us to determine the ontology consistency using the Pellet reasoner, and to what extent the ontology developed fulfills the requirements.

6.2.2 Development

The O4TBSS was developed in five Sprints.

6.2.2.1 First Sprint: Searching for existing ontologies that fulfilled the need

According to the NTCP, during epidemiological surveillance of TB, the following information are recorded:

- **Patients and their follow-up:** Captures information about patients and the follow-up of their treatment;
- **Symptoms of the disease:** Contains information that can be used to suspect, infirm or confirm that a patient is suffering from tuberculosis;
- **Laboratory testing:** Models the laboratory examinations that confirm or infirm that a patient is suffering from tuberculosis;
- **Epidemiology:** Contains a set of indicators used to provide information for a better monitoring of the disease;
- **Drugs:** Contains information on the medication used for TB treatment;
- **Sensitization:** Captures information on patients and population sensitization;
- **Users:** Captures information on all the persons involve in the surveillance;
- **Training and training materials:** Models the management of the training of health workers and their training materials.

The ontology modelling epidemiological surveillance used by the NTCP must contain all this information. We have conducted a review of existing ontologies using Bioportal [137] and Google's Search Engine. Keywords such as "tuberculosis", "tuberculosis surveillance", "ontology for tuberculosis surveillance" and "tuberculosis ontology" were used to carry out searches. In summary, we proceeded as follows:

- Firstly, we searched for ontologies modelling the TB surveillance on the Bioportal repository using the keywords "tuberculosis" and "tuberculosis surveillance." A total of 38 ontologies were found using the keyword "tuberculosis" and 48 ontologies were found using the keyword "tuberculosis surveillance." These ontologies were examined and all excluded because they did not focus on epidemiological surveillance of tuberculosis.

- Secondly, we used the keywords "ontology for tuberculosis surveillance" and "tuberculosis ontology" to search for existing ontologies using Google's Search Engine. Scientific papers obtained were analyzed. A total of 12 scientific papers were initially identified, 9 of these papers were excluded because they did not focus on tuberculosis ontology and four papers were retained. The first one entitled "A Tuberculosis Ontology for Host Systems Biology" [80] focuses on clinical terminology. The ontology presented has been made available in a csv format; "RepTB: a gene ontology based drug repurposing approach for tuberculosis" [101] focuses on drug repurposing; "An ontology for factors affecting tuberculosis treatment adherence behavior in sub-Saharan Africa" [95] focuses on the factors that influence TB treatment behaviour in sub-Saharan Africa; and "An Ontology based Decision support for Tuberculosis Management and Control in India" [2] which presents the use of an ontology for TB management in India. Although these papers are about ontologies of TB, only one ontology is available for download in a csv format and this ontology covers just the clinical aspects of epidemiological surveillance.

At the end of the first Sprint, we have noted that no existing ontology covers the domain that we want to represent. This justifies the development of a new ontology.

6.2.2.2 Second Sprint: knowledge extraction from EPICAM source code

To develop the new ontology, the first domain resource we used is the EPICAM source code. In fact, source code is any fully executable description of a software designed for a specific domain such as medical, industrial, military, communication, aerospace, commercial, scientific, etc. In the software design process, a set of knowledge related to the domain is captured and integrated in the source code [13, 14, 15].

In a previous work, we extracted knowledge from the source code of the EPICAM platform and used this knowledge to construct an ontology (named ontoEPICAM), modelling epidemiological surveillance of tuberculosis in Cameroon [15]. This ontology is composed of 329 terms with 97 classes, 117 DataProperties and 115 ObjectProperties. Given that this ontology models the epidemiological surveillance system of tuberculosis in Cameroon, it is yet to be evaluated to see if it is complete. That is why, we evaluated this ontology given two criteria: (1) the completeness of the modelled domains, which measures if all the domains covered by epidemiological surveillance are well covered by the ontology; (2) the completeness of the ontology for each domain involved in the epidemiological surveillance, which measures if each domain of interest is appropriately covered in this ontology.

The keywords were identified from ontoEPICAM terms and used to carry out searches of existing ontologies on Bioportal repository and Google's Search Engine. The ontologies found were examined using the browsing tool integrated in Bioportal. The figure 39 is an example of browsing the "Human Disease Ontology". We found 275 ontologies. For each term, we noted the list of ontologies obtained. For the ontologies found in the BioPortal repository, the BioPortal ontology visualization tool was used to visualize the terms that are presented in the ontology. If an ontology contains the relevant terms, it is selected. In many cases, two ontologies have the same terms when searching using certain keywords e.g., "patient", "doctor", "nurse", "tuberculosis", etc.

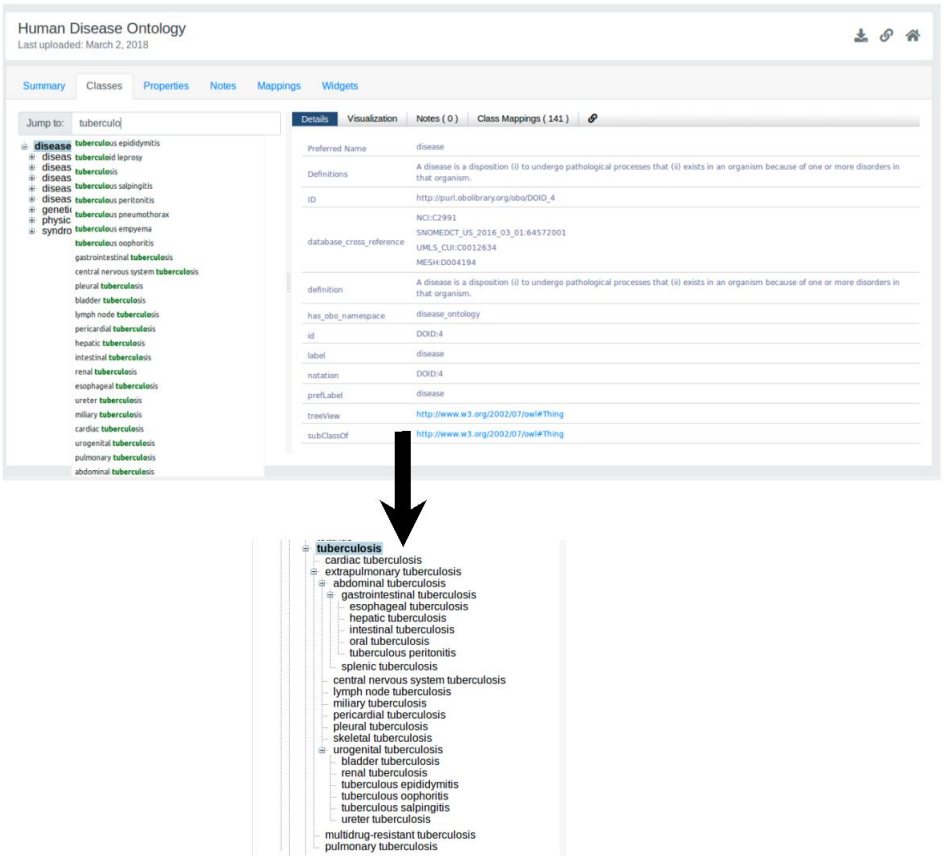


Figure 39: Example of browsing Human Disease Ontology (DOID) using Bioportal visualization tool

Then, the most complete were selected. The ontologies not present in the BioPortal repository were examined using Protege. The ontology in csv file was examined using LibreOffice Calc. The table 6.1 presents the ontologies selected for our purpose. In this table, the keyword column presents the keywords that were used to find the ontology. The ontology column presents the ontology selected given the keyword. The covered domain column presents the domain of epidemiological surveillance covered by the ontology and the description column presents a brief description of the ontology

In the table 6.2, presenting the comparison of selected ontologies with the EPICAM ontology: patients information, characteristics and Follow-up is represented by "Patient"; symptoms of the disease is represented by "Symp"; training and training materials is represented by Training. Comparing the ontologies selected using ontoEPICAM terms (see table 6.2), we found that only ontoEPICAM takes into account all the aspects of epidemiological surveillance. However, by considering the completeness of each domain covered by epidemiological surveillance, we remarked that the ontologies selected are more complete. For example, information about patient follow-up is more complete in Mental Health Management Ontology (MHMO) than in ontoEPICAM. In the

Keywords	Ontology	Covered domains	Ontology description
Epidemiological surveillance	Epidemiology Ontology (EPO)	Epidemiology	This is an ontology describing the epidemiological, demographics and infection transmission process [103].
Tuberculosis symptoms	Symptom Ontology (Symp)	Tuberculosis sign and symptoms	Symp aims to understand the relationship between signs and symptoms and capture the terms relative to the signs and symptoms of a disease [1].
Tuberculosis	Human Disease Ontology (DOID)	Patients and their follow-up, Epidemiology	Human Disease Ontology is an ontology that represents a comprehensive hierarchically controlled vocabulary for human disease representation [118].
Tuberculosis ontology	A Tuberculosis Ontology for Host Systems Biology	Patients, symptoms, laboratory testing	Tuberculosis Ontology for Host Systems Biology focuses on clinical terminology of tuberculosis diagnosis and treatment. It is available in a csv format [80].
Patient	Adherence and Integrated Care[2]	Patient and their follow-up	This ontology is an ontology that defines the medication adherence of patient.
Patient	Presence Ontology (PREO)[3]	Patient and their follow-up	This ontology defines relationships that model the encounters taking place every day among providers, patients, and family members or friends in environments such as hospitals and clinics.
Patient	Mental Health Management Ontology (MHMO)	Patient and their follow-up	The Mental Health Management Ontology is an ontology for mental healthcare management [142]

Table 6.1: The list of ontologies selected for our purpose.

next Sprint, we will show how knowledge has been extracted from the ontologies presented in table 6.1 and combined with ontoEPICAM to build the Ontology for Tuberculosis Surveillance.

Ontologies	Patients	Symp	Lab testing	Epidemiology	Drugs	users	Sensitization	Training
ontoEPICAM	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Epidemiology Ontology	No	No	No	Yes	No	No	No	No
Symptom Ontology	No	Yes	No	No	No	No	No	No
Adherence and Integrated Care	Yes	No	No	No	No	Yes	No	No
Presence Ontology	Yes	No	No	No	No	Yes	No	No
Human Disease Ontology	Yes	No	No	No	No	No	No	No
Mental Health Management Ontology	Yes	Yes	No	No	No	Yes	No	No
A Tuberculosis Ontology For Host Systems Biology	Yes	Yes	Yes	No	Yes	No	No	No

Table 6.2: Comparison of selected ontologies with the EPICAM ontology.

6.2.2.3 Third Sprint: Ontology construction

The ontologies selected in the second Sprint were used to construct the O4TBSS. To do so, ontological knowledge was extracted using either ontofox [140] or Protege. By using ontofox, we specified the source ontology, the classes, the keyword "includeAllChildren" to extract terms of the ontology hierarchy branch and the keyword "includeAllAxioms" to extract all annotations. For the ontologies not presented in ontofox such as "Adherence and Integrated Care ontology", Protege software was used for their examination, identification of irrelevant terms and the deletion of the latter. Knowledge obtained were imported in Protege and examined term by term with the help of an epidemiologist to evaluate each term and identify redundancies. Redundant terms identified were removed. Additional terms were extracted from ontoEPICAM to enrich the ontology obtained. The Pellet reasoner in Protege permitted us to verify the consistency of the ontology obtained. The table 6.3 and the figure 40 respectively present the metric and a part of the ontology obtained.

#	Ontologies	Classes	DataProperties	ObjectProperties	Total
	O4TBSS	865	123	13	1001
1	Epidemiology Ontology	95	0	0	95
2	Symptom Ontology	12	0	0	12
3	Adherence and Integrated Care	246	12	2	260
4	Presence Ontology	205	25	5	235
5	Human Disease Ontology	22	14	0	36
6	Mental Health Management Ontology	143	64	0	217
7	A Tuberculosis Ontology For Host Systems Biology	125	0	0	125
8	ontoEPICAM	17	8	6	31

Table 6.3: O4TBSS terms and terms imported from 7 other ontologies sources and enriched with EPICAM terms

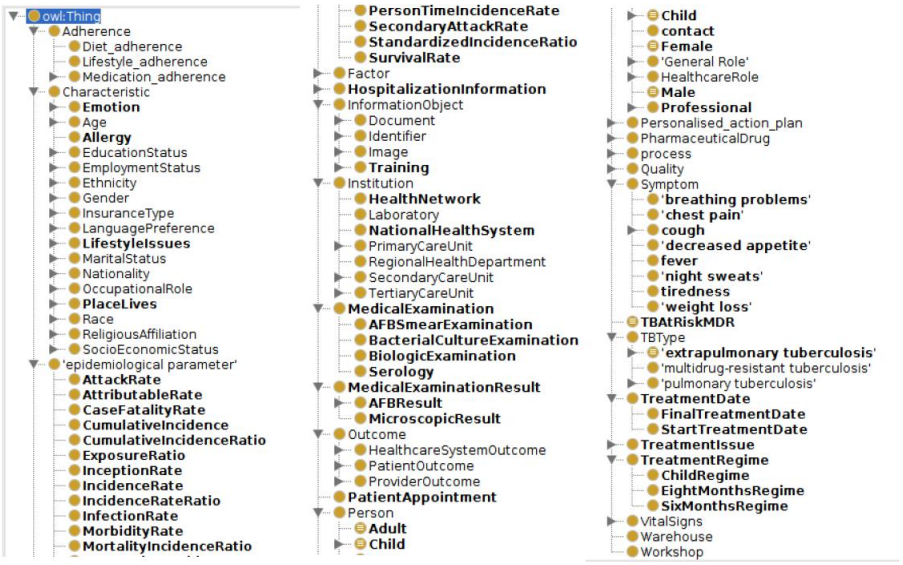


Figure 40: A screenshot of O4TBSS obtained after the third Sprint

6.2.2.4 Fourth Sprint: Ontology enrichment

After building the ontology, we decided to populate it with data gathered from EPICAM database. But we remarked that some data contained in the database can be considered as concepts/property. For example, the occupation of patients was already represented in the ontology as a class with a list of occupations as subclasses. Some occupations (specific to Cameroon like taxi drivers) were not represented in the ontology. Then, with a SQL query, we extracted these knowledge composed of 70 classes and enriched our ontology. The actual version of the ontology is composed of 1068 terms, with 935 classes, 123 ObjectProperties and 13 DataProperties. The complete ontology and the source code write for its population is available on github[4].

6.2.2.5 Fifth Sprint: Ontology population

The purpose of ontology population is to complete the ontology built with instances. It consists of extracting and systematically listing all the instances contained in the database that can reflect a concept or a relationship of the domain to be modelled. As a matter of fact, we developed an inte-

[4]https://github.com/jiofidelus/ontologies/tree/master/O4TBSS

grator (see the architecture presented by the figure 38) which permits us to import and manage all the data from the database to the ontology in Java. The source code of this integrator is available on github[5]. A flat view of the database was created by making a simple SQL query. This query permits us to gain access to information and the information obtained was populated in the ontology. To keep the relation between the tables in the database, the tuples identification in the database were used as the identification of these instances in the ontology. For example, the TB case with ID "TB-CASE_14f7ee" is linked to its appointment with ID "RDV_14f5e7a" in the database. Then, in the ontology, their identifications will also be "TBCASE_14f7ee" and "RDV_14f5e7a". The complete ontology and the source code write for its population is available on github[6].

6.3 Use cases

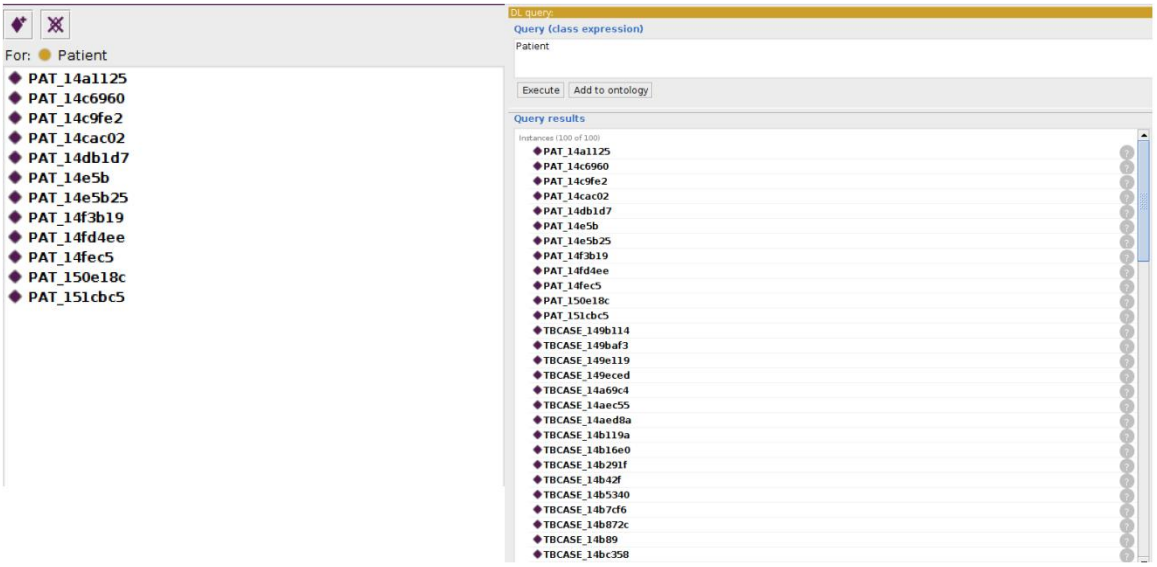
In section 6.2, we presented O4TBSS, an ontology that we built for epidemiological surveillance of tuberculosis. This ontology was developed using OWL and populated with data of epidemiological surveillance of TB in Cameroon. Given that O4TBSS supports OWL-ontological reasoning, this section presents two use cases in which the reasoning mechanism permits us to derive new knowledge from existing knowledge. To proceed these use cases, we populated the ontology with 100 patients among which 88 tuberculosis patients. Then, we used the DL query implemented the DL query tab in Protege software to query the ontology and the Pellet reasoner engine.

6.3.1 Use case 1: inferring patient instances

The first use case on the use of the O4TBSS is the inference about patients who come to hospital with health problems. In fact, the epidemiologist with which we work believes that an extensive set of patient data would reveal subtle patterns if these patterns can be identified. The epidemiologist revealed that all the patients who come for consultation are important in their job because he/she wants to know the characteristics of the patients suffering from TB and the one who does not suffer from TB to see what are the differences between these patients. However, the EPICAM platform does not provide the information on the patients who come for consultation because the main goal of the platform was to follow the TB patients. Given that in the ontology the "TB Patient" is subclass of "Patient", the first use case consists of showing the result that is obtained using an inference system and without an inference system. The figure 41 presents on the left the list of all patients when the inference system is not used and on the right, the list of all patients when the inference is used. The reasoner uses the set of assertions and knowledge accumulated in the ontology to answer semantic queries, in particular inferred patients.

This use case shows that using O4TBSS permits the epidemiologists to identify from a set of patients data the TB patients and non TB patients. The characteristics of these patients can then be used to identify risk factors.

[5]https://github.com/jiofidelus/ontologies/tree/master/O4TBSS
[6]https://github.com/jiofidelus/ontologies/tree/master/O4TBSS/populatingO4TSS



The list of all patients without inference **A part of the list of patients with inference**

Figure 41: Request of the list of patients without the inference system (on the left) and using the inference system (on the right)

6.3.2 Use case 2: automatic detection of TB-MDR susceptible patients by reasoning on ontology

The TB-MDR is generally caused by an inadequate treatment of tuberculosis, which can give rise to an epidemic of TB difficult to cure. In fact, poor adherence or non adherence of patients to TB treatment is a major cause of treatment failure in Africa. Poor adherence is the failure of patients to take medication or follow a diet and lifestyle in accordance with the prescription of the clinician. Patients with poor adherence to TB treatment over a period of time have a high risk to become resistant to prescribed drugs [95]. According to the NTCP, the patients who did not come to their rendez-vous to get the medications are those who will later develop resistance to drugs and come back with TB-MDR. A health worker revealed that often, some patients will follow the first 4 months of treatment, feeling better, they will not come back in the last two months and will come later with TB-MDR. According to the epidemiologist, these patients and their characteristics must be identified at time and an action must be taken.

The actual version of the EPICAM platform does not consider the TB-MDR patients. However, the information on the following appointments of the patients are stored in the database. To get access to this information, a SQL request must be made. However, given that the database is flat, to get access to other information with the link to the patients, a joined request with 6 tables must be done and a source code written to filter patient information (e.g., patient health center, district and/or region). The current use case shows how a simple DL query with the inference system permitted to get all the patients at risk to become TB-MDR (figure 42). A simple click permits access to the patients' characteristics.

This second use case shows that the ontology can be used to classify patients according to their behavior. It can also be used to detect by inferring the other types of patients. For example, the

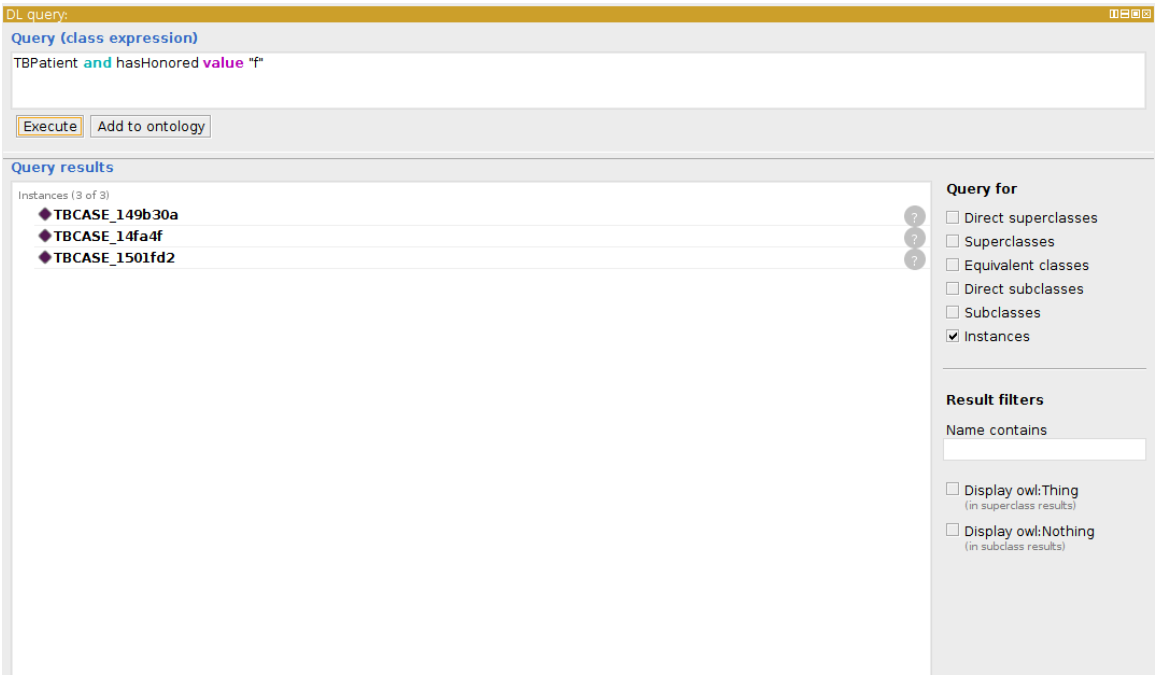


Figure 42: Inferring the patients at risk of TB-MDR

positive microscopy, negative microscopy and what is the difference between them, identify risk factors according to many parameters such as time, location, etc.

6.3.3 Other useful feature of O4TBSS

One of the major benefits of using an ontology is the possibility of using reasoner to automatically compute class hierarchy. The ontology we have developed in this thesis also facilitates the checking for class subsumption. The reasoner is used to automatically compute a classification hierarchy given the class definition. The example of figure 43 shows some class hierarchies obtained from asserted classes by reasoning.

6.4 Conclusion

In this chapter, we reported the development of an ontology for tuberculosis surveillance. This ontology can be used for the annotation of clinical and epidemiological data of tuberculosis. Our motivation was to provide a model for epidemiological data which permits stakeholders involved in epidemiological surveillance of TB to have access to all needed information. The goal being to infer new knowledge from asserted ones using the reasoning mechanism. During the development of O4TBSS, we found many biomedical ontologies that we classified in three main groups. The first group was made of large ontologies such as "Human Disease ontology", modeling one aspect of O4TBSS. These ontologies were not completely reused because they were too large and the parts involved in the epidemiological surveillance of TB were too small. The second group

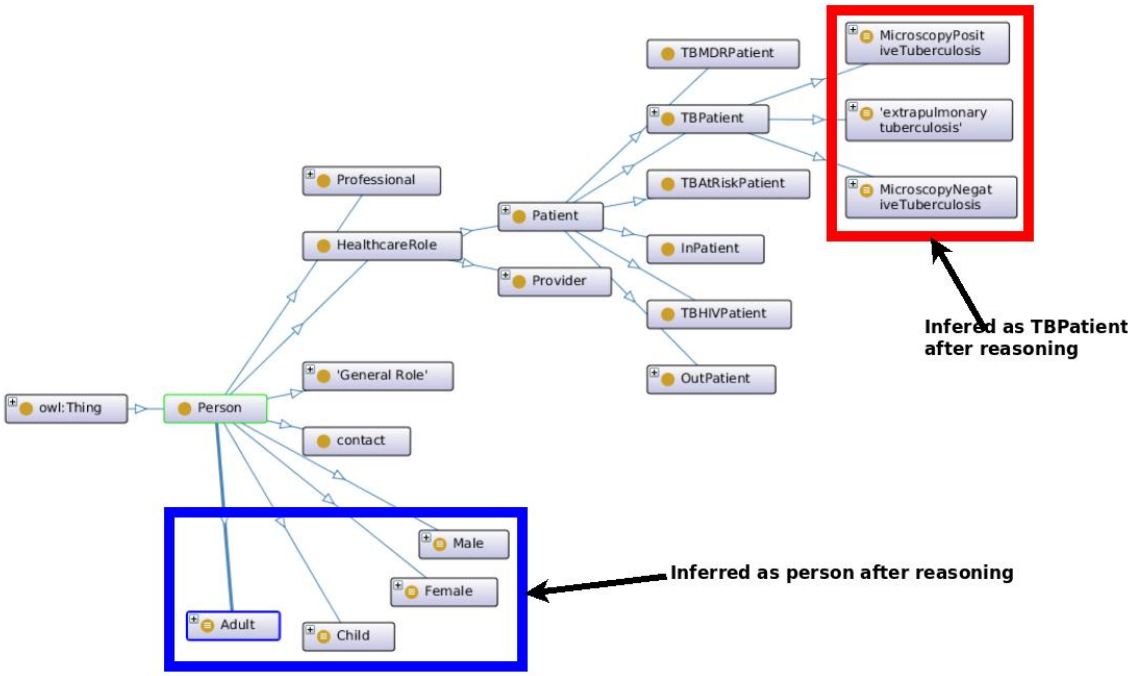


Figure 43: The Inferred Hierarchy alongside the Asserted Hierarchy after classification has taken place. Note the inferred subclasses of the classes Patient and TBPatient

was made of ontologies such as "Mental Health Management Ontology" not focusing on the tuberculosis, but containing relevant knowledge that can be reused. From the first and the second groups, TB knowledge was extracted to build O4TBSS. The third group was made of ontologies such as "Tuberculosis Ontology for Host Systems Biology" specialized in tuberculosis. Some of these ontologies have not been used because they were not available for download and on the other hand, what they described was different from what we were modeling. Knowledge extracted from EPICAM source code presented in chapter 5 was used to enrich the ontology obtained.

Even if our main goal was to infer new knowledge, it should be noted that the ontology developed can also be used for data exchange and integration. It can easily be integrated with other ontologies. This ontology may serve as an example to model epidemiological surveillance of other infectious diseases.

7 Conclusion

At the end of this thesis, we present the summary of the research and the future directions.

7.1 Research summary

In this thesis, we highlighted the problems encountered during the development/use of epidemiological surveillance systems. Then, we focused on two of them: the problem of failed software due to an unsystematic transfer of business requirements to the implementation and the problem of furnishing needed information to stakeholders. Therefore, we proposed a semantic-aware epidemiological surveillance system. Our main objective being to design a semantic-aware tuberculosis surveillance system. This system was built in two phases with two complementary modules:

- **The data collection and management module:** Based on Model Driven Architecture, this module permitted us to model tuberculosis surveillance systems in Cameroon and generate the EPICAM platform. During the pilot phase in 2014 and 2015, the EPICAM platform was deployed in twenty five pilot hospitals distributed in the ten regions of Cameroon. This system allows the National Tuberculosis Control Program (NTCP) in Cameroon to register and follow-up around 3900 patients, representing 15.6% annual number of TB cases in Cameroon. According to user' feedback: at the central level, the EPICAM platform allow a better visibility of the health problems and an early detection of issues; at the peripheral level, the system allows the improvement of patients management and follow-up by health workers.

  Given the success of the pilot phase, the NTCP have adopted the EPICAM platform and extended it in twenty new health centers during the years 2016 and 2017.

- **The knowledge management module:** The goal of this module is to support tuberculosis surveillance by allowing stakeholders to retrieve all needed information. To this end, during this thesis, we have developed an Ontology for Tuberculosis Surveillance System (O4TBSS). Enriched with the data acquired during epidemiological surveillance of TB, O4TBSS permitted derive new information given asserted ones.

Globally, in this thesis, we proposed:

- A Model Driven Architecture-based approach for the development of epidemiological surveillance systems. Thereafter, we have developed an open source tool permitting the modeling and the generation of epidemiological surveillance systems. This tool were used to model and generate a platform for epidemiological surveillance of tuberculosis in Cameroon;
- Hidden Markov Models (HMM) based approach for ontology learning from Java source code. Two models were also proposed: one for the modeling of concepts properties and axioms and the other for the modeling of rules. This approach and these models were used for ontology learning from the source code of EPICAM. The knowledge learned was validated by Ontology Recommender tool and domain experts;
- An Ontology for Tuberculosis Surveillance System developed by combining the knowledge learned from EPICAM source code, EPICAM database and existing biomedical ontologies. This ontology was populated with the data contained in the EPICAM database and use cases permitted to show how new information can be obtained by the mean of inference system;
- An architecture for a semantic-aware epidemiological surveillance system. This architecture describes how an ontology can be integrated in an epidemiological surveillance system in order to provide information that is not explicitly registered in the system to stakeholders.

7.2 Discussion and future works

Future works follow five different directions: the development of the semantic search engine on top of O4TBSS; the filtering of epidemiological information according to user profile; the building an ontology network for epidemiological surveillance of TB; re-estimating the parameters of the HMMs presented in this thesis using specialized algorithms; and the development of a source code knowledge graph.

The development of the semantic search engine on top of O4TBSS. In this thesis, we have made the specification and the design of the module permitting to stakeholders to obtain all needed information. Thereafter, the ontology which populated with data obtained during epidemiological surveillance of TB was developed. However, the experimentation was done using the Protege software which is not the appropriate tool for querying and displaying results. In many cases, decision makers prefer results in the form of statistical tables, graphics and maps. Then, we are planning to develop a search engine, integrating O4TBSS and permitting stakeholders to use keywords to access relevant information in an appropriate form.

Filtering information according to user profile. The computerization of epidemiological surveillance systems allow the collection and storage of large volume of data [62, 110]. If the information to furnish to the users is not filtered, this may lead to the problem of information overload. Information overload (infobesity or infoxication) is defined as the difficulty that a person may have to understand an issue and make decisions caused by the presence of too much information. According to the literature, this problem stresses and affects the performance of health

professionals and undermines effectiveness [74]. Given that the users of epidemiological surveillance systems are from different domains, have different profiles and preferences on information, we will design an architecture for filtering epidemiological information according to user profile.

Building an ontology network for epidemiological surveillance of tuberculosis. Epidemiological surveillance systems may need additional data (e.g., pluviometric data, population data) stored in external data sources to explain some phenomenon (e.g., TB outbreak). In the future work, we will build an ontology network permitting access to all the relevant data sources necessary to give relevant information to stakeholders.

Re-estimating the HMMs parameters. The performance of HMMs in knowledge extraction depends on its training. In this thesis, we have particularly focused on knowledge extraction from Java source code using the Viterbi algorithm. Future work will be devoted to the training of the models using specialized algorithms such as Baum-welch or Viterbi training in order to determine the best values of the HMMs parameters.

Building a source code knowledge graph to solve software development problems. The popularity of computer applications and the huge growth of new software development technologies has brought about the development of many applications and services [23] such as e-epidemiology and e-health platforms. Large softwares consist of many modules (small programs), possibly written by different programmers [23, 39]. In developing and using large software, some problems could be encountered:

- Software requirements could be articulated on the web through historical email messages, discussion forums, etc. Once asserted, there are generally no "software requirements specification documents" [116]. In GitHub for example, adding comments and some code updates are not always done in parallel with the updating of the other software artifacts.
- In some Open Source software with large communities, many developers contribute to source code with their own vocabularies [116].
- Sometimes, developers focus on coding features rather than ensuring that they have a solid and complete documentation that facilitates the integration of newcomers [116].
- Changing the needs during the development process is still not managed by software process models. As a consequence, software projects do not always meet their expectations in terms of functionality, cost and delivery schedule [116].
- Integrating the evolution of platforms directly into source code can make new programmers take too much time to grab the source code [109, 116].
- Bugs and security vulnerabilities such as non-bounds-checking functions, input validation, buffer overflow, etc. are difficult to identify and solve.

To solve the above problems, we will develop a source code knowledge graph and a tool for knowledge extraction from source code. The tool will use the HMMs presented and experimented in this thesis to extract knowledge from existing tested and known to work source code in order to build and/or to update the source code knowledge graph. The source code knowledge graph

will describe source code structure, vocabulary, bugs and vulnerabilities. Integrated in Integrated Environment Development such as Eclipse, this knowledge graph will permit to: make the source code readable and understandable to new developers; make the software artifacts coherent; make an early discovering of bugs, security breaches or violations of programming conventions; highlight possible coding errors, the precise source file, line numbers and even subsection of lines that are affected. An immediate feedback will be provided to the developers in issues they might be introducing into source code.

Bibliography

Bibliography

[1] Abdelghany Abdelghany, Nagy Ramadan, and Hesham Hefni. An agile methodology for ontology development. *International Journal of Intelligent Engineering and Systems*, 12:170–181, 04 2019.

[2] Kumar Abhishek and Singh M.P. An ontology based decision support for tuberculosis management and control in india. *International Journal of Engineering and Technology*, 8:2860–2877, 12 2016.

[3] Pekka Abrahamsson, Outi Salo, Jussi Ronkainen, and Juhani Warsta. Agile software development methods: Review and analysis. *CoRR*, abs/1709.08439, 2002.

[4] Maedche Alexander and Volz Raphael. The ontology extraction & maintenance framework text-to-onto. In *International Conference on Data Mining (ICDM), San Jose, USA, November 29 - December 2, 2001*. IEEE, Los Alamitos (CA), 2001.

[5] Maedche Alexander and Staab Steffen. Semi-automatic engineering of ontologies from text. *Proceedings of the 12th Internal Conference on Software and Knowledge Engineering. Chicago, USA*, 2000.

[6] Muhammad Amith, Zhe He, Jiang Bian, Juan Antonio Lossio-Ventura, and Cui Tao. Assessing the practice of biomedical ontology evaluation: Gaps and opportunities. *Journal of Biomedical Informatics*, 80:1–13, 2018.

[7] Muhammad Nabeel Asim, Muhammad Wasim, Muhammad Usman Ghani Khan, Waqar Mahmood, and Hafiza Mahnoor Abbasi. A survey of ontology learning techniques and applications. *Database*, 2018:bay101, 2018.

[8] Blagoj Atanasovski, Milos Bogdanovic, Goran Velinov, Leonid Stoimenov, Aleksandar S. Dimovski, Bojana Koteska, Dragan Jankovic, Irena Skrceska, Margita Kon-Popovska, and Boro Jakimovski. On defining a model driven architecture for an enterprise e-health system. *Enterprise Information Systems*, 12(8-9):915–941, 2018.

[9] Blagoj Atansovski, MiloÅ¡ BogdanoviÄ, Goran Velinov, Leonid Stoimenov, Dragan Sahpaski, Irena Skrceska, Margita Kon-Popovska, Dragan JankoviÄ, and Boro Jakimovski. Transforming an enterprise e-health system from process oriented to model driven architecture. In *7th International Conference on Information Society and Technology ICIST*, pages 159–162, 05 2017.

[10] Mattia Atzeni and Maurizio Atzori. Codeontology: Querying source code in a semantic framework. In *Proceedings of the ISWC 2017 Posters & Demonstrations and Industry Tracks co-located with 16th International Semantic Web Conference (ISWC 2017), Vienna, Austria, October 23rd - to - 25th, 2017.*, 2017.

[11] Mattia Atzeni and Maurizio Atzori. Codeontology: Rdf-ization of source code. In *The Semantic Web - ISWC 2017 - 16th International Semantic Web Conference, Vienna, Austria, October 21-25, 2017, Proceedings, Part II*, pages 20–28, 2017.

[12] Fidèl Jiomekong Azanzi, Laurent Broto, Daniel Hagimont, Suzy Temate, and Maurice Tchuente. Data collection in a degraded network: case of developing countries or countries in crisis. In *6th International Conference on Theory and Practice of Electronic Governance, ICEGOV '12, Albany, NY, USA, October 22-25, 2012*, pages 406–409, 2012.

[13] Fidèl Jiomekong Azanzi and Gaoussou Camara. Knowledge extraction from source code based on hidden markov model: Application to EPICAM. In *14th IEEE/ACS International Conference on Computer Systems and Applications, AICCSA 2017, Hammamet, Tunisia, October 30 - Nov. 3, 2017*, pages 1478–1485, 2017.

[14] Fidèl Jiomekong Azanzi and Gaoussou Camara. An approach for knowledge extraction from source code (KNESC) of typed programming languages. In *Trends and Advances in Information Systems and Technologies - Volume 1 [WorldCIST'18, Naples, Italy, March 27-29, 2018].*, pages 122–131, 2018.

[15] Fidèl Jiomekong Azanzi, Gaoussou Camara, and Maurice Tchuente. Extracting ontological knowledge from java source code using hidden markov models. *Open Computer Science*, 9(1):181–199, 2019.

[16] Hossein Bagherian, Mohammad Farahbakhsh, Reza Rabiei, Hamid Moghaddasi, and Farkhondeh Asadi. National communicable disease surveillance system: A review on information and organizational structures in developed countries. *Acta informatica medica*, 12 2017.

[17] Hossein Bagherian, Mohammad Farahbaksh, Reza Rabiei, Hamid Moghaddasi, and Farkhondeh Asadi. National communicable disease surveillance system: A review on information and organizational structures in developed countries. 25:271â276, 12 2017.

[18] Kent Beck. *Extreme Programming Explained: Embrace Change*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2000.

[19] David W. Binkley, Marcia Davis, Dawn J. Lawrie, and Christopher Morrell. To camelcase or under_score. In *Proceedings of the 27th International Conference on Program Comprehension*, ICPC '19, pages 177–177, Piscataway, NJ, USA, 2019. IEEE Press.

[20] Xavier Blanc. *MDA en action, Ingénierie logicielle guidée par les mod'eles*. EYROLLES, 2005.

[21] Joaquín A Blaya, Sonya Shin, Carmen Contreras, Gloria Yale, Carmen Suarez, Luis Asencios, Jihoon Kim, Pablo Rodriguez, Peter Cegielski, and Hamish S F Fraser. Full impact of laboratory information system requires direct use by clinical staff: cluster randomized controlled trial. *Journal of the American Medical Informatics Association*, 18(1):11–16, 11 2010.

[22] Bernd Blobel and Peter Pharow. A model driven approach for the german health telematics architectural framework and security infrastructure. *International journal of medical informatics*, 76(2-3):169â175, 2007.

[23] Kalina Bontcheva. Learning ontologies from software artifacts: Exploring and combining multiple choices. In Jeff Z. Pan and Yuting Zhao, editors, *Semantic Web Enabled Software Engineering*, volume 17 of *Studies on the Semantic Web*, pages 235–250. IOS Press, 2014.

[24] Bouchra Bouihi and Mohamed Bahaj. An uml to owl based approach for extracting moodleâs ontology for social network analysis. *Procedia Computer Science*, 148:313 – 322, 2019. The Second International Conference on Intelligent Computing in Data Sciences, ICDS2018.

[25] Waylon Brunette, Mitchell Sundt, Nicola Dell, Rohit Chaudhri, Nathan Breit, and Gaetano Borriello. Open data kit 2.0: Expanding and refining information services for developing regions. In *HotMobile : The International Workshop on Mobile Computing Systems and Applications*, 2013.

[26] Marko Brunzel. The xtreem methods for ontology learning from web documents. In Paul Buitelaar and Philipp Cimiano, editors, *Ontology Learning and Population: Bridging the Gap between Text and Knowledge*, volume 167 of *Frontiers in Artificial Intelligence and Applications*, pages 3–26. IOS Press, 2008.

[27] Bernard C K Choi. The past, present, and future of public health surveillance. *Scientifica*, 2012:875253, 08 2012.

[28] Kenneth G. Castro. Tuberculosis Surveillance: Data for Decision-Making. *Clinical Infectious Diseases*, 44(10):1268–1270, 2007.

[29] Farid Cerbah and Nadira Lammari. Ontology Learning from Databases: Some Efficient Methods to Discover Semantic Patterns in Data. In AKA / IOS Press. Serie, editor, *Perspectives in Ontology Learning*, page 30. October 2014.

[30] Philipp Cimiano. *Ontology learning and population from text - algorithms, evaluation and applications*. Springer US, 2006.

[31] Benoît Combemale. Ingénierie Dirigée par les Modèles (IDM) – État de l'art. working paper or preprint, 2008.

[32] Nadine Cullot, Raji Ghawi, and Kokou Yétongnon. DB2OWL : A tool for automatic database-to-ontology mapping. In *Proceedings of the Fifteenth Italian Symposium on Advanced Database Systems, SEBD 2007, 17-20 June 2007, Torre Canne, Fasano, BR, Italy*, pages 491–494, 2007.

[33] Vasa Curcin, Thomas Woodcock, Alan J. Poots, Azeem Majeed, and Derek Bell. Model-driven approach to data collection and reporting for quality improvement. *Journal of Biomedical Informatics*, 52:151 – 162, 2014. Special Section: Methods in Clinical Research Informatics.

[34] Alberto Rodrigues da Silva. Model-driven engineering: A survey supported by the unified conceptual model. *Computer Languages, Systems Structures*, 43:139 – 155, 2015.

[35] Frankel David and Parodi John. *The Mda Journal: Model Driven Architecture Straight From The Masters*. Meghan Kiffer, 2004.

[36] Antonio De Nicola, Alberto Tofani, Giordano Vicoli, and Maria Luisa Villani. An mda-based approach to crisis and emergency management modeling. *International Journal on Advances in Intelligent Systems*, 5:89–100, 01 2012.

[37] Reza Dehnavieh, AliAkbar Haghdoost, Ardeshir Khosravi, Fahime Hoseinabadi, Hamed Rahimi, Atousa Poursheikhali, Nahid Khajehpour, Zahra Khajeh, Nadia Mirshekari, Marziyeh Hasani, Samera Radmerikhi, Hajar Haghighi, Mohammad Hossain Mehrolhassani, Elaheh Kazemi, and Saeide Aghamohamadi. The district health information system (dhis2): A literature review and meta-synthesis of its strengths and operational challenges based on the experiences of 11 countries. *Health Information Management Journal*, 48(2):62–75, 2019. PMID: 29898604.

[38] Klaas Dellschaft and Steffen Staab. Strategies for the evaluation of ontology learning. In *Proceedings of the 2008 Conference on Ontology Learning and Population: Bridging the Gap Between Text and Knowledge*, pages 253–272, Amsterdam, The Netherlands, The Netherlands, 2008. IOS Press.

[39] F. DeRemer and H. H. Kron. Programming-in-the-large versus programming-in-the-small. *IEEE Transactions on Software Engineering*, SE-2(2):80–86, June 1976.

[40] Torgeir Dingsøyr, Sridhar Nerur, VenuGopal Balijepally, and Nils Brede Moe. A decade of agile methodologies. *J. Syst. Softw.*, 85(6):1213–1221, June 2012.

[41] Disciplined-agile. Requirements envisioning: An agile core practice, 2019. `http://agilemodeling.com/essays/initialRequirementsModeling.htm`.

[42] Dragan Djuric, Dragan Gasevic, and Vladan Devedzic. Ontology modeling and MDA. *Journal of Object Technology*, 4(1):109–128, 2005.

[43] Sean R. Eddy. What is a hidden Markov model? *Nature Biotechnology*, 22(10):1315, October 2004.

[44] Allae Erraissi and A Belangour. Data sources and ingestion big data layers: Meta-modeling of key concepts and features. *International Journal of Engineering and Technology(UAE)*, 7:3607–3612, 01 2018.

[45] Muhammad Fahad. ER2OWL: generating OWL ontology from ER diagram. In *Intelligent Information Processing IV, 5th IFIP International Conference on Intelligent Information Processing, October 19-22, 2008, Beijing, China*, pages 28–37, 2008.

[46] Matt Fenwick, Gerard Weatherby, Heidi J. C. Ellis, and Michael R. Gryk. Parser combinators: A practical application for generating parsers for NMR data. In *Tenth International Conference on Information Technology: New Generations, ITNG 2013, 15-17 April, 2013, Las Vegas, Nevada, USA*, pages 241–246, 2013.

[47] Gernot A. Fink. *Markov Models for Pattern Recognition: From Theory to Applications*. Advances in Computer Vision and Pattern Recognition. Springer-Verlag, London, 2 edition, 2014.

[48] G. David Forney. The viterbi algorithm: A personal history. *CoRR*, abs/cs/0504020, 2005.

[49] Monica Franzese and Antonella Iuliano. Hidden markov models. In Shoba Ranganathan, Michael Gribskov, Kenta Nakai, and Christian SchAnbach, editors, *Encyclopedia of Bioinformatics and Computational Biology*, pages 753 – 762. Academic Press, Oxford, 2019.

[50] Ralph R. Frerichs. Epidemiologic surveillance in developing countries. *Annual Review Public Health*, 12:257, 1991.

[51] Gopinath Ganapathy and S. Sagayaraj. To generate the ontology from java source code. *International Journal of Advanced Computer Science and Applications*, 2(2), 2011.

[52] Fabien Gandon, Catherine Faron Zucker, and Olivier Corby. *Le web sémantique*. Dunod, 2012.

[53] Andrés García-Silva, Leyla Jael García-Castro, Alexander García Castro, and Óscar Corcho. Building domain ontologies out of folksonomies and linked data. *International Journal on Artificial Intelligence Tools*, 24(2), 2015.

[54] Dragan Gasevic, Dragan Djuric, and Vladan Devedzic. *Model Driven Engineering and Ontology Development*. Springer Publishing Company, Incorporated, 2nd edition, 2009.

[55] Mirna El Ghosh, Hala Naja, Habib Abdulrab, and Mohamad Khalil. Ontology learning process as a bottom-up strategy for building domain-specific ontology from legal texts. In *Proceedings of the 9th International Conference on Agents and Artificial Intelligence, ICAART 2017, Volume 2, Porto, Portugal, February 24-26, 2017.*, pages 473–480, 2017.

[56] Philippe Glaziou, Katherine Floyd, and Mario Raviglione. Global epidemiology of tuberculosis. *Cold Spring Harbor Laboratory Press*, 5, 2015.

[57] Philippe Glaziou, Katherine Floyd, and Mario Raviglione. Global epidemiology of tuberculosis. *Seminars in Respiratory and Critical Care Medicine*, 39:271–285, 2018.

[58] Asunción Gómez-Pérez, Mariano Fernández-López, and Óscar Corcho. *Ontological Engineering: With Examples from the Areas of Knowledge Management, e-Commerce and the Semantic Web*. Advanced Information and Knowledge Processing. Springer, 2004.

[59] Mokhtaria Hacherouf, Safia Nait Bahloul, and Christophe Cruz. Transforming XML documents to OWL ontologies: A survey. *J. Inf. Sci.*, 41(2):242–259, 2015.

[60] Tamás Hauer, Dmitry Rogulin, Sonja Zillner, Andrew Branson, Jetendr Shamdasani, Alexey Tsymbal, Martin Huber, Tony Solomonides, and Richard McClatchey. An architecture for semantic navigation and reasoning with patient data - experiences of the health-e-child project. In Amit Sheth, Steffen Staab, Mike Dean, Massimo Paolucci, Diana Maynard, Timothy Finin, and Krishnaprasad Thirunarayan, editors, *The Semantic Web - ISWC 2008*, pages 737–750, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[61] Maryam Hazman, Samhaa R. El-Beltagy, and Ahmed Rafea. A survey of ontology learning approaches. *International Journal of Computer Applications*, 22(8):36–43, May 2011.

[62] D. A. Henderson. The Development of Surveillance Systems. *American Journal of Epidemiology*, 183(5):381–386, 02 2016.

[63] Pascal Hitzler, Markus Krötzsch, and Sebastian Rudolph. *Foundations of Semantic Web Technologies*. Chapman & Hall/CRC, 2009.

[64] Pascal Hitzler, Markus Krötzsch, and Sebastian Rudolph. *Foundations of Semantic Web Technologies*. Chapman and Hall/CRC Press, 2010.

[65] HMN. Framework and standards for country health information systems. Technical report, 2008.

[66] Bouchra El Idrissi, Salah Baïna, and Karim Baïna. Ontology learning from relational database: How to label the relationships between concepts? In *Beyond Databases, Architectures and Structures - 11th International Conference, BDAS 2015, Ustroń, Poland, May 26-29, 2015, Proceedings*, pages 235–244, 2015.

[67] S. Iloga, O. Romain, L. Bendaouia, and M. Tchuente. Musical genres classification using markov models. In *2014 International Conference on Audio, Language and Image Processing*, pages 701–705, July 2014.

[68] Irum Inayat, Siti Salwah Salim, Sabrina Marczak, Maya Daneva, and Shahaboddin Shamshirband. A systematic literature review on agile requirements engineering practices and challenges. *Comput. Hum. Behav.*, 51(PB):915–929, October 2015.

[69] Azanzi Jiomekong and Gaoussou Camara. Model-driven architecture based software development for epidemiological surveillance systems. *Studies in health technology and informatics*, 264:531â535, August 2019.

[70] Val Jones, Arend Rensink, and Ed Brinksma. Modelling mobile health systems: an application of augmented MDA for the extended healthcare enterprise. In *Ninth IEEE International Enterprise Distributed Object Computing Conference (EDOC 2005), 19-23 September 2005, Enschede, The Netherlands*, pages 58–69, 2005.

[71] Rupinder Kaur and Jyotsna Sengupta. Software process models and analysis on failure of software development projects. *CoRR*, abs/1306.1068, 2013.

[72] Faten Kharbat and Haya El-Ghalayini. Building Ontology from Knowledge Base Systems. *Data Mining in Medical and Biological Research*, November 2008.

[73] Olga Kolesnikova. Survey of word co-occurrence measures for collocation detection. *Computación y Sistemas*, 20(3):327–344, 2016.

[74] Hanumantha Rao Kolusu. Information overload and its effect on healthcare. Master of biomedical informatics, Oregon Health & Science University, 2015.

[75] Agnieszka Konys. Knowledge systematization for ontology learning methods. In *Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 22nd International Conference KES-2018, Belgrade, Serbia, 3-5 September 2018.*, pages 2194–2207, 2018.

[76] Guy Leonard Kouemou. History and Theoretical Basics of Hidden Markov Models. *Hidden Markov Models, Theory and Applications*, April 2011.

[77] Martin Labský, Vojtech Svátek, Ondrej Sváb, Pavel Praks, Michal Krátký, and Václav Snásel. Information extraction from HTML product catalogues: From source code and images to RDF. In *2005 IEEE / WIC / ACM International Conference on Web Intelligence (WI 2005), 19-22 September 2005, Compiegne, France*, pages 401–404, 2005.

[78] Craig Larman and Victor R. Basili. Iterative and incremental development: A brief history. *Computer*, 36(6):47–56, June 2003.

[79] Nelson K. Y. Leung, Sim Kim Lau, and Nicole Tsang. Reuse existing ontologies in an ontology development process - an integration-oriented ontology development methodology. *IJWS*, 2(3):159–180, 2014.

[80] David M. Levine, Noton Dutta, Josh Eckels, Charles Scanga, Catherine Stein, Smriti Mehra, Deepak Kaushal, Petros Karakousis, and Hugh Salamon. A tuberculosis ontology for host systems biology. *Tuberculosis*, 95(5):570–574, 9 2015.

[81] Yunyao Li, Rajasekar Krishnamurthy, Sriram Raghavan, Shivakumar Vaithyanathan, and H. V. Jagadish. Regular expression learning for information extraction. In *2008 Conference on Empirical Methods in Natural Language Processing, EMNLP 2008, Proceedings of the Conference, 25-27 October 2008, Honolulu, Hawaii, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 21–30, 2008.

[82] Yu Lin, Zuoshuang Xiang, and Yongqun He. Brucellosis ontology (IDOBRU) as an extension of the infectious disease ontology. *J. Biomedical Semantics*, 2:9, 2011.

[83] Francesca A. Lisi. Learning onto-relational rules with inductive logic programming. *CoRR*, abs/1210.2984, 2012.

[84] Mapatano and Piripiri. Quelques erreurs courantes d'analyse d'un syst'eme d'information sanitaire(rd congo). In *Santé Publique 2005*, volume 17, 2005.

[85] Carmen Martinez-Cruz, Ignacio J. Blanco, and M. Amparo Vila. Ontologies versus relational databases: are they so different? a comparison. *Artificial Intelligence Review*, 38(4):271–290, Dec 2012.

[86] John Mathenge Kanyaru, Melanie Coles, Sheridan Jeary, and Keith Phalp. Using visualisation to elicit domain information as part of the model driven architecture approach. *CEUR Workshop Proceedings*, 376, 01 2008.

[87] Martin McHugh, Fergal McCaffery, and Valentine Casey. Barriers to adopting agile practices when developing medical device software. In Antonia Mas, Antoni Mesquida, Terry Rout, Rory V. O'Connor, and Alec Dorling, editors, *Software Process Improvement and Capability Determination*, pages 141–147, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[88] Cameroun MINSANTE. Syst'eme national d'information sanitaire, formulaire de recueil de données pour le rapport mensuel d'activités dans les hôpitaux centraux. Technical report, 2015.

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

[90] Gunter Mussbacher, Daniel Amyot, Ruth Breu, Jean-Michel Bruel, Betty H. C. Cheng, Philippe Collet, Benoit Combemale, Robert B. France, Rogardt Heldal, James Hill, Jörg Kienzle, Matthias Schöttle, Friedrich Steimann, Dave Stikkolorum, and Jon Whittle. The relevance of model-driven engineering thirty years from now. In Juergen Dingel, Wolfram Schulte, Isidro Ramos, Silvia Abrahão, and Emilio Insfran, editors, *Model-Driven Engineering Languages and Systems*, pages 183–200, Cham, 2014. Springer International Publishing.

[91] Oscar Nierstrasz and Jan Kurs. Parsing for agile modeling. *Sci. Comput. Program.*, 97:150–156, 2015.

[92] Natalya F. Noy and Deborah L. McGuinness. Ontology development 101: A guide to creating your first ontology. Technical report, March 2001.

[93] Natalya Fridman Noy, Ray W. Fergerson, and Mark A. Musen. The knowledge model of protégé-2000: Combining interoperability and flexibility. In Rose Dieng and Olivier Corby, editors, *Knowledge Engineering and Knowledge Management Methods, Models, and Tools*, pages 17–32, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[94] Martin O'Connor and Amar Das. Sqwrl: A query language for owl. In *Proceedings of the 6th International Conference on OWL: Experiences and Directions - Volume 529*, OWLED'09, pages 208–215, Aachen, Germany, Germany, 2009. CEUR-WS.org.

[95] Olukunle A. Ogundele, Deshendran Moodley, Christopher J. Seebregts, and Anban W. Pillay. An ontology for tuberculosis treatment adherence behaviour. In *Proceedings of the 2015 Annual Research Conference on South African Institute of Computer Scientists and Information Technologists*, SAICSIT '15, pages 30:1–30:10, New York, NY, USA, 2015. ACM.

[96] Okoro, Sholagberu, and Kolo. Mobile phone ownership among nigerians with diabetes. In *African Health Sciences*, volume 10, 2010.

[97] OMS. Organisation mondiale de la santé (oms). s̈yst‘emes d’informations pour la gestion des programmes sanitaires nationaux. rôle des programmes ayant l’an 1990 pour horizon-programme élargie de vaccination, approvisionnement en eau et salubrité, lutte contre la malnutrition pour atteindre la santé pour tous en l’an 2000. mobilisation des collectivités en vue du développement sanitaire : approches et contraintes.̈ Technical report, 1984.

[98] OMS. Réseau de métrologie sanitaire au cameroun. analyse situationnelle du syst‘eme d’information sanitaire au cameroun. Technical report, 2007.

[99] OMS. Rapport d’évaluation du syst‘eme national d’information sanitaire ivoirien par l’outil du réseau de métrologie sanitaire rms/hmn en 2008. Technical report, 2008.

[100] Tünay Özcan, Semra Kocak, and Philipp Brune. Agile software development with open source software in a hospital environment – case study of an ecrf-system for orthopaedical studies. In Florian Daniel, Peter Dolog, and Qing Li, editors, *Web Engineering*, pages 439–451, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[101] Anurag Passi, Neeraj Rajput, David Wild, and Anshu Bhardwaj. Reptb: a gene ontology based drug repurposing approach for tuberculosis. *Journal of Cheminformatics*, 10, 12 2018.

[102] Branko Perisic. Model driven software development â state of the art and perspectives. In *INFOTEH-JAHORINA*, volume 13, pages 1237–1248, 03 2014.

[103] Catia Pesquita, João Ferreira, Francisco Couto, and Mario J Silva. The epidemiology ontology: an ontology for the semantic annotation of epidemiological resources. *Journal of biomedical semantics*, 5:4, 2014.

[104] H.S. Pinto, A. Gómez-Pérez, and J.P. Martins. Some issues on ontology integration. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI 99) Workshop: KRR5: Ontologies and Problem-Solving Methods: Lesson Learned and Future Trends*, volume 18, 1999.

[105] PNLT and OMS. Minist‘ere de la santé, programme national de lutte contre la tuberculose cameroun. guide technique pour les personnels de sante. Technical report, 2015.

[106] Wullianallur Raghupathi and Amjad Umar. Exploring a model-driven architecture (mda) approach to health care information systems development. *International Journal of Medical Informatics*, 77(5):305 – 314, 2008.

[107] Wullianallur Raghupathi and Amjad Umar. Integrated digital health systems design. *International Journal of Information Technologies and Systems Approach*, 2:15–33, 06 2009.

[108] Abdul Mateen Rajput and Harsha Gurulingappa. Semi-automatic approach for ontology enrichment using umls. *Procedia Computer Science*, 23(Supplement C):78 – 83, 2013. 4th International Conference on Computational Systems-Biology and Bioinformatics, CSBio2013.

[109] C. V. Ramamoorthy, V. Garg, and A. Prakash. Programming in the large. *IEEE Transactions on Software Engineering*, SE-12(7):769–783, 1986.

[110] Chesley L. Richards, Michael F. Iademarco, Delton Atkinson, Robert W. Pinner, Paula Yoon, William R. Mac Kenzie, Brian Lee, Judith R. Qualters, and Thomas R. Frieden. Advances in public health surveillance and information dissemination at the centers for disease control and prevention. *Public Health Reports*, 132(4):403–410, 2017.

[111] Marcos Martínez Romero, Clément Jonquet, Martin J. O'Connor, John Graybeal, Alejandro Pazos, and Mark A. Musen. NCBO ontology recommender 2.0: an enhanced approach for biomedical ontology recommendation. *J. Biomedical Semantics*, 8(1):21:1–21:22, 2017.

[112] Kenneth S. Rubin, Thomas Beale, and Bernd Blobel. *Modeling for Health Care*, pages 125–146. Springer New York, New York, NY, 2005.

[113] Stuart J. Russell and Peter Norvig. *Artificial Intelligence - A Modern Approach, Third International Edition*. Pearson Education, 2010.

[114] Marek Rychlý and Pavlína Tichá. A tool for supporting feature-driven development. In Bertrand Meyer, Jerzy R. Nawrocki, and Bartosz Walter, editors, *Balancing Agility and Formalism in Software Engineering*, pages 196–207, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[115] Dina Salah, Richard F. Paige, and Paul Cairns. A systematic literature review for agile development processes and user centred design integration. In *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, EASE '14, pages 5:1–5:10, New York, NY, USA, 2014. ACM.

[116] Walt Scacchi. Understanding the requirements for developing open source software systems. *IEE Proceedings - Software*, 149(1):24–39, Feb 2002.

[117] Hannes Schlieter, Martin Burwitz, O. Schonherr, and Martin Benedict. Towards model driven architecture in healthcare information system development. In *Wirtschaftsinformatik*, pages 497–511, 03 2015.

[118] Lynn M Schriml, Elvira Mitraka, James Munro, Becky Tauber, Mike Schor, Lance Nickle, Victor Felix, Linda Jeng, Cynthia Bearer, Richard Lichenstein, Katharine Bisordi, Nicole Campion, Brooke Hyman, David Kurland, Connor Patrick Oates, Siobhan Kibbey, Poorna Sreekumar, Chris Le, Michelle Giglio, and Carol Greene. Human Disease Ontology 2018 update: classification, content and workflow expansion. *Nucleic Acids Research*, 47(D1):D955–D962, 11 2018.

[119] Ken Schwaber. *Agile Project Management With Scrum*. Microsoft Press, Redmond, WA, USA, 2004.

[120] Kristie Seymore, Andrew Mccallum, and Ronald Rosenfeld. Learning hidden markov model structure for information extraction. In *AAAI 99 Workshop on Machine Learning for Information Extraction*, pages 37–42, 1999.

[121] Mehrnoush Shamsfard and Ahmad Abdollahzadeh Barforoush. The state of the art in ontology learning: A framework for comparison. *Knowl. Eng. Rev.*, 18(4):293–316, dec 2003.

[122] Feichen Shen and Yugyung Lee. Knowledge discovery from biomedical ontologies in cross domains. *PLOS ONE*, 11(8):1–34, 08 2016.

[123] Thuppahi Sisira De Silva, Don MacDonald, Grace I. Paterson, Khokan C. Sikdar, and Bonnie Cochrane. Systematized nomenclature of medicine clinical terms (SNOMED CT) to represent computed tomography procedures. *Computer Methods and Programs in Biomedicine*, 101(3):324–329, 2011.

[124] Barry Smith, Michael Ashburner, Cornelius Rosse, Jonathan Bard, William Bug, Werner Ceusters, and al. The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration. *Nature biotechnology*, 25(11):1251–1255, November 2007.

[125] Rudi Studer, V. Richard Benjamins, and Dieter Fensel. Knowledge engineering: Principles and methods. *Data Knowl. Eng.*, 25(1-2):161–197, 1998.

[126] Mari Carmen Suárez-Figueroa, Asunción Gómez-Pérez, and Mariano Fernández-López. *The NeOn Methodology for Ontology Engineering*, pages 9–34. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[127] Mari Carmen Suárez-Figueroa, Asunción Gómez-Pérez, and Mariano Fernández-López. The neon methodology framework: A scenario-based methodology for ontology development. *Applied Ontology*, 10(2):107–145, 2015.

[128] William M. Tierney, Marion Achieng, Elaine Baker, April Bell, Paul Biondich, Paula Braitstein, Daniel Kayiwa, Sylvester Kimaiyo, Burke Mamlin, Brian McKown, Nicholas Musinguzi, Winstone Nyandiko, Joseph Rotich, John Sidle, Abraham Siika, Martin Were, Ben Wolfe, Kara Wools-Kaloustian, Ada Yeung, and Constantin Yiannoutsos. Experience implementing electronic health records in three east african countries. In *Studies in Health Technology and Informatics*, volume 160, pages 371–375, 2010.

[129] JAMES HENDLER TIM BERNERS-LEE and ORA LASSILA. The semantic web. *Scientific American:*, 05 2001.

[130] Gerald Töpper, Magnus Knuth, and Harald Sack. Dbpedia ontology enrichment for inconsistency detection. In *Proceedings of the 8th International Conference on Semantic Systems*, I-SEMANTICS '12, pages 33–40, New York, NY, USA, 2012. ACM.

[131] Frank Truyen. The fast guide to model driven architecture the basics of model driven architecture. 01 2006.

[132] Jörg Unbehauen, Sebastian Hellmann, Sören Auer, and Claus Stadler. Knowledge Extraction from Structured Sources. In Stefano Ceri and Marco Brambilla, editors, *Search Computing: Broadening Web Search*, volume 7538 of *Lecture Notes in Computer Science*, pages 34–52. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[133] UNHCR. Syst'eme d'information sanitaire (sis). Technical report, 01 2010.

[134] Vaughan and Morrow. *Manuel épidémiologique pour la gestion de la santé au niveau du district*. OMS, Gen'eve, Suisse, 1991.

[135] Andrew J. Viterbi. Viterbi algorithm. *Scholarpedia*, 4(1):6246, 2009.

[136] Shufeng Wang, Wen Wang, Yanbin Zhuang, and Xianju Fei. An ontology evolution method based on folksonomy. *Journal of Applied Research and Technology*, 13(2):177 – 187, 2015.

[137] Patricia L. Whetzel, Natalya Fridman Noy, Nigam H. Shah, Paul R. Alexander, Csongor Nyulas, Tania Tudorache, and Mark A. Musen. Bioportal: enhanced functionality via new web services from the national center for biomedical ontology to access and use ontologies in software applications. *Nucleic Acids Research*, 39(Web-Server-Issue):541–545, 2011.

[138] WHO. Tuberculosis, October 2020. `https://www.who.int/news-room/fact-sheets/detail/tuberculosis`.

[139] Anna Wróblewska, Teresa Podsiadly-Marczykowska, Robert Bembenik, Grzegorz Protaziuk, and Henryk Rybinski. Methods and Tools for Ontology Building, Learning and Integration ÃÂÂ Application in the SYNAT Project. In Robert Bembenik, Lukasz Skonieczny, Henryk Rybinski, and Marek Niezgodka, editors, *Intelligent Tools for Building a Scientific Information Platform*, volume 390 of *Studies in Computational Intelligence*, pages 121–151. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[140] Zuoshuang Xiang, Mélanie Courtot, Ryan R. Brinkman, Alan Ruttenberg, and Yongqun He. Ontofox: web-based support for ontology reuse. *BMC Research Notes*, 3(1):175, Jun 2010.

[141] Zhuoming Xu, Yuyan Ni, Wenjie He, Lili Lin, and Qin Yan. Automatic extraction of OWL ontologies from UML class diagrams: a semantics-preserving approach. *World Wide Web*, 15(5-6):517–545, 2012.

[142] Diego Bettiol Yamada, Vinicius Tohoru Yoshiura, Newton Shydeo Brandao Miyoshi, Inácia Bezerra de Lima, Gustavo Yukiu Usumoto Shinoda, Rui Pedro Charters Lopes Rijo, Joao Mazzoncini de Azevedo Marques, Maria Manuela Cruz-Cunha, and Domingos Alves. Proposal of an ontology for mental health management in brazil. *Procedia Computer Science*, 138:137 – 142, 2018.

[143] Yang, C. Jun, and Y. Xiangshu L. Use of mobile phones in an emergency reporting system for infectious disease surveillance after the sichuan earthquake in china. In *Bull World Health Organ*, volume 7, 2010.

[144] Shuxin Zhao, Elizabeth Chang, and Tharam S. Dillon. Knowledge extraction from web-based application source code: An approach to database reverse engineering for ontology development. In *Proceedings of the IEEE International Conference on Information Reuse and Integration, IRI 2008, 13-15 July 2008, Las Vegas, Nevada, USA*, pages 153–159, 2008.

[145] Guodong Zhou and Jian Su. Named entity recognition using an hmm-based chunk tagger. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA.*, pages 473–480, 2002.

[146] Lina Zhou. Ontology learning: state of the art and open issues. *Information Technology and Management*, 8(3):241–252, 2007.

List of Tables

LIST OF FIGURES

Appendix

A List of abbreviations

BIRT	Business Intelligence and Reporting Tool
BN	Bayes Network
CDC	Centers for Disease Control and Prevention
CIM	Computational Independent Model
CPC	Centre Pasteur du Cameroun
DBN	Dynamic Bayes Networks
DHIS	District Health Information Software
DSDM	Dynamic Systems Development Method
DSL	Domain Specific Languages
DSM	Domain-Specific Modelling
EHR	Electronic Health Record
EPICAM	Epidemiology in Cameroon
ER	Entity-Relation
FDD	Feature-Driven Development
FSM	Framework Specific Modelling
GSM	Global System for Mobile Communications
HIS	Health Information Systems
HMMs	Hidden Markov Models
ICT	Information and Communication Technologies
JEE	Java Enterprise Edition
KB	Knowledge Base
KR	Knowledge Representation
LOP	Language Oriented Programming
MDA	Model-Driven Architecture
MDSD	Model-Driven Software Development
MDGs	Millennium Development Goals
MDR	MultiDrug-Resistant

MOF	Meta-Object Facility
NTCP	National Tuberculosis Control Program
O4TBSS	ontology for TB surveillance
OCL	Object Constraint Language
OCR	Optical Character Recognition
ODK	Open Data Kit
OL	Ontology Learning
OMG	the Object Management Group
OSD	Open Source Development
OSM	Open Street Map
OWL	Ontology Web Language
PIM	Platform Independent Model
PLHIV	Persons Living with HIV
PSM	Platform Specific Model
RDF	Resource Description Framework
RIF	Rule Interchange Format
RuleML	Rule Markup Language
RUP	Rational Unified Process
SDGs	Sustainable Development Goals
SMS	Short Message Service
SPARQL	SparQL Protocol and RDF Query Language
SQL	Structured Query Language
SQWRL	Semantic Query-Enhanced Web Rule Language
SWRL	Semantic Web Rule Language
TB	Tuberculosis
TPM-	Tuberculosis patient with negative Pulmonary Microscopy
TPM+	Tuberculosis patient with positive Pulmonary Microscopy
UI	User interface
UML	Unified Modelling Language
UNICEF	United Nations International Children's Emergency Fund
UP	Unified Process
URI	Uniform Resource Identifier
US	United States
W3C	World Wide Web Consortium
WHO	World Health Organization
XDR	eXtensively Drug Resistant
XP	Extreme Programming

B List of publications

Journal

A. Jiomekong, G Camara, M Tchuente. Extracting ontological knowledge from Java source code using Hidden Markov Models. 2019 Open Computer Science 9 (1), 181-199.

National conference

Jiomekong Azanzi, Gaoussou Camara. Extraction des connaissances ontologiques du code source Java en utilisant les Chaînes de Markov Cachées. Quatrième Conférence de Recherche en Informatique (CRI 2019).

International conferences

1. Jiomekong Azanzi, Gaoussou Camara. Knowledge Extraction from Source Code Based on Hidden Markov Model: Application to EPICAM. 2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA), 1478-1485.
2. A. Jiomekong, G Camara. An Approach for Knowledge Extraction from Source Code (KNESC) of Typed Programming Languages. 2018 World Conference on Information Systems and Technologies, 122-131.
3. A. Jiomekong, G Camara. Model-Driven Architecture Based Software Development for Epidemiological Surveillance Systems. 2019 Studies in health technology and informatics 264, 531-35.

C Journal paper

Research Article **Open Access**

Azanzi Jiomekong*, Gaoussou Camara, and Maurice Tchuente

Extracting ontological knowledge from Java source code using Hidden Markov Models

https://doi.org/10.1515/comp-2019-0013
Received April 29, 2019; accepted July 25, 2019

Abstract: Ontologies have become a key element since many decades in information systems such as in epidemiological surveillance domain. Building domain ontologies requires the access to domain knowledge owned by domain experts or contained in knowledge sources. However, domain experts are not always available for interviews. Therefore, there is a lot of value in using ontology learning which consists in automatic or semi-automatic extraction of ontological knowledge from structured or unstructured knowledge sources such as texts, databases, etc. Many techniques have been used but they all are limited in concepts, properties and terminology extraction leaving behind axioms and rules. Source code which naturally embed domain knowledge is rarely used. In this paper, we propose an approach based on Hidden Markov Models (HMMs) for concepts, properties, axioms and rules learning from Java source code. This approach is experimented with the source code of EPICAM, an epidemiological platform developed in Java and used in Cameroon for tuberculosis surveillance. Domain experts involved in the evaluation estimated that knowledge extracted was relevant to the domain. In addition, we performed an automatic evaluation of the relevance of the terms extracted to the medical domain by aligning them with ontologies hosted on Bioportal platform through the Ontology Recommender tool. The results were interesting since the terms extracted were covered at 82.9% by many biomedical ontologies such as NCIT, SNOWMEDCT and ONTOPARON.

Keywords: Knowledge Extraction, Ontology Learning, Hidden Markov Models, Java Source Code, Viterbi

1 Introduction

Studer et al. [1] defined an ontology as "A formal, explicit specification of a shared conceptualization". In the context of domain ontologies, conceptualization refers to the abstract model of the domain which is machine readable, and where all the elements are explicitly defined and accepted by the members of a group. Several domain ontologies define and organize relevant knowledge about activities, processes, organizations and strategies, in order to facilitate information exchange between machines and, between a human and a machine [2, 3]. Building domain ontologies requires the access to domain knowledge owned by domain experts or contained in knowledge sources [2, 4]. However, domain experts are not always available for interviews. And in case they are available, the knowledge provided is often incomplete and subjective. In addition, as the domain evolves, the knowledge provided by the experts is likely to be out of date. Therefore, there is a lot of added value in creating domain ontologies from existing knowledge sources such as structured and unstructured documents of the domain: texts [5–8], databases [9–12], XML files [13], existing ontologies [14–16], UML/Metamodel diagrams [17–19], and source code [12, 20–24]. Although source code is often used to extract concepts and relations, its full potential is not exploited to extract, for example, axioms and rules [21, 22]. Indeed, source code is any fully executable description of a software designed for a specific domain such as medical, industrial, military, communication, aerospace, commercial, scientific, etc. It can be used for the collection, organization, storage and communication of information. It is designed to facilitate repetitive tasks or to process information quickly. In software design process, a set of knowledge related to the domain are captured and integrated in the source code.

The extraction of knowledge from structured (relational databases, XML) and unstructured (text, docu-

***Corresponding Author: Azanzi Jiomekong:** University of Yaounde I, Faculty of Science, Yaounde, Cameroon; IRD, Sorbonne Université, UMMISCO, F-93143, Bondy, France;
E-mail: jiofidelus@gmail.com
Gaoussou Camara: LIMA, Université Alioune Diop de Bambey, Sénégal; IRD, Sorbonne Université, UMMISCO, F-93143, Bondy, France; E-mail: gaoussou.camara@uadb.edu.sn

Maurice Tchuente: University of Yaounde I, Faculty of Science, Yaounde, Cameroon; IRD, Sorbonne Université, UMMISCO, F-93143, Bondy, France; E-mail: Maurice.Tchuente@gmail.com

ments, images) sources is also known as ontology learning [25–27] that consists in applying statistical techniques, symbolic techniques or both to (semi-)automatically extract the ontological knowledge from knowledge sources. Several authors have proposed the use of symbolic techniques [12, 20, 28] and statistical techniques [23, 29] to extract generally concepts and properties from source code.

In this paper, we propose an approach for extracting ontological knowledge from Java source code using Hidden Markov Models (HMMs). Our approach is experimented on the EPICAM source code. The EPICAM project[1] aims at building an integrated platform for epidemiological surveillance of tuberculosis in Cameroon. The project started in 2012 and involves partners from the different area: academy (University of Yaounde 1 in Cameroon), clinic (fifty hospitals in Cameroon), epidemiology (Epidemiology and Public Health department of the Centre Pasteur of Cameroon, and the National Tuberculosis Control Program), and industry (MEDES in France).

The rest of this paper is organized as follows. In section 2, we present an overview of ontology learning. Our approach is detailed in section 3. In section 4, we provide the results of the experimentation. The section 5 presents the evaluation of the knowledge extracted. Related works are discussed in section 6. We conclude and present future works in section 7.

2 Ontology Learning

Acquiring knowledge for building an ontology from scratch, or for refining an existing ontology is costly in time and resources. Ontology learning techniques are used to reduce this cost during the knowledge acquisition process. Ontology learning refers to the extraction of ontological knowledge from unstructured, semi-structured or fully structured knowledge sources in order to build an ontology from them with little human intervention [3, 25, 26, 30]. In this section, we present the basic ontological knowledge, knowledge sources generally used for ontology learning, some ontology learning techniques and ontology learning evaluation.

2.1 Basic ontological knowledge

An ontology is composed of these basic components [2]:

- **Concept**, also called *Class*, represents a category of objects. For instance "*Health_facility*" is the concept of all health facilities including health centers and clinics;
- **Individual** is an instance of a concept and corresponds to a concrete object. For example, from the concept "*Person*", "*Bob*" is an individual;
- **Property** is used to describe the characteristics of individuals of a concept. They are composed of DataProperties and ObjectProperties. DataProperties are properties whose values are data types. For instance, "*age*" of type "*Integer*" can be a property of an instance of the concept "*Person*". ObjectProperties are special attributes whose values are individuals of concepts. For instance, "*examined_in*" defines a relationship between the concept "*Person*" and the concept "*Health_facility*" ("A person is examined in a health facility");
- **Class/Property hierarchy** is one of the most important relation used to organize concepts and properties in the ontology. It is used to organize concepts/properties through which inheritance mechanisms can be applied. For instance, "*Patient*" is subClassOf "*Person*" is a hierarchical relation between these two classes. Class/Property taxonomies are generally used to construct the so called lightweight ontologies or taxonomies;
- **Axiom** is used to model statements that are always true. Heavyweight ontologies add axioms and constraints to lightweight ontologies. Axioms and constraints clarify the intended meaning of the terms in the ontology. For example, the assertion "the concepts "*Men*" and "*Women*" are disjoint" is an axiom;
- **Rule** is a statement in the form $\frac{P_1,\ldots,P_n}{P}$, this means that if the statement P is true, then, the statements $P_1, \ldots, P_n$ are true. Rules are used for knowledge inference purposes.

2.2 Knowledge sources for ontology learning

The process of developing an ontology requires knowledge acquisition from any relevant sources. There are several possible sources of knowledge: domain experts or unstructured, semi-structured, and structured sources [4].

2.2.1 Domain experts

A domain expert is a person knowledgeable of a domain. To get knowledge from domain experts, a knowledge engineer conducts interviews. This process might lead to

knowledge loss or even worse, introduce errors because misunderstandings that arises frequently in human communication.

2.2.2 Unstructured knowledge sources

Unstructured knowledge sources contain knowledge that do not have a pre-defined organization. These are all kinds of textual resources (Web pages, manuals, discussion forum postings, specifications, analysis and conception documents, source code comments) and multimedia contents (videos, photos, audio files) [3, 5, 6, 8, 23, 25, 26]. Unstructured sources are the most recurrent and can permit us to extract a more complete knowledge. However, the unstructured sources are easily accessible to human information processing only. For example, extracting formal specifications from arbitrary texts is still considered a hard problem because sentences might be ambiguous and, in some cases, no unique correct syntactic analysis is possible [31].

2.2.3 Structured knowledge sources

Structured knowledge sources contain knowledge described by a schema. It is advantageous to use these knowledge sources because they contain directly accessible knowledge [31]. Some structured knowledge sources include:

- Ontologies: Before constructing an ontology from scratch, one may look at other ontologies that could be reused [4, 15, 16];
- Knowledge bases: In knowledge bases, one can generate discovered rules as input to develop a domain ontology [25, 32];
- Database : Terms to be used to build an ontology can be extracted from a database schema [9–12, 25].

2.2.4 Semi-structured knowledge sources

Semi-structured knowledge sources contain knowledge having a structure that already reflects part of the semantic interdependencies. This structure facilitates the extraction of a schema [31]. Some examples of semi-structured knowledge sources are:

- Folksonomies/thesaurus: It is advantageous to extract knowledge from folksonomies or/and thesaurus to build an ontology because they reflect the vocabulary of their users [33, 34];
- XML (Extensible Markup Language): The aim of XML data conversion to ontologies is the indexing, integration and enrichment of existing ontologies with knowledge acquired from XML documents [13];
- UML/meta-model: To learn an ontology from UML or/and meta-model, one approach is to extract OWL classes and properties from diagrams or to use Ontology UML Profile (OUP) which, together with Ontology Definition Meta-model (ODM), enable the usage of Model Driven Architecture (MDA) standards in ontological engineering [18];
- Entity-relation diagram: They can be used to learn ontologies because they are used to describe the information managed by the databases [35];
- Source code [12, 21–23, 28]: Generally, in source code, the names of data structures, variables, functions are close to the terms of the domain.

A lot of work has been done on the extraction of ontological knowledge from texts, databases, XML files, vocabularies, and the use of ontologies to build or enrich other ontologies. This has resulted in a wide range of models, techniques and tools for the generation of knowledge structure that can be considered as an intermediate process when constructing ontologies. It should be noted that few works go beyond extracting concepts and properties from source code whereas axioms and rules are also key elements of ontologies.

2.3 Ontology learning techniques

To extract knowledge from knowledge sources, many techniques are used [3, 25, 26, 36]. Shamsfard and Barforoush [26] proposed a classification of these techniques by considering symbolics, statistics and multi-strategies.

2.3.1 Symbolic techniques

In symbolic techniques, the extraction process consists of examining text fragments that match some predefined rules, looking for lexico-syntactic patterns corresponding for instance to taxonomic relations or scanning for various types of templates related to ontological knowledge. A symbolic method can be rule-based, linguistic-based or pattern-based.

1. Rule-based models are represented as a set of rules where each rule consists of a condition and an action [30].

- *Logical rules* may be used to discover new knowledge by deduction (deduce new knowledge from existing ones) or induction (synthesize new knowledge from experience). For example, inductive logic programming can be used to learn new concepts from knowledge sources [5, 25, 26, 37];
- *Association rules* aim at finding correlations between items in a dataset. This technique is generally used to learn relations between concepts [5, 8, 25, 26] and can be used to recognize a taxonomy of relations [25] or to discover gaps in conceptual definitions [5, 26, 38].

2. *Linguistic* approaches (syntactic analysis, morpho-syntactic analysis, lexico-syntactic pattern parsing, semantic processing and text understanding) are used to derive knowledge from text corpus [25, 26]. This technique can be used to derive an intentional description of concepts in the form of natural language description [38].
3. *Pattern/Template-driven* approach allows to search for predefined keywords, templates or patterns. Indeed, a large class of entity extraction tasks can be accomplished by the use of carefully constructed regular expressions [39].

Although very powerful for particular domains, symbolic techniques are inflexible because of their strong dependency on the structure of the data. Symbolic techniques are precise and robust, but can be complex to implement, and difficult to generalize [26].

2.3.2 Statistic-based techniques

Statistic analysis for ontology learning is performed from input data to build a statistical model [3, 25, 26, 30]. Several statistical methods for extracting ontological knowledge have been identified in the literature:

1. *Co-occurrence or collocation detection* identifies the occurrence of some words in the same sentence, paragraph or document. Such occurrences hint a potential direct relation between words [40]. These techniques can be used to discover terms that are siblings to each other [24].
2. *Clustering* can be used to create groups of similar words (clusters) which can be regarded as representing concepts, and further hierarchically organize these as clusters. This technique is generally used for learning concepts by considering clusters of related terms as concepts and learning taxonomies by organizing these groups hierarchically [5]. Ontology alignment can use agglomerative clustering to find candidate groups of similar entities in ontologies [38].
3. Hidden Markov Models (HMMs) define a generative statistical models that are able to generate data sequences according to rather complex probability distributions and that can be used for classifying sequential patterns [41–43]. Zhou and Su [44] have used HMM for Named Entity Recognition; Maedche and Staab [8] have used the n-gram models based on HMMs to process documents at the morphological level before supplying them to term extraction tools. Labsky et al. [29] present the use of HMMs to extract information on product offered by companies from HTML files.

2.3.3 Multi-Strategy learning

Multi-Strategy learning techniques leverage the strengths of the above techniques to extract a wide range of ontological knowledge from different types of knowledge sources [25, 26, 30]. for example, Maeche and Staab [8] present the use of clustering for concept learning and association rules to learn relations between these concepts.

2.4 Ontology learning evaluation

After the extraction process, the evaluation phase permits to know whether the knowledge extracted is accurate and to conclude on the quality of the knowledge source. The evaluation of ontological knowledge is coined by several authors in the literature [45, 46]. Dellschaft and Staab [46] have proposed two ways to evaluate ontological knowledge: (1) In manual evaluation by human experts, the knowledge is presented to one or more domain experts who have to judge to what extent it is correct; (2) The comparison of the knowledge to existing reference vocabularies/ontologies to ensure that it covers the studied domain.

3 Ontology learning from Java source code using Hidden Markov Models

Source code contains well-defined words in a language that everyone understands (for example the elements generally found on the user interface), some statements with a particular lexicon specific to the programming language and to the programmer. For example, in Java programming

language, the term "class" is used to define a class, the terms "if", "else", "switch", "case" are used to define the business rules (candidate to become rules). Other terms defined by the programmer such as "PatientTuberculeux" are used to represent the names of classes (candidate to be concept); the term "examenATB" is used to define the relation (ObjectProperty) with cardinality (candidate to become axiom) between the classes "PatientTuberculeux" and "Examen"; and the group of terms "int agePatient" is used to define a property (DataProperty) of the class "PatientTuberculeux". This section shows how to define, train and use Hidden Markov Models (HMMs) for knowledge extraction from Java source code.

3.1 Hidden Markov Models

A Markov Chain is a random process having a finite set of states, and only the current state influences where it goes next [41]. Hidden Markov Models are particular types of Markov Chain composed of a finite state automaton with edges between any pair of states that are labeled with transition probabilities. It also describes a 2-stage statistical process in which the behavior of the process at a given time t is only dependent on the immediate predecessor state. It is characterized by the probability between states $P(q_t|q_1, q_2, \ldots, q_{t-1}) = P(q_t|q_{t-1})$ and for every state at time t an output or observation o_t is generated. The associated probability distribution is only dependent on the current state q_t and not on any previous states or observations: $P(o_t|o_1, \ldots, o_{t-1}, q_1, \ldots, q_t) = P(o_t|q_t)$ [41, 43, 47–49]. HMMs are generally used for pattern recognition, automatic voice processing, automatic natural language processing, character recognition [41].

A first order HMM perfectly describes the source code because it can be seen as a string sequence typed by a programmer in which the current word (corresponding to an assign hidden state) depends on the previous word. In this HMM, the observed symbol depends only on the current state [41–43]. Equation 1 presents the joint probability of a series of observations $O_{1:T}$ given a series of hidden states $Q_{1:T}$. The HMM of Fig. 1 shows how the source code can be modeled using a HMM. In this figure, the observations are the words ("public", "class", "Patient", etc.) typed by the programmers and each of these words are labeled by the hidden states "PRE", "TARGET", "POST", and "OTHER".

$$P(O_{1:T}, Q_{1:T}) = P(q_1)P(o_1|q_1)\prod_{t=2} P(q_t|q_{t-1})P(o_t|q_t) \quad (1)$$

Filtering, smoothing, prediction, and the most likely explanation are four uses of HMMs. The probability that a string O is emitted by a HMM M is calculated as the sum of all possible paths by the equation 2.

$$P(O \mid M) = \sum_{q_1,\ldots,q_l} \prod_{k=1}^{l+1} P(q_{k-1} \to q_k)P(q_k \uparrow o_k) \quad (2)$$

Where q_0 and q_{l+1} are limited to q_I and q_N respectively and o_{l+1} is an end of word. The observable output of the system is the sequence of symbols emitted by the states, but the underlying state sequence itself is hidden.

In the most likely explanation, the goal is to find the sequence of hidden states $V(O \mid M)$ that best explains the sequence of observations (equation 3) [41–43]. To this end, the sequence of states $V(O \mid M)$ which has the greatest probability to produce an observation sequence is searched.

For example, in automatic translation, one may want the most probable string sequence that corresponds to the string to be translated. In this case, instead of taking the sum of the probabilities, the maximum must be chosen (equation 3).

$$P(O \mid M) = max_{\, q_1 \ldots q_l \in Q^l} \prod_{k=1}^{l+1} P(q_{k-1} \to q_k)P(q_k \uparrow o_k) \quad (3)$$

Before using the model, its parameters (transition probabilities, emission probabilities and initial probabilities) must be calculated using statistical learning, Baum-Welch algorithm or Viterbi training [41].

3.2 Source code versus HMM

During software development, it is recommended to write the source code according to good programming practices, including naming conventions [50]. These practices inform programmers on how to name variables, organize and present the source code. This organization can be used to model source code using HMMs (see Fig. 1). For example, from Java source code, we can say that at a time t, the programmer enters a word (e.g. "public" at the beginning of a Java source file). Thus, the keyword "public" at time t conditions the next word at time $t+1$ which in this case can be "class", "int", etc. We can say that PRE and $TARGET$ are the hidden states and "public" and "class" are respectively their observations.

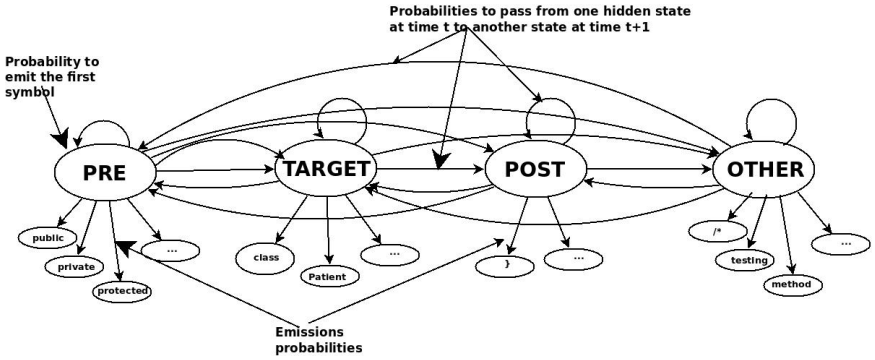


Figure 1: An example of HMM modeling the Java source code

Source code contains several types of files: files describing data, files processing data, user interface files and configuration files.

3.2.1 Files describing data

These files describe the data to be manipulated and equally, some constraints on this data (e.g., data types). In Java EE for example, there are entities whose names are close to the terms of the domain that will be transformed into tables in the database. These files often contain certain rules to verify the reliability of the data. Thus, from these files, we can retrieve concepts, properties, axioms and rules.

3.2.2 Files containing data processing

Located between user interface files and data description files is the data processing files of the source code consisting of:

- **Control:** For example, restricting certain data from certain users (e.g., only the attending physician has the right to access the data), checking the validity of a field (checking whether the data entered in an "age" field is of type integer);
- **Calculation:** For example, converting a date of birth into an age, determining the date of the next appointment of a patient, calculating the body mass index of a patient based on his/her weight and height.

These are the algorithms implementing the business rules to be applied to the data. They are thus good candidates for axioms and rules extraction.

3.2.3 User interfaces files

The User interfaces are composed of files which describe the information that will be presented to users for data viewing or recording. Unlike the first two files types, these files contain the words of a human-readable vocabulary that can be found in a dictionary. User interfaces usually provide:

- Translations allowing navigation from one language to another, control for users to enter the correct data;
- An aid allowing users to know for example, the role of a data entry field.

User Interfaces are therefore good candidates for concepts and their definitions, properties, axioms and rules extraction.

3.2.4 Configuration files

These files allow developers to specify certain information such as the type and path of a data source, different languages used by users, etc. For instance, from these files, the languages labels (e.g. English, French, Spanish) for terms can be extracted.

The files we just presented generally contain comments that can be useful for knowledge extraction or ontology documentation. Knowledge extraction from user interfaces/web interfaces has already been addressed in [12, 24], knowledge extraction from text has been presented in [5, 7, 8, 23]. In this article, we will focus on knowledge extraction from files describing data and their processing.

3.3 Knowledge extraction process

To extract knowledge from Java source code, we designed a method divided into five main steps: data collection, data

preprocessing, entity labeling, formal language translation, and knowledge validation.

3.3.1 Data collection

The data collection step consist of the extraction of a dataset necessary for the next steps. In Java files, statements for importing third-party libraries and comments are deleted. We proposed the definition of a regular expression that allow them to be identified.

3.3.2 Data preprocessing

The purpose of data preprocessing is to put data in a form compatible with the tools to be used in the next steps. During this phase, potentially relevant knowledge will be identified and retrieved, and some entities will be recoded. The problem of extracting knowledge from the source code has been reduced to the problem of syntactic labeling. This is to determine the syntactic label of the words of a text [42]. In our case, it will be a matter of assigning a label to all the words of the source code and extracting the words marked as target words. This problem can be solved using HMMs [42, 43]. In the following paragraphs, we will first present the HMM structure for source code modeling. Then, we will show how this HMM is trained and finally, how it is used to extract the knowledge from Java source code.

HMMs structure definition. To define the structure of the HMMs, we manually studied the organization of the source code of Java language. Generally, data structures, attributes, and conditions are surrounded by one or more specific words. Some of these words are predefined in advance in the programming language. To label the source code, we have defined four labels, corresponding to four hidden states of the HMM:

- **PRE:** Corresponding to the preamble of the knowledge. This preamble is usually defined in advance;
- **TARGET:** The target, (i.e. the knowledge sought) may be preceded by one or more words belonging to the PRE set. The knowledge we are looking for are the names of classes, attributes, methods, and the relationships between classes. They are usually preceded by a meta-knowledge which describes them. For example, the meta-knowledge "class" allows for concept identification;
- **POST:** Any information that follows the knowledge sought. In some cases, POST is a punctuation character or braces;
- **OTHER:** Any other word in the source code that neither precedes nor follows the knowledge sought.

An example of HMM annotated with labels is given by Fig. 1. Concepts, properties, axioms, and rules are usually arranged differently in the source code. We propose the definition of two HMMs which permit them to be identified: one to identify concepts, properties, axioms and the other one to identify rules.

Learning Model Parameters. There are several techniques to determine the parameters of a HMM: Statistical learning on data, specialized algorithms such as Baum-Welch or Viterbi training [41, 42]. In this paper, we have chosen statistical learning on data to train the HMMs modeled in the previous paragraphs. Thus, we assumed that we have access to T source code files labeled f_t knowing that f_t is not just a sequence of words, but a sequence of words pairs with the word and its label (see Fig. 1) modeled by the equation 4. To train the model, we assume that we can define the order in which the different words are entered by the programmer. We assume that before entering the first word, the programmer reflects on the label of that word and as a function of it, defines the label of the next word and so on. For example, before entering the word *public*, the programmer knows that its label is *PRE* and that the label of the next word is *TARGET*. Thus, the current word depends only on the current label, the following label depends on the previous label, and so on. The process continues until the end of the file.

$$\begin{aligned} f_t &= [(w_1^t, e_1^t), \ldots, (w_d^t, e_d^t)], \\ words(f_t) &= [w_t^t, \ldots, w_d^t], \\ labels(f_t) &= [e_1^t, \ldots, e_d^t]. \end{aligned} \tag{4}$$

In the equation 4, w_i and e_i are words and labels of f_i files respectively. In practice, w_i are words contained in the source code (observations) and e_i are the labels of w_i used as hidden states.

From the training data, we can extract statistics on:

- The first label $P(q_1)$ (equation 5). A priori probability that the first label is equal to the word $'a'$ is the number of times the first label in each file of the source code is the word $'a'$ divided by the number of source code files.

$$P(Q_1 = a) = \frac{\sum_t freq(e_1^t = a, f_t)}{T} \tag{5}$$

- The relation between a word and its label $P(O_k \mid q_k)$ (equation 6). The conditional probability that the k^{th} word is $'w'$, knowing that the label is $'b'$ corresponds to the number of times the word $'w'$ associated with

the label $'b'$ in the source code file f_t normalized with the fact that the label $'b'$ is associated with any other word in f_t source code. For example, "Patient" can be a concept, an attribute, but cannot be a rule.

$$P(O_k = w \mid q_k = b) = \frac{\alpha + \sum_t freq((w, b), f_t)}{\beta + \sum_t freq(('\star, b), f_t)} \quad (6)$$

To avoid zero probabilities for observations that do not occur in the training data, we added smoothing terms (α and β).

– The relation between the adjacent syntactic label is $P(q_k \mid q_{k+1})$ (equation 7). The probability that q_{k+1} is equal to label $'a'$ knowing that q_k is equal to label $'b'$ (previous hidden state) is the number of times $'a'$ follows $'b'$ in the source code of the training data divided by the number of times that $'b'$ is followed by any other label.

$$P(q_{k+1} = a \mid q_k = b) = \frac{\alpha + \sum_t freq(b, a), label(f_t)}{\beta + \sum_t freq(b, \star'), label(f_t)} \quad (7)$$

To avoid zero probabilities for transitions that do not occur in the training data, we added smoothing terms (α and β).

Let us consider the HMM in Fig. 1. Then, training data to identify concepts and attributes would be: [("public", PRE), ("class", TARGET), ("Patient", TARGET), ("extends", TARGET), ("ImogEntityImpl", TARGET), ("{", OTHER), (...), ("int", TARGET), ("age", TARGET), ...]. Tab. 1 presents the initial vector, which is the probability that the first label is PRE, TARGET, POST, or OTHER; Tab. 2 presents the transition vector containing the frequencies that a state follows another state; and Tab. 3 presents the emission vector containing the frequencies that a state emits an observation.

Knowledge extraction. The model previously defined and trained can be applied to any Java source code in order to identify *TARGET* elements. It will be necessary to find from the files f_1, ..., f_n, a sequence of states q_1, ..., q_n that is plausible. For this, equation 3 will be used to determine the most plausible string sequence. From this string, the hidden states will be identified and the targets (words that are labeled *TARGET*) will be extracted. In our approach, we used Viterbi algorithm which provides an efficient way of finding the most plausible string sequence of hidden states [51, 52]. The algorithm 1 gives an overview of the Viterbi Algorithm. More details can be found in [41].

Any source code can then be submitted to the HMM trained and a table similar to Tab. 10 containing the probability for the hidden states to emit a word from the source code is built.

Let $M = (\pi, A, B)$ *our HMM*
With π the vector of start probabilities, A the matrix of state-transition probabilities, and B the matrix of observation probabilities
Let $\delta_t(i) =$
$max_{q_1,...,q_{t-1}} P(O_1, ..., O_t, q_1, ...q_{t-1}, q_t = i|M)$
1. Initialization
$\delta_1(i) := \pi_i b_i(O_1) \quad \psi_1(i) := 0$
2. Recursion
For all times t, t_1, ..., $T - 1$:
$\delta_{t+1}(j) := max_i\{\delta_t(i)a_{ij}\}b_j(O_{t+1})$
$\psi_{t+1}(j) := argmax_i\{\delta_t(i)a_{ij}\}$
3. Termination
$P^\star(O|M) = P(O, q^\star|M) = max_i\delta_T(i)$
$q_T^\star := argmax_j\delta_T(j)$
4. Back-Tracking of the Optimal Path
for all times t, $t = T - 1, ..., 1$:
$q_t^\star = \psi_{t+1}(q_{t+1}^\star)$

Algorithm 1: The Viterbi algorithm [41, 52]

Recoding variables. Programmers usually use expressions made up of words from a specific lexicon, sometimes encoded with "ad hoc" expressions, requiring specific processing to assign a new name or a label understandable by humans before using. These words are generally divided into words or groups of words according to the naming conventions of the programming language. For example, we can have "PatientTuberculeux" → "Patient tuberculeux", "agePatient" → "Age Patient", "listeExamens" → "liste Examens", etc. Therefore, during the recoding, these names are separated in order to find their real sense in human understandable language.

3.3.3 Entities labeling

The extraction of relevant terms has yielded knowledge and meta-knowledge. This knowledge and meta-knowledge will permit us identify to which ontological components they may belong to. For example, the code: "class Patient extends Person int age", submitted to a trained HMM to identify concepts and relations will identify three meta-knowledge ("class", "extends" and "int") that will be used to identify two concepts (Patient and Person), one attribute of type integer and a hierarchical relation between "Patient" and "Person". From the extracted knowledge, two candidates to be concepts are related if one is declared in the structure of the other. One may identify three types of relations:

Table 1: The initial vector - probability to have a state as the first label

f(PRE)	f(TARGET)	f(POST)	f(OTHER)

Table 2: An example of a transition table

States	PRE	TARGET	POST	OTHER
PRE	f(PRE,PRE)	f(PRE,TARGET)	f(PRE,POST)	f(PRE,OTHER)
TARGET	f(TARGET,PRE)	f(TARGET,TARGET)	f(TARGET,POST)	f(TARGET,OTHER)
POST	f(POST,PRE)	f(POST,TARGET)	f(POST,POST)	f(POST,OTHER)
OTHER	f(OTHER,PRE)	f(OTHER,TARGET)	f(OTHER,POST)	f(OTHER,OTHER)

- ObjectProperty: If two classes 'A' and 'B' are candidates to be concepts and 'b' of type B is declared as attribute of class 'A', then classes 'A' and 'B' are related. The attribute 'b' is an ObjectProperty having 'A' as domain and 'B' as range.
- DatatypeProperty: If a class 'A' is a candidate to be a concept and contains the attributes 'a' and 'b' of basic data types (integers, string, boolean, etc.), then, 'a' and 'b' are DatatypeProperty having the class 'A' as domain;
- Taxonomy (subClassOf): If two classes 'A' and 'B' are candidates to be concepts and the class 'B' extends the class 'A' (in Java, the keyword "extends" is used), then, one can define a taxonomic relation between the classes 'B' and 'A'.

3.3.4 Translation in a formal language

Once all relevant knowledge are identified in the previous phase, they are automatically translated to a machine readable language. We use OWL for concepts, properties and axioms, and SWRL for rules.

3.3.5 Knowledge evaluation

After the extraction process, the evaluation phase permits us to know if this knowledge is relevant to the related domain and to conclude on the relevance in using source code as a knowledge source. Given that the knowledge extracted is ontological knowledge, two evaluation techniques will be used: (1) Manual evaluation by human experts in which the knowledge extracted is presented to one or more domain experts who have to judge to what extent these knowledge are correct; (2) The comparison of the knowledge extracted (alignment) to gold standards which will be existing ontologies.

3.4 HMMs definition, training and use

To extract knowledge from Java source code, two HMMs have to be defined and trained: a HMM for concepts, properties, and axioms identification, and a HMM for rules identification. All the algorithms for HMMs training and usage have been coded in Java[2].

3.4.1 HMM structure for concepts, properties and axioms

The HMM used to identify concepts, properties and axioms is defined by:

1. $PRE = \{public, private, protected, static, final\}$, the set of words that precedes TARGET;
2. $TARGET = \{package, class, interface, extends, implements, abstract, enum, w_i\}$, $\forall i, w_{i-1} \in PRE \,||\, w_{i-2} \in PRE \wedge w_{i-1} \in PRE$, the set of all words that we are seeking;
3. $POST = \{\{, ;, \}\}$, the set of words that follow TARGET;
4. $OTHER = \{w_i\}, w_i \notin PRE, \wedge w_i \notin TARGET, \wedge w_i \notin POST$, the set of all other words.

Each HMM state emitted a term corresponding to a word from the source code. We have seen that the observation emitted by the PRE set can be enumerated. However, the observation of $TARGET$ and $OTHER$ sets cannot be enumerated because they depend on the programmer. Then, we considered $data$ to be all the observations emitted by $TARGET$ and $other$ to be all the observations emitted by $OTHER$. We obtained the HMM presented by an initial vector (e.g., Tab. 4) a transition vector (e.g., Tab. 5), and an observation vector (e.g., Tab. 6).

Table 3: An example of an observation table

	package	pac	;	public	class	patient	...
PRE	f(PRE,package)	f(PRE, pac)	f(PRE,;)	f(PRE,public)	f(PRE,class)	f(PRE,patient)	...
TARGET	f(TARGET,package)	f(TARGET, pac)	f(TARGET,;)	f(TARGET,class)	f(TARGET,patient)	...	
POST	f(POST,package)	f(POST, pac)	f(POST,;)	f(POST,public)	f(POST,class)	f(POST,patient)	...
OTHER	f(OTHER,package)	f(OTHER, pac)	f(OTHER,;)	f(OTHER,public)	f(OTHER,class)	f(OTHER,patient)	...

3.4.2 HMM structure for rules

Rules can be contained in conditions. Then, we will exploit the structure of source code to extract the rules. For example, the portion of code (if (agePatient> 21) {Patient = Adult}) is a rule determining whether a patient is an adult or not. It must therefore be extracted.

The HMM to identify the rules is composed of:

1. PRE = {"}", ";", "{"}, the set of words that precede one or more TARGET;
2. $TARGET = \{if, else, switch, w_i\} \mid \exists k, r \in N \mid w_{i-k} \in PRE \wedge wi + r \in POST$: the set of all words that follow PRE and precede $POST$;
3. $POST$ = {"}"}, the end of the condition;
4. $OTHER = \{w_i\} \mid w_i \notin PRE, TARGET, POST$: the set of all other words.

We can identify the beginning and the end of a condition represented here by the sets PRE and $POST$ respectively. Note that all the observations emitted by $TARGET$ and $OTHER$ sets cannot be fully enumerated. Therefore, we have considered *data* to be all the observations emitted by $TARGET$, and $other$ to be all the observations emitted by $OTHER$.

3.4.3 Statistical learning of the HMMs

LearnJava source code (composed of 59 files and 2663 statements) was downloaded from github[3] and from this source code, we used statistical learning on data presented in section 3.3.2 to calculate the values of the HMMs parameters[4]. Tabs 4, 5, 6, 7, 8, 9 present the initialization, transition and observation vectors respectively obtained after the training step.

3.4.4 Knowledge extraction

Once the HMMs are built, we can apply them to the source code of any Java applications in order to extract the knowledge. To do this, the most likely state sequence (equation 3) that produced this source code is calculated. To calculate the most likely state sequence, we have implemented the Viterbi algorithm [41, 51, 52] in Java[5]. In fact, we have exploited the structure of the HMM in the context of dynamic programming. It consists of breaking down the calculations into intermediate calculations which are structured in a table. An example of Viterbi table is given by the Tab. 10. Every element of the table is being calculated using the previous ones. From this table, the Viterbi path is retrieved by getting the frame with the highest probability in the last column and given this frame, to search all the frames that were used to build it. All the elements whose labels are $TARGET$ are extracted as candidates.

4 Experimentation

This section presents the experimentation of the approach described in section 3. This experimentation consists in extracting ontological knowledge from EPICAM source code composed of 1254 Java files and 271782 instructions. Fig. 2 presents a screenshoot of some concepts from the EPICAM source code.

4.1 Knowledge extraction from EPICAM

To extract ontological knowledge from EPICAM source code, we proceeded step by step using the method presented in section 3.

4.1.1 Data collection

The source files of EPICAM platform are composed of statements, imported libraries and comments. Data collection involves removing the imported libraries and comments. To this end, we defined the regular expression $import[\backslash\backslash u0000\backslash - \backslash\backslash ufffff]$*?;|$\backslash\backslash$(.) * $\backslash n$|(/$\backslash\backslash$* [$\backslash\backslash u0000\backslash - \backslash\backslash ufffff$]*?$\backslash\backslash$*/) to identify them. Once identified, we wrote a Java program to delete them.

Table 4: The initial vector of the HMM for concepts, properties and axioms extraction

PRE	TARGET	POST	OTHER
0.0	1.0	0.0	0.0

Table 5: Transition vector of the HMM for concepts, properties and axioms extraction

	PRE	TARGET	POST	OTHER
PRE	0.1686	0.8260	0.0027	0.0027
TARGET	0.0008	0.7523	0.2461	0.0008
POST	0.0603	0.0033	0.0234	0.9130
OTHER	0.7364	0.1133	0.0025	0.1478

Table 6: Observation vector of the HMM for concepts, properties and axioms extraction

	public	private	protected	static	final	data	{	;	}	other
PRE	0.6417	0.1684	0.0053	0.1124	0.0722	0.0	0.0	0.0	0.0	0.0
TARGET	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
POST	0.0	0.0	0.0	0.0	0.0	0.0	0.6678	0.3256	0.0066	0.0
OTHER	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0

Table 7: The initial vector of the HMM for rules extraction

PRE	TARGET	POST	OTHER
0.0	0.0	0.0	1.0

Table 8: Transition vector of the HMM for rules extraction

	PRE	TARGET	POST	OTHER
PRE	0.0667	0.7999	0.0667	0.0667
TARGET	0.0010	0.9321	0.0659	0.0010
POST	0.0172	0.0172	0.0172	0.9484
OTHER	0.0072	0.0001	0.0001	0.9926

Table 9: Observation vector of the HMM for rules extraction

	{	}	;	if	else
PRE	0.8462	0.0769	0.0769	0.0	0.0
TARGET	0.0	0.0	0.0	0.0185	0.0031
POST	0.0	1.0	0.0	0.0	0.0
OTHER	0.0	0.0	0.0	0.0	0.0

	switch	data	other
PRE	0.0	0.0	0.0
TARGET	0.0010	0.9774	0.0
POST	0.0	0.0	0.0
OTHER	0.0	0.0	1.0

4.1.2 Data preprocessing

Data preprocessing consists in extracting the elements likely to be relevant from the source code and recoding them if necessary. We have used the HMMs defined and trained in section 3.4. These HMMs were applied to the source code of EPICAM by calculating the values of the Viterbi table (see Tab. 10). Once the table is built, we searched the Viterbi path by getting the frames with the highest probability in the last column and using this frame, we search all the frames that were used to build it. Once the Viterbi path is identified, all the elements labeled *TARGET* are extracted.

Fig. 3 presents the set of candidates for concepts, properties, and axioms identified and Fig. 4 presents the set of candidates for rules identified.

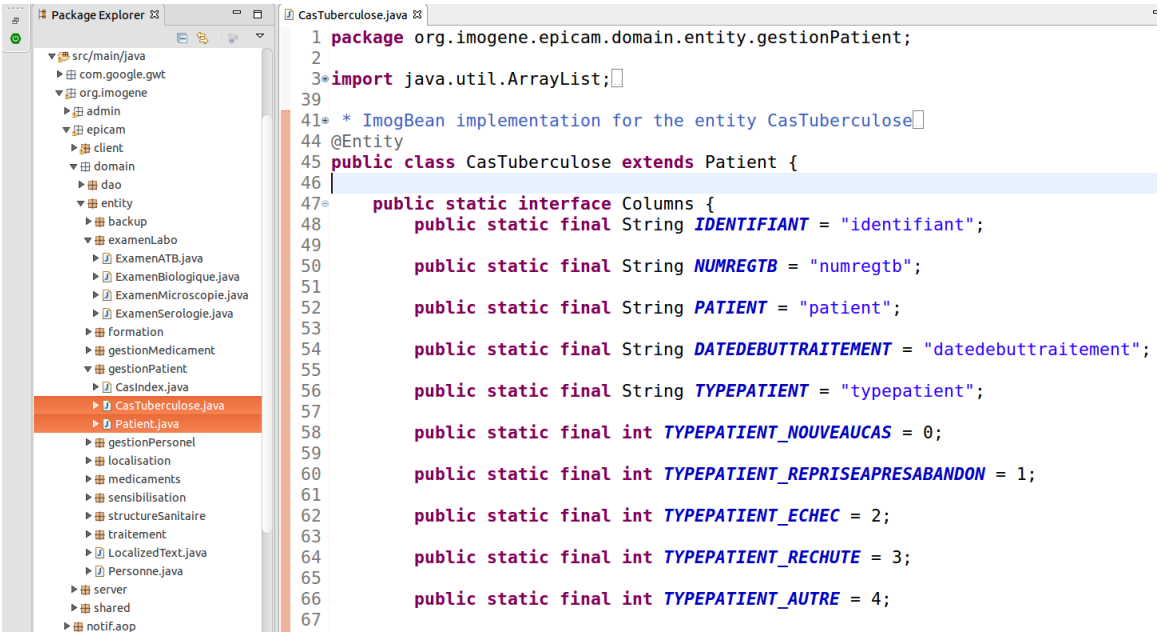


Figure 2: An overview of the Java source code of the EPICAM project

Table 10: The Viterbi table (α table) built using EPICAM source code

	package	org.epicam	;	public	...	}
PRE	0	$\alpha(PRE, 2)$	$\alpha(PRE, 3)$	$\alpha(PRE, 4)$	...	$\alpha(PRE, t)$
TARGET	1	$\alpha(TARGET, 2)$	$\alpha(TARGET, 3)$	$\alpha(TARGET, 4)$	...	$\alpha(TARGET, t)$
OTHER	0	$\alpha(OTHER, 2)$	$\alpha(OTHER, 3)$	$\alpha(OTHER, 4)$	...	$\alpha(OTHER, t)$

4.1.3 Recoding terms and rules

To recode the candidates extracted, we used Java naming conventions. All the candidates were browsed and for the candidates containing the keywords of the programming language, these keywords were removed. For example, consider the term *CasTuberculoseEditorWorkflow* that was extracted from the source code; the terms *Editor* and *Workflow* are keywords of Google Web Toolkit, the technology used to build the EPICAM platform. Then, the terms *Editor* and *Workflow* are removed and the term *CasTuberculose* is retained as candidate.

After the recoding, we moved to the next step which is the translation into formal language.

4.1.4 Entities identification and translation into OWL

Data preprocessing phase produced a file containing only the meta-knowledge (e.g "package", "class", "extends", "if", "switch") and the knowledge (e.g "patientManagement.Patient", "Patient" or "serology"). We wrote a Java program to browse these files in order to identify the knowledge that may be useful. Meta-knowledge allow the identification of the candidates as concepts, properties and axioms. For example, if the string "package minHealth.Region.District.hospitals.patientRecord ... class Patient extends Person ... int age ... List<Exam> listExam" is extracted, then, the following ontological knowledge is identified:

- **"package minHealth.Region.District.hospitals.patientRecord:"** This is used to identify the class hierarchy;
- **"class Patient extends Person":** This expression means that "Patient" and "Person" are candidates that will become concepts and there is a hierarchical relation between concepts "Patient" and "Person";
- **"int age; List <Exam> listExam":** This expression means that "age" and "listExam" are properties of the concept "Patient"; the following axiom is also defined: a patient has only a single age (i.e. age is a functional property).

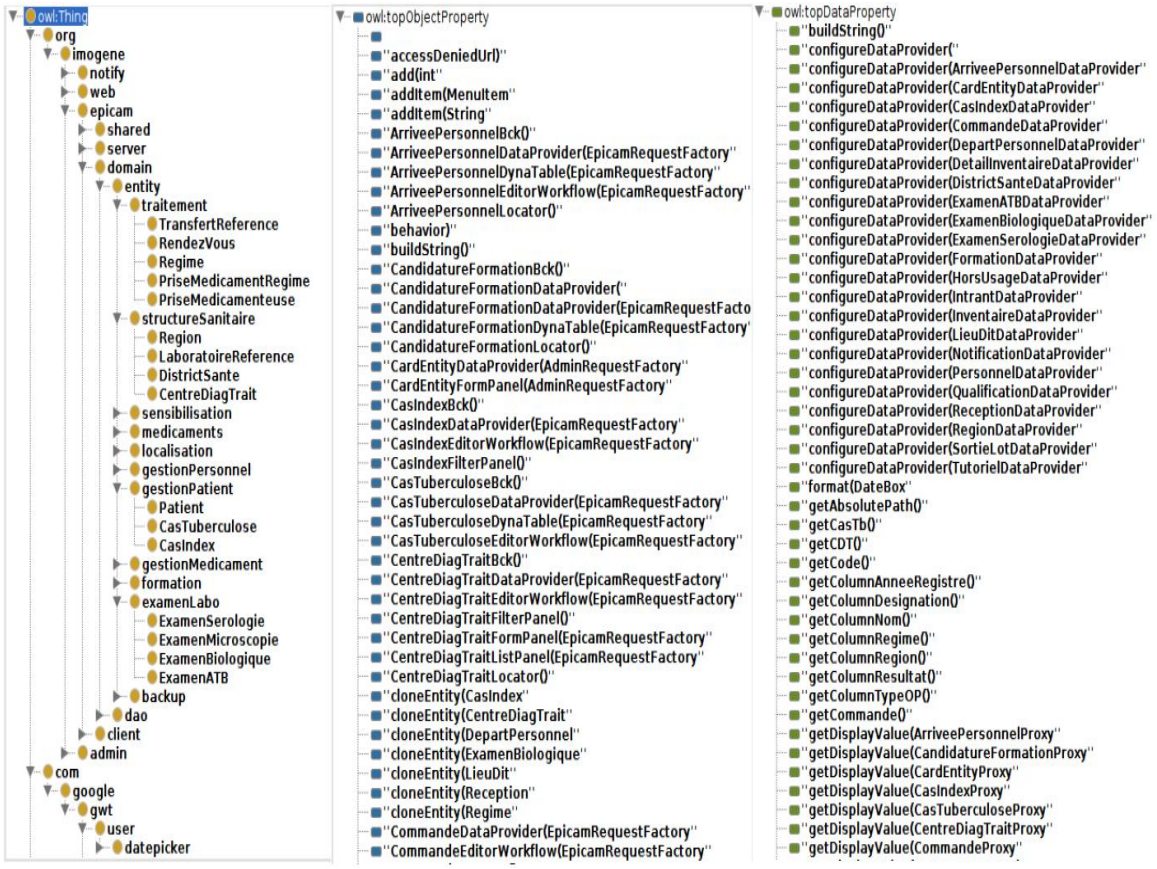


Figure 3: An excerpt of candidates extracted for concepts, properties and axioms

```
if (AccessManager.canDirectAccessPatient()&&AccessManager.canReadPatient()){
Commandcommand=newCommand(){publicvoidexecute(){LocalSession.get().setSearchCriterions
(null,null);History.newItem(TokenHelper.TK_LIST+"/patient/",true);

If(AccessManager.canDirectAccessCasTuberculose()&&AccessManager.canReadCasTuberculose()){
Commandcommand=newCommand(){publicvoidexecute(){LocalSession.get().setSearchCriterions
(null,null);History.newItem(TokenHelper.TK_LIST+"/castuberculose/",true);

if(AccessManager.canDirectAccessExamenATB()&&AccessManager.canReadExamenATB()){
Commandcommand=newCommand(){publicvoidexecute(){LocalSession.get().setSearchCriterions(null,null);
History.newItem(TokenHelper.TK_LIST+"/examenatb/",true);

if(AccessManager.canCreatePatient()&&AccessManager.canEditPatient())patient=
newImogMultiRelationBox<PatientProxy>(patientDataProvider,EpicamRenderer.get(),null);
else patient = newImogMultiRelationBox<PatientProxy>(false,patientDataProvider,EpicamRenderer.get(),null);

if(poidsMin.getValueWithoutParseException()==null&&poidsMin.isValid())delegate.recordError
(BaseNLS.messages().error_required(),null,"poidsMin");//poidsMinshallbesuperiororequalto'0'

switch(typeCAs){case0:nouveauCas++;LOGGER.debug("xxxxxxxxxNombredenouveauxcas:
"+nouveauCas);break;case1:repriseTrait++;LOGGER.debug("xxxxxxxxxNombrederetraitementcas:
"+repriseTrait);break;case2:echecs++;break;case3:rechuttes++;break;default:break;
```

Figure 4: An excerpt of candidates extracted for rules identification

After the identification of entities, we proposed a second Java program[6] to automatically translate them into an OWL ontology[7].

In the same way, rules were also extracted and translated into Semantic Web Rule Language[8]. An example of a rule specifying the rights of a doctor on patient data is given by:

doctorsRule = "Personnel (?pers) ∧ personnel_login (?pers, login) ∧ personnel_passwd (?pers, passwd) ∧ Patient (?p) ∧ RendezVous (?rdv) ∧ hasRDV (?rdv, ?p) ∧ patient_nom (?p, ?nom) ∧ patient_age (?p, ?age) ∧ patient_sexe (?p, ?sexe) ∧ patient_telephoneUn (?p, ?telephone) ∧ rendezVous_dat eRendezVous (?rdv, ?datardv) ∧ rendezVous_honore (?rdv, ?honore) ∧ rendezVous_honore (?rdv, Non) → sqwrl:select (?nom, ?age, ?sexe, ?telephone, ?datardv, ?honore)";

4.2 Analysis of the elements extracted

The extraction process produced a set of candidates (Figs 3 and 4), but also false positives (Tab. 11 presents the statistics). The false positives consist of the set of candidates that belong to the *PRE*, *POST* or *OTHER* sets that normally should not be extracted as observations of *TARGET*. We wrote a Java program to identify and delete them.

Tab. 11 presents the statistics of candidates/group of candidates that were extracted. After the extraction process, we obtained different types of candidates/group of candidates:

- **Irrelevant candidates/group of candidates:** These are utility classes and temporary variables. Utility classes are classes that the programmer defines to perform certain operations. These classes usually contain constants and methods. The names of these classes are usually not related to the domain. Temporary variables (e.g., the variables used in a loop) are used temporarily in the source code and are not related to the domain.
- **Relevant candidates/group of candidates:** These are knowledge found. These candidates are composed of synonyms (candidates of identical meaning) and redundancies (candidates that come up several times). We wrote a Java program to identify and remove redundancies candidates automatically.

We also extracted candidates conditions to be rules. As we did with the candidates to be concepts, properties and axioms, false positives were identified and deleted. From the rules extracted, we found:

- **Irrelevant conditions:** These are conditions that are not really important. For example, testing whether a temporary variable is positive or is equal to a certain value. These conditions were the most numerous;
- **Relevant conditions:** Conditions corresponding to a business rule (e.g., testing if a user has access right to certain data).

Table 11: Statistics on candidates extracted

Candidates	Relevant	Irrelevant
Concepts	1840 (72.87%)	685 (27.13%)
Properties	38355 (81.42%)	8755 (18.58%)
Axioms	3397 (83.22%)	685 (16.78%)
Rules	1484 (07.89%)	17332 (92.11%)

5 Evaluation

The concepts, properties and axioms extracted were translated into an OWL ontology. The extracted rules are represented in SWRL. We used the Protege editor to provide a graphical visualization of the ontology and rules to human experts for their evaluation. Fig. 5 presents an overview of the ontology obtained.

Three experts from the tuberculosis surveillance domain involved in the EPICAM project were invited to evaluate the knowledge extracted. They are from three different organizations in Cameroon (Centre Pasteur of Cameroon, National Tuberculosis Control Program and a hospital in Yaounde). The domain experts were asked to check first if the terms extracted are relevant to the tuberculosis clinical or epidemiological perspectives. Second, they analyzed the axioms and rules. First of all, they found that the terminology was relevant to the tuberculosis. However, they suggested to correct some typos caused by the names of the classes and attributes given by programmers. Axioms and rules were generally correct. Some rules were suggested to be updated as the business rules have evolved (e.g. user access to patient data has been improved taking into account their post such as epidemiologist, physician, nurse or administrative staff).

In line with the experts validation, we evaluated the coverage of the ontology terms by taking reference on other ontologies in the biomedical domain. We used BioPortal [53] as a biomedical ontology repository. BioPortal contains more than 300 ontologies including a large number of medical terminologies such as SNOMED (Systematized Nomenclature of Medicine) [54]. BioPortal has an Ontology Recommender module that is used to find the best ontologies for a biomedical text or a set of keywords [55]. This task is done according to four criteria: (1) the extent to which the ontology covers the input data; (2) the acceptance of the ontology in the biomedical community; (3) the level of detail of the ontology classes that cover the input data; (4) and the specialization of the ontology to the domain of the input data. We gave as input keywords to the Recommender the set of terms (concepts and properties)

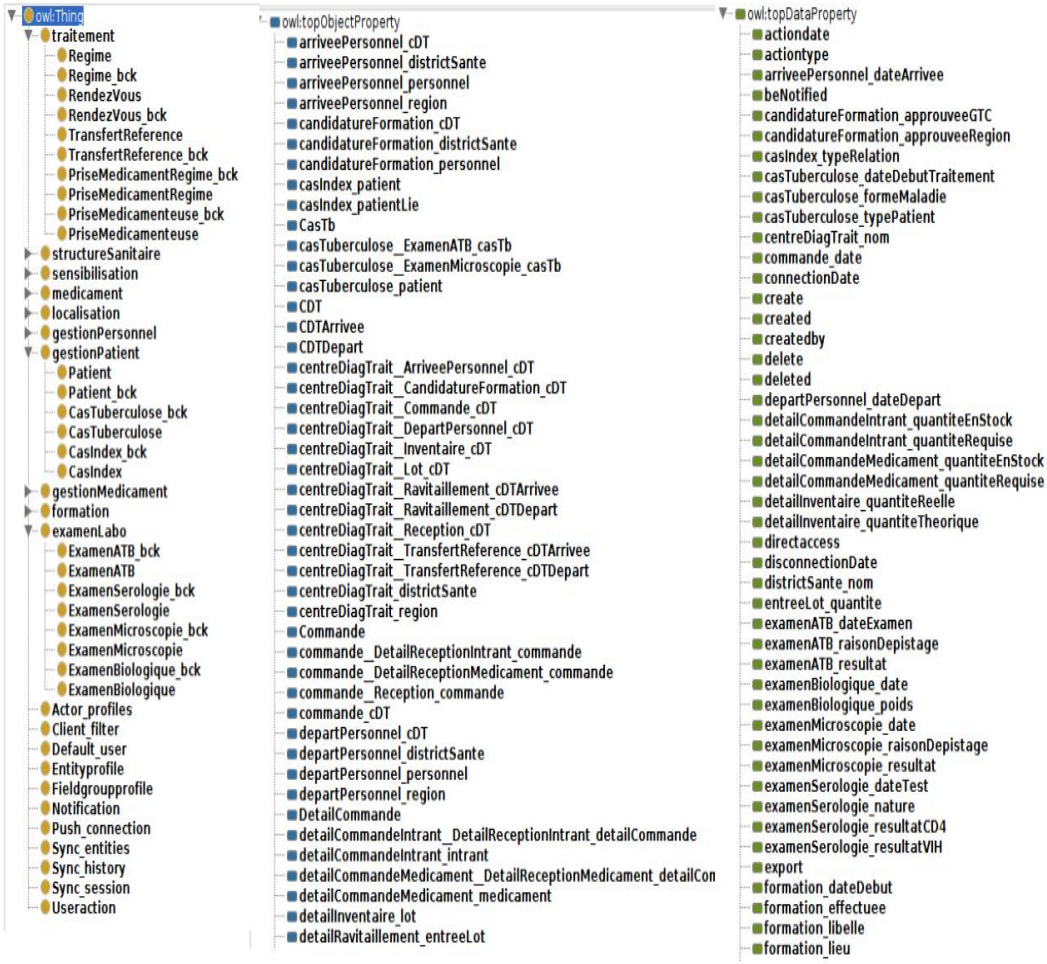


Figure 5: An overview of the generated OWL ontology

of the ontology extracted by our HMM. Fig. 6 shows that the ontology terms are covered by many biomedical ontologies. In the first line of the recommended ontologies, we could see that NCIT, SNOWMEDCT, ONTOPARON (accepted by the community with a score of 75.6%) cover the terms from our ontology with a score of 82.9%, have a level of details of 64% and the level of specialization of 40%. We came to the conclusion that terms extracted by our HMM are relevant to the biomedical domain.

At the end of the evaluation, we conclude that EPICAM source code contains ontological knowledge that can be used as a relevant basis to build and/or enrich an ontology for the tuberculosis surveillance domain.

6 Related work

Despite the large amount of available source codes and the fact that they may contain relevant knowledge of the domain [12, 21–23] addressed by the software, the number of existing work on knowledge extraction from these knowledge sources is quite low. Parser-based approach and machine learning techniques are the commonly used in knowledge extraction from source code.

6.1 Parser-based approach

A straightforward solution to extract knowledge from source code is to use a parser. There are works in this direction for generating knowledge base (RDF triples) or extracting ontological knowledge (concepts and properties) from source codes using parsers. For instance, CodeOntology [20, 56] parser is able to analyze Java source code and serialize it into RDF triples. From these triples, highly expressive queries using SPARQL (SPARQL Protocol and RDF Query Language) can be executed for different software engineering purposes including the searching of specific software component for reuse. Ganapathy and Sagayaraj

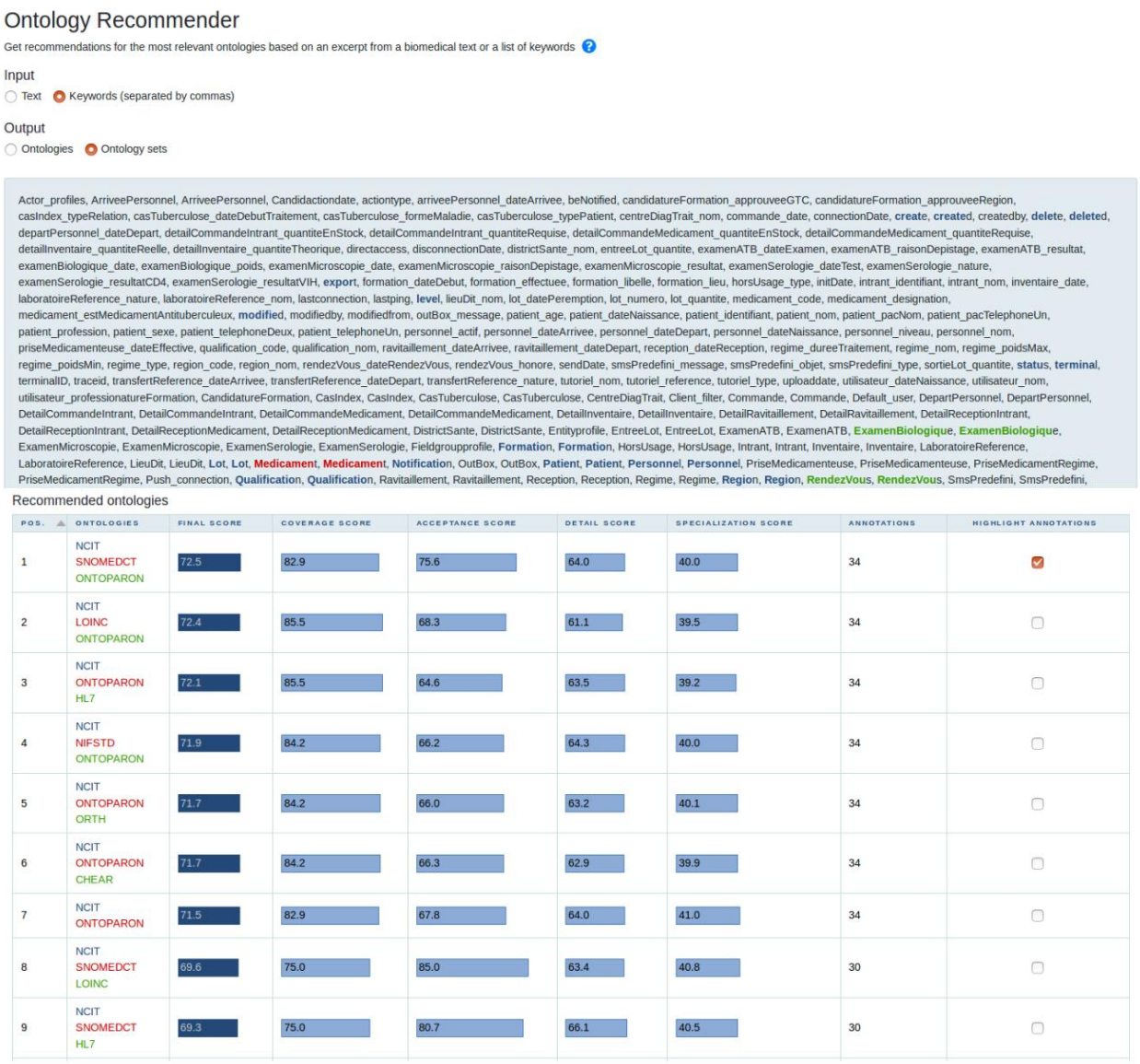


Figure 6: The Ontology Recommender output from the extracted ontology terms

[28] used QDox[9] generator to generate an ontology that will further enable the developers to reuse source code efficiently. QDox generator is a parser that can be used for extracting classes, attributes, interfaces and method definition from Java source code. In the approach proposed by [12], the authors defined the components parts of the source code and break down the source code into these components. The source code is browsed and the different components are analyzed in order to take an appropriate action which is the extraction of knowledge sought. This knowledge can be used in supplementing and assisting ontology development from database schemas.

Beyond RDF triples, terms, concepts and properties extraction, existing parsers do not provide services for axioms and rules extraction. To overcome these limits, they need to be improved. However, building and/or updating parsers for programming languages is a non-trivial, laborious and time-consuming task [57, 58].

6.2 Machine learning-based approach

Machine learning approaches are also proposed to extract knowledge from source code.

Kalina Bontcheva and Marta Sabou [23] have presented an approach for ontology learning from software artifacts such as software documentation, discussion forums and source code by using the language processing

facilities provided by GATE 2 platform[10]. GATE 2 is an Open source software developed in Java for building and deploying Human Language Technology application such as parsers, morphology, tagging, Information Retrieval tools, Information Extraction components, etc. To extract concepts from source code, Kalina Bontcheva and Marta Sabou used the GATE key phrase extractor, which is based on TF.IDF (term frequency/inverted document frequency). The TD.IDF approach is an unsupervised machine learning technique which consists of finding words/phrases that are characteristic of the given text, while ignoring phrases that occur frequently in the text simply because they are common in the language as a whole. When using TF.IDF on the source code, high frequency terms specific to the programming language can be eliminated and only terms specific to the given software project would be selected as relevant to the domain (ontology concept). This approach is used to extract concept. However, ontological knowledge is also made up of properties, axioms and rules.

Labsky et al. [29] presented an approach for information extraction on product offered by companies from their websites. To extract information from HTML documents, they used Hidden Markov Models to annotate these documents. Tokens modelled by this HMM include words, formatting tags and images. The HMM is modelled using four states: the target state (T) which is the slot to extract, the prefix and the suffix state (P, S) which constitute the slot's context, and the irrelevant tokens modelled by a single background state (B). This approach permitted the extraction of slots and the relation between nearby slots. For example product image often follows its name. Unlike the authors approach which consists of terms extraction, our approach uses meta-data extracted from source code in order to identify to which ontological component every term/group of terms corresponds to.

7 Conclusion and future work

In this paper, we proposed an approach for knowledge extraction from Java source code using Hidden Markov Models (HMMs). We experimented this approach by extracting ontological knowledge from EPICAM, a tuberculosis epidemiological surveillance platform developed in Java. Evaluation by domain experts (clinicians and epidemiologists) permitted us to show the relevance of the knowledge extracted. In line with the experts validation, we evaluated the coverage of terms extracted by reference ontologies in biomedical domain. We used Ontology Recommender from BioPortal repository. The results of the evaluation shows that the terms are well covered by many biomedical ontologies (e.g., NCIT, SNOWMEDCT, ONTOPARON).

Our goal in this paper was twofold: (1) to show that source code contains ontological knowledge that could be used in domain ontology engineering and (2) to show how to define, train and use HMMs to extract these knowledge. Since we have used the statistical learning on data approach to calculate the parameters of the HMMs, our future work consists of experimenting the Baum-welch and Viterbi training approaches. The performance of these three approaches will be evaluated and compared to the parser approach.

Notes

[1] http://www.medes.fr/fr/nos-metiers/la-e-sante-et-l-epidemiologie/la-tele-epidemiologie/projet-epicam.html

[2] https://github.com/jiofidelus/source2onto

[3] https://github.com/mafudge/LearnJava

[4] https://github.com/jiofidelus/source2onto/blob/master/code2onto-model/src/main/java/cm/uy1/training/HMMTrainingData.java

[5] https://github.com/jiofidelus/source2onto/blob/master/code2onto-model/src/main/java/cm/uy1/modelUse/KnowledgeExtractionHMM.java

[6] https://github.com/jiofidelus/source2onto/blob/master/code2onto-model/src/main/java/cm/uy1/helper/OWLHelper.java

[7] https://github.com/jiofidelus/ontologies/blob/master/epicam/epicam.owl

[8] https://github.com/jiofidelus/ontologies/blob/master/epicam/epicamrules.owl

[9] https://github.com/paul-hammant/qdox

[10] https://gate.ac.uk/

References

[1] Studer R., Benjamins V.R., Fensel D., Knowledge Engineering: Principles and Methods, Data Knowl. Eng., 1998, 25(1-2), 161–197, 10.1016/S0169-023X(97)00056-6

[2] Gómez-Pérez A., Fernández-López M., Corcho Ó., Ontological Engineering: With Examples from the Areas of Knowledge Management, e-Commerce and the Semantic Web, Advanced Information and Knowledge Processing, Springer, 2004, 10.1007/b97353

[3] Konys A., Knowledge systematization for ontology learning methods, in Knowledge-Based and Intelligent Information & Engineering Systems, Proceedings of the 22nd International Conference KES-2018, Belgrade, Serbia, 3-5 September 2018., 2018, 2194–2207, 10.1016/j.procs.2018.07.229

[4] Suárez-Figueroa M.C., Gómez-Pérez A., Fernández-López M., The NeOn Methodology framework: A scenario-based methodology for ontology development, Applied Ontology, 2015, 10(2),

107–145, 10.3233/AO-150145
[5] Cimiano P., Ontology learning and population from text - algorithms, evaluation and applications, Springer US, 2006, 10.1007/978-0-387-39252-3
[6] Ghosh M.E., Naja H., Abdulrab H., Khalil M., Ontology Learning Process as a Bottom-up Strategy for Building Domain-specific Ontology from Legal Texts, In Proceedings of the 9th International Conference on Agents and Artificial Intelligence, ICAART 2017, Volume 2, Porto, Portugal, February 24-26, 2017., 2017, 473–480, 10.5220/0006188004730480
[7] Alexander M., Raphael V., The Ontology Extraction & Maintenance Framework Text-To-Onto, In International Conference on Data Mining (ICDM), San Jose, USA, November 29 - December 2, 2001, IEEE, Los Alamitos (CA), 2001
[8] Alexander M., Steffen S., Semi-automatic engineering of ontologies from text, Proceedings of the 12th Internal Conference on Software and Knowledge Engineering. Chicago, USA, 2000
[9] Cerbah F., Lammari N., Ontology Learning from Databases: Some Efficient Methods to Discover Semantic Patterns in Data, in A..I.P. Serie, ed., Perspectives in Ontology Learning, 2014, 30
[10] Cullot N., Ghawi R., Yétongnon K., DB2OWL : A Tool for Automatic Database-to-Ontology Mapping, In Proceedings of the Fifteenth Italian Symposium on Advanced Database Systems, SEBD 2007, 17-20 June 2007, Torre Canne, Fasano, BR, Italy, 2007, 491–494
[11] Idrissi B.E., Baïna S., Baïna K., Ontology Learning from Relational Database: How to Label the Relationships Between Concepts?, In Beyond Databases, Architectures and Structures - 11th International Conference, BDAS 2015, Ustroń, Poland, May 26-29, 2015, Proceedings, 2015, 235–244, 10.1007/978-3-319-18422-7_21
[12] Zhao S., Chang E., Dillon T.S., Knowledge extraction from web-based application source code: An approach to database reverse engineering for ontology development, In Proceedings of the IEEE International Conference on Information Reuse and Integration, IRI 2008, 13-15 July 2008, Las Vegas, Nevada, USA, 2008, 153–159, 10.1109/IRI.2008.4583022
[13] Hacherouf M., Bahloul S.N., Cruz C., Transforming XML documents to OWL ontologies: A survey, Journal of Information Science, 2015, 41(2), 242–259, 10.1177/0165551514565972
[14] Leung N.K.Y., Lau S.K., Tsang N., Reuse existing ontologies in an ontology development process - an integration-oriented ontology development methodology, International Journal of Web Science, 2014, 2(3), 159–180, 10.1504/IJWS.2014.066435
[15] Pinto H., Gómez-Pérez A., Martins J., Some Issues on Ontology Integration, In Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI 99) Workshop: KRR5: Ontologies and Problem-Solving Methods: Lesson Learned and Future Trends, volume 18, 1999
[16] Smith B., Ashburner M., Rosse C., Bard J., Bug W., Ceusters W., al., The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration, Nature biotechnology, 2007, 25(11), 1251–1255, 10.1038/nbt1346
[17] Bouihi B., Bahaj M., An UML to OWL based approach for extracting Moodle's Ontology for Social Network Analysis, Procedia Computer Science, 2019, 148, 313 – 322, https://doi.org/10.1016/j.procs.2019.01.039, the Second International Conference on Intelligent Computing in Data Sciences, ICDS2018
[18] Djuric D., Gasevic D., Devedzic V., Ontology Modeling and MDA, Journal of Object Technology, 2005, 4(1), 109–128, 10.5381/jot.2005.4.1.a3
[19] Xu Z., Ni Y., He W., Lin L., Yan Q., Automatic extraction of OWL ontologies from UML class diagrams: a semantics-preserving approach, World Wide Web, 2012, 15(5-6), 517–545, 10.1007/s11280-011-0147-z
[20] Atzeni M., Atzori M., CodeOntology: RDF-ization of Source Code, In The Semantic Web - ISWC 2017 - 16th International Semantic Web Conference, Vienna, Austria, October 21-25, 2017, Proceedings, Part II, 2017, 20–28, 10.1007/978-3-319-68204-4_2
[21] Azanzi F.J., Camara G., Knowledge Extraction from Source Code Based on Hidden Markov Model: Application to EPICAM, In 14th IEEE/ACS International Conference on Computer Systems and Applications, AICCSA 2017, Hammamet, Tunisia, October 30 - Nov. 3, 2017, 2017, 1478–1485, 10.1109/AICCSA.2017.99
[22] Azanzi F.J., Camara G., An Approach for Knowledge Extraction from Source Code (KNESC) of Typed Programming Languages, In Trends and Advances in Information Systems and Technologies - Volume 1 [WorldCIST'18, Naples, Italy, March 27-29, 2018]., 2018, 122–131, 10.1007/978-3-319-77703-0_12
[23] Bontcheva K., Learning Ontologies from Software Artifacts: Exploring and Combining Multiple Choices., In J.Z. Pan, Y. Zhao, eds., Semantic Web Enabled Software Engineering, volume 17 of Studies on the Semantic Web, IOS Press, 2014, 235–250
[24] Brunzel M., The XTREEM Methods for Ontology Learning from Web Documents., In P. Buitelaar, P. Cimiano, eds., Ontology Learning and Population: Bridging the Gap between Text and Knowledge, volume 167 of Frontiers in Artificial Intelligence and Applications, IOS Press, 2008, 3–26
[25] Asim M.N., Wasim M., Khan M.U.G., Mahmood W., Abbasi H.M., A survey of ontology learning techniques and applications, Database, 2018, 2018, bay101, 10.1093/database/bay101
[26] Shamsfard M., Barforoush A.A., The state of the art in ontology learning: a framework for comparison, The Knowledge Engineering Review, 2003, 18(4), 293–316, 10.1017/S0269888903000687
[27] Unbehauen J., Hellmann S., Auer S., Stadler C., Knowledge Extraction from Structured Sources, in S. Ceri, M. Brambilla, eds., Search Computing: Broadening Web Search, volume 7538 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, 34–52, 10.1007/978-3-642-34213-4_3
[28] Ganapathy G., Sagayaraj S., To Generate the Ontology from Java Source Code, International Journal of Advanced Computer Science and Applications, 2011, 2(2), 10.14569/IJACSA.2011.020218
[29] Labský M., Svátek V., Sváb O., Praks P., Krátký M., Snásel V., Information Extraction from HTML Product Catalogues: From Source Code and Images to RDF, in 2005 IEEE / WIC / ACM International Conference on Web Intelligence (WI 2005), 19-22 September 2005, Compiegne, France, 2005, 401–404, 10.1109/WI.2005.78
[30] Zhou L., Ontology learning: state of the art and open issues, Information Technology and Management, 2007, 8(3), 241–252, 10.1007/s10799-007-0019-5
[31] Hitzler P., Krötzsch M., Rudolph S., Foundations of Semantic Web Technologies, Chapman and Hall/CRC Press, 2010
[32] Kharbat F., El-Ghalayini H., Building Ontology from Knowledge Base Systems, Data Mining in Medical and Biological Research, 2008, 10.5772/6407
[33] García-Silva A., García-Castro L.J., Castro A.G., Corcho Ó., Building Domain Ontologies Out of Folksonomies and Linked Data,

International Journal on Artificial Intelligence Tools, 2015, 24(2), 10.1142/S021821301540014X

[34] Wang S., Wang W., Zhuang Y., Fei X., An ontology evolution method based on folksonomy, Journal of Applied Research and Technology, 2015, 13(2), 177 – 187

[35] Fahad M., ER2OWL: Generating OWL Ontology from ER Diagram, In Intelligent Information Processing IV, 5th IFIP International Conference on Intelligent Information Processing, October 19-22, 2008, Beijing, China, 2008, 28–37, 10.1007/978-0-387-87685-6_6

[36] Hazman M., El-Beltagy S.R., Rafea A., A Survey of Ontology Learning Approaches, International Journal of Computer Applications, 2011, 22(8), 36–43

[37] Lisi F.A., Learning Onto-Relational Rules with Inductive Logic Programming, CoRR, 2012, abs/1210.2984

[38] Wróblewska A., Podsiadly-Marczykowska T., Bembenik R., Protaziuk G., Rybinski H., Methods and Tools for Ontology Building, Learning and Integration Application in the SYNAT Project, in R. Bembenik, L. Skonieczny, H. Rybinski, M. Niezgodka, eds., Intelligent Tools for Building a Scientific Information Platform, volume 390 of Studies in Computational Intelligence, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, 121–151, 10.1007/978-3-642-24809-2_9

[39] Li Y., Krishnamurthy R., Raghavan S., Vaithyanathan S., Jagadish H.V., Regular Expression Learning for Information Extraction, in 2008 Conference on Empirical Methods in Natural Language Processing, EMNLP 2008, Proceedings of the Conference, 25-27 October 2008, Honolulu, Hawaii, USA, A meeting of SIGDAT, a Special Interest Group of the ACL, 2008, 21–30

[40] Kolesnikova O., Survey of Word Co-occurrence Measures for Collocation Detection, Computación y Sistemas, 2016, 20(3), 327–344

[41] Fink G.A., Markov Models for Pattern Recognition: From Theory to Applications, Advances In Computer Vision and Pattern Recognition, Springer-Verlag, London, 2 edition, 2014

[42] Russell S.J., Norvig P., Artificial Intelligence - A Modern Approach, Third International Edition, Pearson Education, 2010

[43] Seymore K., Mccallum A., Rosenfeld R., Learning Hidden Markov Model Structure for Information Extraction, In AAAI 99 Workshop on Machine Learning for Information Extraction, 1999, 37–42

[44] Zhou G., Su J., Named Entity Recognition using an HMM-based Chunk Tagger, In Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA., 2002, 473–480

[45] Amith M., He Z., Bian J., Lossio-Ventura J.A., Tao C., Assessing the practice of biomedical ontology evaluation: Gaps and opportunities, Journal of Biomedical Informatics, 2018, 80, 1–13, 10.1016/j.jbi.2018.02.010

[46] Dellschaft K., Staab S., Strategies for the Evaluation of Ontology Learning, In Proceedings of the 2008 Conference on Ontology Learning and Population: Bridging the Gap Between Text and Knowledge, IOS Press, Amsterdam, The Netherlands, The Netherlands, 2008, 253–272

[47] Eddy S.R., What is a hidden Markov model?, Nature Biotechnology, 2004, 22(10), 1315, 10.1038/nbt1004-1315

[48] Franzese M., Iuliano A., Hidden Markov Models, in S. Ranganathan, M. Gribskov, K. Nakai, C. SchAnbach, eds., Encyclopedia of Bioinformatics and Computational Biology, Academic Press, Oxford, 2019, 753 – 762, https://doi.org/10.1016/B978-0-12-809633-8.20488-3

[49] Kouemou G.L., History and Theoretical Basics of Hidden Markov Models, Hidden Markov Models, Theory and Applications, 2011, 10.5772/15205

[50] Binkley D., Davis M., Lawrie D., Morrell C., To camelcase or under_score, in 2009 IEEE 17th International Conference on Program Comprehension, 2009, 158–167, 10.1109/ICPC.2009.5090039

[51] Forney G.D., The Viterbi Algorithm: A Personal History, CoRR, 2005, abs/cs/0504020

[52] Viterbi A.J., Viterbi algorithm, Scholarpedia, 2009, 4(1), 6246, 10.4249/scholarpedia.6246

[53] Whetzel P.L., Noy N.F., Shah N.H., Alexander P.R., Nyulas C., Tudorache T., Musen M.A., BioPortal: enhanced functionality via new Web services from the National Center for Biomedical Ontology to access and use ontologies in software applications, Nucleic Acids Research, 2011, 39(Web-Server-Issue), 541–545, 10.1093/nar/gkr469

[54] Silva T.S.D., MacDonald D., Paterson G.I., Sikdar K.C., Cochrane B., Systematized nomenclature of medicine clinical terms (SNOMED CT) to represent computed tomography procedures, Computer Methods and Programs in Biomedicine, 2011, 101(3), 324–329, 10.1016/j.cmpb.2011.01.002

[55] Romero M.M., Jonquet C., O'Connor M.J., Graybeal J., Pazos A., Musen M.A., NCBO Ontology Recommender 2.0: an enhanced approach for biomedical ontology recommendation, Journal of Biomedical Semantic, 2017, 8(1), 21:1–21:22, 10.1186/s13326-017-0128-y

[56] Atzeni M., Atzori M., CodeOntology: Querying Source Code in a Semantic Framework, In Proceedings of the ISWC 2017 Posters & Demonstrations and Industry Tracks co-located with 16th International Semantic Web Conference (ISWC 2017), Vienna, Austria, October 23rd - to - 25th, 2017., 2017

[57] Fenwick M., Weatherby G., Ellis H.J.C., Gryk M.R., Parser Combinators: A Practical Application for Generating Parsers for NMR Data, In Tenth International Conference on Information Technology: New Generations, ITNG 2013, 15-17 April, 2013, Las Vegas, Nevada, USA, 2013, 241–246, 10.1109/ITNG.2013.39

[58] Nierstrasz O., Kurs J., Parsing for agile modeling, Science of Computer Programming, 2015, 97, 150–156, 10.1016/j.scico.2013.11.011