



UNIVERSIDAD
NACIONAL
DE COLOMBIA
SEDE MEDELLÍN

Doctoral Thesis
Doctoral Program in Engineering
-Systems and Informatics-

**Metamodel for personalized adaptation of pedagogical strategies using metacognition
in Intelligent Tutoring Systems**

Manuel Fernando Caro Piñeres
mfcarop@unal.edu.co

Tutor:
Jovani Alberto Jiménez Builes Ph.D.

Departamento de Ciencias de la Computación y la Decisión
Facultad de Minas
Universidad Nacional de Colombia -Medellín
2015

Note thesis approval

Approved as to style and content by:

MICHAEL T. COX (Ph.D.)
Thesis committee Member

DARSANA P. JOSYULA (Ph.D.)
Thesis committee member

GUSTAVO A. RAMIREZ (Ph.D.)
Thesis committee member

ACKNOWLEDGMENTS

My first thanks go to God for His infinite love. I believe that all understanding and knowledge comes from God.

I thank my wife, Keyla Acosta, and my daughters, Karla and Manuela, for their unconditional love and unwavering support. They made tremendous sacrifices so that I could conclude this stage of my life and for that I'm eternally grateful. I could not have done this without them.

I would like to thank my advisor and mentor Jovani Jiménez. For his valuable and timely guidance, as well as their dedication to successfully complete this work.

I thank my extended family in Montería and Cereté for their support and encouragement. They provided all the support, love, care and dedication, an example of strength, work, determination and persistence that occurred during lifetime. I especially thank to Rodrigo Caro, Sol Piñeres, Carmen Caro and Jairo Rodríguez.

I'm indebted to a number of people for their advice and generosity over the years. I thank them for keeping me grounded. I'm grateful to Darsana Josyula for her support and for our extended discussions on many topics. She appreciated my work and allowed me to conduct the research internship under her mentorship in the United States, which was one of the most enriching experiences in my life. Her patience and friendship are invaluable to me. Michael Cox taught me the essence of research on metacognition in computation. Today I appreciate his patience and mentorship. I would like to thank to Catriona Kennedy for her contributions in the development of M++.

Finally, I thank the faculty, students and staff in the Departamento de Informática at Universidad de Córdoba for an extraordinary education.

TABLE OF CONTENTS

1	INTRODUCTION.....	15
1.1	MOTIVATION	15
1.2	CHALLENGES.....	16
1.3	THESIS PROJECT	16
1.3.1	<i>Research problem.....</i>	<i>16</i>
1.3.2	<i>Research question.....</i>	<i>18</i>
1.3.3	<i>Objectives</i>	<i>19</i>
1.3.4	<i>Methodology</i>	<i>19</i>
1.3.5	<i>Contributions.....</i>	<i>21</i>
1.3.6	<i>Document organization.....</i>	<i>22</i>
2	THEORETICAL BACKGROUND	23
2.1	MODELING OF PEDAGOGICAL STRATEGIES IN ITS.....	23
2.1.1	<i>Pedagogical strategies</i>	<i>23</i>
2.1.2	<i>Learning theories and pedagogical strategies</i>	<i>23</i>
2.2	INTELLIGENT TUTORING SYSTEMS (ITS)	25
2.2.1	<i>Expert Module.....</i>	<i>25</i>
2.2.2	<i>Student Module.....</i>	<i>26</i>
2.2.3	<i>Tutor module</i>	<i>26</i>
2.2.4	<i>User Interface Module.....</i>	<i>27</i>
2.2.5	<i>Pedagogical models in ITS.....</i>	<i>27</i>
2.2.6	<i>Personalized adaptation of pedagogical strategies in ITS.....</i>	<i>30</i>
2.3	METACOGNITION IN ITS.....	31
2.3.1	<i>Metacognition in intelligent systems</i>	<i>31</i>
2.3.2	<i>Models, frameworks and architectures of metacognition in intelligent systems</i>	<i>35</i>
2.3.3	<i>Support of metacognitive components.....</i>	<i>39</i>
2.4	FRAMEWORK OF MODEL DRIVEN ARCHITECTURE (MDA).....	41
2.4.1	<i>Models.....</i>	<i>41</i>
2.4.2	<i>Transformation Model.....</i>	<i>42</i>
2.4.3	<i>Metamodel.....</i>	<i>44</i>
2.5	CONCLUSION OF THE CHAPTER	44
3	METAMODEL FOR PEDAGOGICAL MODULE.....	46
3.1	METAMODEL FOR PEDAGOGICAL MODULE IN INTELLIGENT SYSTEMS (METAGOGIC)	46
3.1.1	<i>Metacore package</i>	<i>49</i>
3.1.2	<i>Planner package.....</i>	<i>51</i>

3.1.3	<i>Advisor package</i>	54
3.1.4	<i>Assessment package</i>	56
3.1.5	<i>User package</i>	58
3.2	CONCLUSION OF THE CHAPTER	61
4	METAMODEL FOR METACOGNITION SUPPORT IN INTELLIGENT SYSTEMS..	62
4.1	METAMODEL FOR METACOGNITION SUPPORT IN INTELLIGENT SYSTEMS (MISM).....	62
4.1.1	<i>metacore Package</i>	65
4.1.2	<i>Self-Regulation package</i>	68
4.1.3	<i>selfregulation.monitoring package</i>	68
4.1.4	<i>selfregulation.control package</i>	71
4.1.5	<i>Metamemory package</i>	73
4.1.6	<i>metamemory.monitoring package</i>	73
4.1.7	<i>metamemory.control Package</i>	75
4.1.8	<i>Meta-comprehension package</i>	76
4.2	CONCLUSION OF THE CHAPTER	78
5	MOF-BASED METAMODEL FOR PERSONALIZATION OF PEDAGOGICAL STRATEGIES USING METACOGNITION IN ITS.....	79
5.1	MOF-BASED METAMODEL.....	79
5.1.1	<i>Elements of the conceptual architecture</i>	80
5.2	M ₂ - METAMODEL FOR PERSONALIZATION OF PEDAGOGICAL STRATEGIES USING METACOGNITION IN ITS (MPPSM).....	80
5.2.1	<i>General overview</i>	81
5.2.2	<i>Structure and organization</i>	82
5.2.3	<i>Semantic definitions for elements in MPPSM</i>	103
5.2.4	<i>Mapping Approach for MPPSM</i>	104
5.3	CONCRETE SYNTAX FOR THE DESIGN OF METACOGNITIVE FUNCTIONS IN ITS	109
5.3.1	<i>MetaThink tool</i>	110
5.4	EXAMPLE OF USE: DESIGN OF A METACOGNITIVE MODEL BASED ON MPPSM USING M++ ..	112
5.5	VALIDATION.....	114
5.5.1	<i>Empirical validation of M++</i>	114
5.5.2	<i>Configuration of the experiment</i>	114
5.5.3	<i>Data analysis</i>	115
5.5.4	<i>Model validation</i>	116
5.6	CONCLUSION OF THE CHAPTER	118
6	INTELLIGENT TUTORING SYSTEM FOR TEACHING INTRODUCTION TO PROGRAMMING - FUNPRO.....	120
6.1	OBJECT-LEVEL	121
6.1.1	<i>Multi-level Pedagogical model in FUNPRO</i>	122
6.1.2	<i>Personalization of pedagogical strategies</i>	127

6.1.3	<i>Learning environment</i>	132
6.2	VALIDATION.....	147
6.2.1	<i>Validation of self-regulation for monitoring and control of personalization of pedagogical strategies</i>	148
6.2.2	<i>Validation of metamemory in FUNPRO</i>	151
6.3	CONCLUSIONS OF THE CHAPTER	153
7	EVALUATION	155
7.1	ANSWERS TO RESEARCH QUESTIONS.....	155
7.2	CONTRIBUTIONS OF THE THESIS.....	160
7.3	PUBLICATIONS.....	161
7.3.1	<i>Articles published in international journals</i>	161
7.3.2	<i>Articles published in national journals</i>	161
7.3.3	<i>Papers presented at international events</i>	162
7.3.4	<i>Papers presented at national events</i>	162
7.3.5	<i>Book Chapters</i>	162
7.4	CONCLUSIONS OF THE CHAPTER	162
8	CONCLUSIONS AND FUTURE WORKS	163
8.1	CONCLUSIONS	163
8.2	FUTURE WORKS.....	165
8.2.1	<i>MPPSM metamodel</i>	165
8.2.2	<i>M++ and MetaThink</i>	165
8.2.3	<i>FUNPRO</i>	166
9	REFERENCES	167

LIST OF ACRONYMS

AI	Artificial Intelligence
AOL	Advance Of Learning
ART	Available Resources in Retrieval
AS	Artificial System
ATL	ATLAS Transformation Language
BLU	Basic Learning Units
CBR	Case-Based Reasoning
CIM	Computational Independent Model
CL	Collaborative Learning
COP	Certainty of Optimal Performance
CSRD	Certainty of Satisfying the Retrieval constraints
DSVL	Domain-Specific Visual Language
EMF	Eclipse Modeling Framework
EMOF	Essential MOF
FUNPRO	<i>FUNdamentos de PROgramación</i>
GDL	Goal Driven Learning
GUI	Graphical User Interface
IL	Introspective Learning
IML	Introspective Multi-strategy Learning
IP	Instructional Plan
ITS	Intelligent Tutoring System
IS	Intelligent System
ISM	Implementation Specific Model
LE	Learning by Experience
LO	Learning Objective
LTM	Long-Term Memory
MAS	Multi-Agent System
MCL	The Meta-Cognitive Loop
MDA	Model Driven Architecture
MDD	Model Driven Development
MDE	Model Driven Engineering
MPPSM	Metamodel of Personalized adaptation of pedagogical strategies using metacognition in Intelligent Tutoring Systems
MOF	Meta Object Facility
MP	Model Platform
OGI	On Going Learning
OMG	Object Management Group
PIM	Platform-Independent Model

PSM	Platform Specific Model
RDBS	Relational Database Schema
QVT	Query/View/Transformation
RL	Reinforcement Learning
SQL	Structured Query Language
SRL	Self-Regulated Learning
UML	Unified Modeling Language
URT	Unavailable Resources in Retrieval
WM	Working Memory

LIST OF FIGURES

FIGURE 2.1 ITS CLASSIC MODEL (<i>JEREMIĆ, JOVANOVIĆ, & GAŠEVIĆ, 2012; PHOBUN & VICHEANPANYA, 2010</i>)	25
FIGURE 2.2. BASIC STRUCTURE OF A PEDAGOGICAL MODEL IN AN ITS, BASED ON (<i>BEZERRA, 2012; PHOBUN & VICHEANPANYA, 2010</i>)	28
FIGURE 2.3. STRUCTURE OF AN INSTRUCTIONAL PLAN, BASED ON (<i>AGUILAR ET AL., 2011; ARIAS ET AL., 2009; ESCUDERO & FUENTES, 2010; LEGASPI, SISON, & NUMAO, 2004A; VICCARI & JIMÉNEZ, 2007</i>)..	29
FIGURE 2.4. METACOGNITIVE LOOP (<i>NELSON & NARENS, 1990</i>)	32
FIGURE 2.5. METACOGNITIVE ELEMENTS (<i>VOCKELL, 2004</i>).....	32
FIGURE 2.6. META-AQUA (<i>COX & RAM, 1999; GORDON, HOBBS, & COX, 2007</i>).....	36
FIGURE 2.7. MCL (<i>ANDERSON ET AL., 2006; SCHMILL ET AL., 2011</i>).....	36
FIGURE 2.8. SIMPLE MODEL FOR METAREASONING (<i>COX & RAJA, 2012</i>)	37
FIGURE 2.9. DMF (<i>KENNEDY & SLOMAN, 2003; KENNEDY, 2010</i>)	38
FIGURE 2.10. MIDCA (<i>COX ET AL., 2011</i>).....	38
FIGURE 2.11. RELATIONSHIP BETWEEN MDA, MDD AND MDE, GRAPHIC BASED ON (<i>KLEPPE, WARMER, & BAST, 2003</i>)	41
FIGURE 2.12. MOF ARCHITECTURE (<i>BRAGANÇA & MACHADO, 2008</i>).....	41
FIGURE 2.13. TRANSFORMATION STRUCTURE IN MDA (<i>KLEPPE ET AL., 2003; KOCH & GMBH, 2006</i>).....	43
FIGURE 2.14. TRANSFORMATIONS IN MDA, GRAPHIC BASED ON (<i>JOUAULT & KURTEV, 2006; MENS & VAN GORP, 2006</i>).....	43
FIGURE 2.15. SIMPLIFIED MOF METAMODEL AT META-METAMODELING LAYER (M3), (<i>OMG, 2013; RENSINK & NEDERPEL, 2008</i>).....	44
FIGURE 3.1. PACKAGE MODEL IN METAGOGIC METAMODEL; SOURCE: THE AUTHOR.	49
FIGURE 3.2. METACORE PACKAGE MODEL IN METAGOGIC; SOURCE: THE AUTHOR.	50
FIGURE 3.3. PLANNER PACKAGE MODEL IN METAGOGIC METAMODEL ; SOURCE: THE AUTHOR.....	52
FIGURE 3.4. ADVISOR PACKAGE MODEL IN METAGOGIC METAMODEL; SOURCE: THE AUTHOR.	55
FIGURE 3.5. ASSESSMENT PACKAGE MODEL IN METAGOGIC METAMODEL; SOURCE: THE AUTHOR.....	57
FIGURE 3.6. USER PACKAGE MODEL IN METAGOGIC METAMODEL; SOURCE: THE AUTHOR.....	60
FIGURE 4.1. PACKAGE MODEL IN MISM METAMODEL.....	64
FIGURE 4.2. INTERNAL STRUCTURE OF METACORE PACKAGE IN MISM METAMODEL (<i>CARO ET AL., 2014</i>)	65
FIGURE 4.3. INTERNAL STRUCTURE OF SELFREGULATION . MONITORING PACKAGE IN MISM METAMODEL (<i>CARO ET AL., 2014</i>)	69
FIGURE 4.4. INTERNAL STRUCTURE OF SELFREGULATION . CONTROL PACKAGE IN MISM METAMODEL. (<i>CARO ET AL., 2014</i>).....	72
FIGURE 4.5. INTERNAL STRUCTURE OF METAMEMORY . MONITORING PACKAGE IN MISM METAMODEL (<i>CARO ET AL., 2014</i>).....	74
FIGURE 4.6. INTERNAL STRUCTURE OF METAMEMORY . CONTROL PACKAGE IN MISM METAMODEL (<i>CARO ET AL., 2014</i>).....	76
FIGURE 4.7. INTERNAL STRUCTURE OF METACOMPREHENSION . MONITORING PACKAGE IN MISM METAMODEL (<i>CARO ET AL., 2014</i>)	77

FIGURE 5.1. CONCEPTUAL ARCHITECTURE OF MOF-BASED METAMODEL FOR PERSONALIZATION OF PEDAGOGICAL STRATEGIES USING METACOGNITION IN ITS; SOURCE: THE AUTHOR.	79
FIGURE 5.2. GENERAL OVERVIEW OF THE METACOGNITIVE LOOP IN MPPSM; SOURCE: THE AUTHOR.....	81
FIGURE 5.3. ORGANIZATION OF PACKAGES IN MPPSM METAMODEL.....	82
FIGURE 5.4. ECORE SPECIFICATION OF METACORE PACKAGE IN MPPSM.....	83
FIGURE 5.5. SPECIFICATION OF THE MPPSM.METAGOGIC.CORE PACKAGE; SOURCE: THE AUTHOR.....	86
FIGURE 5.6. THE MPPSM.MISM.CORE SPECIFICATION IN ECORE	87
FIGURE 5.7. THE MPPSM.MISM.CORE INTEGRATION DIAGRAM. CLASSESS IMPORTED FROM OTHER PACKAGES IN WHITE COLOR; SOURCE: THE AUTHOR.....	87
FIGURE 5.8. THE SELF-REGULATION PACKAGE SPECIFICATION IN ECORE	88
FIGURE 5.9. DEPENDENCY DIAGRAM OF SELFREGULATION.MONITORING PACKAGE.....	89
FIGURE 5.10. INTERNAL STRUCTURE OF MPPSM.MISM.SELFREGULATION.MONITORING PACKAGE. CLASES IMPORTED FROM OTHER PACKAGES IN WHITE COLOR; SOURCE: THE AUTHOR.	89
FIGURE 5.11. DEPENDENCY DIAGRAM OF SELFREGULATION.CONTROL PACKAGE.	90
FIGURE 5.12. INTERNAL STRUCTURE OF SELFREGULATION.CONTROL PACKAGE. CLASESS IMPORTED FROM OTHER PACKAGES IN WHITE COLOR; SOURCE: THE AUTHOR.....	90
FIGURE 5.13. INTERNAL STRUCTURE OF SELFREGULATION.CONTROL PACKAGE.....	91
FIGURE 5.14. DEPENDENCY DIAGRAM OF METAMEMORY.MONITORING PACKAGE.....	91
FIGURE 5.15. INTERNAL STRUCTURE OF METAMEMORY.MONITORING PACKAGE. CLASESS IMPORTED FROM OTHER PACKAGES IN WHITE COLOR; SOURCE: THE AUTHOR.....	92
FIGURE 5.16. DEPENDENCY DIAGRAM OF METAMEMORY.CONTROL PACKAGE.....	93
FIGURE 5.17. INTERNAL STRUCTURE OF METAMEMORY.CONTROL PACKAGE. CLASESS IMPORTED FROM OTHER PACKAGES IN WHITE COLOR; SOURCE: THE AUTHOR.	93
FIGURE 5.18. SELF-MODEL SPECIFICATION IN MPPSM; SOURCE: THE AUTHOR.	94
FIGURE 5.19. SPECIFICATION OF THE MPPSM.METAGOGIC.CORE PACKAGE IN MPPSM	95
FIGURE 5.20. SPECIFICATION OF THE MPPSM.METAGOGIC.CORE PACKAGE IN MPPSM. CLASES IMPORTED FROM OTHER PACKAGES IN WHITE COLOR; SOURCE: THE AUTHOR.	96
FIGURE 5.21. SPECIFICATION OF THE MPPSM.METAGOGIC.PLANNER PACKAGE IN MPPSM	96
FIGURE 5.22. DEPENDENCY DIAGRAM OF MPPSM.METAGOGIC.PLANNER PACKAGE	97
FIGURE 5.23. PLANNER PACKAGE MODEL. CLASESS IMPORTED FROM OTHER PACKAGES IN WHITE COLOR; SOURCE: THE AUTHOR.....	98
FIGURE 5.24. ADVISOR PACKAGE SPECIFICATION IN ECORE.....	98
FIGURE 5.25. DEPENDENCY DIAGRAM OF MPPSM.METAGOGIC.ASSESSMENT PACKAGE	99
FIGURE 5.26. CLASS DIAGRAM OF METAGOGIC.ADVISOR PACKAGE. CLASES IMPORTED FROM OTHER PACKAGES IN WHITE COLOR; SOURCE: THE AUTHOR.	99
FIGURE 5.27. ASSESSMENT PACKAGE SPECIFICATION IN ECORE	100
FIGURE 5.28. DEPENDENCY DIAGRAM OF MPPSM.METAGOGIC.ASSESSMENT PACKAGE	100
FIGURE 5.29. ASSESSMENT PACKAGE MODEL. CLASESS IMPORTED FROM OTHER PACKAGES IN WHITE COLOR; SOURCE: THE AUTHOR.....	101
FIGURE 5.30. USER PACKAGE SPECIFICATION IN ECORE	101
FIGURE 5.31. DEPENDENCY DIAGRAM OF MPPSM.METAGOGIC.USER PACKAGE	102
FIGURE 5.32. USER PACKAGE MODEL. CLASESS IMPORTED FROM OTHER PACKAGES IN WHITE COLOR; SOURCE: THE AUTHOR.....	102
FIGURE 5.33. ENDOGENOUS TRANSFORMATION REPRESENTATION IN MPPSM	105
FIGURE 5.34. EXOGENOUS TRANSFORMATION MODEL IN MPPSM, BASED ON (BEZIVIN ET AL., 2006)	106
FIGURE 5.35. MAIN ELEMENTS IN M++ NOTATION; (CARO, JOSYULA, JIMÉNEZ, KENNEDY, & COX, 2015)	110

FIGURE 5.36. PLUGIN-METATHINK GRAPHICAL USER INTERFACE; SOURCE: THE AUTHOR.....	111
FIGURE 5.37. METATHINK TOOLBAR.....	112
FIGURE 5.38. EXAMPLE OF A METACOGNITIVE MODEL GENETRATED FOR FUNPRO - ITS; SOURCE: THE AUTHOR.	112
FIGURE 5.39. EXAMPLE OF A METACOGNITIVE MODEL GENETRATED FOR AN ITS CORRESPONDING WITH FIGURE 5.38	113
FIGURE 5.40. RESULT OF THE VAIABLES: (A) USEFULNESS NOTATION AND (B) INTENTION TO USE IN THE EMPIRICAL STUDY.....	116
FIGURE 5.41. (A) SECTION OF MPPSM (LAYER M ₂ IN MOF) WITH OBJECT-LEVEL SPECIFICATION A PARTIAL VIEW OF INTROSPECTIVE MONITORING PROCESS AT META-LEVEL; (B) METACOGNITIVE MODEL AT M ₁ CONFORMS TO PARTIAL VIEW OF MPPSM IN SECTION A; (C) USER MODEL CONFORMS WITH THE METACOGNITIVE MODEL IN SECTION B.....	117
FIGURE 6.1. ARCHITECTURE OF DOUBLE-LOOP OF REASONING IN FUNPRO; SOURCE: THE AUTHOR.	120
FIGURE 6.2. MULTI-LEVEL PEDAGOGICAL MODEL IN LAYER M ₁ ACCORDING TO MPPSM METAMODEL AT M ₂	123
FIGURE 6.3. ONTOLOGY IN THEORY LEVEL; SOURCE: THE AUTHOR.....	124
FIGURE 6.4. ONTOLOGY IN TEACHING METHOD LEVEL; SOURCE: THE AUTHOR.	124
FIGURE 6.5. ONTOLOGY IN PEDAGOGICAL TACTIC LEVEL; SOURCE: THE AUTHOR.	125
FIGURE 6.6. ONTOLOGY IN ACTIVITY LEVEL; SOURCE: THE AUTHOR.....	126
FIGURE 6.7. ONTOLOGY IN RESOURCE LEVEL; SOURCE: THE AUTHOR.....	126
FIGURE 6.8. LEVEL M ₁ CONTAINS THE ONTOLOGICAL REPRESENTATION OF PEDAGOGICAL STRATEGY IN FUNPRO; LEVEL M ₁ CORRESPONDS TO MPPSM METAMODEL AT LEVEL M ₂	127
FIGURE 6.9. ONTOLOGY IN STUDENT MODEL	129
FIGURE 6.10. WELCOME PAGE IN FUNPRO WITH MENU SECTION, LOGIN SECTION AND WORKSPACE	132
FIGURE 6.11. IDENTIFICATION OF STUDENT’S PROFILE IN FUNPRO	133
FIGURE 6.12. DESCRIPTION OF THE INTERFACE COMPONENTS IN FUNPRO.....	134
FIGURE 6.13. EXPLANATION OF LESSON –“SENTENCIA SI”-. LEFT SIDE AN EXPLANATION FOR A VERBAL STUDENT. RIGHT SIDE AN EXPLANATION FOR A VISUAL STUDENT.....	135
FIGURE 6.14. NAVIGATION MODEL: A) NAVIGATION STYLE TAB FOR GLOBAL STUDENTS, B) NAVIGATION STYLE BUTTONS (NEXT-PREVIOUS) FOR SEQUENTIAL STUDENTS	136
FIGURE 6.15. METACOGNITIVE MODEL IN M++ OF THE MRP.....	138
FIGURE 6.16. BASIC FLOW OF INFORMATION IN THE INTROSPECTIVE MONITORING IMPLEMENTATION - FUNPRO	139
FIGURE 6.17. FLOW DIAGRAM OF FUNPRO WITH DIFFERENT SECTIONS REGARDING INFORMATION RETRIEVAL.....	143
FIGURE 6.18. RELATION BETWEEN RESOURCE ASSESSMENT AND (AVERAGE OF CHANGE AND PERFORMANCE AVERAGE); “Y” AXIS CORRESPONDS TO THE SCORE.....	150
FIGURE 6.19. COMPARISON BETWEEN RETRIEVAL RATES IN FUNPRO. SECTION A SHOWS THE PERFORMANCE OF FUNPRO WITHOUT USING METAMEMORY. SECTION B SHOWS THE PERFORMANCE OF FUNPRO USING METAMEMORY.....	153

LIST OF TABLES

TABLE 2.1. TUTOR MODULE TASKS IN AN ITS	28
TABLE 2.2. THE IMPLEMENTATION STRATEGIES OF LEARNING IN INTELLIGENT SYSTEM (PREPARED BY THE AUTHORS)	34
TABLE 2.3. SUPPORT OF METACOGNITIVE COMPONENTS.....	39
TABLE 3.1. PEDAGOGICAL MODEL CLASSIFICATION.....	47
TABLE 3.2. CONCEPTS INCLUDED IN PLANNER PACKAGE IN METAGOGIC METAMODEL.....	53
TABLE 3.3. CONCEPTS INCLUDED IN ADVISOR PACKAGE IN METAGOGIC METAMODEL	55
TABLE 3.4. CONCEPTS INCLUDED IN ASSESSMENT PACKAGE IN METAGOGIC METAMODEL.....	58
TABLE 3.5. CONCEPTS INCLUDED IN USER PACKAGE IN METAGOGIC METAMODEL	60
TABLE 4.1. METACOGNITIVE MODEL CLASSIFICATION IN MISM METAMODEL.....	62
TABLE 4.2. CONCEPTS INCLUDED IN METACORE PACKAGE IN MISM METAMODEL.....	66
TABLE 4.3. CONCEPTS INCLUDED IN SELFREGULATION . MONITORING PACKAGE IN MISM METAMODEL	69
TABLE 4.4. CONCEPTS INCLUDED IN SELFREGULATION . CONTROL PACKAGE IN MISM METAMODEL.....	72
TABLE 4.5. CONCEPTS INCLUDED IN METAMEMORY.MONITORING PACKAGE IN MISM METAMODEL.....	74
TABLE 4.6. CONCEPTS INCLUDED IN METACOMPREHENSION . MONITORING PACKAGE IN MISM METAMODEL ...	77
TABLE 5.1. LIST OF CONCEPTS IN MISM.METACORE AND METAGOGIC.METACORE	83
Table 5.2. Knowledge about UML and ontology notation	115
Table 5.3. Perception of usability	115
Table 5.4. ITS-FUNPRO mapping table	117
TABLE 6.1. LEARNING STYLES MODELED IN FUNPRO	121
TABLE 6.2. EQUIVALENCE OF FUNCTIONS BETWEEN THE MODULES.....	121
TABLE 6.3. REASONING TASKS IN FUNPRO	128
TABLE 6.4. CHANGES IN PEDAGOGICAL STRATEGIES - PRETEST AND POSTTEST MEAN AND STANDARD DEVIATION (SD).....	149
TABLE 6.5. STUDENTS' PRETEST AND POSTTEST MEAN AND STANDARD DEVIATION (SD).....	151
TABLE 6.6. PERFORMANCE METRICS USED FOR METAMEMORY	151
TABLE 6.7. SESSION CONFIGURATION	152

ABSTRACT

The modeling process of metacognitive functions in Intelligent Tutoring Systems (ITS) is a difficult and time-consuming task. In particular when the integration of several metacognitive components, such as self-regulation and metamemory is needed. Metacognition has been used in Artificial Intelligence (AI) to improve the performance of complex systems such as ITS. However the design ITS with metacognitive capabilities is a complex task due to the number and complexity of processes involved. The modeling process of ITS is in itself a difficult task and often requires experienced designers and programmers, even when using authoring tools. In particular the design of the pedagogical strategies for an ITS is complex and requires the interaction of a number of variables that define it as a dynamic process.

This doctoral thesis presents a metamodel for the personalized adaptation of pedagogical strategies integrating metamemory and self-regulation in ITS. The metamodel called MPPSM (*Metamodel of Personalized adaptation of Pedagogical Strategies using Metacognition in intelligent tutoring systems*) was synthesized from the analysis of 40 metacognitive models and 45 ITS models that exist in the literature. MPPSM has a conceptual architecture with four levels of modeling according to the standard Meta-Object Facility (MOF) of Model-Driven Architecture (MDA) methodology.

MPPSM enables designers to have modeling tools in early stage of software development process to produce more robust ITS that are able to self-regulate their own reasoning and learning processes. In this sense, a concrete syntax composed of a graphic notation called M++ was defined in order to make the MPPSM metamodel more usable. M++ is a Domain-Specific Visual Language (DSVL) for modeling metacognition in ITS. M++ has approximately 20 tools for modeling metacognitive systems with introspective monitoring and meta-level control. MPPSM allows the generation of metacognitive models using M++ in a visual editor named MetaThink.

In MPPSM-based models metacognitive components required for monitoring and executive control of the reasoning processes take place in each module of an ITS can be specified. MPPSM-based models represent the cycle of reasoning of an ITS about: (i) failures generated in its own reasoning tasks (e.g. self-regulation); and (ii) anomalies in events that occur in its Long-Term Memory (LTM) (e.g. metamemory).

A prototype of ITS called FUNPRO was developed for the validation of the performance of metacognitive mechanism of MPPSM in the process of the personalization of pedagogical strategies regarding to the preferences and profiles of real students. FUNPRO uses self-regulation to monitor and control the processes of reasoning at object-level and metamemory for the adaptation to changes in the constraints of information retrieval tasks from LTM.

The major contributions of this work are: (i) the MOF-based metamodel for the personalization of pedagogical strategies using computational metacognition in ITS; (ii) the M++ DSVL for modeling metacognition in ITS; and (iii) the ITS prototype called FUNPRO (*FUNdamentos de PROgramación*) that aims to provide personalized instruction in the subject of *Introduction to Programming*.

The results given in the experimental tests demonstrate: (i) metacognitive models generated are consistent with the MPPSM metamodel; (ii) positive perceptions of users with respect to the proposed DSVL and it provide preliminary information concerning the quality of the concrete syntax of M++; (iii) in FUNPRO, multi-level pedagogical model enhanced with metacognition allows the dynamic adaptation of the pedagogical strategy according to the profile of each student.

Keywords: metacognition, metamodel, ITS, adaptation of pedagogical strategies, tutor module, MOF.

1 INTRODUCTION

The purpose of this chapter is to provide an overview of the thesis by presenting the motivation and challenges, which inspired this research; the proposal details: the problem, research questions, hypotheses, objectives and methodology. Finally, the contributions and a vision of the structure of the document are presented.

1.1 Motivation

Metacognition is a field of study that emerged from cognitive science and psychology in the 1970s with the work of Flavell and Wellman (*Flavell & Wellman, 1977*). Metacognition from cognitive science is defined as mental awareness and regulation of one's thinking (*Jozefowicz, Staddon, & Cerutti, 2009*). Metacognition involves two executive processes performed by the subject over his cognitive processes: monitoring and control (*Anderson, Oates, Chong, & Perlis, 2006; Nelson & Narens, 1990*). Several authors (*Gaeta, Mangione, Orciuoli, & Salerno, 2011; Vockell, 2004*) have identified the following three major classes of metacognition: (i) Self-regulation that relates to the learners' ability to make adjustments to their own learning processes (*Soh & Blank, 2008*) in response to the perception about their current state of learning (*Azevedo, Witherspoon, Chauncey, Burkett, & Fike, 2009; Josyula, Hughes, Vadali, & Donahue, 2009*); (ii) Metamemory that refers to the processes involved in self-regulation or self-awareness of memory (*Nelson, Narens, & Dunlosky, 2004; Nelson & Narens, 1990*); and (iii) Meta-comprehension that addresses the abilities to adjust the cognitive activities in order to promote more effective comprehension and understanding of information (*Cox, 2005; Pule & Anderson, 2009*).

The term computational metacognition in Artificial Intelligence (AI) refers to the ability of an intelligent system to monitor and control its own learning and reasoning processes (*Cox & Raja, 2012*). Intelligent Tutoring Systems (ITS) are a particular type of Intelligent System (IS), which are used as educational tools in teaching and learning processes. An ITS can be defined as a cognitive tool (*Cheng, 2011; Wang, 2011*) formed by a software application, that uses AI techniques for representing knowledge (*Aamodt, 1994; Zhiping, 2009*), claiming that students interact with system, developing concepts and facilitating learning (*Cheng, 2011; McLaren, Deleeuw, & Mayer, 2011; Soh & Blank, 2008*).

This work is focused on the use of metamemory and self-regulation in order to improve some processes of personalization of pedagogical strategies in ITS. Our motivation is to provide a new metamodel-based approach for the integration of metacognitive capacities in ITS. A metamodel can facilitate the integration of metacognitive capacities in ITS by suggesting functional and semantic relationships between variables (*Sun, Zhang, & Mathews, 2006*) that affect the performance of the system in the personalization of pedagogical strategies. The metamodels are accompanied by a set of transformations that can generate models from them, clearly and efficiently.

1.2 Challenges

In the literature, different models of metacognition are applied to ITS. However, many of these models have a narrow focus, because they do not address comprehensively the elements of metacognition. The design of a new ITS with metacognitive capability is a difficult and time consuming task (*Gaeta et al., 2011; Soh & Blank, 2008*), due to the diversity and complexity of the available metacognitive models such as EM-ONE (*Singh, 2005*), Meta-AQUA (*Cox & Ram, 1999*), and CLARION (*Sun et al., 2006*).

The modeling process of metacognition in an ITS is a difficult task in terms of the diversity of constituent elements and to the complexity of the relationships among them; particularly, with the integration of several metacognitive components such as self-regulation with metamemory or meta-comprehension in a new system is necessary. Moreover, computational models (*Alonso, Arnold, & Havasi, 2010; Kennedy, 2010; Shapiro, Rapaport, Kandefer, Johnson, & Goldfain, 2007*) of metacognition do not present formalisms of software engineering methodologies that allow the development of an ITS in a systematic way. The focus of current metacognitive models on specific domains, poses difficulty in the adaptation of elements of the model to other domains.

In the other hand, the design of mechanisms for the formulation of pedagogical strategies in ITS is a complex task due to the number of variables involved. In particular, the selection of the methods or pedagogic tactics to be used for the development of a certain lesson requires ITS holds an extensive repertoire of pedagogical knowledge.

Therefore, to overcome these types of problems, we propose a metamodel-based approach for the integration of metacognitive capacities in ITS. Metamodels define language, structure and rules to be used for the design of different types of models, therefore, the metamodels are known as models of models. Thus in a metamodel, it can be specified metacognitive components required for monitoring and executive control of the reasoning processes that take place in each module of an ITS (*Molina, Gallardo, Redondo, Ortega, & Giraldo, 2013*). A metamodel approach enables designers to have tools, from an early stage of the development process to produce more robust ITS that are able to self-regulate their learning processes.

1.3 Thesis project

This thesis is focused on the formulation of a metamodel-based approach for the generation of models for personalized adaptation of pedagogical strategies integrating metamemory and self-regulation in the tutor module of ITS.

1.3.1 Research problem

The identified problem is related to the modeling process of personalized adaptation of pedagogical strategies using computational metacognition in ITS. The construction of ITS

models is itself a difficult task (Li, 2011; Zhang, Geng, Jiang, & Yang, 2009) and often requires experienced designers and programmers, even when using authoring tools. In particular, the process of personalized adaptation of pedagogical strategies in ITS is complex (Payne et al., 2009; Zhang et al., 2009) and requires the interaction of a number of variables that define it as a dynamic process (Bezerra, 2012).

Computational metacognition has been widely used in AI for designing robust Intelligent Systems. The modeling process of metacognitive capacities in ITS is often difficult and consumes time (Gaeta et al., 2011; Lee & Baylor, 2006; Soh & Blank, 2008). Specially, when modeling process involves simultaneous and flexible integration of metacognitive components such as meta-memory and self-regulation. In literature, some approach referred to the adaptation of teaching strategies in ITS are found. In these approaches, specific mechanisms of metacognition are implemented; such as Cox and Ram (Cox & Ram, 1999; Cox, 1996) who worked the concepts of meta-comprehension system using case-based reasoning (CBR). Soh and Blank (Soh & Blank, 2008) proposed an instructional planner based on CBR using introspection, which is a form of self-regulation. More recently Gaeta (Gaeta et al., 2011) developed a web learning environment based on self-regulation, but adapting the content through the organization of learning objects.

Few ITS models have incorporated some elements of metacognition as a mechanism for adaptation improvement. But adaptations are focused on content and not on pedagogical strategies and elements of metacognition have been incorporated in an isolated form.

These approaches fail to integrate simultaneously metacognitive components such as self-regulation and metamemory, to improve the personalized adaptation of pedagogical strategies in ITS.

The design of new ITS with metacognitive support is a time-consuming and difficult task (Gaeta et al., 2011; Kennedy, 2010; Soh & Blank, 2008) due to the great diversity and complexity of the available metacognitive models, many of them of a general nature (Cox, Oates, & Perlis, 2011), such as: Theoretical framework for the operation of human memory (Nelson & Narens, 1990), Meta-AQUA (Cox & Ram, 1999; Cox, 1995), theoretical framework CLARION (Sun et al., 2006), The metacognitive Loop (MCL) (Anderson et al., 2006; Haidarian et al., 2010), Simple model for meta-reasoning (Cox & Raja, 2012), EM-ONE Architecture (Singh, 2005), metacognition Distributed Framework (Kennedy & Sloman, 2003; Kennedy, 2010), a dual-cycle integrated metacognitive architecture (MIDCA) - (Cox et al., 2011) and Meta-level control agent architecture (Anita Raja & Lesser, 2007).

The design of ITS based on available metacognitive models is difficult also, because some models are theoretical (Cox et al., 2011; Cox & Raja, 2012; Nelson & Narens, 1990) and computational models (Kennedy & Sloman, 2003; Kennedy, 2010) do not present themselves formalisms of software engineering methodologies that allow the development of ITS in a systematic way.

None of the described proposals uses Model Driven Architecture (MDA) to address the problems of complexity and integration of metacognitive modeling. Similarly, the proposals in the review of the state of art about the modeling of the adaptation of pedagogical strategies in ITS do not contemplate the use of metamodels. Metamodels can

generate models with clear specifications, that allow consistently address the complexity of this type of processes.

After discovering the problem completely, it can be conclude that only very few of the current ITS incorporate metacognitive strategies in the adaptation of pedagogical strategies. In these few ITS, metacognition has been implemented in isolated form and without integration of aspects such as self-regulation and meta-memory. Moreover, the large number and diversity of metacognitive models make difficult for designers to work when modeling metacognitive aspects in ITS.

None of these approaches in reviewing the state of the art presents guidelines that belong to Software Engineering which facilitate the modeling process of custom adaptation of pedagogical strategies in ITS with integration of metamemory and self-regulation.

Therefore, to overcome these problems it was proposed a Metamodel for personalized adaptation of pedagogical strategies, by using metacognition in ITS.

The metamodel is configured according to the (Meta-Object Facility) MOF standard of MDA methodology. MOF standard provides a sequence of transformations and refinement of models. The transformations allow designers to have general schemes that facilitate the integration metacognitive components in the personalized process of adaptation of pedagogical strategies in ITS.

1.3.2 Research question

In the context of the identified problem the following research question is formulated:

RQ. How to design a metamodel for personalized adaptation of pedagogical strategies in ITS with integration of self-regulation and metamemory?

Following the systematization of the research problem is presented, which consists of a set of questions intended to decompose the main problem into less complex problems:

SRQ1. Which should be the specifications of a pedagogical model, so that has properties and methods for improving processes related to personalized adaptation of pedagogical strategies in ITS?

SRQ2. What kind of structural variant and invariant properties that have meta-cognitive models, can be used for integration of metamemory and self-regulation in processes related to personalized adaptation of pedagogical strategies in ITS?

SRQ3. What are MDA techniques necessary for designing a metamodel containing the specifications required for the modeling of personalized adaptation of pedagogical strategies by using metacognition in ITS?

SRQ4. What are the components and specifications of an MDA-based metamodel that allows the creation of personalized adaptation models of pedagogical strategies by using metacognition in ITS?

SRQ5. Which indicators should be taken into account by a prototype to validate the metamodel designed for generating personalized adaptation models of pedagogical strategies by using metacognition in ITS?

1.3.3 Objectives

1.3.3.1 General

- To design a MOF-based metamodel for the generation of models for personalized adaptation of pedagogical strategies with the integration of metamemory and self-regulation in the tutor module of ITS.

1.3.3.2 Specific objectives

- To identify the components and methods that have pedagogical models, so that allow the improvement of processes related to personalized adaptation of pedagogical strategies in ITS.
- To characterize the structural properties that have meta-cognitive models, to be used in the integration of metamemory management and self-regulation on processes associated with personalized adaptation of pedagogical strategies in ITS.
- To identify the MDA techniques necessary for designing a metamodel with the specifications required for the modeling of personalized adaptation of pedagogical strategies using metacognition in ITS
- To design the logical structure of MOF-based metamodel for personalized adaptation of pedagogical strategies integrating metamemory and self-regulation in ITS.
- To validate the metamodel designed for the generation of personalized adaptation models of pedagogical strategies using metacognition in ITS with the development of a prototype and its application in an educational environment.

1.3.4 Methodology

The methodology used in this research begins exploring the theoretical framework and state of the art of the latest research in the areas of study, activity that will continue in each of the phases. Mainly the methodology consists of five phases, each one pointing to the respective specific objective. The phases are:

In the first phase we identify the properties and methods that have pedagogical models, which permit to improve processes related to personalized adaptation of pedagogical

strategies in ITS. The result will be a model of the Tutor Module, which is focused on personalized adaptation of pedagogical strategies and identifying characteristics of the pedagogical model and instructional planning mechanism.

The second phase identifies and characterizes the structural properties, which have meta-cognitive models. This will be used in the integration of metamemory and self-regulation in processes related to personalized adaptation of pedagogical strategies in ITS. The result will be a conceptual metamodel for the integration of meta-memory and self-regulation in ITS

Then, in the third phase the components and specifications of an MDA-based metamodel are defined. This phase will generate as a result, the basic components of the structure of a metamodel that enables the modeling of personalized adaptation of pedagogical strategies using metacognition in ITS.

In the fourth phase metamodel is defined, using the defined components and structure developed. This phase results in an MDA-based metamodel, that allows the generation of personalized adaptation models of pedagogical strategies using metacognition in ITS.

Finally it is implemented and validated the metamodel through the construction of a prototype of ITS. The product is the functional description of the prototype, the analysis of data validation, including case studies and a chapter of this thesis document.

The research plan established consists of five phases, which at the same time are divided into a number of activities, as follows:

Phase 1: Identification of properties and methods of pedagogical models used in ITS.

- Identification of features and ways to represent pedagogical models in ITS.
- Design of the pedagogical model that composes the Tutor Module.
- Design of the educational planning mechanism for Tutor Module

Phase 2: Definition of a structural model for the integration of metacognition in the pedagogical model of ITS

- Determination of the components and processes associated with meta-memory in ITS.
- Determination of the components and processes related to self-regulation in ITS.
- Identification of integrative elements between metamemory and self-regulation in ITS.
- Design a conceptual metamodel for the integration of metacognition in the pedagogical model of ITS.

Phase 3: Configuration and components specification of a MDA-based metamodel for personalized adaptation of pedagogical strategies in ITS, with integration of metacognition.

- Determination of the components of a MDA-based *framework* for ITS.
- Design of a MDA-based framework for modeling personalized adaptation of pedagogical strategies using metacognition in ITS.

- Specification of metamodel abstractions for personalized adaptation of pedagogical strategies using metacognition in ITS.
- Determination of structural and semantic relations among the components of the metamodel.

Phase 4: Design of the logical structure of MDA-based metamodel for personalized adaptation of pedagogical strategies using metacognition in ITS.

- Design of a Metamodel Based on MDA for personalized adaptation of pedagogical strategies using metacognition in ITS

Phase 5: Validation of the proposed metamodel for the dynamic adaptation of pedagogical strategies using metacognition in ITS, by building a prototype.

- Prototype development using MDA-based metamodel for modeling custom adaptation of pedagogical strategies using metacognition in ITS.
- Validating the metamodel developed by building a prototype of ITS and its application in a case study.
- Measurement and analysis of statistical data and qualitative and empirical evidence of validation. Empirical work will be done through tests involving two groups of students, one with the ITS without metacognitive functions and the other group with ITS with metacognitive functions. Then will be compared the results of the personalized adaptation in both groups.

1.3.5 Contributions

The main contribution in this dissertation is a metamodel for personalized adaptation of pedagogical strategies in ITS, using in an integrated manner the meta-memory and self-regulation. The metamodel is based on the Software Engineering technique called Model Driven Architecture (MDA).

The proposed contribution, according to the literature reviewed is unpublished and innovative. Due the metamodels have not been used to facilitate the design of mechanisms for personalized adaptation of pedagogical strategies, using metacognition in ITS. Thus, the approach based on metamodels supports designers to deal with dynamic complexity of ITS modeling, providing guidance on the design and integration of metacognitive components.

The main achievements resulting from the development of this metamodel-based approach can be summarized as follows:

1. Theoretical work:

- Metamodel for flexible integration of metacognitive components related to metamemory and self-regulation in the personalized adaptation of pedagogical strategies processes in ITS, based on MOF Standard.

- Metamodel for the configuration of pedagogical models for the Tutor module in ITS.

2. Practical work:

- Development of a prototype of ITS with improvements in personalized adaptation of pedagogical strategies using metacognition.

3. Empirical work:

- The efficiency of the proposed metamodel for generating models of personalized adaptation of pedagogical strategies using metacognition in ITS. Empirical work will be done through tests involving two groups of students, one with the ITS without metacognitive functions and the other group with ITS with cognitive functions. Then will be compared the results of the personalized adaptation in both groups.

1.3.6 Document organization

This thesis is structured as follows. Chapter “Theoretical background” describes theoretical framework and a review of the general state of the art on research areas covered in the thesis. Chapter “Metamodel for pedagogical module” presents a metamodel which describes the concepts commonly used in modeling of pedagogical modules in ITS. Chapter “Metamodel for metacognition support in IS” describes the design and validation of a general purpose metamodel for metacognition support in IS. Chapter “MOF-based metamodel for personalization of pedagogical strategies using metacognition in ITS” presents a MOF-based metamodel called MPPSM, which is the main objective of this thesis. Chapter “Intelligent Tutoring System for teaching Introduction to Programming – FUNPRO” describes a prototype of ITS that aims to provide personalized instruction in the subject of *Introduction to Programming*. Chapter “Evaluation” presents answers to the research questions formulated in this doctoral thesis. Chapter “Conclusions and future works” presents the conclusions of this doctoral thesis.

2 THEORETICAL BACKGROUND

This chapter provides the theoretical framework and a review of the state of art on research areas covered by this thesis. The first section provides a contextualization regarding the pedagogical strategies and a description of learning theories included in this thesis. In this first section, the basic architecture of ITS and the personalized adaptation of pedagogical strategies in ITS are also discussed.

The second section presents a description of the main kinds of metacognition and an analysis of the most referenced architectures in the area of computational metacognition.

The third section describes the principles and elements of the Model Driven Architecture (MDA).

2.1 Modeling of pedagogical strategies in ITS

This section describes the theoretical support covered in this dissertation regarding the pedagogical strategies, learning theories and ITS.

2.1.1 Pedagogical strategies

The instructional plan configures the pedagogic strategy used for each student. The purpose of the pedagogic strategies is to facilitate the instruction and learning of students (*Woo et al., 2006*). Pedagogic strategies are of a general nature (*Dick, Carey, & Carey, 2005*) referring to abstract teaching methods (*Mizoguchi, Hayashi, & Bourdeau, 2010*). Pedagogic strategies are oriented toward configurations of activities and interfaces between the student and the medium imparting learning.

In educational environments, the pedagogic strategies are action plans designed to manage issues related to sequencing and organizing the instructional content (*Woo et al., 2006; Woo, Evens, Michael, & Rovick, 1991*) specifying learning activities, deciding how to deliver the content (*Mizoguchi et al., 2010*) and employing pedagogic tactics (*Bezerra, 2012*).

The pedagogical strategies are the set of actions performed by who teaches (the teacher) to facilitate the training and learning of students in various disciplines (*Ezechil & Coman, 2012*). The basic components of a pedagogical strategy are: the environment, the audience, pedagogical tactics (*Woo et al., 2006*) to be employed and the resources associated with such tactics (*Bezerra, 2012; Ding, Liu, & Deng, 2010*).

2.1.2 Learning theories and pedagogical strategies

The pedagogical strategies are implemented under the criteria of learning theories (*Chang-long, 2009; Opdenakker & Van Damme, 2006; Ozdamli, 2012*) otherwise it would be limited to sequence of activities and tasks without clear educational purpose (*Irfan & Shaikh, 2008; Ozdamli, 2012*). The following are learning theories that have influenced modern education.

2.1.2.1 Constructivism

The constructivist-learning paradigm focuses on the notion of subjective reality (*Palmer, 2005; Solomonidou, 2009*), where the knowledge is only an image or representation of the world (*von Glasersfeld, 1984; Von Glasersfeld, 1996*). In this order of ideas, inside the constructivism, learning is defined as learning to learn (*Jozefowicz et al., 2009; Segal, 2001; Von Foerster & Poerksen, 2002*), this concept of learning is one of the most important current trends in education.

The paradigm indicates that the student must construct knowledge for themselves and with the help of others, making the role of mediators or pairs (*Gong, Zhao, Wang, & Sun, 2009; Wen, 2004*). Similarly, constructivism indicates that can only be learned new concepts when these are in some way related to previously acquired knowledge (*Jozefowicz et al., 2009; Wen, 2004*).

Due to the participation of others in the learning of the individual and the relationship of new knowledge with existing knowledge, is that this paradigm is called historical-social constructivism (*Makgato, 2012; J. Payne & Israel, 2010*).

Therefore, always rely on him "generalized other" to our physical and mental development (*Makgato, 2012; Ovalle & Jiménez, 2004; Wen, 2004*). Knowledge and learning are not located in the corners of the cerebral cortex but rather in social encounters (*Ausubel, 1978; Vygotsky, 1978*) that positively enrich, frighten, oppress and liberate the human existence (*Makgato, 2012; Wen, 2004*).

Based on the social nature of learning, Vygotsky (*Vygotsky, 1978*) proposed the zone of proximal development (ZPD). The ZPD can be defined as the difference between the knowledge and skills already possessed by the student (real learning) and those that can get to learn by supporting someone more qualified (*Solomonidou, 2009; Wiemer-hastings & Glasswell, 2003*). The ZPD is one of the aspects that have influenced modern pedagogy.

2.1.2.2 Behaviorism

Behaviorism has its origins in the early 1950s with the work of Skinner (*Skinner, 1950, 1954*) at Harvard. In this theory prevailing conditions external to the subject, that promote learning, over the internal (*Ozdamli, 2012*). Behaviorism is primarily concerned with observable behavior (*Watson, 1913*) and may be subject to measurement, generally rejecting the participation of mental processes, emotions and consciousness in learning (*Richardson, 2011; Solomonidou, 2009; Watson, 1930*). This theory is one of the pillars in the relationship of the reinforcement-stimuli (*Skinner, 1954*).

The reinforcement should be given immediately after the stimulus, but in animals, the reinforcement can be negative or positive, with humans only is used positive reinforcement (*Bonarini, Lazaric, Montrone, & Restelli, 2009; Vassiliades, Cleanthous, & Christodoulou, 2011*).

Students are taught so that induces them to adopt new ways of behavior according to specific pathways (Bezerra, 2012). In this paradigm, learning is guided; therefore, the contents are presented in a linear manner to the students (Aguilar et al., 2011).

Due to the sequential structure of education, it does not promote independence and autonomous learning in the students.

2.2 Intelligent tutoring systems (ITS)

An ITS is a particular type of Intelligent System (IS), whose main function is to provide individualized instruction to students. Thus, it is necessary to know the needs and behavior of the student in order to infer that pedagogical strategy should be applied at a given moment.

In the literature, there is a considerable consensus since the early 1980s that the ITS consists of four basic components (Aguilar et al., 2011; Landowska, 2010; Nwana, 1990). See Figure 2.1. Initially (Bonnet, 1985; Hayes-Roth, 1982) described the expert module, student module, tutor module and finally (Aleven, Kay, & Mostow, 2010; Burns & Capps, 1988; Mandl & Lesgold, 1988) identified and added in several works the module of graphical user interface (GUI).

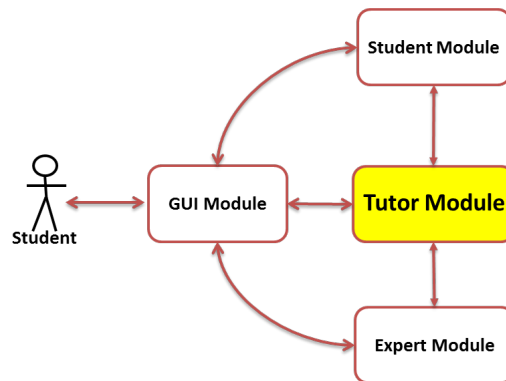


Figure 2.1 ITS classic model (Jeremić, Jovanović, & Gašević, 2012; Phobun & Vicheanpanya, 2010)

2.2.1 Expert Module

This module simulates the knowledge of a human expert in a specific domain of knowledge (Phobun & Vicheanpanya, 2010). The expert module contains the structure of knowledge and educational content (Prentzas, Hatzilygeroudis, & Garofalakis, 2002). Specifically, the knowledge base is constructed from a conceptual network of knowledge units, which are structured in hierarchical or relational form (Yong & Zhijing, 2003). In ITS with multiple domains, the expert module perform the data acquisition process when the user chooses a domain (Priya, Subhashini, & Akilandeswari, 2012).

2.2.2 Student Module

A great number of researches on ITS, are focused on this module (Kim & Shinn, 2010). The student models tend to be complex and multivariate (Landowska, 2010).

The student model is designed based on the following information: knowledge (Alexei Samsonovich, 2009), attitudes (Roll, Ryu, et al., 2006), cognitive skills (Aleven, Roll, McLaren, Ryu, & Koedinger, 2005; Conati, 2000; Peterson, Rayner, & Armstrong, 2009) and metacognitive skills (Gaeta et al., 2011; Kaelbling, Littman, & Moore, 1996; A. Moore, Macarthur, & Conlan, 2011), the emotional or affective state (Gui-mei & Guang-Xing, 2010; A. Moore et al., 2011), their learning progress and preferences (Landowska, 2010; Rishi, Govil, & Sinha, 2007; Soh & Blank, 2008).

The dynamic construction of the student model is the core of any ITS (Duan & Ren, 2011). This is because the ITS aims to provide personalized instruction to students (Duan & Ren, 2011; Ovalle & Jiménez, 2004). Therefore, ITS monitor at all times the student's actions and progress; this information serves as the basis for the model of each student.

Based on the student model, the ITS select pedagogical strategies and the most appropriate resources (Espinosa, Sánchez, Valdivia, & Pérez, 2007; Ganjanasuwan & Sanrach, 2006; Qadoori, 2010; Viccari & Jiménez, 2007) to improve the level of student learning. So, the whole system of adaptation of ITS depends on the student model contained in the Student module (Kinnebrew, Biswas, Sulcer, Taylor, & Sta, 2010; Lian, 2011; Lopes & Fernandes, 2009; Phobun & Vicheanpanya, 2010).

Computationally, the student models have been addressed with Bayesian networks, neural networks, relational databases (Bravo, Joolingen, & Jong, 2009; Landowska, 2010), Case-Based Reasoning (Arias, Jiménez, & Ovalle, 2009; Barros et al., 2011; Rishi & Chaplot, 2010; Zouhair, En-naïmi, Boukachour, Person, & Bertelle, 2010), fuzzy logic (Aguilar et al., 2011; Kim, Gil, & Rey, 2008) and semantic approaches and ontologies (Bittencourt, Costa, Silva, & Soares, 2009; Duan & Ren, 2011; Karampiperis & Sampson, 2004; Hua Wang, 2011).

2.2.3 Tutor module

The tutorial module has educational functions. It is responsible for guiding the teaching-learning process and decides what pedagogical actions must be done, how and when (Arias et al., 2009; Priya et al., 2012; Qadoori, 2010; Rongmei & Lingling, 2009).

The individualized education process consists of determining the Learning Objectives (LO) and the set of tasks, taking into account the characteristics of each student (Landowska, 2010; Lian, 2011; Xiao & Greer, 2009). The set of tasks to be performed by the student is designed in a way that allows acquiring the concepts or skills established in LO (Aguilar et al., 2011; Jeremić et al., 2012; Roll, Aleven, McLaren, & Koedinger, 2011b).

There is not a standard set of tasks to be performed by each student, since it depends on the characteristics of each one (Aguilar et al., 2011; Gaeta et al., 2011; Lian, 2011). For each particular student are established LO and a specified sequence of actions to achieve those objectives (Aleven, McLaren, Koedinger, & Roll, 2006; Duan & Ren, 2011).

Finally, the elements to be considered in the design of student assignments (plans) are (Aguilar *et al.*, 2011; Arias *et al.*, 2009; Gaeta *et al.*, 2011): characteristics of students, LO and available resources.

2.2.4 User Interface Module

This module is responsible for the communication between the system and the user (Duan & Ren, 2011; Escudero & Fuentes, 2010; Priya *et al.*, 2012), its main goal is to show the learning topics to the students (Aguilar *et al.*, 2011; Arias *et al.*, 2009; Gaeta *et al.*, 2011; Lian, 2011).

Depending on the interface design, the user interaction with the system can be more or less comprehensible (Muñoz-Merino, Fernández Molina, Muñoz-Organero, & Delgado Kloos, 2012; Wang, Xuejing, He, Zheng, & Wang, 2010). The interface design may affect the level of acceptance that the student has to ITS (Escudero & Fuentes, 2010; Gulz & Haake, 2006; Ozdamli, 2012). This module transforms the system interventions in a representation that is readable for the user, encoding the user input in the information that the system uses internally (Cabada, Barrón Estrada, & Reyes García, 2011; Snaider, Mccall, & Franklin, 2011; Soh & Blank, 2008).

2.2.5 Pedagogical models in ITS

The primary objective of the ITS is to provide personalized instruction (Rongmei & Lingling, 2009; Z. Wang *et al.*, 2010). Therefore, the main module of an ITS is the tutor module (Rongmei & Lingling, 2009; Soh & Blank, 2008; Yu-Liang Ting, 2012). The tutor module is also known in the literature as Instructional Planner (Aguilar *et al.*, 2011; Arias *et al.*, 2009; Viccari & Jiménez, 2007).

In ITS, the pedagogical model contained in the tutor module is responsible for determining the LO and select the most appropriate pedagogical strategies to guide the learning process for a particular student (Barros *et al.*, 2011; Bezerra, 2012; K. S. Cheung, Lam, Lau, & Shim, 2010; Seridi, Sari, Khadir, & Sellami, 2006).

The pedagogical model of an ITS should have at least a bank of learning theories (H. Chen, 2009; Espinosa *et al.*, 2007; Palmer, 2005; Silva, Buxton, & Campbell, 2003), a bank of teaching strategies (Magnisalis, Demetriadis, & Karakostas, 2011; Muldner & Conati, 2007; Seridi *et al.*, 2006) and a set of rules or mechanism to determine the relationship between the theories and strategies (Iglesias, Martínez, Aler, & Fernández, 2009; Prentzas *et al.*, 2002), which determine the pedagogical knowledge of ITS (see Figure 2.2).

The configuration of the set of pedagogical knowledge rules determine the capabilities of ITS to adapt in a personalized way the Instructional Plan (IP) (Aguilar *et al.*, 2011; Viccari & Jiménez, 2007; Woo *et al.*, 2006).

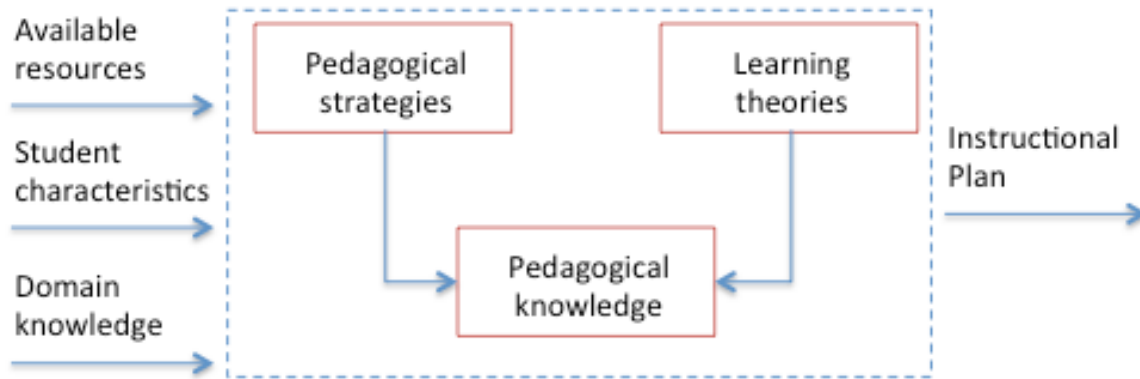


Figure 2.2. Basic structure of a pedagogical model in an ITS, based on (Bezerra, 2012; Phobun & Vicheanpanya, 2010)

Below is a list of the main tasks that execute a Tutor Module in an ITS (Table 2.1).

Table 2.1. Tutor module tasks in an ITS	
Tasks performed by a tutor module in an ITS	
(Aguilar et al., 2011)	To make decisions and control the ITS
(V. Payne et al., 2009)	To respond to requests for help from students
(Bittencourt et al., 2009)	To define instructional plan
(Aguilar et al., 2011; Bittencourt, Costa, Almeida, Fonseca, & Maia, 2007; Bittencourt et al., 2009)	To adapt pedagogical strategies to be used in training sessions according to the characteristics of each student
(Jeremić et al., 2012)	To decide how to present the learning resources to students
(Aguilar et al., 2011; Gaeta et al., 2011; Jeremić et al., 2012)	To detect the learning progress of each student
(Escudero & Fuentes, 2010; Jeremić et al., 2012; Roll, Aleven, et al., 2006; Viccari & Jiménez, 2007)	To intervene when students make mistakes
(Koedinger, Aleven, Roll, & Baker, 2009; V. Payne et al., 2009; Viccari & Jiménez, 2007)	To assess student's performance

The general process of running an IP is as follows (Aguilar et al., 2011; Elorriaga & Fernandez-Castro, 2000; Woo et al., 2006).

- The system identifies the student logged. The Tutor Module activates the corresponding student model to adapt the teaching session.

- Tutor Module provides the LO as student characteristics.
- Tutor Module offers a range of activities for the student to achieve the LO that have been established for him (IP or teaching plan). The concepts to teach and the resources available for such concepts are obtained from expert module. At the same time, the preferences and indicators related to the level and learning style of the student are obtained from student module.
- Tutor Module executes the lesson plan designed for the individual student and verifies the student's responses and performance. If the student's performance is not as expected, then the Tutor Module re-plans the activities.

The factors taken into account in the designs of plans (Yu-fen Chen, Juang, Feng, Chou, & Chan, 2004; Feng, Huang, Yang, & Mei, 2006; Legaspi, Sison, & Numao, 2004c; Lopes & Fernandes, 2009), for students are: characteristics students, learning objectives and available resources. The basic structure of an IP can be seen in Figure 2.3.

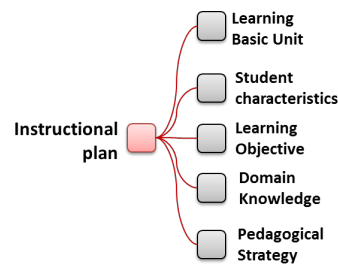


Figure 2.3. Structure of an instructional plan, based on (Aguilar et al., 2011; Arias et al., 2009; Escudero & Fuentes, 2010; Legaspi, Sison, & Numao, 2004a; Viccari & Jiménez, 2007)

- *Basic Learning Units (BLU)*. Set of topics that conform a subject or course (Escudero & Fuentes, 2010). The BLU are organized according to the sequence or order set by the IP (Arias et al., 2009; Ovalle & Jiménez, 2004).
- *Learning Objectives (LO)*. Represent those goals to be achieved by the student when complete a BLU (Arias et al., 2009; Escudero & Fuentes, 2010; Viccari & Jiménez, 2007).
- *Student characteristics*. Indicators related with student that can influence the design of the plan, generally include the learning level and learning style. Student characteristics are considered to present the knowledge with some degree of abstraction.
- *Knowledge*. It is content (learning objects) represented by figures, diagrams, formulas, videos, exercises, problems solved, examples, animations and simulations, among others.
- *Methodology*. It has to do with the set of strategies selected for the student develops the BLU.

2.2.6 Personalized adaptation of pedagogical strategies in ITS

In the literature, various approaches for addressing personalized adaptation of pedagogical strategies in ITS's are found. Early versions of ITS's incorporated pedagogical strategies in static plans developed by an expert (*Viccari & Jiménez, 2007*). The next step in the evolution of the adaptation of pedagogical strategies is given by the implementation of algorithms for the production of IP (*J. Jones, 1992; Liu, 1988; Specht & Augustin, 1998; E. Wang & Kim, 2009; Woo et al., 1991*), these plans were difficult to develop, maintain and modify.

Several approaches and proposals were presented by various authors in order to improve the personalized adaptation of pedagogical strategies in ITS (*Espinosa et al., 2007; Graham, 2011; Karampiperis & Sampson, 2004; Yu & Zhiping, 2008*).

Karampiperis and Sampson (*Karampiperis & Sampson, 2004*) proposed an adaptive IP model, based on the use of ontologies. Although the IP is able to re-planning itself when the student has trouble achieving learning objectives, re-planning occurs in terms of resources and not in pedagogical strategies.

Arias, Jiménez and Ovalle propose a model of instructional planning using Case-Based Reasoning (CBR) (*Arias et al., 2009*) the model allows to adapt the instruction to the specific needs of each student. The plan is constantly redesigned to define and identify methods that can be used to guide the learner to acquire knowledge.

A limitation of this study is that the activity plans can be generated incomplete. Because of that, the characteristics of the cases do not cover adequately the entire solution space; it would take a large repository of cases that can be adapted to the characteristics of each student.

More recently, adaptation strategies have focused on the pedagogical model of the ITS tutor module. Barros (*Barros et al., 2011*) presented a pedagogical model designed using an ontology called pedagogical ontology. The pedagogical model contains the knowledge of how to teach and serves as a resource for the development of IP according to the characteristics of each student. Instructional strategies contained in the model are based on learning theories: cognitive, situated, Socratic, constructivist and behaviorist. However, the model does not have mechanisms to auto adjust the established strategies by the ontology.

Aguilar (*Aguilar et al., 2011*) proposes a model of instructional planner, of two levels, based on multi-agent systems (MAS) and fuzzy logic. This model does not base the design of pedagogical strategies in learning theories, instead, are specified by pedagogical experts in the form of rules. If the student does not achieve the established learning objectives for each lesson, the ITS can maintain the student in the same lesson or turn his/her back, in case of having very bad performance. This proposal has the disadvantage of keeping the initial plan and does not reconfigures it, instead the ITS returns, maintains, or move forward the student in the content.

The works described do not incorporate the use of metacognition as a regulating mechanism for adaptation processes of pedagogical strategies in ITS. The fact of not using

metacognition causes ITS less robust (*Haidarian et al., 2010*) regarding the detection of errors or failures in their reasoning processes.

However, have arisen proposals which have implemented the use of metacognition as a mechanism of self-improvement on ITS. Soh and Blank (*Soh & Blank, 2008*) presented an Instructional Planner based on CBR using introspection, which is one of the mechanisms of metacognition. The pedagogical strategies are stored as cases and then, are recovered and adapted as the difficulties presented by the student to achieve the learning objectives. But the adaptation is performed after completion of the lesson and not during the learning process.

Gong in (*Gong et al., 2009*) developed a web environment for cultivating metacognition in students. The environment includes the use of a set of metacognitive strategies to enhance metacognitive skills in students. However, the Web environment does not implement metacognition to improve itself, neither describes IP specifications.

Gaeta in (*Gaeta et al., 2011*) presents a Web learning environment based on self-regulation. In this environment adaptation takes place by a mechanism called Learning Path. However, the Web environment does not provide support to the adaptation of pedagogical strategies instead it organizes a set of learning objects according to user characteristics.

Thus, the current works do not incorporate metacognition on ITS in an integral way because they are focused on some components, being self-regulation the most addressed component (*Gaeta et al., 2011; Soh & Blank, 2008*).

Neither of the approaches found in the literature review presents an integral use of the components of metacognition to monitor and control the process of personalized adaptation of pedagogical strategies on ITS.

2.3 Metacognition in ITS

This section describes the theoretical foundations related to the implementation of metacognition in intelligent tutoring systems; including a description of the concepts of self-regulation, metamemory and meta-comprehension.

2.3.1 Metacognition in intelligent systems

Metacognition is a field of study that emerged from cognitive science and psychology in the 1970s with the work of Flavell (*Flavell & Resnick, 1976; Flavell & Wellman, 1977*). Metacognition from cognitive science is defined as mental awareness and regulation of one's thinking (*Josyula, Vadali, Donahue, & Hughes, 2009; Veenman, Hout-Wolters, & Afflerbach, 2006*). In metacognition are two executive processes performed by the subject over their cognitive processes, these processes are the monitoring and control (*Anderson et al., 2008; Cox, 2005; Nelson & Narens, 1990*).

Especially in the early 1990s, metacognition became into a field of study by specialized AI community (Christodoulou & Keravnou, 1998; Cox, 1997; R Oehlmann, Edwards, & Sleeman, 1995). Since then, metacognition has been widely used in AI for designing robust IS.

The term metacognition in AI refers to the ability of intelligent systems to monitor and control their own learning and reasoning processes (Anderson et al., 2006; Cox & Raja, 2012; Schmill et al., 2011; Singh, 2005); therefore, metacognition in AI often referred by some authors as meta-reasoning (Anderson et al., 2008; Cox et al., 2011; Cox, 2005).

A first contribution in the field of metacognition occurred when was presented the theoretical framework for the operation of human memory (Nelson & Narens, 1990), (see Figure 2.4), which were introduced the three key principles of metacognition:

- Cognitive processes can be divided into two or more levels.
- The meta-level contains a dynamic model of the object-level.
- There are two dominant relations called control and monitoring.

Today the two-tier architecture (Nelson & Narens, 1990) is the basis for the architectural design of metacognition in IS, see Figure 2.4.

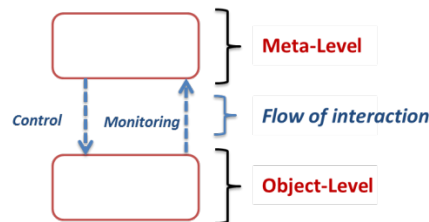


Figure 2.4. Metacognitive loop (Nelson & Narens, 1990)

Metacognition has two elements or components (Vockell, 2004): metamemory and self-regulation, on which are grouped all metacognitive processes, see Figure 2.5.

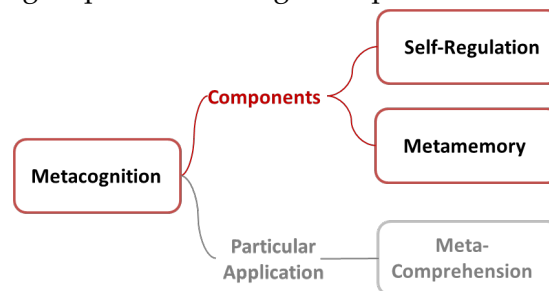


Figure 2.5. Metacognitive elements (Vockell, 2004)

Several authors consider the metacomprehension as a practical application of metacognition (Kolodner, Owensby, & Guzdzial, 2004). Therefore in this thesis is defined similarly.

2.3.1.1 Metamemory

The metamemory is one of the components of metacognition (Azevedo et al., 2009), in AI refers to the capabilities and strategies that can use an IS, to improve their own memory (Cox & Raja, 2012). Thus, metamemory refers to the processes involved in self-regulation or self-awareness of memory (Nelson et al., 2004; Nelson & Narens, 1990).

Metacognitive processes or skills related to metamemory, work monitoring and controlling each memory activities (Flavell & Resnick, 1976; Flavell & Wellman, 1977) and are grouped into three phases (Nelson & Narens, 1990): acquisition, retention and retrieval. In the acquisition phase metacognitive processes occur in two stages: Before Learning (AOL - Advance of Learning) and lifelong learning (OGL - Ongoing Learning) (Metcalfe & Dunlosky, 2008; Nelson et al., 2004; Nelson & Narens, 1990).

In AOL and OGL, the monitoring activities of the memory functions are performed using metacognitive judgments; these judgments have a high impact on the predictability of the difficulty of a problem (Metcalfe & Dunlosky, 2008; Nelson & Narens, 1990).

2.3.1.2 Self-regulation

Self-regulation in AI refers to the ability of an intelligent system to make adjustments of their own learning processes (Soh & Blank, 2008). Adjustments are produced in response to the perception of the intelligent system about its current state of learning (Rishi et al., 2007; Zhiping, Yu, & Tianwei, 2011).

The concept of self-regulated learning (SRL) comes from a pedagogical approach that is based on students take control of their own learning process (de Bruin, Thiede, Camp, & Redford, 2011; Kinnebrew et al., 2010). SRL involves monitoring and control of the learning process in intelligent system (Josyula, Hughes, et al., 2009; Schmill et al., 2011). SRL enables systems to detect anomalies in the process of learning and work proactively to respond to them (Anderson & Perlis, 2004; Josyula, Vadali, et al., 2009; Kinnebrew et al., 2010). Self-regulation is activated when IS not satisfied one of the goals established or when the system has difficulty in obtaining the expected return (Anderson et al., 2008; Cox et al., 2011; Fox & Leake, 1994).

There are four approaches for the implementation of SRL in intelligent systems, these are: introspective learning (IL) (Cox, 1996; Fox & Leake, 1994; Rudiger Oehlmann, 1995; Soh, 2007), reinforcement learning (RL) (Celiberto, Matsuura, de Mantaras, & Bianchi, 2010; Hwang, Lin, Hsu, & Yu, 2011; Maeda & Hanaka, 2008; Vassiliades et al., 2011), learning by experience (LE) (Yuh-jen Chen & Chen, 2009; Rishi & Chaplot, 2010; Saberi & Mohammad, 2008; Zouhair et al., 2010) and cooperative learning (CL) (Abbasi & Abbasi, 2008; Haitao, Weidong, Wenyan, & Xiaoming, 2000; J. Li, Sheng, & Ng, 2011).

IL is the most implemented for self-regulation. Based on the introspective planning (Cox, 1996; R Oehlmann et al., 1995), which consists in self-questioning by the IS (Fox & Leake, 1994; Soh, 2007), with respect to experience (Cox, 1996; Roll, Aleven, et al., 2006). Thus, the system generates responses to questions raised and produces self-action plan that is

stored in a CBR system (*Bhat & Kolodner, 2009; Kolodner et al., 2003; Soh & Blank, 2008*). Thus the intelligent system can self-regulate their learning process.

RL is the kind of learning that acquires a system after facing a problem, using the technique of trial and error (*Celiberto et al., 2010; Kaelbling et al., 1996; Vassiliades et al., 2011*). This kind of learning is generally employed in dynamic environments (*Abbasi & Abbasi, 2008; Qiang & Zhongli, 2011; Vassiliades et al., 2011*) and is achieved after multiple iterations (*Hwang et al., 2011*).

LE is a learning technique used in intelligent systems, which is based on solving new problems by adapting solutions given to similar problems in the past (*Yuh-jen Chen & Chen, 2009; Gadhiok, Amanna, Price, & Reed, 2011; Kolodner et al., 2003; Zouhair et al., 2010*). This technique is generally based on the CBR (*Kolodner, Cox, & Gonzalez-Perez, 2005; Soh & Blank, 2008*).

CL is a kind of learning used in MAS (*Abbasi & Abbasi, 2008; Aguilar et al., 2011*) and is based on a set of policies of communication and collaborative work, which is made among system agents (*Vassiliades et al., 2011; Zouhair et al., 2010*).

The learning theories described in section 2.1.2 (constructivism and behaviorism) and the four SRL approaches described in this section are related as follows: (i) Constructivist practice are the process of collaborative learning (CL) and deep personal introspection (IL) into one's own learning process (*Brooks & Brooks, 1993, 1999*), where the new information is linked with prior knowledge (LE); and (ii) In the behaviorism the learning is a change in external behavior achieved through using reinforcement and repetition (RL).

Below is a table with computational implementations used for each kind of learning in intelligent systems.

Table 2.2. The implementation strategies of learning in Intelligent System (prepared by the authors)

Learning	Implementation
IL	CBR
	Nearest Neighbor
	Rules techniques
RL	Q-Learning
	Bayesian Networks
	Ontologies
	Neural Networks
CL	Multi-Agents Systems (MAS)
	Markov (MDP)
	D-Trees
	Ontologies
LE	CBR
	Similarity Measures

To conclude this section, note that all implementations of RL, are developed using Q-learning algorithms (*Abbasi & Abbasi, 2008; Hsu & Juang, 2011; Huang, Chung, Chang, & Ren, 2009; Hongbing Wang, Zhou, Zhou, Liu, & Li, 2010; Watkins & Dayan, 1992*), while CBR is the most commonly computational approach implemented in experience-based learning such as IL (*Arias et al., 2009; Cheng, 2011; Cox, 1997; Livingston, 2003*) and LE (*Gadhiok et al., 2011; Soh & Blank, 2008*).

2.3.1.3 Meta-comprehension

The meta-comprehension is a specific application of metacognition (*Keener, 2011; Ozuru, Kurby, & McNamara, 2012; Pule & Anderson, 2009*), in other words, is the process of executive control of comprehension (*de Bruin et al., 2011; Gaeta et al., 2011*). In AI, meta-comprehension refers to the ability of intelligent systems to control the degree in which comprise the information being communicated (*Benes, 2004; Gaeta et al., 2011*).

Due to meta-comprehension is a particular application of metacognition; this is not a structural part in this thesis.

2.3.2 Models, frameworks and architectures of metacognition in intelligent systems

As a result of the review of this literature, are described a series of models that have been referenced in the development of intelligent systems with metacognitive support.

2.3.2.1 Meta-AQUA

Meta-AQUA (*Cox & Ram, 1999; Cox, 2007*) is a model based on the theory of Introspective Multi-strategy Learning (IML) (*Cox, 1996*) and a cognitive model of introspection. The main functionality of the Meta-AQUA system is the “story understanding”, which is considered as the ground level. The meta-level is structured by the implementation of IML (*Cox & Ram, 1999*), which is based on Case Based Reasoning (CBR) (*Aamodt, 1994; Kolodner, 1992*). The learning strategy in Meta-AQUA is implemented using a model of goal-driven learning (GDL) (*Cox & Ram, 1999; Cox, 2005*) and produces structures called meta-explanations (Figure 2.6).

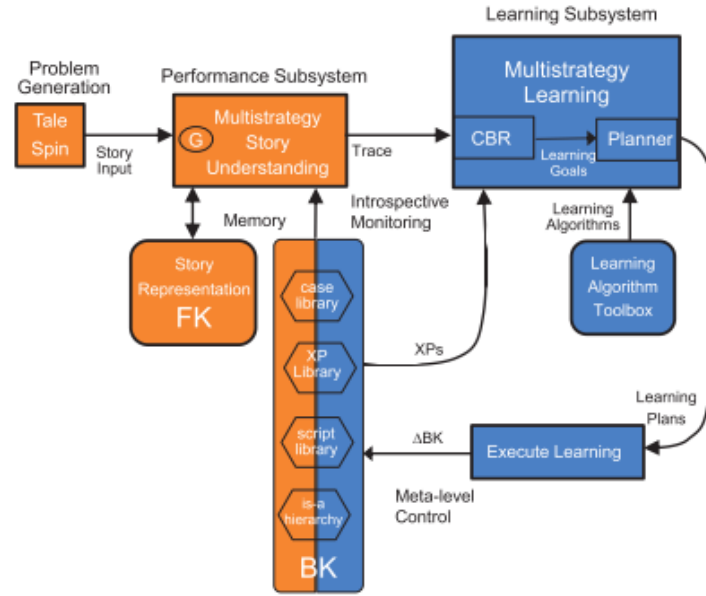


Figure 2.6. Meta-AQUA (Cox & Ram, 1999; Gordon, Hobbs, & Cox, 2007)

2.3.2.2 CLARION theoretical framework

CLARION (Sun *et al.*, 2006) is an overall architecture of the mind. The architecture is used to construct models of specific metacognitive processes such as self-monitoring and self-regulation (of cognitive processes). Clarion is used to capture experimental data related to meta-cognition with humans.

2.3.2.3 The MetaCognitive Loop (MCL)

MCL (Anderson *et al.*, 2006; Haidarian *et al.*, 2010) is an architecture focused on detection of anomalies in learning process and how to respond to them. MCL presents a general architecture and has three sets of ontologies (Noy & McGuinness, 2000), which are: ontology for anomaly types, failure ontology for use in assessment and response ontology for selecting repair types to guide. Figure 2.7.

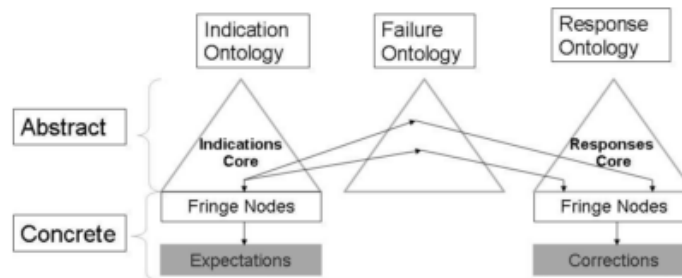


Figure 2.7. MCL (Anderson *et al.*, 2006; Schmill *et al.*, 2011)

2.3.2.4 Simple model for meta-reasoning

Cox and Raja (Cox & Raja, 2012) proposed a simple model for meta-reasoning. This model presents a double cycle of reasoning. The first cycle, refers to the traditional conception of cognitive science and AI about the reasoning in IS. In this cycle the intelligent agent receives perceptions of the environment, it makes decisions (reasoning) and acts making changes on the environment (Anita Raja & Lesser, 2007). On the other hand, the second cycle of the simple model refers to the perception that the metal-level has about object-level. The metal-level makes decisions (meta-reasoning) about the information that comes from the object-level see Figure 2.8.

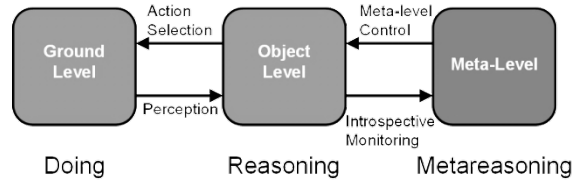


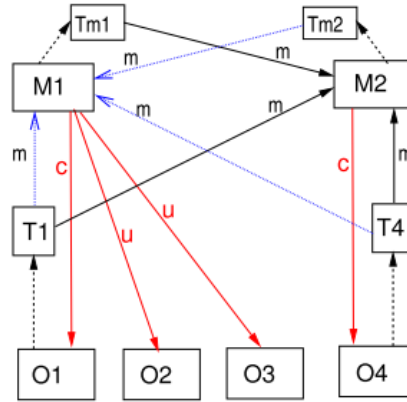
Figure 2.8. Simple model for metareasoning (Cox & Raja, 2012)

2.3.2.5 EM-ONE Architecture

EM-ONE (Singh, 2005) is a cognitive architecture which purpose is to support the kinds of commonsense thinking that is required to produce a possible scenario in a system. "Mental critics" (Singh, 2005) are used as a mechanism of operation in this architecture, which are procedures that recognize problems in the current situation. The Mental critics use commonsense narratives to suggest courses of action, ways to deliberate about the circumstances and consequences of those actions. Also, it can propose ways to reflect upon their mistakes when things go wrong. In EM-ONE there are mental critics for answering the problems in the world, and other mental critics for answering the problems in the EM-ONE system itself.

2.3.2.6 Distributed metacognition Framework - DMF

This is a conceptual architecture for a distributed metacognition with context-awareness and diversity; see Figure 2.9. A distributed metacognitive architecture is one in which all meta-level reasoning components can be monitored and controlled by other components of meta-level (Kennedy & Sloman, 2003; Kennedy, 2010).



A snapshot of distributed metacognition

Figure 2.9. DMF (Kennedy & Sloman, 2003; Kennedy, 2010)

2.3.2.7 A metacognitive integrated dual-cycle architecture (MIDCA)

MIDCA (Cox et al., 2011) is a novel architecture that incorporates both a perception-action cognitive cycle and a monitor-control metacognitive cycle (Cox et al., 2011). In meta-level the agent recognizes the problem, explains what causes the problem, and generates a new goal to remove the cause (Cox., 2007). The meta-level reasoner can change the goals, the processes and the input. MIDCA is based in a previous work of Norman (Norman & Shallice, 1986) who designed a cognitive architecture; see Figure 2.10.

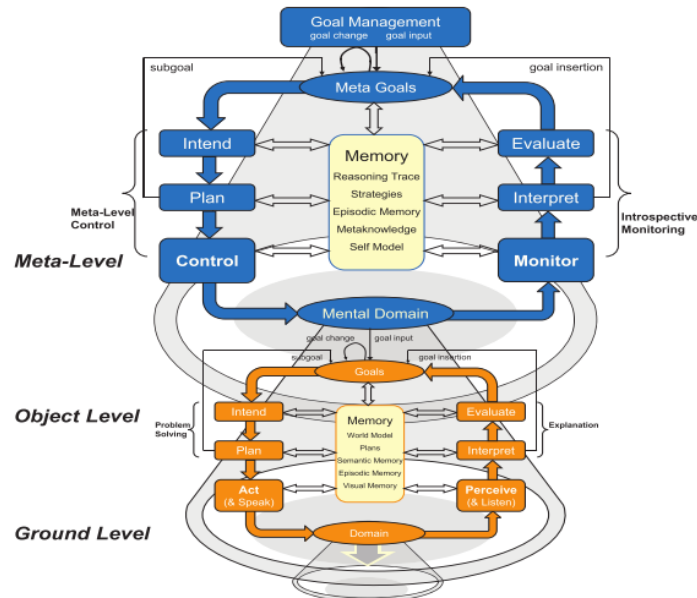


Figure 2.10. MIDCA (Cox et al., 2011)

2.3.2.8 Meta-level control agent architecture (Framework)

This architecture was implemented in MAS and is centered in making more effective the meta-level control decisions. This framework is a precursor of the distributed metacognition (*Anita Raja & Lesser, 2007*). The meta-level control uses an abstract representation of the agent state. The framework uses decision trees to support the make-decision process at metal-level.

2.3.3 Support of metacognitive components

With reference to the support of the main components of metacognition as metamemory, meta-comprehension and self-regulation, it is appreciated that the majority of the architecture does not provide support for the three components, see Table 2.3.

Table 2.3. Support of metacognitive components

Model	Meta-memory	Meta-comprehension	Self-regulation
Meta-AQUA (<i>Cox & Ram, 1999; Cox, 2007</i>)	Memory awareness	Story understanding – Meta-XP (meta-explanation)	Story understanding
CLARION Architecture (<i>Sun et al., 2006</i>)	-	-	Meta-level can act as an executive function
The Meta-Cognitive Loop (MCL) (<i>Anderson et al., 2006; Schmill et al., 2011</i>)	Basic mechanisms of short-term memory	Basic comprehension of object-level process	Anomaly detection – monitoring and control
Simple model for meta-reasoning (<i>Cox & Raja, 2012</i>)	<i>No evidence presented</i>	<i>No evidence presented</i>	Introspective monitoring
EM-ONE Architecture (<i>Singh, 2005</i>)	Metamemory based on CBR	Mental critics that use commonsense narratives	Commonsense thinking
Meta-level control Agent architecture – MLCAA (<i>Cox &</i>	<i>No evidence presented</i>	<i>No evidence presented</i>	Effective meta-level control decisions

Raja, 2012)

Distributed metacognition Framework DMF (Kennedy & Sloman, 2003; Kennedy, 2010)	Distribute memory system	No evidence presented	Context-awareness and diversity
A metacognitive integrated dual-cycle architecture (MIDCA) –(Cox et al., 2011)	Memory mechanism which can access both the object-level and the meta-level	Object-level explanation	Introspective monitoring and meta-level control

In relation with metamemory, it could be appreciated in Table 2.3 that Meta-AQUA has a complex multifaceted memory (Cox & Raja, 2012; Cox, 2005) and has the capability to reason about memory events. While, in MCL are leaved out aspects referred to metamemory strategies that can be used to learn from detected failures (Schmill et al., 2007). Moreover, MCL has only basic mechanisms of short-term memory, which in the meta-level are matched with long-term memory. EM-ONE implements a metamemory strategy based on a CBR system. MIDCA has a memory mechanism that can be accessed from the object-level and the meta-level. The rest of architectures do not present a clearly support to control and monitor the memory process.

Regarding to meta-comprehension, Meta-AQUA, EM-ONE and MIDCA are the architectures that offer adequate support. Meta-AQUA uses introspection (Cox & Raja, 2012; Cox & Ram, 1999) to represents traces of reasoning with (meta-explanation). EM-ONE has a strategy known as mental critics (Singh, 2005) that use commonsense narratives to suggest courses of action to deliberate about the circumstances and consequences of those actions.

With respect to Self-regulation, it can be clearly appreciated that all architectures provide full support for this component of metacognition. In MIDCA the meta-level can act as an executive function in a similar manner to CLARION. CLARION and MCL have a better development of the meta-cognitive processes than the rest of architectures. Note that Mata-AQUA, EM-ONE and MIDCA, are the most complete metacognitive architectures, because provide support to three main components of metacognition that are: metamemory, meta-comprehension and self-regulation.

2.4 Framework of model driven architecture (MDA)

The Model Driven Architecture (MDA) is an approach from the Object Management Group's (OMG) (OMG, 2013) for the development of model-driven software (MDD). MDD is a development paradigm of Model-Driven Engineering (MDE) that uses models as the primary artifact of the development process of systems. MDE is a software development methodology that uses models at different levels of abstraction for developing systems. Figure 2.11 shows the relationships between MDA, MDD and MDE.

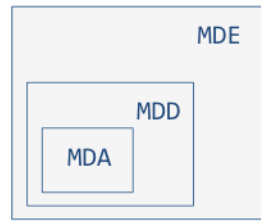


Figure 2.11. Relationship between MDA, MDD and MDE, graphic based on (Kleppe, Warmer, & Bast, 2003)

The MDA is based on MOF, which provides a framework for the management of metadata and a set of metadata services to enable the development and interoperability among models (Bragança & Machado, 2008). Figure 2.12 shows an example from Bragança and Machado (Bragança & Machado, 2008), which describes the MOF metadata architecture for modeling the schema of a database. The figure shows the relationships among models in different layers of the MOF architecture.

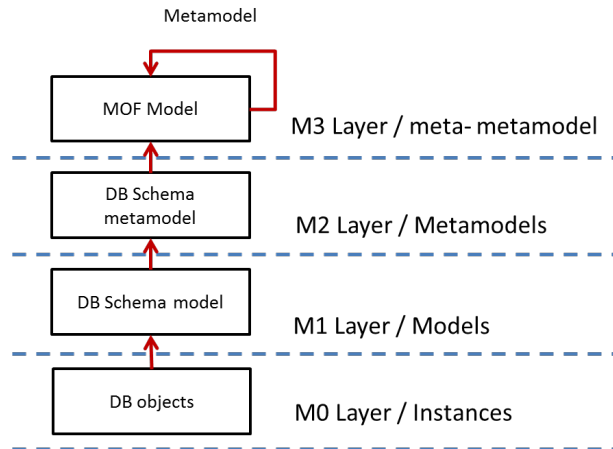


Figure 2.12. MOF Architecture (Bragança & Machado, 2008)

MDA is based on the following elements and principles:

2.4.1 Models

The models are used to develop the system abstractions (*Bragança & Machado, 2008*) at various levels and from different perspectives. In MDA there are four basic types of models (*Chitfroush, Yazdandoost, & Ramsin, 2007*): (i) computational independent model (CIM), which serves as the model of the problem domain and excludes any reference to implementation details or description of computer system; (ii) platform-independent model (PIM) which describes the system from several perspectives regardless of operating platform; (iii) platform specific model (PSM), this model provides a platform dependent description of the same system described in PIM and is constructed through the transformation of PIM according to a Model Platform (MP); and (iv) implementation specific model (ISM), which specifies the details of implementation.

2.4.2 Transformation Model

In MDA, the development of a system is viewed as a sequence of transformations and model refinement (*Chitfroush et al., 2007*). Transformation is a series of steps that allow refinement of models (*Bragança & Machado, 2008; Chitfroush et al., 2007*). Model transformations play an important role in the MDA approach. The objective is to obtain a model that contains enough features for automatic generation of executable code. The execution of models' transformations establishes the links of traceability between the CIM, PIM and PSM models.

Model transformation is the process of transforming one model into another model. The first model is called source model, and the second is target model (*Yonglin, Weiping, Qun, & Yifan, 2009*). Both models can have the same or different metamodels.

MDA uses languages to represent transformations of models. QVT (Queries, Views, Transformations) (*Omg, 2008*) and ATL (ATLAS Transformation Language) (*Jouault, Allilaire, Bézivoin, & Kurtev, 2008*) are two representative transformation languages in MDA.

Figure 2.13 shows a transformation structure in MDA using QVT.

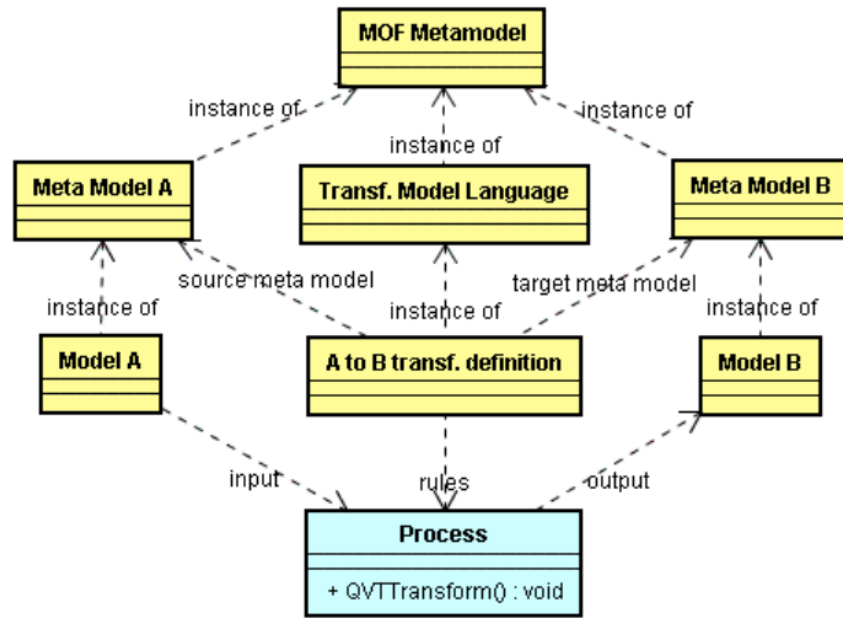


Figure 2.13. Transformation structure in MDA (Kleppe et al., 2003; Koch & GmbH, 2006)

The models' transformations recommended by MDA are essentially the CIM transformations to PIM and PIM transformations to PSM (Bragança & Machado, 2008; Raghupathi & Umar, 2008). Figure 2.14 shows the mapping between models in MDA. Mapping is the specification of a mechanism for transforming an input model into an output model.

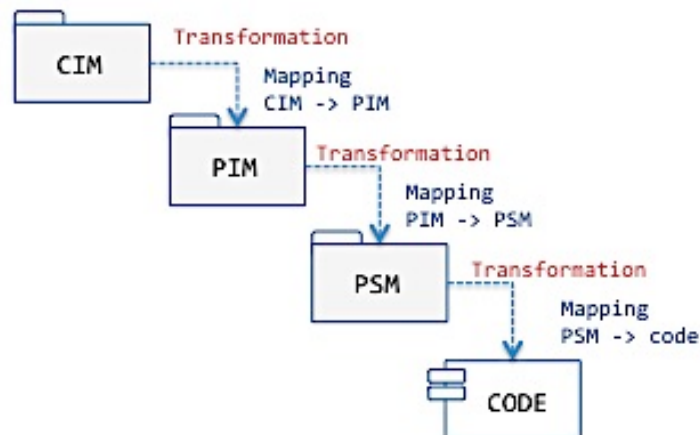


Figure 2.14. Transformations in MDA, graphic based on (Jouault & Kurtev, 2006; Mens & Van Gorp, 2006)

Transformations can be classified into endogenous and exogenous (Mens & Van Gorp, 2006). Endogenous transformations are transformations between models expressed in the same metamodel (Jouault & Kurtev, 2006; Mens & Van Gorp, 2006). Exogenous transformations are transformations between models expressed using different metamodels (Jouault & Kurtev, 2006; Mens & Van Gorp, 2006).

2.4.3 Metamodel

The models themselves are expressed by metamodels that allow meaningful integration and transformation among models, specifically through tools (*Chitforoush et al., 2007*).

Therefore, a metamodel is a model of a model. MDA architecture is based on a metamodeling of four layers: a) meta-Metamodeling layer, which corresponds to MOF and defines an abstract language for specifying metamodels b) metamodel layer, which consists in metamodels that are defined in the standard MOF c) layer model, which includes real-world models d) layer of "real world" which includes real-world things.

Figure 2.15 shows a simplified view of MOF metamodel at M₃ layer. As can be seen the basic concepts of MOF are:

- Classes
- Attributes
- Association between classes

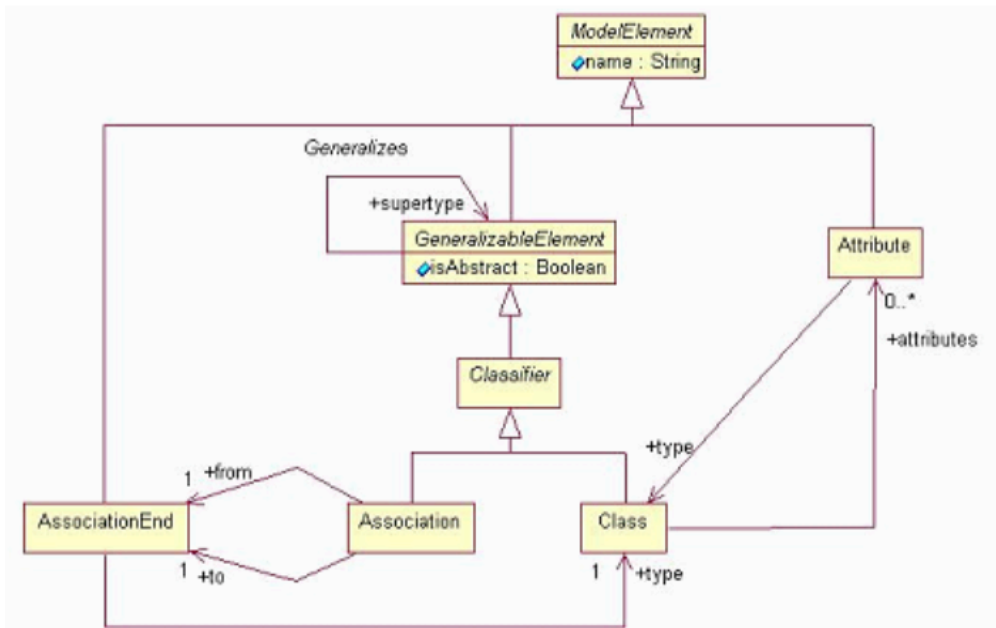


Figure 2.15. Simplified MOF metamodel at meta-metamodeling layer (M₃), (*OMG, 2013; Rensink & Nederpel, 2008*)

2.5 Conclusion of the chapter

This chapter presented the theoretical framework and a review of the general state of the art on research areas covered in the theses which are: pedagogical strategies, learning theories, ITS, pedagogical models in ITS, metacognition in intelligent systems and MDA framework.

The pedagogical strategies are the set of actions performed by who teaches to facilitate the instruction and the learning process of the students. Actions include sequencing and organizing the instructional content, specifying learning activities, deciding how to deliver the content and employing pedagogic tactics.

Learning theories are paradigms that aim to describe how learning occurs. This thesis covers learning theories of constructivism and behaviorism. Constructivism indicates that the learner must construct knowledge for themselves and with the help of others, making the role of mediators or pairs. Moreover, in the behaviorist paradigm learning is guided; therefore, prevailing external conditions that promote learning to the learner.

An Intelligent Tutoring System (ITS) is an intelligent system that provides personalized instruction to learners. In ITS, the pedagogical model contained in the tutor module is responsible for selecting the appropriate pedagogical strategies to guide the learning process of a particular student.

The term metacognition in AI refers to the ability of intelligent systems to monitor and control their own learning and reasoning processes. Metamemory and self-regulation are different forms of metacognition implemented in intelligent systems.

MDA is an approach for the development of model-driven software. MDA is based on MOF standard. The MOF standard provides a framework based on metamodels to enable the development and interoperability of models.

After reviewing the state of the art and theoretical framework, it could not be found researches that described metamodels for personalized adaptation of pedagogical strategies, with the integration of self-regulation and metamemory in ITS. The main contribution of this thesis is to design a metamodel that enables the integration of self-regulation and metamemory in the process of customizing pedagogical strategies in ITS.

3 METAMODEL FOR PEDAGOGICAL MODULE

This chapter presents a metamodel which describes the concepts commonly used in modeling of pedagogical modules in ITS. The metamodel contains concepts and relationships that are present in the following tasks related to the design of instructional strategies: instructional planning, assessment of instruction and advice on learning activities.

3.1 Metamodel for Pedagogical Module in Intelligent Systems (METAGOGIC)

METAGOGIC is a metamodel for pedagogic strategy modeling in ITS. METAGOGIC was designed based on the analysis of pedagogical models contained in 25 intelligent tutoring systems (*Set I*). The metamodel was validated and refined using a second set of 20 intelligent tutoring systems (*Set VS I*).

A 6-step metamodeling process adapted from FAML (Framework for Agent-oriented Modeling Language)(*Beydoun et al., 2009*) was used to create the METAGOGIC metamodel. Metamodeling is a technique promoted by the Object Management Group (OMG) (OMG, 2013) with the goal to automate the process of model generation in software engineering. Adaptations in the methodology of metamodeling with respect to FAML include: (i) addition of step 0 for the collection of pedagogical models; (ii) inclusion of the task, generalization of concepts, in step 5; and (iii) inclusion of validation techniques in step 6.

The 6-step metamodeling process is a guide that contains detailed instructions on the tasks and processes performed at each stage of metamodeling, see (Caro, Josyula, Cox, & Jiménez, 2014) for more detail. The goal of each step is as follows:

- Step 0: Identifying sources of information and collection of pedagogical models in ITS.
- Step 1: Classification (into sets) of pedagogical models according to the type of metacognition.
- Step 2: Extraction of concepts related to pedagogical strategies in each set created in step 1.
- Step 3: Selection of the concepts commonly used in the models.
- Step 4: Classification of the concepts selected in step 3.
- Step 5: Identification of relationships between selected concepts.
- Step 6: Creating the metamodel of personalization of pedagogical strategies based on steps 4 and 5.

Table 3.1 shows the intelligent tutoring systems analyzed to design METAGOGIC.

Table 3.1. Pedagogical model classification

	ITS model	T cited	Y published
<i>Set I for metamodel development</i>			
1	AutoTutor (Graesser, Chipman, Haynes, & Olney, 2005; Graesser, Wiemer-Hastings, Wiemer-Hastings, & Kreuz, 1999)	678	1999, 2055
2	EML (Koper, 2001)	621	2001
3	*AH systems (Brusilovsky, 2003)	264	2003
4	Eon (Murray, 1998)	216	1998
5	Why2-Atlas (Vanlehn et al., 2002)	195	2002
6	Cognitive Tutor (Aleven et al., 2005)	173	2006
7	Betty ' s Brain (Leelawong, Biswas, & Isis, 2008)	145	2008
8	EUME (Amorim, Lama, Sánchez, Riera, & Vila, 2006)	120	2006
9	ABITS (Capuano, Marsella, & Salerno, 2000)	106	2000
10	eTeacher (Schiaffino, Garcia, & Amandi, 2008)	92	2008
11	Help Tutor (Roll, Aleven, McLaren, & Koedinger, 2011a)	80	2011
12	SlideTutor (Crowley & Medvedeva, 2006)	74	2006
13	AIP-O (Karampiperis & Sampson, 2004)	69	2004
14	SWBES (Bittencourt et al., 2009)	64	2009
15	SmartTutor (B. Cheung, Hui, Zhang, & Yiu, 2003)	63	2003
16	ActiveMath (Melis & Siekmann, 2004)	62	2004
17	Curriculum Tree (Chan, 1992)	61	1992
18	ZOSMAT (Keleş, Ocak, Keleş, & Gülcü, 2009)	44	2007
19	u-Museum (C. Chen & Huang, 2012)	34	2012
20	Genetics Cognitive Tutor (Corbett, Kauffman, Maclaren, Wagner, & Jones, 2010)	30	2010
21	CIRCSIM (Woo et al., 2006)	29	2006
22	Gaze tutor (D'Mello, Olney, Williams, & Hays, 2012)	26	2012

23	DEPTHS (<i>Jeremić et al., 2012</i>)	20	2012
24	eTutor (<i>Heift, 2010</i>)	20	2010
25	ILMDA (<i>Soh & Blank, 2008</i>)	15	2008
Set VS I to be used for validation			
1	*Fuzzy MAS-IP-ITS (<i>Aguilar et al., 2011</i>)	1	2011
2	AIP-W-GA (<i>Lopes & Fernandes, 2009</i>)	3	2009
3	ALLEGRO (<i>Viccari & Jiménez, 2007</i>)	6	2007
4	CSPM (<i>Legaspi, Sison, & Numao, 2004b</i>)	9	2004
5	*GIP (<i>Tan, 1996</i>)	7	1996
6	IMS-LD (<i>Vidal-castro, Sicilia, & Prieto, 2012</i>)	4	2012
7	*BN-CBR-ITS (<i>Ding et al., 2010</i>)	1	2010
8	II-RPS (<i>Ganjanasuwan & Sanrach, 2006</i>)	2	2006
9	*IP-ANN (<i>Seridi et al., 2006</i>)	7	2006
10	Ekit (<i>Escudero & Fuentes, 2010</i>)	3	2010
11	METEOR (<i>Kazi, Haddawy, & Suebnukarn, 2012</i>)	3	2012
12	IPASS (<i>Yu-fen Chen et al., 2004</i>)	2	2004
13	*KM-ITS (<i>Priya et al., 2012</i>)	2	2012
14	IWT (<i>Gaeta et al., 2011</i>)	2	2011
15	METIOREW (<i>Rahman & Farag, 2011</i>)	4	2011
16	INES (<i>Mikic-Fonte, 2010</i>)	2	2010
17	*Graph (<i>Rollande & Grundspenkis, 2012</i>)	1	2012
18	EVIE-m (<i>Pachoulakis, Profit, & Kapetanakis, 2012</i>)	2	2012
19	AMT (<i>L. Zhang et al., 2014</i>)	3	2014
20	EMASPEL (<i>Ben Ammar, Neji, Alimi, & Gouardères, 2010</i>)	36	2010

* Abbreviations used in the table to reference architectures with long names.

METAGOGIC is organized in five packages: `metacore`, `planner`, `advisor`, `assessment` and `users`; see package diagram in Figure 3.1. A package diagram shows how a system is divided into logical groupings and shows the dependencies among these groupings. The icon (---->) represents a dependency.

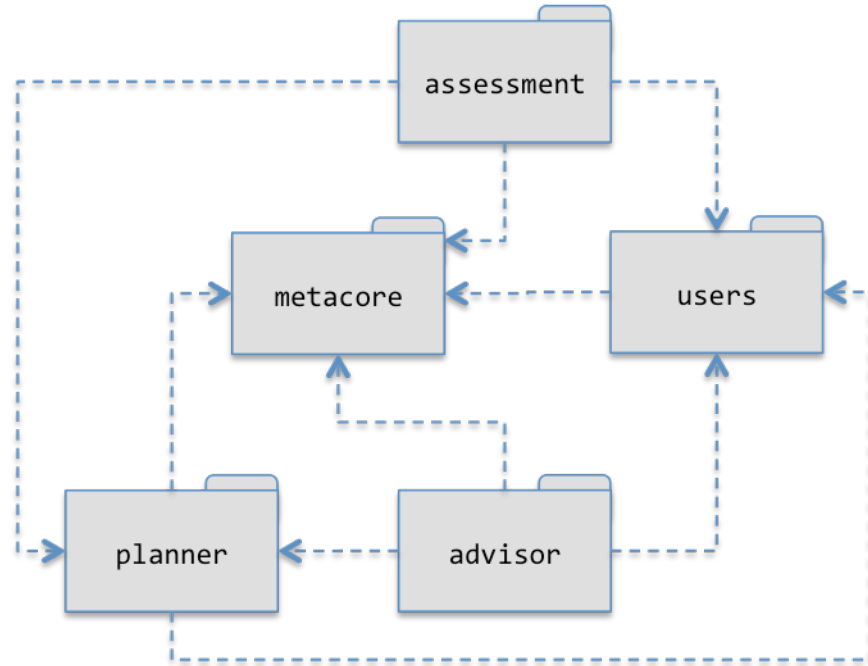


Figure 3.1. Package model in METAGOGIC metamodel; Source: the author.

The METAGOGIC metamodel is presented in 5 diagrams (see Figures 3.2 – 3.6) covering three main functions of a tutor module in an ITS: `Planner`, `Advisor`, `Assessment`; including `metacore` and `User` packages.

3.1.1 Metacore package

The `metacore` package contains the concepts and relationships commonly used in the three main functions of the module tutor in an ITS. Figure 3.2 shows the internal structure of `metacore` package.

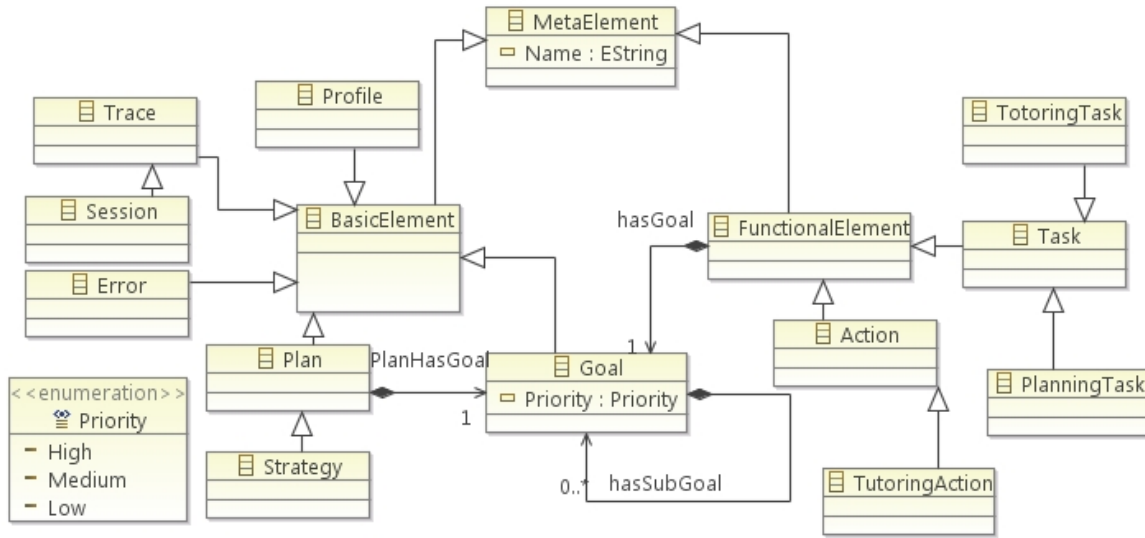


Figure 3.2. Metacore package model in METAGOGIC; Source: the author.

The structure of the package is composed of two types of elements (Figure 3.2): functional elements and basic elements.

(i) Functional elements

The functional elements are tasks that enable an ITS to perform pedagogical reasoning. The pedagogical reasoning refers to the processes and decisions that have to do with the recommendation of pedagogical strategies, personalized tutoring, assessment processes and personalization of the learning environment in an ITS.

Action and Task are two types of functional elements in metacore package:

- Action relates to mechanisms that an intelligent tutor system uses to interact with the user.
- Tasks represent sets of processes inherent to the pedagogical reasoning. METAGOGIC supports two types of tasks: Planning Task and Tutoring Task. Planning Task allows an ITS building plans for adaptation of different aspects of the learning process. Moreover, Tutoring Task enables the ITS to guide the process of tutoring a student in a personalized way.

(ii) Basic elements

The basic elements are common to all four packages that make the metamodel. The basic elements are abstract and are the basis for the generation of particular elements that

interact with the functional elements in packages: `Planner`, `Advisor`, `Assessment` and `User`. For example `Tutoring Session` is an instance of `Session` in `Planner` package.

The metacore package contains the following basic elements: `Error`, `Goal`, `Plan`, `Profile`, `Session`, and `Trace`.

- `Error` is a discrepancy that occurs between the system's expectations and the actual observations.
- `Goal` is an objective that a task or process tries to achieve.
- `Plan` is an organized set of tasks or actions that an ITS performs to achieve an objective.
- `Profile` is a record that stores relevant data that represent the performance of any component of the system or user behavior.
- `Session` maintains records of actions that users perform on the system.
- `Trace` represents records generated by the processes involved in the pedagogical reasoning. `Traces` are elements that can store both data and data structure with rules used in the reasoning process.

3.1.2 Planner package

The main objective of this package is to select the most appropriate pedagogical strategy for each student. The class `Pedagogical Strategy` represents a pedagogical strategy. `Pedagogical Strategy` is a plan and consists of three basic components: `Context`, `Pedagogical Approach` and `Learning Activity`. Figure 3.3 shows the structure of the package.

(i) Context class

The `Context` specification contains the general input data used to configure the pedagogical strategy. The context of the pedagogical strategy identifies three main aspects: (i) the student who will configure the strategy; (ii) the course in which the student is enrolled; and (iii) the lesson in which the strategy will be contextualized.

- `Student`: Student profile is based on the next aspects: learning styles and performance on the course.
- The term learning styles refers to the concept that individuals differ in regard to what mode of instruction or study is more effective for them (*Demirbas & Demirkan, 2007; Tulbure, 2012*). The learning style of the student is one of the most important characteristic to be considered for adaptation of learning in ITS (*Arias et al., 2009*).
- `Course`: A course consists of one or more lessons.
- `Lesson`: Each lesson has a structure that varies according to the student profile.

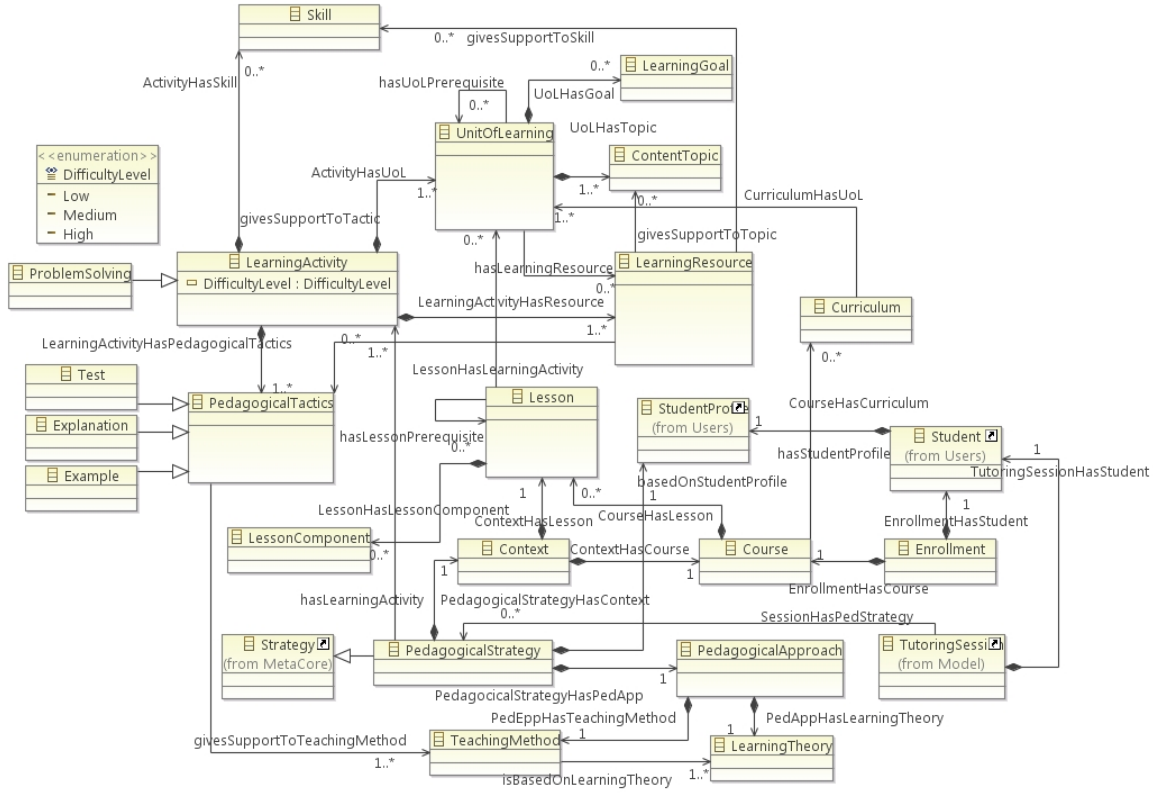


Figure 3.3. Planner package model in METAGOGIC metamodel ; Source: the author.

(ii) PedagogicalApproach class

The pedagogical approach addresses the strategy from learning theories and teaching methods. The pedagogical approach is set from the context of the pedagogical strategy, so it is possible having for each student an individualized pedagogical approach. The pedagogical approach is composed by navigation style, pedagogical theory and the teaching method.

- **LearningTheory** is composed of a diverse set of theoretical frameworks, which try to explain how individuals access knowledge. Many features of pedagogical theories can be partially modeled computationally. In our case we have only included those characteristics that can be modeled computationally, as the type of content sequencing, the type of assistance provided to students and the type of evaluation.
- **TeachingMethod**: A teaching method comprises the principles that imply an orderly logical arrangement of tactics and activities used in lessons of a course. The

teaching methods are based on pedagogical theories; each method may contain all or part of the pedagogical principles of theory that is derived.

- The teaching methods are modeled as classes that are composed of a set of pedagogical tactics and which have an organization of activities, based on a theory of learning

(iii) LearningActivity class

The `LearningActivity` is the third component of the pedagogical strategy and corresponds to the organization and presentation of the content of a `Lesson`.

For each student the instructional activity defines: (i) the most appropriate pedagogic tactics to address the contents of the lesson; and (ii) the format and the order in which the learning resources will be presented in a specific lesson. `LessonComponent`, `PedagogicTactic` and `LearningResource` compose `LearningActivity`.

- `LessonComponents` represents the sections in which the lesson activities are organized. Some students cannot use some components of the lesson because their learning style, e.g. students with reflective style of learning could not use the component activities of the lesson.
- `PedagogicTactics` are composed of actions and resources which are used in the interaction with the student (*Bezerra, 2012*) for providing a personalized teaching.
- `LearningResources` are digital objects such as images, animations, simulations, web pages, and more. Learning resources are the carriers of the content of the lesson and have different formats.

Table 3.2 shows the full list of all classes that are part of the *Planner* package.

Table 3.2. Concepts included in planner package in METAGOGIC metamodel

Concept	Short definition
<code>ContentTopic</code>	Each of the themes that are part of the contents of a course or lesson.
<code>Context</code>	It is a component which function is to contextualize a pedagogical strategy in terms of the course and lesson.
<code>Course</code>	It is a teaching unit managed by one or more tutors (teachers) and has enrolled a group of student. A course has educational objectives, skills that students must acquire and a set of topics related to an area of study.
<code>Curriculum</code>	Curriculum refers to a complete academic program that is addressed in one or more courses. The curriculum contains a planned sequence of instruction and instructional goals.
<code>Enrollment</code>	This class represents the registration of students enrolled in

	a course.
Example	It is a particular type of pedagogical tactic where an example is used to guide the student in learning.
Explanation	It is a kind of pedagogical tactic in which sets of statements are structured so that students understand a topic in detail.
LearningActivity	Activities are designed to create conditions for learning.
LearningGoal	The learning goals refer to skills, knowledge and attitudes that a learner must acquire in a course or lesson.
LearningResource	Learning resources are digital objects such as images, animations, simulations, web pages, and more. Learning resources contain knowledge related to a topic of study.
LearningTheory	Learning theory are theoretical approaches, which describe how information is processed and knowledge is acquired during the learning process.
Lesson	It is a period of time, which has as objective that students learn a particular topic or acquire some skills.
LessonComponent	Each of the sections in which a lesson is structured. For example: Introduction, Explanation, Evaluation and Conclusion.
PedagogicalApproach	It refers to the set of practices and strategies used to teach.
PedagogicalStrategy	The pedagogic strategies are action plans designed to manage issues related to sequencing and organizing the instructional content.
PedagogicalTactic	Pedagogical tactics are composed of actions and resources, which are used in the interaction with the student for providing a personalized teaching.
ProblemSolving	It is an activity where students learn by solving problems.
Skill	It is a cognitive or behavioral ability that a student must acquire in a lesson or course.
TeachingMethod	A teaching method comprises the steps and principles used in the teaching process.
Test	It is an examination that takes a student to determine the level of knowledge about a topic.
UnitOfLearning	This represents a set of content and skills to structure a course or lesson.

3.1.3 Advisor package

This package contains the concepts used to configure and generate feedback in ITS. In class diagram shown in Figure 3.4 it can be seen that **Feedback** is the main class of Advisor package and has a basic structure consisting of a message. Feedback is generated according to the actions that the student performs during a tutoring session; the

FeedbackTrace class allows the system to keep an updated record of each generated feedback.

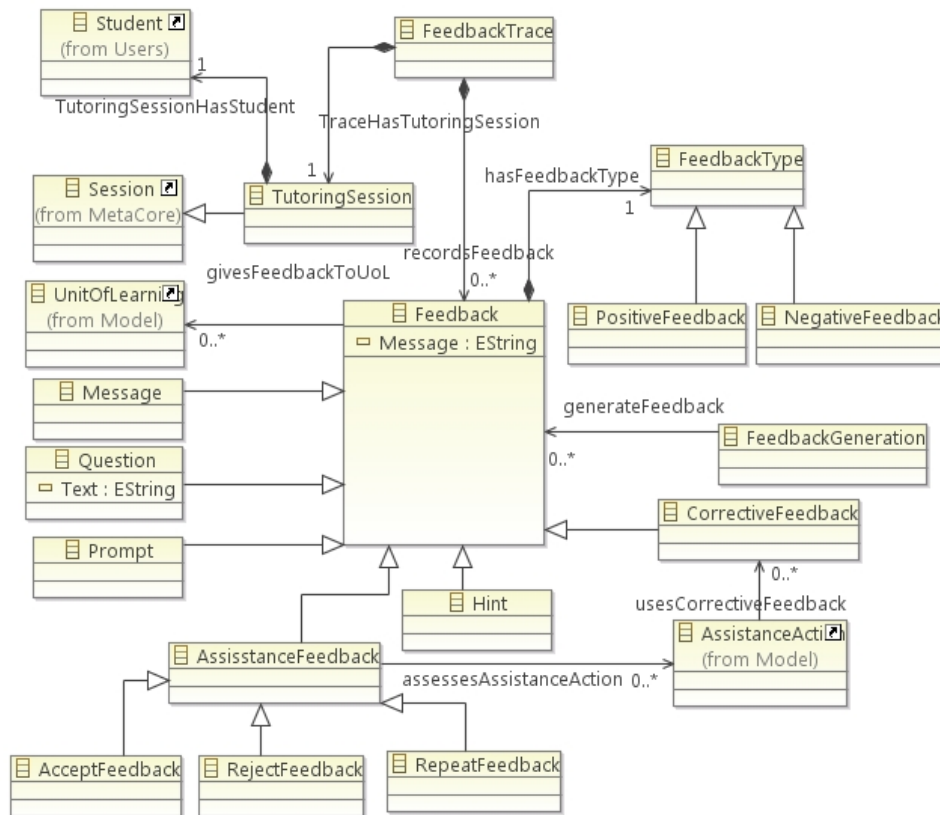


Figure 3.4. Advisor package model in METAGOGIC metamodel; Source: the author.

METAGOGIC supports two types of feedback commonly found in the literature: **PositiveFeedback** and **NegativeFeedback**. **PositiveFeedback** is messages that are sent to the student indicating that an action was successful. **NegativeFeedback** indicates faults made by the student; usually this kind of feedback is used to indicate an error in an answer on a test.

Feedback can be presented to students in different ways depending on the type of system, METAGOGIC supports the following four ways: **Message**, **Question**, **Prompt** and **Hint**. **Message** is a feedback that can be displayed to the student in the form of audio or text in a popup window. The full list of all classes that are part of the **Advisor** package is shown in Table 3.3.

Table 3.3. Concepts included in advisor package in METAGOGIC metamodel

Concept	Short definition
AcceptFeedback	This action represents that an agent accepts a feedback.
AssistanceFeedback	AssistanceFeedback is a particular type of feedback that

	provides information that guides the agent to resolve an error.
CorrectiveFeedback	Feedback used to improve the performance of an agent on a task. This type of feedback is given after making a mistake.
Feedback	Feedback is a mechanism used to give information to an agent (e.g. student, teacher or system) about their performance on a task.
FeedbackGeneration	Process that generates feedback in response to an action of an agent.
FeedbackTrace	It is a record that stores each of the feedback that the system has sent or received.
FeedbackType	It is a generic class that represents each of the types of feedback that a system processes.
Hint	It is a type of feedback that has short text format and is used in graphic user interfaces.
Message	It is a type of feedback that can be sent to an agent in the form of pop-up menu, audio, video or information flow.
NegativeFeedback	Feedback reports an error that an agent has committed but does not provide information on how to fix it.
PositiveFeedback	Feedback reports an error that an agent has committed. This type of feedback uses messages that encourage the agent to seek solutions to the error.
Prompt	It is a kind of feedback that has text format and is used in systems with textual interfaces.
Question	Feedback that use questions to collect information about the user or a particular process.
RejectFeedback	It is an action that represents an agent that rejects a feedback.
RepeatFeedback	It is an action that represents an agent that requests a feedback again.
TutoringSession	Period of time in which a tutor develops a lesson based on some instructional objectives and using various pedagogical strategies.

3.1.4 Assessment package

The objective of this package is grouping classes related to the assessment process of students. METAGOGIC has the **Assessment Session** class that allows keep a record of each assessment that a student performs in a tutoring session. Assessments are based on one or more assessments methods and contain a series of questions. The metamodel

supports two basic methods of assessment: **Test** and **Problem**. Figure 3.5 shows the relationships among the classes in the **Assessment** package.

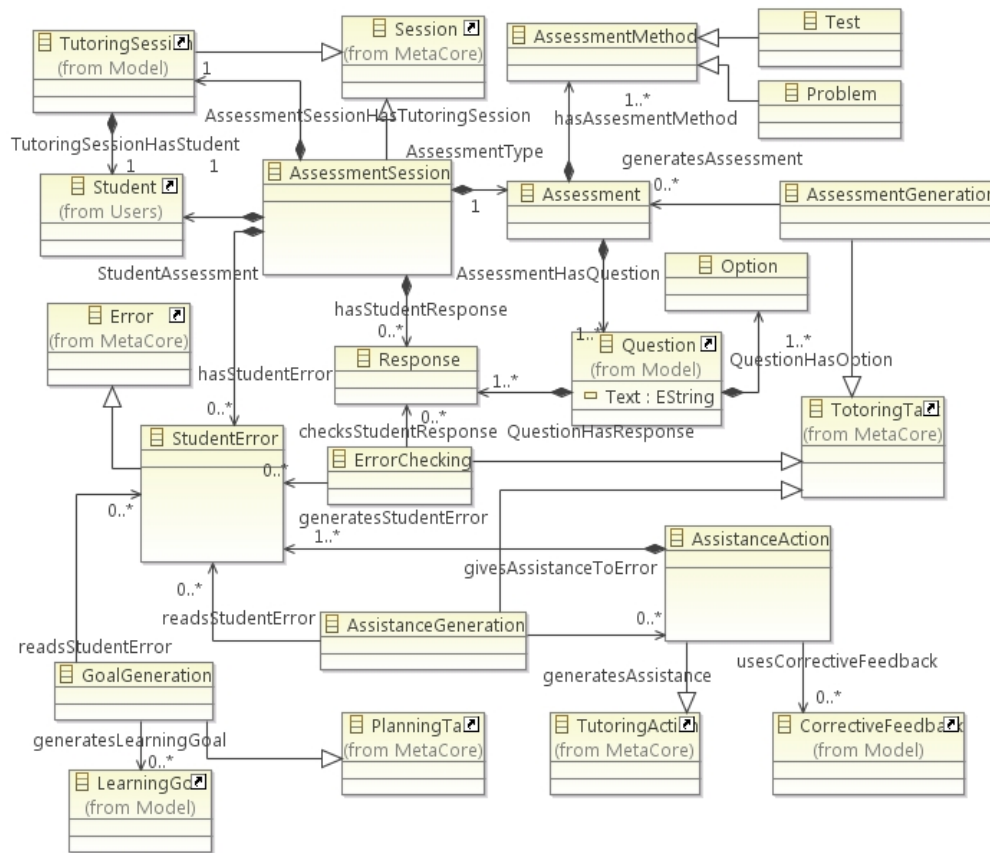


Figure 3.5. Assessment package model in METAGOGIC metamodel; Source: the author.

This package contains the following tasks to manage assessments in an intelligent tutor: **AssessmentGeneration**, **ErrorChecking**, **GoalGeneration** and **AssistanceGeneration**.

- **AssessmentGeneration** is a tutoring task for the personalized generation of assessments based on the profile of the student and the student's performance in the tutoring session.
- **ErrorChecking** is a tutoring task that monitors the responses of students to the evaluation questions in order to find errors. Class **StudentError** represents the response errors and is recorded in the **AssessmentSession** associated with the student.
- **GoalGeneration** is a planning task, which reads each error made by a student in a **TutoringSession**, and then this task generates the learning goals that the student must achieve in the tutoring session.

- **AssistenaceGeneration** is a tutoring task that aims to recommend actions for assistance to deal with errors made by the student in the tutoring session. **StudentError** and **LearningGoal** are the inputs of this **AssistenaceGeneration**.

See Table 3.4 for more information about the classes included in the package.

Table 3.4. Concepts included in assessment package in METAGOGIC metamodel

Concept	Short definition
Assessment	This class represents the different processes of measurable verification of knowledge and skills of a student.
AssessmentGeneration	Tutoring task that generates a new assessment for the student.
AssessmentMethod	Generic class that represents the methods used in the assessments to students.
AssessmentSession	AssessmentSession class keeps a record of each assessment that a student performs in a tutoring session.
AssistanceAction	Action called to assist the student to solve a problem or error.
AssistanceGeneration	It is a tutoring task that allows the generation of assistance activities to guide a student in solving a problem.
ErrorChecking	Tutoring task that monitors the responses of students to the evaluation questions in order to find errors
GoalGeneration	This planning task generates the learning goals that the student must achieve in the tutoring session
Option	Component representing the response options of a question on an exam.
Problem	Assessment method based on problem solving.
Response	Answer given by a student to a question in an examination.
StudentError	Student commits error when trying to solve a problem.
Test	It is an assessment method that consists of a set of questions about a topic or skill.

3.1.5 User package

This package contains the necessary components to model user profiles of an ITS. The main components of the **User** package are **TeacherProfile** and the **StudentProfile**.

The `TeacherProfile` class allows knowing the preferences for system configuration and educational skills of the teacher. The Figure 3.6 shows the relationships among the classes included in the package.

The student profile is composed of `CognitiveProfile`, `LearningProgress`, `BackgroundKnowledge`, `AffectiveState` and `Preference`.

- `CognitiveProfile` stores the learning style and skills of the student. The skills can be cognitive or mastering a study area.
- `LearningProgress` stores information related to student performance in the tutoring sessions. The content and skills in which the student presents poor performance are recorded in the *EducationalNeed* class.
- `BackgroundKnowledge` represents the record of knowledge that a student has before starting a new lesson. `BackgroundKnowledge` is kept updated as the student progresses in lessons. For the first lesson of a course, the students perform the diagnostic evaluation to determine the initial level of knowledge.
- `AffectiveState` allows intelligent tutors detect and store the students' motivational state during an instructional session. METAGOGIC contains support for the two most referenced affective states in the specialized literature (Interest and Boredom). However, other affective states can be instantiated from the `AffectiveState` class according to the characteristics of each intelligent tutor.
- `Preference` is related to the settings that the user makes to the system in terms of customizing the GUI, but may also include search preferences and resources.

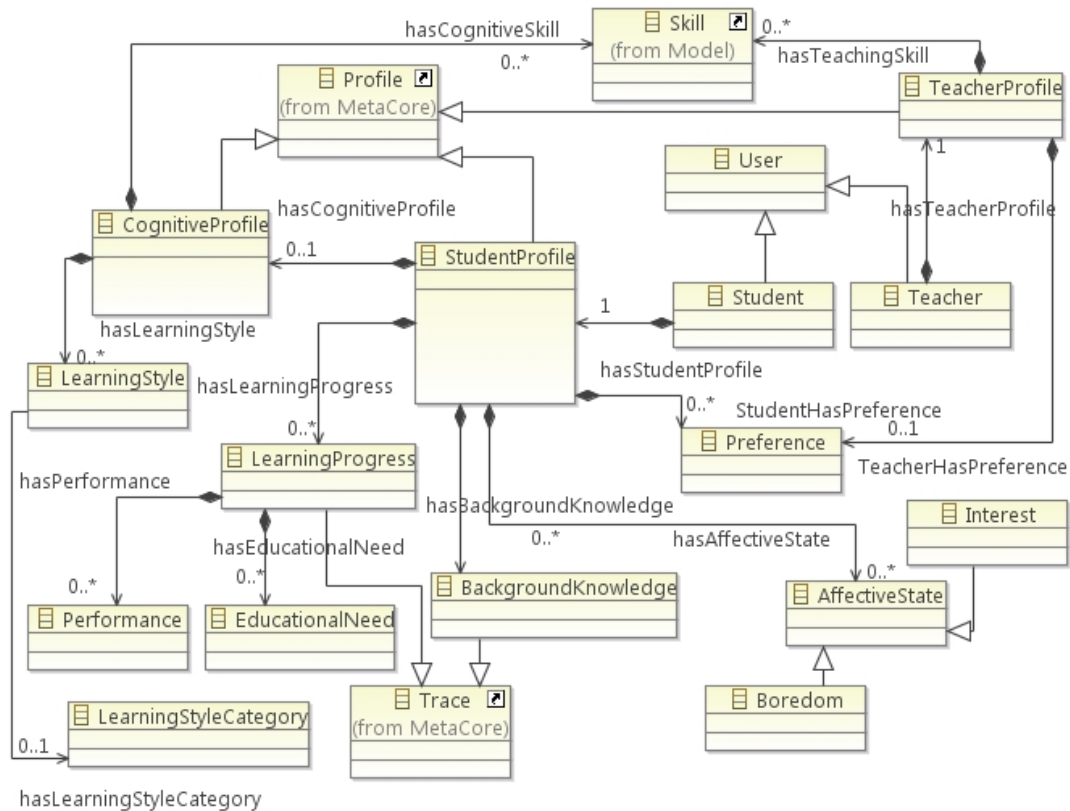


Figure 3.6. User package model in METAGOGIC metamodel; Source: the author.

See Table 3.5 for more information about the classes included in the package.

Table 3.5. Concepts included in user package in METAGOGIC metamodel

Concept	Short definition
AffectiveState	A class that aims to detect and to store the students' motivational state during an instructional session
BackgroundKnowledge	Record of knowledge that a student has before starting a new lesson
Boredom	It is an affective state indicating that the student has no interest in the lesson.
CognitiveStyle	Cognitive Profile stores the learning style and skills of the student.
EducationalNeed	This class stores data related to the educational needs that have been identified in students.
Interest	It is an affective state indicating that the student is interested in the lesson.
LearningProgress	Information related to student's performance in the tutoring sessions.

LearningStyle	Class aims to store and update the student's learning style.
LearningStyleCategory	Representing the category to which belongs a learning style.
Performance	This class stores the performance of a student in a lesson or course.
Preference	Each setting that the user makes to the system
Student	Class that stores and manages data from a teacher.
StudentProfile	Class containing the model of the student in the system.
Teacher	Class that stores and manages data from a teacher.
TeacherProfile	Class containing the model of the teacher in the system.
User	Each of the agents that interact with the system.

3.2 Conclusion of the chapter

In this chapter a metamodel for designing the *module tutor* of an ITS is presented. The metamodel is called METAGOGIC and was synthesized from 45 module tutor models found in the literature. The METAGOGIC metamodel is the result from the first specific objective of this thesis regarding to identify the components and methods that have pedagogical models for the improvement of processes related to personalized adaptation of pedagogical strategies in ITS.

The metamodel is structured into five packages: `MetaCore`, `Planner`, `Advisor`, `Assessment` and `User`. The `MetaCore` package allows the reuse of components and simplifies the design of metamodel because brings together the elements common to the other four packages. The `Planner` package contains the necessary elements for an intelligent tutoring system that recommends the most appropriate pedagogical strategy for each student. The `Advisor` package has the elements related to the generation of feedback between the tutorial system and the user. The `Assessment` package aims to group classes related to the assessment process of students in learning sessions and the `User` package contains the necessary components to model user's profile in an ITS.

4 METAMODEL FOR METACOGNITION SUPPORT IN INTELLIGENT SYSTEMS

The main objective of this chapter is to present the design of a metamodel for metacognition support in Intelligent Systems. The metamodel integrates the concepts and relationships related to the three following types of metacognition: self-regulation, metamemory and meta-comprehension.

4.1 Metamodel for metacognition support in Intelligent Systems (MISM)

MISM is a comprehensive and general purpose metamodel that covers and describes a broad range of commonly referenced concepts in metacognitive models in the area of AI. MISM was synthesized from the analysis of 20 metacognitive models (*Set I*) with application in intelligent systems. A second set of 20 metacognitive models (*Set VS I*) was used for the metamodel refinement and concepts coverage validation. The Table 4.1 shows the models used to synthesize and validate MISM. The entire process of metamodeling used for synthesizing MISM, see (Caro et al., 2014) for more detail.

Table 4.1. Metacognitive model classification in MISM metamodel

	Model	T cited	Y published
<i>Set I for metamodel development</i>			
1	Meta-AQUA (Cox & Ram, 1999)	85	1999
2	Modeling meta-cognition in a cognitive architecture (MCLARION) (Sun et al., 2006)	48	2006; 2007
3	MIDCA (Cox, Oates, & Perlis, 2011)	13	2011
4	E-SOAR (Laird, 2008)	217	2008
5	MCL (Anderson et al., 2006; Schmill et al., 2011)	51	2006; 2011
6	Dormobile (Self, 1994)	41	1994
7	M-SNePS (Shapiro et al., 2007)	24	2007
8	MAMID Cognitive-Affective Architecture (Hudlicka, 2005)	19	2005
9	ITS-SR-CBR (Soh & Blank, 2008)	14	2008
10	Metareasoning and meta-level learning in a hybrid knowledge-based architecture (MMLHKA) (Christodoulou & Keravnou, 1998)	10	1998
11	Decentralized Metacognition in Context-Aware Autonomic Systems	10	2010

	(DMCAAS) (Kennedy, 2010)		
12	GMU BICA (Alexei Samsonovich & Ascoli, 2006; Alexei Samsonovich & Jong, 2005)	41	2005; 2006
13	Autognotic (Stroulia & Goel, 1995)	66	1995
14	REM (Murdock & Goel, 2001, 2008)	42	2001; 2008
15	Augur (J. K. Jones & Goel, 2012)	5	2012
16	Epilog (Morbini & Schubert, 2008)	17	2008
17	MGSS* - Othello (Russell & Wefald, 1989)	84	1989
18	H-CogAff (Sloman & Chrisley, 2003)	155	2003
19	Know Thyself (Pasquali, Timmermans, & Cleeremans, 2010)	33	2010
20	MRA (Pěchouček, Štěpánková, Marik, & Jaroslav, 2003)	14	2003
<i>Set VS I to be used for validation</i>			
1	MCEL (Azevedo, 2002)	80	2002
2	Metacognitive neural network (MNN) (Sateesh & Suresh, 2012)	33	2012
3	INCA (Oentaryo & Pasquier, 2008)	6	2008
4	Metacognitive behavior in adaptive agents (MAAA) (Thompson, Cohen, & Freeman, 1995)	5	1995
5	Multi-Level Introspection framework (MLIF) (Krause, Schermerhorn, & Scheutz, 2012)	5	2012
6	Meta-cognitive architecture for planning in uncertain environments (MAPUE) (Cannella, Chella, & Pirrone, 2013)	1	2013
7	Representing Metacognitive Experience (MPE) (Oehlmann, Edwards, & Sleeman, 1995)	15	1995
8	Imitative Consciousness (Moura & Sarma, 2005)	17	2005
9	IDA (Franklin, 2000)	50	2000
10	CMattie (Zhaohua Zhang, Franklin, & Dasgupta, 1998)	52	1998
11	Cognitive Tutor (Walker, Koedinger, McLaren, & Rummel, 2006)	9	2006
12	Meta-Radar (Capraro, Wicks, & Schneible, 2010)	1	2010

13	The Constructor Metacognitive Architecture (*TCMA) (Alexei Samsonovich, 2009)	7	2009
14	On-line (Anita Raja, Alexander, & Mappillai, 2006)	6	2006
15	CAILE (Linn, Segedy, Jeong, Podgursky, & Biswas, 2009)	5	2009
16	MAVEN (Kim et al., 2008)	4	2008
17	Metacognitive Classifier ACT-R (Vinokurov, Lebiere, Herd, & Reilly, 2011)	4	2011
18	Metacognitive Radio (Gadhiok et al., 2011)	3	2011
19	HICA-SRL (Alexei Samsonovich, 2010)	2	2010
20	MJ-CBR (Caro, Jimenez, & Paternina, 2012)	1	2012

* Abbreviations used in the table to reference architectures with long names.

MISM is organized in four packages: **metacore**, **selfregulation**, **metamemory** and **metacomprehension**; see package diagram in Figure 4.1. A package diagram shows how a system is divided into logical groupings and shows the dependencies among these groupings. The icon (---->) represents a dependency.

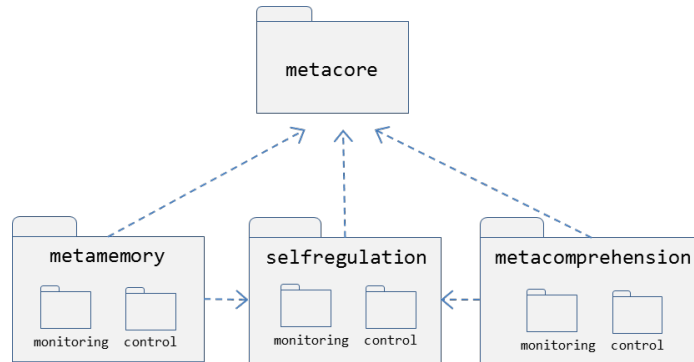


Figure 4.1. Package model in MISM metamodel

All the packages contain two sub-packages named **monitoring** and **control**. The **monitoring** sub-package contains tasks, elements and relationships necessary to perform the functions of monitoring the reasoning and memory processes that run in an intelligent system. The **control** sub-package contains tasks, elements and relationships necessary for meta-level intervention in the reasoning and memory processes performed by an intelligent system.

The MISM metamodel is presented in 6 diagrams (see Figures 4.2 - 4.7) covering three type of metacognition: *self-regulation*, *metamemory* and *meta-comprehension*. Each type of metacognition is divided into monitoring and control process: *metacore*, *selfregulation.monitoring*, *selfregulation.control*, *meta-memory.monitoring*, *metamemory.control*, *meta-comprehension.monitoring* and *metacomprehension.control*.

4.1.1 metacore Package

The concepts and relationships commonly used for the three types of metacognition compose the *metacore*. *metacore* is a package that allows the reuse of components (relationships and concepts), reducing the complexity in the design of the structure of metacognition components. Figure 4.2 shows the internal structure of *metacore* package.

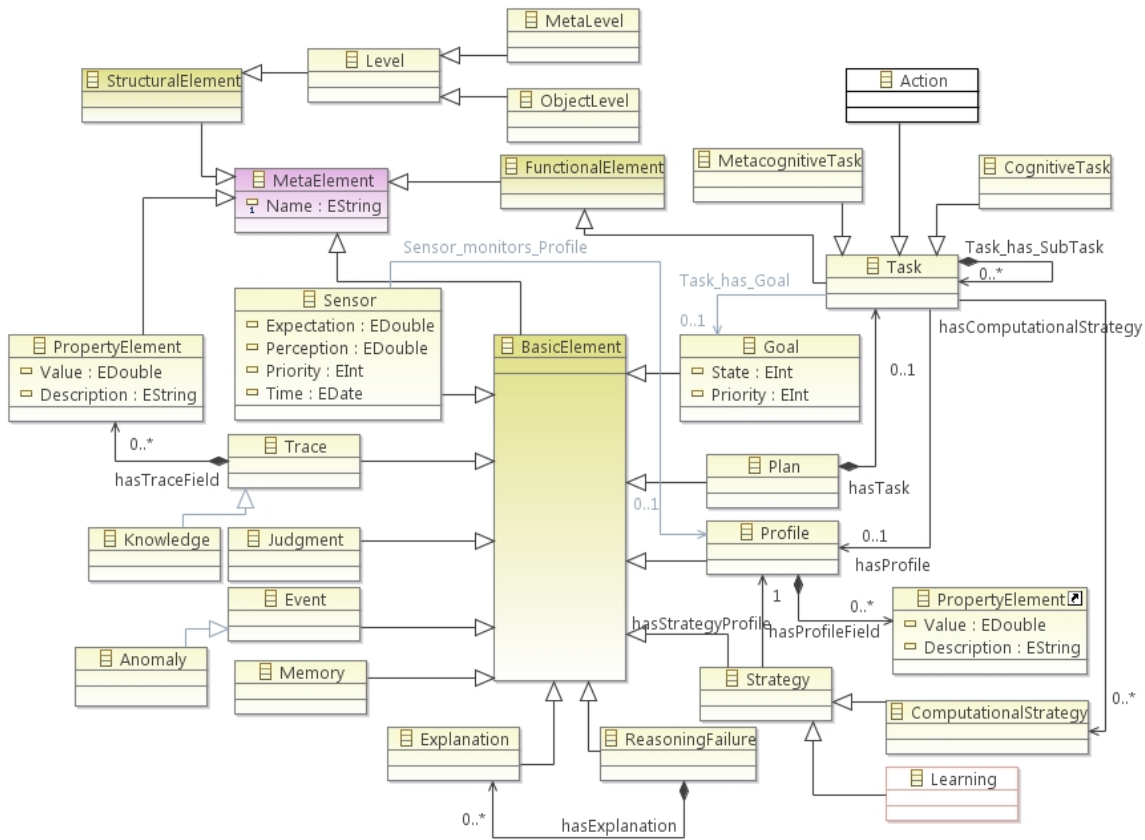


Figure 4.2. Internal structure of *metacore* package in MISM metamodel (Caro et al., 2014)

The structure of the metamodel is composed of three types of elements: structural elements, functional elements and basic elements.

i) Structural elements

Structural elements are containers into which the functional and basic elements are embedded; the main structural element is the `Level`. Structural elements are composed of two cognitive levels named `ObjectLevel` and `MetaLevel`.

- `ObjectLevel` is an abstraction level that contains the model that an intelligent system has for reasoning about the world to solve problems.
- `MetaLevel` is an abstraction level of representation of the reasoning of an intelligent system. The meta-level includes the components, knowledge and mechanisms necessary for a system to monitor and control their own learning and reasoning processes.

ii) Functional elements

The functional elements are tasks that enable reasoning and decision-making. The main functional element is the `Task`. A `Task` is composed of a finite set of organized instructions. Every `Task` has at least one goal and only one output.

- `CognitiveTask` is a kind of `Task` that enables the processing (transformation, reduction, elaboration, storage and retrieval) of information by applying knowledge and decision making in order to meet the objectives of the system. All cognitive tasks are object-level components.
- `MetacognitiveTask` is a kind of `Task` that may be used to explain errors in some reasoning task or to select among cognitive “algorithms” in order to perform the reasoning at object-level. All metacognitive tasks are meta-level elements.
- Actions are tasks that define the behavior of a system in an environment.

iii) Basic elements

Basic elements are those that are common to metacognition types addressed in this research. Basic elements are: `Event`, `Strategy`, `Goal`, `Constraint`, `Judgment`, `Expectation` and `Sensor`. The Table 4.2 shows the definition of each concept included in `metacore` package.

Table 4.2. Concepts included in `metacore` package in MISM metamodel

Concept	Short definition
<code>Action</code>	Action refers to the process by which agents actually perform each task in the plan.

Anomaly	An anomaly is an unusual event that occurs in the object-level. Anomalies are candidates to become failures of reasoning.
BasicElement	Basic elements are those that are common to metacognition types addressed in this research: self-regulation, metamemory and meta-comprehension.
CognitiveTask	Cognitive tasks are actions that enable the processing (transformation, reduction, elaboration, storage and retrieval) of information by applying knowledge and decision making in order to meet the objectives of the system.
ComputationalStrategy	A computational strategy is an algorithm or set of algorithms used to perform some task.
Event	The events represent actions that are performed in the object-level.
Explanation	Explanations contain the identified causes of some reasoning failure.
FunctionalElement	The functional elements are tasks that enable reasoning and decision-making.
Goal	Goals are objectives that drive a task or process
Judgment	Metacognitive judgments represent assessments performed in the meta-level about events that occur in object-level. These judgments provide information that the system uses to determine whether it is able to attempt a solution for a reasoning failure.
Knowledge	This concept represents the structures used to store the acquired knowledge.
Learning	Set of processes performed to acquire new knowledge.
Level	This concept represents each of the levels of abstraction that form a cognitive architecture.
Memory	This concept represents each of the memory types present in the natural intelligence.
MetacognitiveTask	The metacognitive task may be to explain errors in the cognitive task or it may be to select among cognitive "algorithms" to perform the reasoning
MetaElement	<i>MetaElement</i> is an abstract concept that occupies the upper level of the metamodel and of which the other concepts inherit some properties.
MetaLevel	It is a level of representation of reasoning of a system. The meta-level includes the components and mechanisms necessary for a system to monitor and control its own learning and reasoning processes.

ObjectLevel	The object-level contains a model for reasoning about the world to solve problems
Plan	Organized set of tasks performed to achieve a goal.
Profile	Profiles are records that store important data about the performance of a functional element. The performance profile is used to evaluate the results of the functional element.
PropertyElement	This element allows users to add new properties to each concept of the metamodel.
ReasoningFailure	It is an anomaly in a cognitive task. Usually reasoning failure is related to an unfinished task or a discrepancy between the expected result and the real result of the task.
Sensor	Sensors are associated with the CognitiveTasks. A Sensor monitors computational data generated by a CognitiveTask and is composed of the following structure: $\langle id, observation, expectation, P, S \rangle$; where id is a unique identifier, $observation$ is the value perceived by the Sensor from computational data, $expectation$ is an expected value for $observation$ attribute, P is the priority level for focus attention $p \in P$ and $S = \{low, medium, high\}$ and S is the Sensor state, $s \in S$ and $S = \{active, inactive\}$.
Strategy	A strategy is a high level plan of finite actions designed to achieve a particular goal.
StructuralElement	Structural elements are containers into which the functional and basic elements are embedded; the main structural element is the Level.
Task	A task is a piece of computation, which represents a process that must be completed. Tasks have objectives, inputs and outputs.
Trace	Trace represents the records generated by cognitive and metacognitive tasks. The Traces are elements that can store structures and rules used in CognitiveTask and MetacognitiveTask.

4.1.2 Self-Regulation package

The package of self-regulation is composed of two sub-packages: `selfregulation.monitoring`, `selfregulation.control`.

4.1.3 `selfregulation.monitoring` package

Monitoring package includes mechanisms for detecting reasoning failures at the object-level. The main purpose of monitoring is to provide enough information to make effective decisions in the meta-level control. Each reasoning task made in the object-level has a performance profile that is continuously updated in the meta-level. The performance profile is used to evaluate the results of each reasoning task.

The main monitoring tasks of `selfregulation.monitoring` package on MIMS are: `ProfileGeneration`, `FailureDetection`, `FailureExplanation` and `GoalGeneration`.

Figure 4.3 shows the internal structure of `selfregulation.monitoring` sub-package.

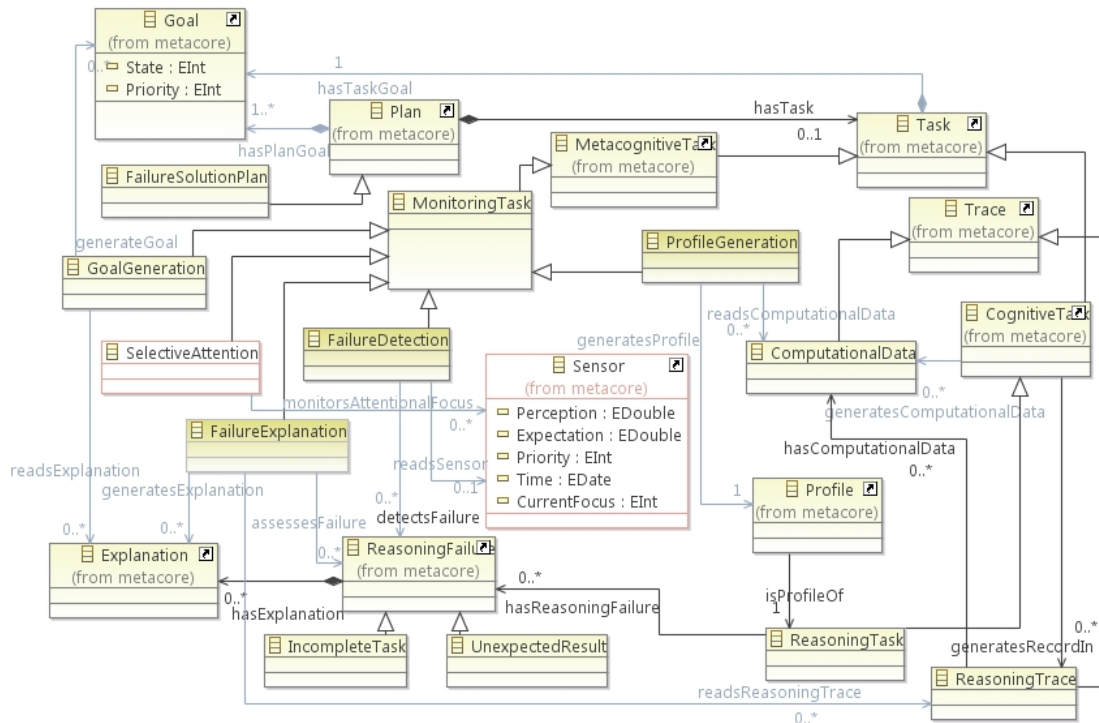


Figure 4.3. Internal structure of `selfregulation.monitoring` package in MISM metamodel (Caro et al., 2014)

Table 4.3 presents a short definition of each concept included in `selfregulation.monitoring` package.

Table 4.3. Concepts included in `selfregulation.monitoring` package in MISM metamodel

Concept	Short definition
ComputationalData	The computational data are numerical values produced during the execution of some cognitive task after performing

	some computational operations. This data type can contain both the output generated by the task as well as the partial data from computational processing.
FailureDetection	It is a metacognitive task that allows the detection of failures in the reasoning processes that occur at the object-level.
FailureExplanation	It is a metacognitive task that allows the generations of explanations for the failures of reasoning identified in reasoning processes are performed at object-level.
FailureSolutionPlan	It corresponds to a plan created in the meta-level in order to solve some reasoning failure detected in the object-level.
GoalGeneration	This metacognitive task allows the generation of new goals in order to deal with failures of reasoning at the object-level.
IncompleteTask	It is one of the possible causes of failure of reasoning. This occurs when a task cannot be fully developed and is detained in any of their instructions.
MonitoringTask	Monitoring tasks include mechanisms for detecting reasoning failures in object-level. The main purpose of monitoring is to provide enough information to make effective decisions in the meta-level control. The monitoring process is done through information feedback that is gathered at the meta-level from the object-level.
ProfileGeneration	It is a metacognitive task that allows the creation of profiles that contain relevant information about the reasoning processes that take place at the object-level.
ReasoningTask	It is a particular type of cognitive task that allows the system to generate conclusions from existing knowledge to solve problems and make decisions using logical techniques.
ReasoningTrace	It is an element that can store data and reasoning structures (e.g. rules) used in the processes of reasoning.
SelectiveAttention	It is a mechanism that allows the meta-level to focus the attention on a specific event that occurs at the object-level. Selectiveattention assigns levels of importance to each event that occurs at the object-level.
UnexpectedResult	It is one of the possible causes of failure of reasoning. This occurs when a task generates a different output to the

expected output.

When a cognitive task is running, then it generates computational data. `ProfileGeneration` reads the computational data and generates a `Profile` of the `CognitiveTask`. Each `CognitiveTask` in the object-level has a performance `Profile` in the meta-level; thus the meta-level is always informed of the status of the reasoning made in the object-level.

The `Sensor` has the function of monitoring the profiles of cognitive tasks in order to detect disturbances or anomalies that may represent reasoning failures produced by the cognitive task. `FailureDetection` reads the properties of a `Sensor`. If the `Sensor` finds a discrepancy between observations and expectations regarding the performance of the `CognitiveTask`, then `FailureDetection` detects a `ReasoningFailure` in the `CognitiveTask` monitored. `FailureExplanation` generates an `Explanation` of the cause of the `ReasoningFailure`, using as inputs, the assessment of the failure and reading of the `ReasoningTrace`. `GoalGeneration` produces new goals based on the `Explanation` for solving the failure detected. A plan to solve the `ReasoningFailure` is built based on the new `Goal`. The plan is called `FailureSolutionPlan`.

4.1.4 selfregulation.control package

The main function of the `selfregulation.control` sub-package is to recommend to object-level the best computational strategy to resolve a reasoning failure; in this way meta-level control improves the quality of decisions made by the IS. The meta-level control decides whether to invoke a task, which task to invoke, and how much resource to invest in the reasoning process (Dannenhauer, Cox, Gupta, Paisner, & Perlis, 2014). Therefore, the main control tasks of this package on MISM are: `ControlActivation` and `StrategySelection`. Figure 4.4 shows the class diagram corresponding to `selfregulation.control` package.

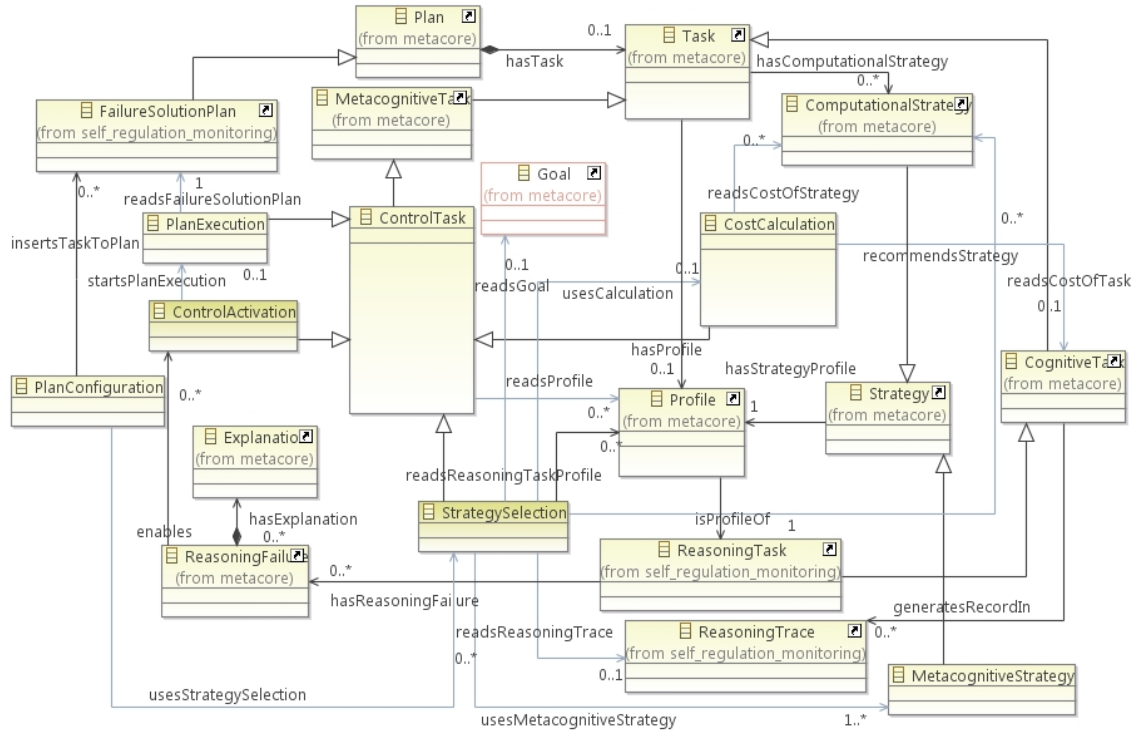


Figure 4.4. Internal structure of `selfregulation.control` package in MISM metamodel. (Caro et al., 2014)

Table 4.4 presents a short definition of each concept included in `selfregulation.control` package.

Table 4.4. Concepts included in `selfregulation.control` package in MISM metamodel

Concept	Short definition
ControlTask	Control tasks are intended to intervene in the processes taking place in the object-level. Metacognitive control is performed to solve reasoning failures or improve processes in the object-level.
PlanExecution	This metacognitive task acts as an engine that executes each of the tasks that constitute the solution plan for a reasoning failure.
ControlActivation	When a reasoning failure is detected then the meta-level control mechanism is activated. The implementation of the failure solution plan is the main action started by <code>controlactivation</code> task.
StrategySelection	Once a reasoning failure is detected and explained by meta-level, then this metacognitive task assesses the strategies available at the object-level to select the most appropriate to

<code>CostCalculation</code>	address the reasoning failure. It is a metacognitive task to estimate the cost of execution of a cognitive task. Cost information generated by this task is used for selecting the most appropriate and less costly cognitive task.
<code>PlanConfiguration</code>	This metacognitive task allows meta-level to add goals and new tasks to the plans generated to solve reasoning failures occurred at the object-level.
<code>MetacognitiveStrategy</code>	It is a particular type of high-level strategy that aims to consciously improve the process of reasoning and learning.

A `FailureSolutionPlan` can activate the metacognitive control. `ControlActivation` task starts `PlanExecution`. `StrategySelection` is one of the tasks that comprise the plan. `StrategySelection` task reads profiles of cognitive tasks and uses `MetacognitiveStrategy` to recommend computational strategies. The computational strategies are recommended to the `CognitiveTask` in order to solve the `ReasoningFailure`.

4.1.5 Metamemory package

Metamemory package contains tasks and metacognitive components involved in self-regulation or self-awareness of memory. This package contains components used to model processes of reasoning about events in memory; for example, storage and retrieval. Metamemory package is structured in the following sub-packages: `metamemory.monitoring`, `metamemory.control`.

4.1.6 `metamemory.monitoring` package

Monitoring package includes mechanisms for detecting events in memory and performing deep search processes on the meta-level knowledge about the object-level. The main monitoring tasks of metamemory package on MISM are: `ProfileGeneration`, `EventDetection`, `EventIdentification`, `FailureDetection`, `FailureExplanation`, `JudgmentTriggering`, `DeeperReasoning` and `GoalGeneration`. Figure 4.5 shows the internal structure of `metamemory.monitoring` package.

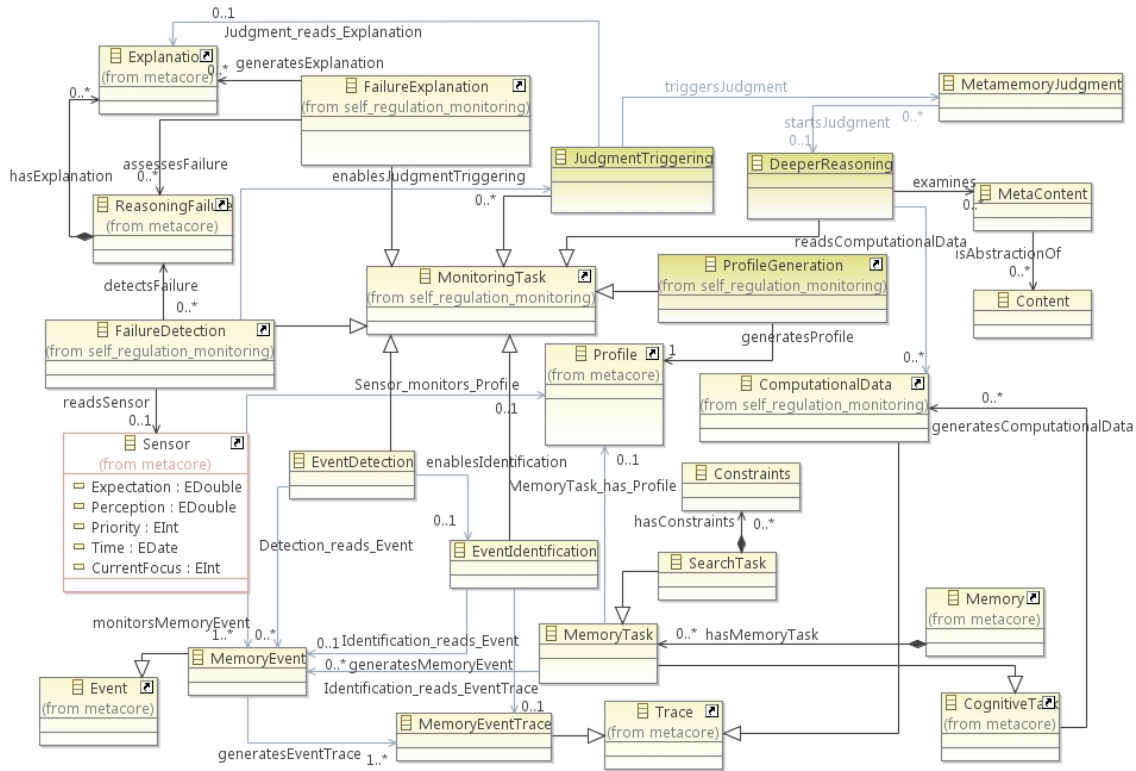


Figure 4.5. Internal structure of `metamemory.monitoring` package in MISM metamodel (Caro et al., 2014)

Table 4.5 presents a short definition of each concept included in `metamemory.monitoring` package.

Table 4.5. Concepts included in `metamemory.monitoring` package in MISM metamodel

Concept	Short definition
Constraints	In a memory event, the constraints refer to the information requirements that must be satisfied so that the <code>Event</code> fulfills the goals. If the information constraints of an event are different from the constraints required to execute a search by default, then the meta-level detects a change in the constraints of the event.
Content	Content represents a unit of information stored in the memory.
DeeperReasoning	If any change in the constraints of an information retrieval task is detected in event memory, then the meta-level decides to launch a deeper reasoning process. The reasoning involves the examination and assessment of the performance of the information retrieval task with similar constraints in the past.

EventDetection	When a new memory event trace is stored in the meta-level, the monitoring process starts the meta-level.
EventIdentification	When a new memory event is detected, the meta-level proceed to identify this event.
JudgmentTriggering	This metacognitive task triggers judgments depending on the knowledge that the meta-level has about the processes that are performed in memory.
MemoryEvent	In a cognitive system, when a process calls a search task in the memory, then a memory event is triggered.
MemoryEventTrace	The meta-level stores traces of all the events that occur in memory.
MemoryTask	This concept represents any task that runs a process on memory.
MetaContent	This is the knowledge that the meta-level possess about the content of the memory.
MetamemoryJudgment	Metamemory judgments represent assessments performed in the meta-level about events that occur in memory. These judgments provide information that the system uses to determine whether it is able to attempt retrieval or storage.
SearchTask	This metacognitive task includes processes associated with accessing of stored information

ProfileGeneration reads the computer data that are generated by a MemoryTask; then a Profile in the meta-level for the MemoryTask is generated. In MISM, the processes operating on the memory such as the retrieval and storage of information are considered as MemoryEvent. MemoryEvent are monitored by sensors to detect anomalies or discrepancies between expectations and observations about the performance of memory tasks. FailureDetection task evaluates the anomalies and identifies possible ReasoningFailures. The FailureExplanation task generates an Explanation of the possible cause of the ReasoningFailure. JudgmentTriggering reads the Explanations and triggers a MetamemoryJudgment about the ReasoningFailure. For example, if the ReasoningFailure task has relation with data that can not be retrieved from memory then MetamemoryJudgment can represent that the system knows that there is not sufficient information for the search.

4.1.7 **metamemory.control** Package

Control package include processes for the recommendation of search strategies on memory. The main control tasks in metamemory.control package on MISM are: StrategySelection and PlanExecution. In metamemory, StrategySelection works the same way as in self-regulation but with the additional inputs of search task constraints and metamemory judgments. Additional inputs in the metacognitive control

are inherent to memory functions, for example, the meta-level using a `MetamemoryJudgment` may: (i) assess whether or not the information is being stored; and (ii) consider making a deeper search for information. `PlanExecution` maintains the same structure as the self-regulation package. `SearchStrategy` is a strategy of searching for information that may be used by a search task.

Figure 4.6 shows the internal structure of `metamemory.control` package.

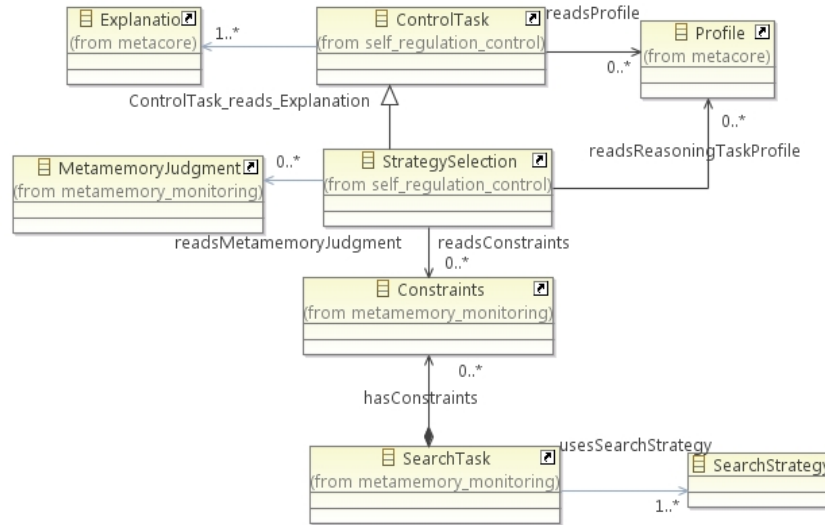


Figure 4.6. Internal structure of `metamemory.control` package in MISM metamodel (Caro et al., 2014)

4.1.8 Meta-comprehension package

Meta-comprehension package groups the component and metacognitive tasks related to self-regulation or self-awareness of a topic. Figure 4.7 shows the internal structure of `metamemory.control` package.

In meta-comprehension the source of the topic can be: (i) external to the system, such as sensory input; and (ii) or internal, such as reasoning trace generated by a `CognitiveTask`. The particular concepts that were identified in the self-regulation package in the meta-comprehension component were: `StoryUnderstanding`, `ReasoningKnowledgeTrace`, `MetaExplanation` and `UnusualEvent`.

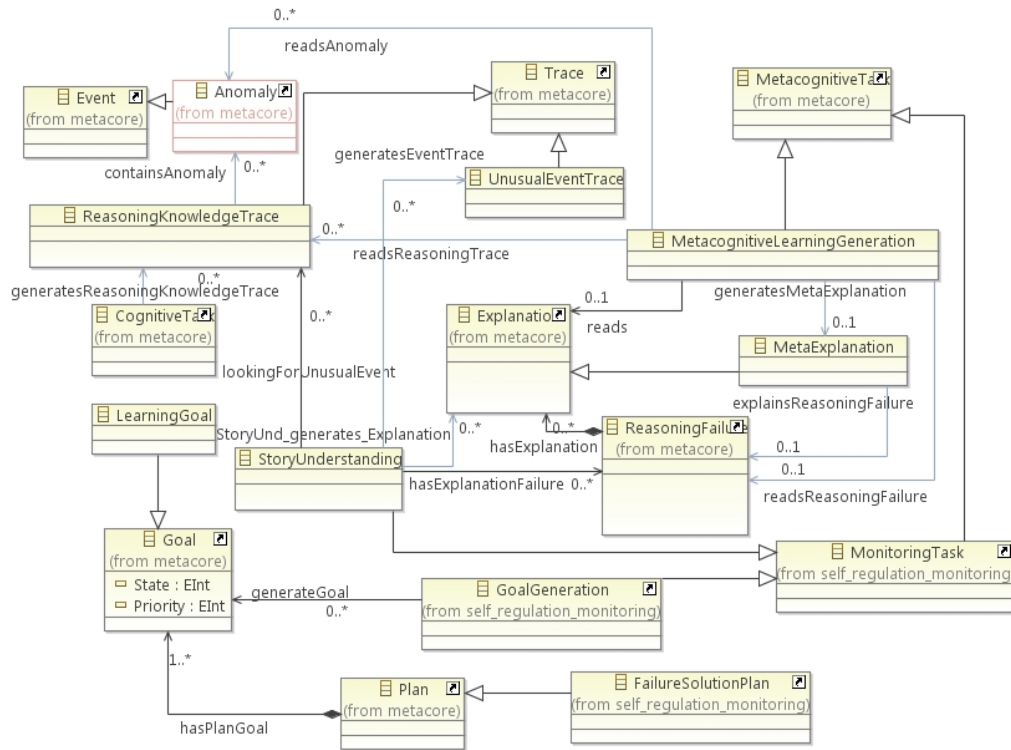


Figure 4.7. Internal structure of `metacomprehension.monitoring` package in MISM metamodel (Caro et al., 2014)

Table 4.6 presents a short definition of each concept included in `metacomprehension.monitoring` package.

Table 4.6. Concepts included in `metacomprehension.monitoring` package in MISM metamodel

Concept	Short definition
<code>LearningGoal</code>	It is a specific learning goal that system wants to achieve.
<code>MetacognitiveLearningGeneration</code>	The process by which the meta-level acquires new knowledge.
<code>MetaExplanation</code>	Meta-explanation refers to the explanation of an error in an explanation of a reasoning failure.
<code>ReasoningKnowledgeTrace</code>	This represents the record of logical operations and knowledge structures used in reasoning processes in the object-level.
<code>StoryUnderstanding</code>	Metacognitive task trying to understand the cause of a reasoning failure.
<code>UnusualEventTrace</code>	When the meta-level attempts to understand the cause of a reasoning failure, then a record

The `StoryUnderstanding` task refers to the process of analyzing the reasoning trace (`ReasoningKnowledgeTrace`) of a cognitive task in order to understand the causes of bad decisions that caused a `ReasoningFailure` in the system. Usually these tasks look for `UnusualEvent` in the `ReasoningKnowledgeTrace`. `MetaExplanation` is a metacognitive task that aims to explain the errors in the explanations given to a `ReasoningFailure`. This task is very important because erroneous explanations of `ReasoningFailure` can lead to erroneous solutions that hinder the functioning of the entire system.

4.2 Conclusion of the chapter

This chapter presents the results from the second specific objective of this thesis regarding to characterize the structural properties that have meta-cognitive models, to be used in the integration of metamemory management and self-regulation in intelligent systems. In this sense, the design and validation of a general purpose metamodel named `MISM` was presented. `MISM` is sufficient to describe a broad range of commonly referenced concepts in AI metacognitive models that exist in the literature. It was presented in Unified Modeling Language (UML) format for an easier understanding.

`MISM` was synthesized from the analysis of 20 metacognitive models (*Set I*) with application in intelligent systems. A second set of 20 metacognitive models (*Set VS I*) was used for the metamodel refinement and concepts coverage validation.

`MISM` is organized in four packages: `metacore`, `selfregulation`, `metamemory` and `metacomprehension`. The `metacore` package facilitates the reuse of elements in different metacognitive components. `MISM` facilitates the integration of metacognitive components in the design of intelligent systems because it is based on independent packages that share common design elements in `metacore`.

5 MOF-BASED METAMODEL FOR PERSONALIZATION OF PEDAGOGICAL STRATEGIES USING METACOGNITION IN ITS

This chapter presents the design of a MOF-based metamodel for the generation of models for personalized adaptation of pedagogical strategies integrating metamemory and self-regulation in ITS, which is the main objective of this thesis. Initially the MOF-based metamodel is presented, and then a concrete syntax and visual modeling tool for the metamodel are introduced. Finally the methods used for validation of the metamodel are described.

5.1 MOF-based metamodel

The metamodel proposed has a conceptual architecture with four levels of modeling according to the standard MOF (see Figure 5.1) that allows the definition of models: level M1, for instance, (a UML class diagram for a concrete application) based on metamodels (level M2, for instance, UML), which in turn are all defined by means of a universal object-oriented and auto-defined meta-metamodel (level M3).

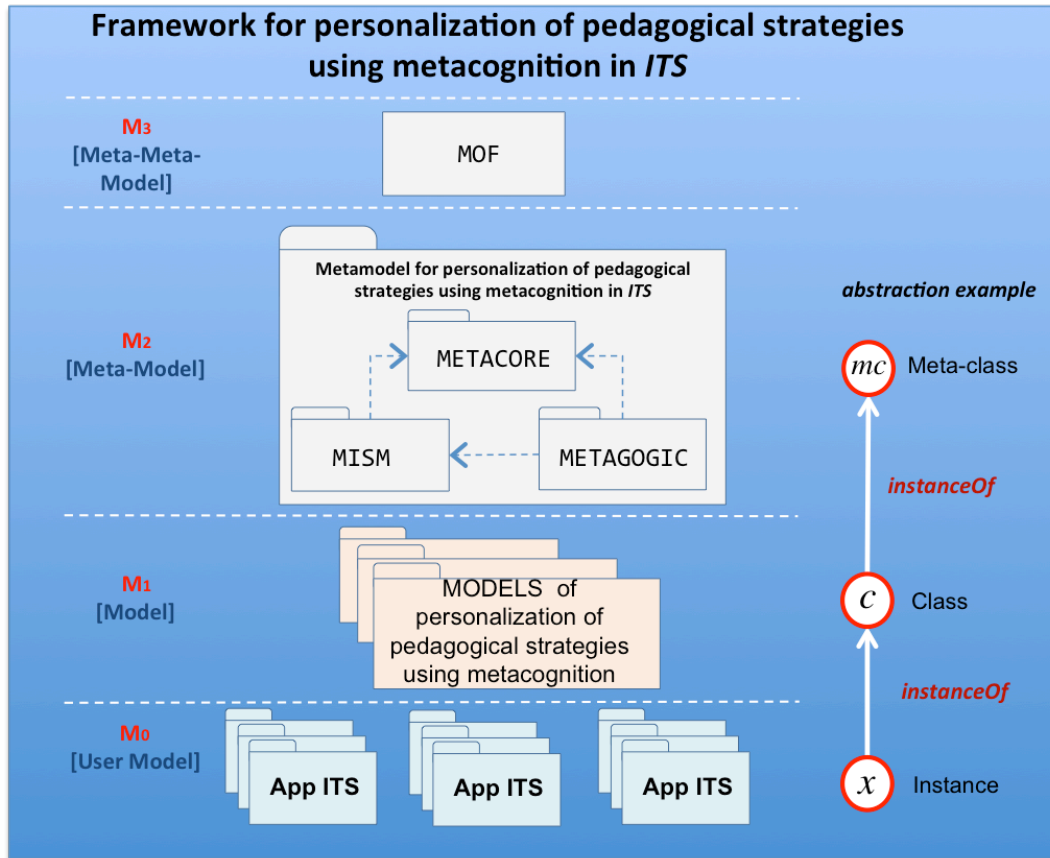


Figure 5.1. Conceptual architecture of MOF-based metamodel for personalization of pedagogical strategies using metacognition in ITS; Source: the author.

5.1.1 Elements of the conceptual architecture

As can be observed in Figure 5.1, the architecture of the metamodel is organized into four levels according to the MOF standard.

- **Meta-MetaModel Level (M_3).** This level comprises meta-metamodel (MOF 2.0) that is used for the implementation of the metamodel for personalization of pedagogical strategies using metacognition in ITS (M_2).
- **Metamodel Level (M_2).** The metamodel for personalization of pedagogical strategies using metacognition in ITS (MPPSM) is positioned at the M_2 -level in the MOF metamodeling framework. Therefore, a Model that is positioned at the M_1 -level can be modeled by the metamodel. MPPSM Metamodel is specified using MOF standard and implemented in Eclipse Modeling Framework (EMF).
- **Model Level (M_1).** This level contains the conceptual models of ITS that are implemented by designers according to the metamodel specified at M_2 level. A MPPSM-based model (M_1 level) is a Metacognitive Model for monitoring and controlling the reasoning failures in ITS.

In the MOF metamodeling framework, the derivation of a model from its metamodel is called a 'conformance.' Through the conformance process, a realization of concept in the MPPSM metamodel in a new instance (object) in the model at the M_1 level can be achieved.

- **User Model Level (M_0).** The user model at the M_0 -level is the target model that is the aim of the MPPSM Metamodel. The derived target model represents an ITS in the real-world. In MOF, the domain concept used in a metamodel is presented as a Class. The data for a Class is presented as an Object. As such, the data for the Object are in turn presented as an Instance in User Model. End-Users manipulate real data using ITS applications generated by a modeling framework from M_1 , i.e. users can create and use models of entities from real world (M_0), using the conceptual model (M_1).

5.2 M_2 - Metamodel for Personalization of Pedagogical Strategies using Metacognition in ITS (MPPSM)

The MPPSM metamodel (M_2 level) provides the conceptual support necessary to design models of personalized adaptation of pedagogical strategies integrating metamemory and self-regulation in ITS in an integrated and consistent way and also avoids the development of specific tools for the design of each new kind of metacognitive capability required.

5.2.1 General overview

The MPPSM metamodel represent the cycle of reasoning of an ITS about: (i) failures generated in its own reasoning tasks (self-regulation); and (ii) anomalies in events that occur in its Long-Term Memory (LTM) (metamemory). The Figure 5.2 shows a general overview of the metacognitive loop in MPPSM.

The reasoning cycle inputs for self-regulation are the computational data generated by the reasoning task and the output consists of recommendations, which may vary according to the reasoning task. While for metamemory, the reasoning cycle inputs are the memory events that occur in LTM and the output consists of recommendations that may vary according to the memory events.

Self-regulation in this thesis is focused on the reasoning process that allows choosing the best strategy to correct a reasoning failure and metamemory is centered on the reasoning process that allows adaptation to anomalies related to retrieving information from LTM.

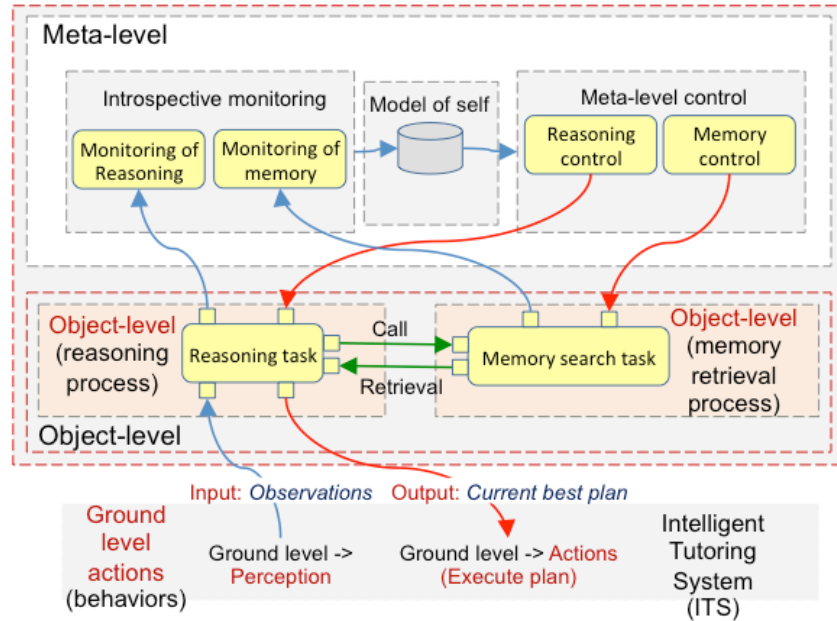


Figure 5.2. General overview of the metacognitive loop in MPPSM; Source: the author.

The MPPSM metamodel consists of the integration of MISM and METAGOGIC metamodels:

- MISM metamodel represent the meta-level and contains all the necessary elements to support metacognitive processes related to self-regulation and metamemory in an IS.

- METAGOGIC metamodel represent the object-level and contains all the necessary elements to model pedagogical strategies in an ITS.

MISM and METAGOGIC were explained in detail in previous chapters; therefore this section will be focused into aspects of design that allowed the integration of metamodels.

5.2.2 Structure and organization

The MPPSM metamodel has been designed using the Eclipse ECORE (Merks, Eliersick, & Grose, 2004; Steinberg, Budinsky, Paternostro, & Merks, 2008) and SIRIUS (International, 2003; Steinberg et al., 2008) Frameworks and it is divided into three main packages: **metacore**, **metagogic** and **mism**. A package in MPPSM is a mechanism for grouping related metamodel elements together in order to manage complexity and facilitate reuse. Figure 5.3 shows the internal organization of packages in MPPSM.

ECORE is an implementation of the standard (Essential MOF) EMOF (OMG, 2011) included in EMF (Clayberg & Rubel, 2008; Steinberg et al., 2008).

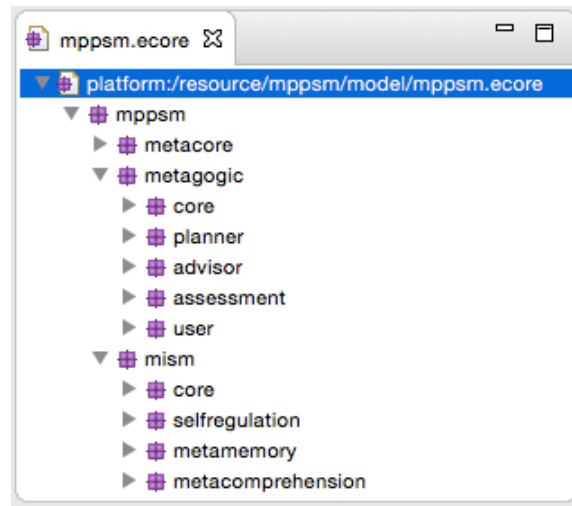


Figure 5.3. Organization of packages in MPPSM metamodel

5.2.2.1 Specification of **mppsm.metacore** package

The MISM and METAGOGIC metamodels share a common package called **metacore** but with some differences in the amount and types of concepts according to the nature of each metamodel. Table 5.1 shows the concepts included in the **metacore** package into each metamodel.

Table 5.1. List of concepts in mism.metacore and metagogic.metacore

Metacore package Integration		
<i>mism.metacore</i>	<i>metagogic.metacore</i>	<i>mppsm.metacore</i>
Action, Anomaly, BasicElement, CognitiveTask, ComputationalStrategy, Explanation, Event, FunctionalElement, Goal, Knowledge, Learning, Level, Memory, MetacognitiveTsk, MetaElement, Metalevel, ObjectLevel, Plan, Profile, ReasoningFailure, Sensor, Strategy, StructuralElement, Task, Trace	Action, BasicElement, Error, FunctionalElement, Goal, MetaElement, Plan, PlanningTask, Profile, Session, Strategy, Task, Trace, TutoringAction, TutoringTask	Action, BasicElement, CognitiveTask, Error, FunctionalElement, Goal, Level, MetacognitiveTsk, MetaElement, MetaLevel, MetareasoningTask, ObjectLevel, Plan, Profile, ReasoningTask, Strategy, StructuralElement, Task, Trace

The concepts and relationships that are common to MISM and METAGOGIC were used to create a common package allowing integration of the metamodels. The *mppsm.metacore* contains fundamental metamodel *classes* needed by the other packages. The Figure 5.4 shows the *classes* that constitute the *mppsm.metacore* package in MPPSM.

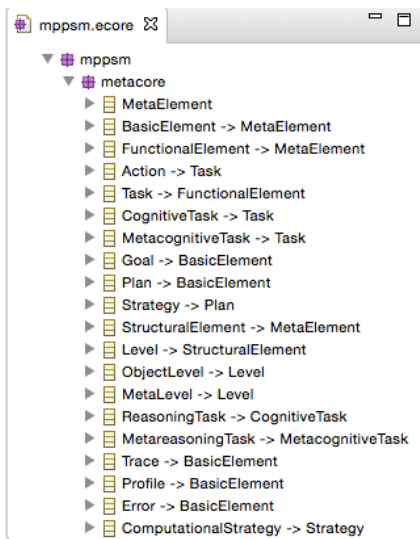


Figure 5.4. Ecore specification of metacore package in MPPSM

The elements of the `metacore` package are described in a formal way to avoid ambiguities, which could lead to design errors. The concepts that form the `metacore` are very important in the metamodel due to enable integration of the packages that comprise the structure of MPPSM metamodel.

5.2.2.2 Basic types of elements in MPPSM

In this section, the basic concepts of the formalism to describe the structural model of MPPSM are described. The MPPSM metamodel is composed of three types of elements: structural elements, functional elements and basic elements.

Definition 1. $T = \{S, F, B\}$ is the set containing the basic types of elements in MPPSM, where:

S represents the structural elements of the model.

F represents the functional elements of the model.

B represents the basic elements of the model.

Definition 2. $S = \{OL, ML\}$ is the set containing the structural elements in MPPSM; this is a system generated from MPPSM specification that is composed of two cognitive levels named `object-level (OL)` and `meta-level (ML)`.

Definition 3. $F = \{RT, MT\}$ is the set of functional elements (F), where:

`Reasoning tasks (RT)` are actions that enable the processing (transformation, reduction, elaboration, storage and retrieval) of information by applying knowledge and decision making processes in order to meet the objectives of the system.

The rule that supports this definition is shown below.

Rule 1. All reasoning tasks are object-level components.

$RT(rt): rt$ is a reasoning task

$OL(x): x$ is an object-level component

$\forall rt (RT(rt) \rightarrow OL(rt))$

`Metareasoning task (MT)` is a high level cognitive task used to monitor and to control reasoning task at object-level, also it may be used to select among cognitive “algorithms” to perform the reasoning.

The rule that support this definition is the following:

Rule 2. All meta-reasoning tasks are meta-level components.

MT (mt): mt is a meta-reasoning task
 ML (x): x is a meta-level component

$$\forall mt (MT(mt) \rightarrow ML(mt))$$

Definition 4. Basic elements (**B**) consist of the set of elements that participate and interact in the metacognitive model. $B = \{F, S, T, P, G\}$, with:

F is the error, $f \in F$ and $F = \{unexpected-result, uncompleted-task\}$. The errors are associated with violations of the expectations a system has about the performance of the cognitive process.

$S = \{s_1, ..., s_n\}$ is the set of strategies that a system has available to achieve specific or general goals, with $S \neq \emptyset$. The number of available strategies depends on the particular implementation of each system.

T is the set of traces generated by reasoning and metacognitive tasks, $t \in T$ and $T = \{Reasoning-trace, Computational-data\}$. The Reasoning-traces are elements that can store structures and rules used in the processes of reasoning. The Computational-data store data generated by the cognitive tasks.

P is the set of performance profiles used to evaluate the results of each reasoning task or strategy.

G is the set of objectives that drive a task or process. $G = \{ID, a, t, s, r\}$ is the set of components that represents the structure of a goal, where: ID is the unique identifier of the goal; a is the action to be performed, $a \in T$ and T is the set of cognitive and metacognitive tasks available for the system; t is the target of the action a ; s is the state of the goal, $s \in S$ and $S = \{starting, waiting, working, finished\}$; and r represents the final result of the goal, $r \in R$ and $R = \{satisfied, unsatisfied\}$.

Figure 5.5 shows the specification of metacore package in the MPPSM model. The resulting package is smaller and less complex in its specification regarding MISM and METAGOGIC; because specific concepts for each metamodel were not included.

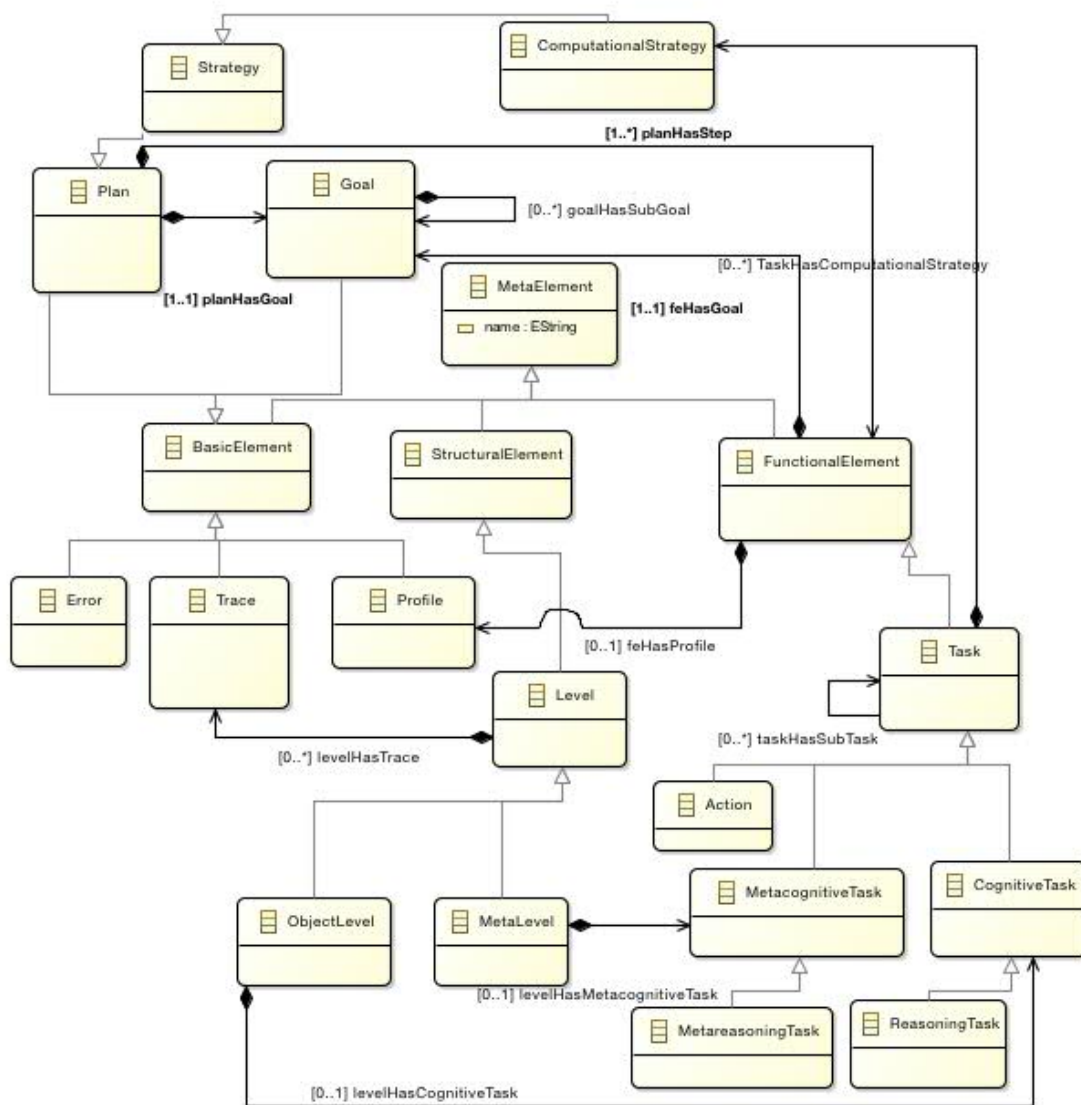


Figure 5.5. Specification of the mppsm.metagodic.core package; Source: the author.

5.2.2.3 Specification of mppsm.mism package

This package contains the necessary classes to design metacognitive capabilities for IS. The mppsm.mism package defines the specifications of metacognitive mechanisms for monitoring and controlling the following types of metacognition: self-regulation and metamemory. The mppsm.mism package is organized into three packages: core, selfregulation and metamemory.

5.2.2.3.1 The mppsm.mism.core package

The main objective of this package is to simplify the complexity level of mppsm.mism package. The mppsm.mism.core package combines the classes that are common to the

subpackages: `selfregulation` and `metamemory`. Figure 5.6 shows the Ecore specification of the package.

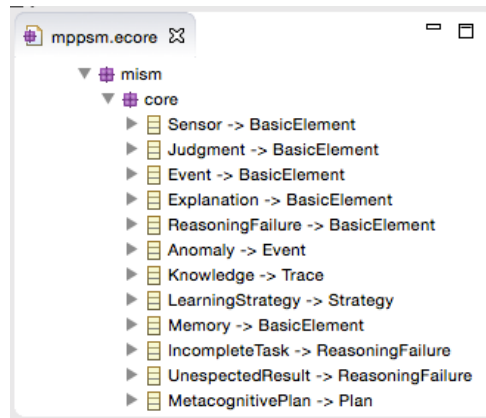


Figure 5.6. The `mppsm.mism.core` specification in Ecore

The Figure 5.7 shows the integration among the concepts of packages: `mppsm.metacore` and `mppsm.mism.core`. The integration among packages is done using generalization relationships. Concepts from `mppsm.metacore` are included in white color to enrich the diagram of the package.

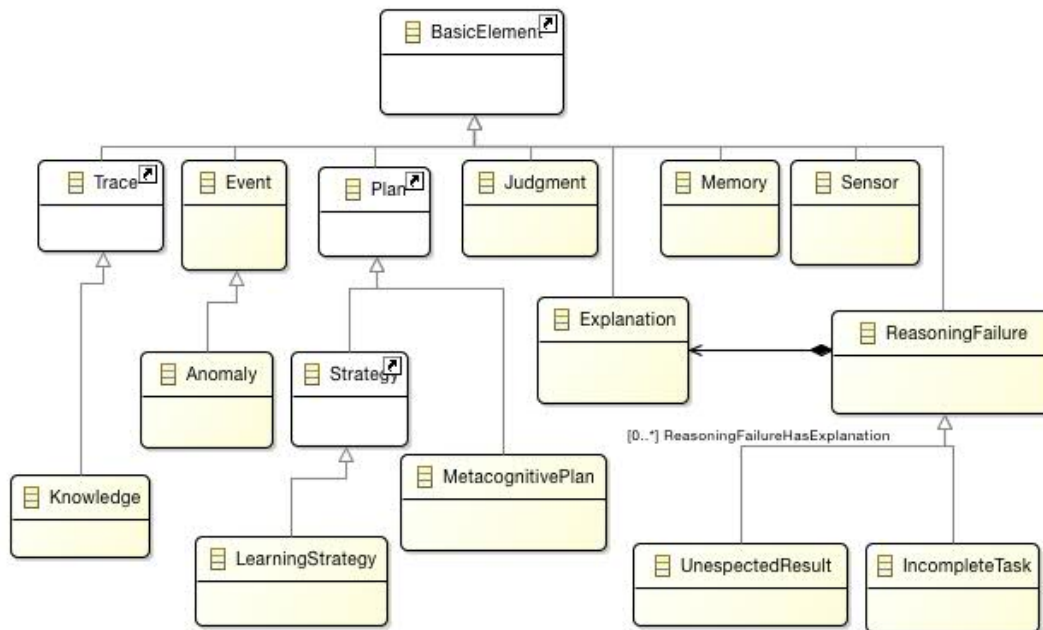


Figure 5.7. The `mppsm.mism.core` integration diagram. Classes imported from other packages in white color; Source: the author.

The main class that makes the integration between the packages `mppsm.metacore` and `mppsm.mism.core` is the class `BasicElement`. The classes `Reasoning Failure`, `Explanation` and `Sensor` inherit from the class `BasicElement` and they are used in the processes of monitoring and control of both metamemory as self-regulation packages.

5.2.2.3.2 The `mppsm.mism.selfregulation` package

The `selfregulation` package contains the specifications of self-regulation mechanisms for monitoring and controlling the reasoning processes that take place in the level-object of an intelligent system. This package has classes that enable to design models for detecting and correcting reasoning failures at object-level.

The `selfregulation` package is organized into two subpackages representing the two meta-reasoning mechanisms that have been incorporated into `mppsm` metamodel, these are: `monitoring` and `control`. Figure 5.8 shows the internal organization of the classes into de package.

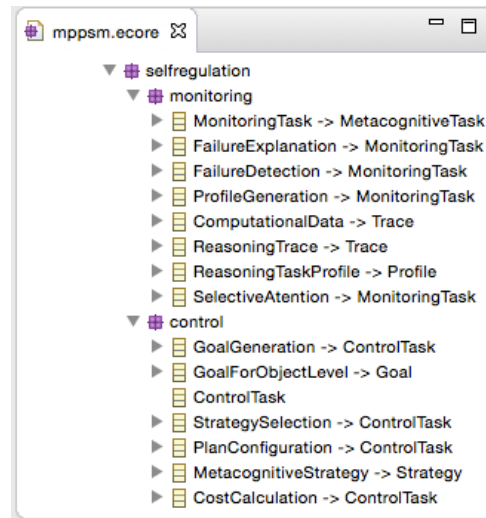


Figure 5.8. The Self-Regulation package specification in ECORE

5.2.2.3.3 The `mppsm.mism.selfregulation.monitoring` package

Introspective monitoring includes mechanisms for detecting reasoning failures at the object-level. The main purpose of monitoring is to provide enough information to make effective decisions in the meta-level control. The monitoring process is done through information feedback that is gathered at the meta-level from the object-level.

The `mppsm.mism.selfregulation.monitoring` package is integrated with packages `mppsm.metacore` and `mppsm.mism.core` see Figure 5.9.

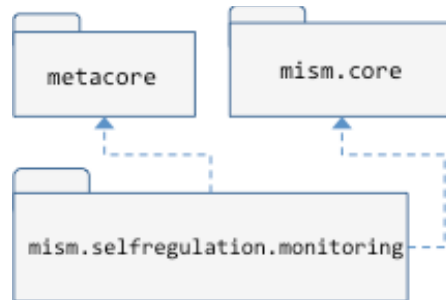


Figure 5.9. Dependency diagram of `selfregulation.monitoring` package.

The `Trace`, `Profile` and `MetacognitiveTask` classes allow the integration with the `mppsm.metacore` package. The integration with the `mppsm.mism.core` package is made by the following classes: `Explanation`, `ReasoningFailure` and `Sensor`. Major details about integration are explicit in the class diagram shown in Figure 5.10.

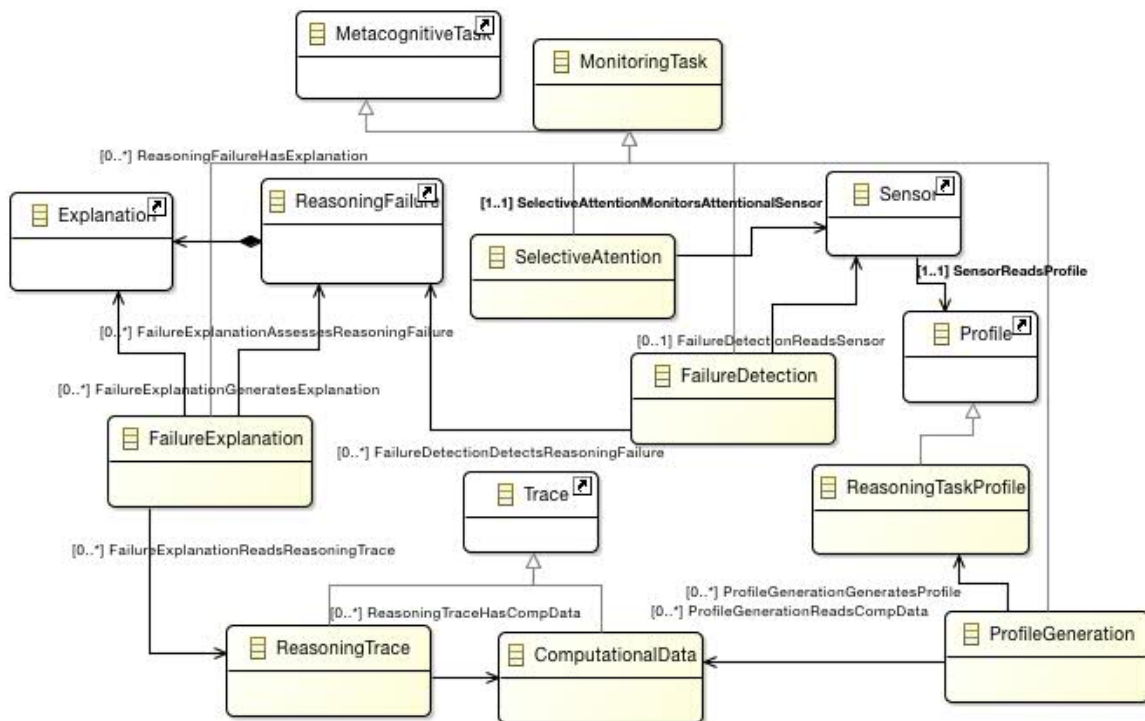


Figure 5.10. Internal structure of `mppsm.mism.selfregulation.monitoring` package. Clases imported from other packages in white color; Source: the author.

5.2.2.3.4 The mppsm.mism.selfregulation.control package

The metacognitive control aims to improve the quality of decisions about what kind of reasoning process is necessary and how much time it will take. In metacognitive control the cost of each strategy required to achieve a goal in the object-level is evaluated. The control package has classes that enable a system to decide whether has reasoned enough time to make a decision.

The `mppsm.mism.selfregulation.control` package has a dependency relationship with the `mppsm.metacore` package as figure 5.11 shows.

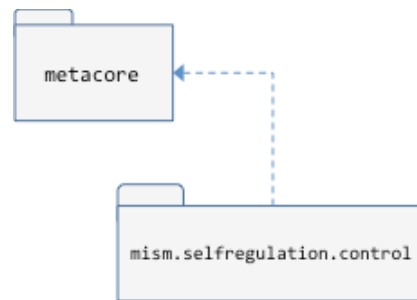


Figure 5.11. Dependency diagram of `selfregulation.control` package.

Figure 5.12 shows the class diagram of `mppsm.mism.selfregulation.control` package, the classes integrated from `mppsm.metacore` package looks in white color. The `MetacognitiveTask` class is the core of the integration between the `mppsm.mism.selfregulation.control` and `mppsm.metacore` packages. In Figure 5.12 it can be seen that 4 of the 6 classes that compose the package inherit functionalities from `ControlTask`, which is a generalization of `MetacognitiveTask`.

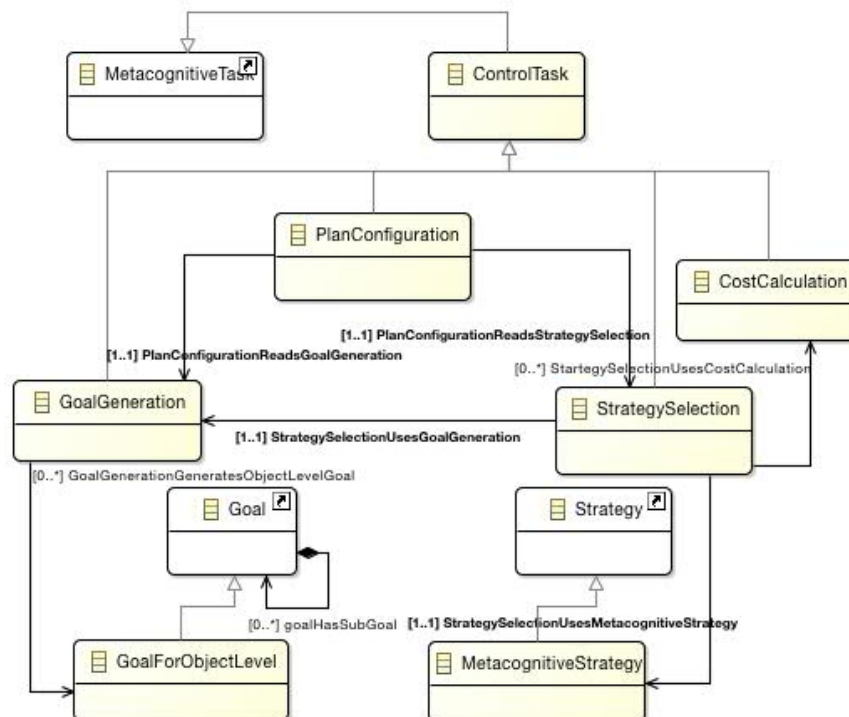


Figure 5.12. Internal structure of `selfregulation.control` package. Classes imported from other packages in white color; Source: the author.

5.2.2.4 The `mppsm.mism.metamemory` package

This package has the classes needed to design models to monitor and control events in the memory of an intelligent system. The events can be triggered by the storage or retrieval operations from memory. This kind of metacognition is important because it directly affects the learning process of a system. The `metamemory` package is organized into two subpackages representing the metamemory mechanisms that have been incorporated into `mppsm` metamodel, these are: `monitoring` and `control`. Figure 5.13 shows the internal representation of the package in Ecore.

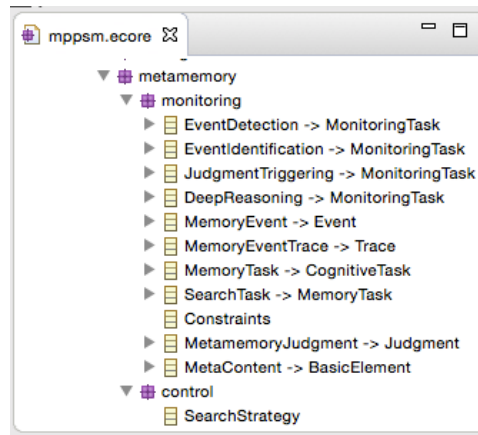


Figure 5.13. Internal structure of `selfregulation.control` package.

5.2.2.4.1 The `mppsm.mism.metamemory.monitoring` package

Monitoring package includes mechanisms for detecting events in memory (e. g. LTM or Working Memory (WM)) and performing search processes on the meta-level knowledge about the object-level. The `mppsm.mism.metamemory.monitoring` package has integration with `mppsm.metacore` and `mppsm.mism.core` packages see Figure 5.14.

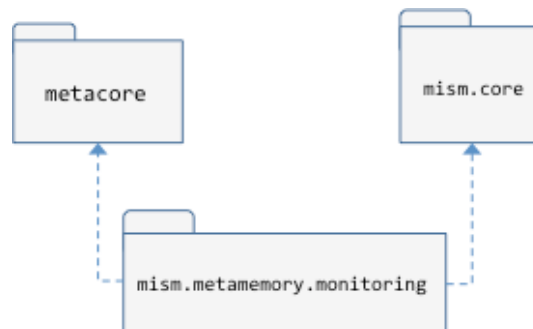


Figure 5.14. Dependency diagram of `metamemory.monitoring` package.

Integration with the `mppsm.metacore` package is done using generalization relationships from classes: `BasicElement`, `CognitiveTask` and `Trace`. On the other hand, the classes `MonitoringTask`, `Judgment`, `Event` allow the integration with `mppsm.mism.core` package. The `MonitoringTask` class is the most important within the package because it contains the features that are common to all monitoring functions of memory. The designers according to characteristics of each system define these monitoring functions. Figure 5.15 shows the class diagram of `mppsm.mism.metamemory.monitoring` package in MPPSM.

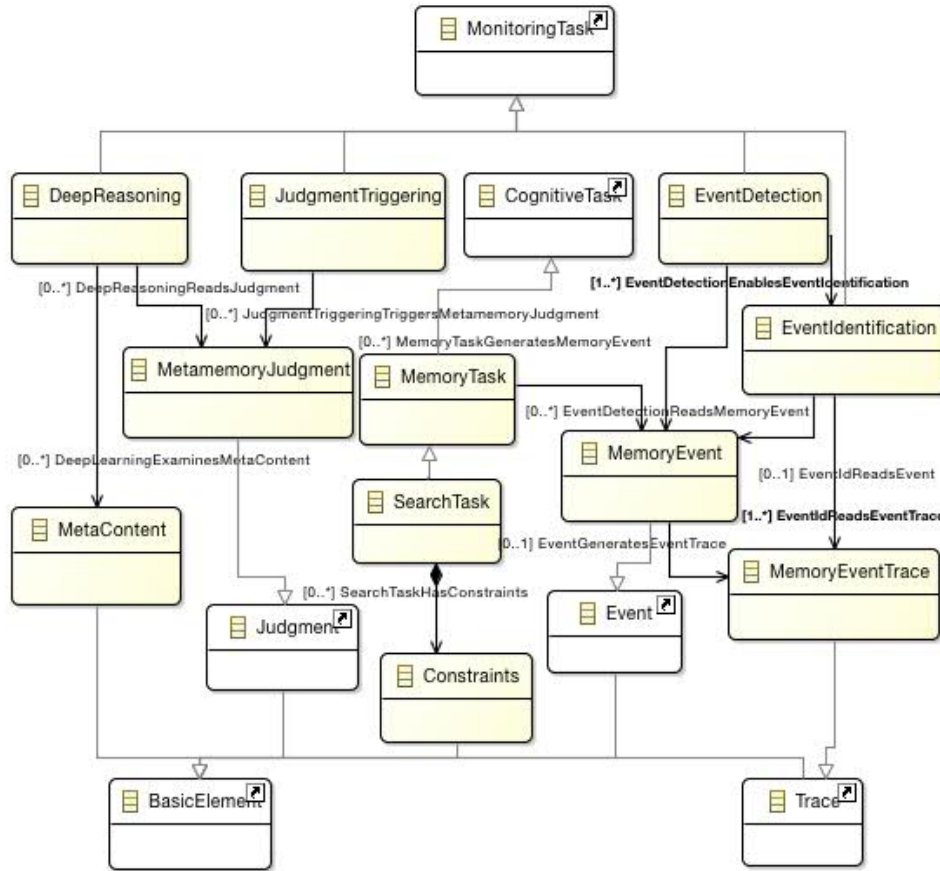


Figure 5.15. Internal structure of `metamemory.monitoring` package. Classes imported from other packages in white color; Source: the author.

5.2.2.4.2 The `mppsm.mism.metamemory.control` package

The meta-level control contains a schema with information about search strategies available at the object-level. A major meta-level control function is to recommend the most appropriate search strategy for the constraints of information retrieval from memory.

The `mppsm.mism.metamemory.control` package is integrated with `mppsm.metacore` and `mppsm.mism.metamemory.monitoring` packages. Figure 5.16 shows dependencies of `mppsm.mism.metamemory.control` package.

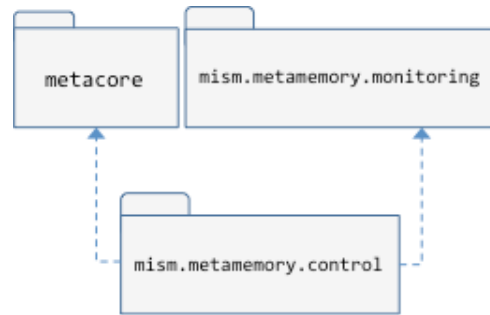


Figure 5.16. Dependency diagram of `metamemory.control` package.

The Integration with the `mppsm.metacore` packages is done through `Strategy` class, while the integration with `mppsm.mism.metamemory.monitoring` is done by a reference from `SearchTask` class to `SearchStrategy` class. The Figure 5.17 shows the class diagram of `mppsm.mism.metamemory.control` package in MPPSM.

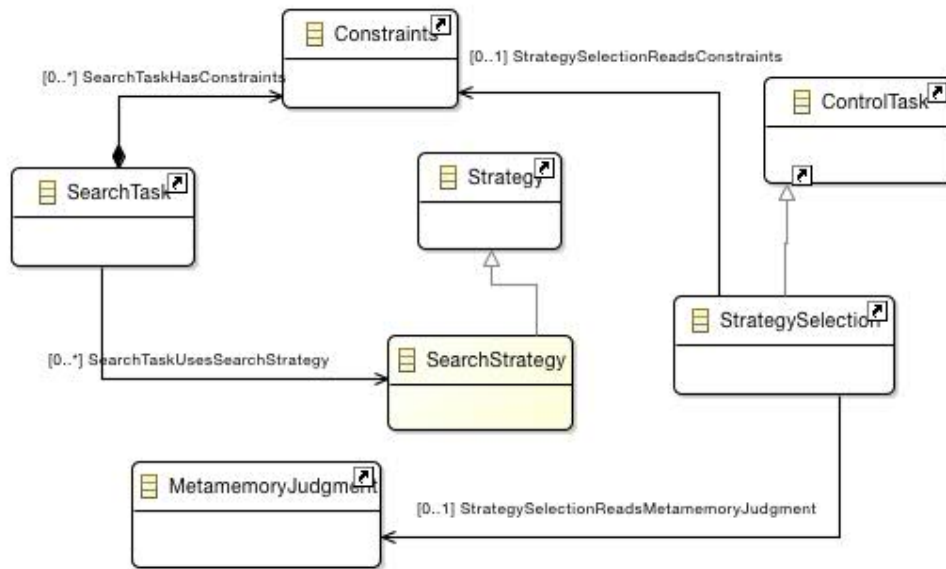


Figure 5.17. Internal structure of `metamemory.control` package. Classes imported from other packages in white color; Source: the author.

5.2.2.5 Specification of the interface between the meta-level and object-level in MPPSM

The meta-level keeps an updated model of the object-level called “Self-model”. The Self-model allows the meta-level to have awareness about reasoning processes that are conducted at the object-level. The main element that composes the object-level is the `ReasoningTask` class. The *reasoning tasks* generate *computational data* and a *reasoning trace*. Both *computational data* and *reasoning trace* are inherited from the `Trace` class. The *computational data* are numerical values produced during the execution of some *reasoning task*. This data type can contain both the output generated by the task as well as the partial

data from computational processing. The *reasoning tasks* have available a set of strategies to achieve its goals.

ReasoningTask and Strategy have profiles at meta-level. The profiles are constantly updated and are used by the meta-level to make decisions related to the performance of object-level. Figure 5.18 shows the organization of the classes conforming the self-model of the object-level in MPPSM.

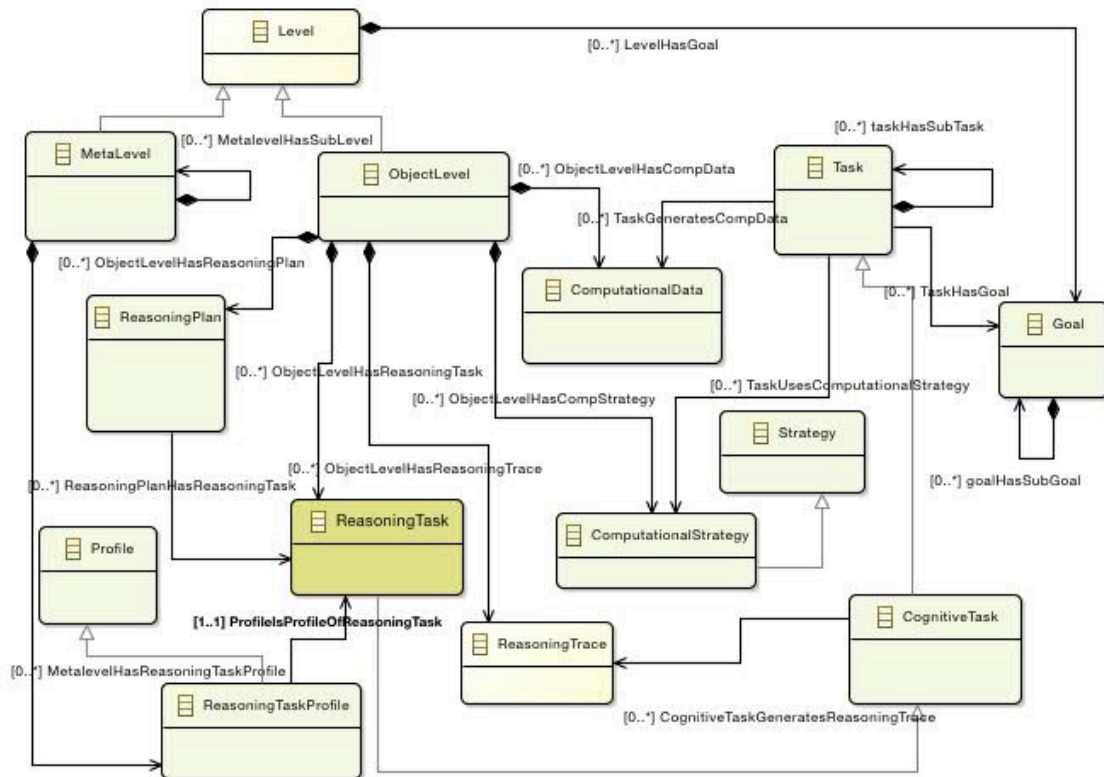


Figure 5.18. Self-model specification in MPPSM; Source: the author.

Profile Generation class performs the interface between the meta-level and the object-level. ProfileGeneration reads the ComputationalData produced in the object-level and generates an updated profile for reasoning tasks. In the meta-level, the Sensor class monitors each of the profiles of the reasoning tasks looking for anomalies in their performance.

The FailureDetection class reads a Sensor in search of discrepancies between *observations* and *expectations*. When a discrepancy is found, then the FailureDetection class generates a description of the ReasoningFailure.

FailureExplanation generates an Explanation of the cause of the ReasoningFailure having as inputs the assessment of the failure and reading the

ReasoningTrace. GoalGeneration produces new goals based on the Explanation for solving the failure detected.

5.2.2.6 Specification of `mppsm.metagogic` package

The `mppsm.metagogic` package is organized into five packages: `core`, `planner`, `advisor`, `assessment` and `user`.

5.2.2.6.1 The `mppsm.metagogic.core` package

This package is designed to simplify the complexity of `metagogic` package. The `mppsm.metagogic.core` package contains the common concepts from packages: `planner`, `advisor`, `assessment` and `user`.

The Figure 5.19 shows the classes that constitute the `mppsm.metacore` package in MPPSM metamodel.

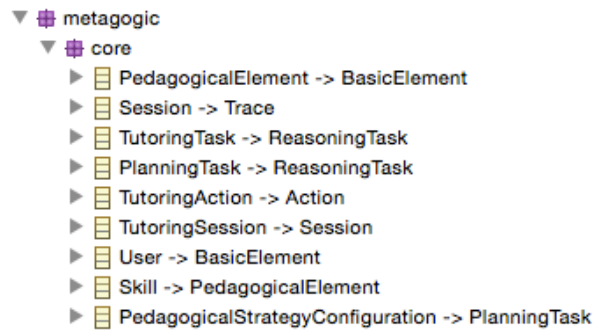


Figure 5.19. Specification of the `mppsm.metagogic.core` package in MPPSM

The Figure 5.20 clearly shows the integration between the concepts of packages: `mppsm.metacore` and `mppsm.metagogic.core`. The Integration between packages is done through the use of the generalization relationships. The concepts belonging to `mppsm.metacore` are included in white color to enrich the diagram of the package.

Following some aspects related to the integration of concepts into the package are listed base on Figure 5.20. The planning tasks and tutoring tasks inherit the attributes and functionalities from `ReasoningTask` class. This means that these tasks can be monitored and controlled by the meta-level. `User`, `Skill` and `PedagogicalElement` are of type `BasicElement`. In this case `PedagogicalElement` is the root of the elements used to design the pedagogical model of an ITS in MPPSM.

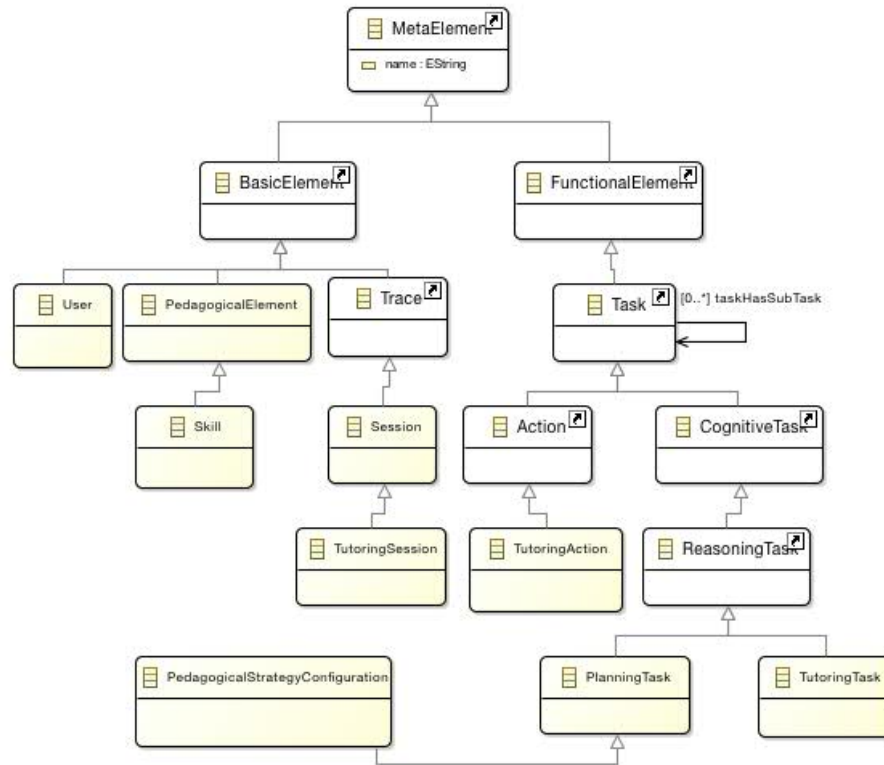


Figure 5.20. Specification of the `mppsm.metagoric.core` package in MPPSM. Classes imported from other packages in white color; Source: the author.

5.2.2.6.2 The `mppsm.metagoric.planner` package

The Figure 5.21 shows the classes that constitute the `mppsm.metagoric.planner` package in MPPSM.

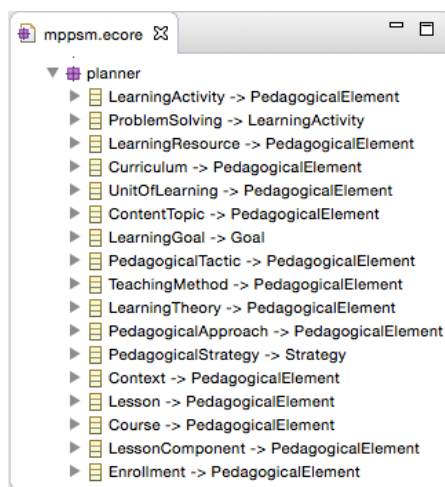


Figure 5.21. Specification of the `mppsm.metagoric.planner` package in MPPSM

This package allows the adaptation of pedagogical strategies in ITS including: (i) selection of educational resources according to the characteristics of a student; and (ii) managing pedagogical knowledge by using classes like `LearningTheory`, `PedagogicalApproach` and `PedagogicalStrategy`.

The `mppsm.metagodic.planner` package is integrated with `mppsm.metagodic.core` and `mppsm.metacore` packages. Dependencies between packages are shown in Figure 5.22.

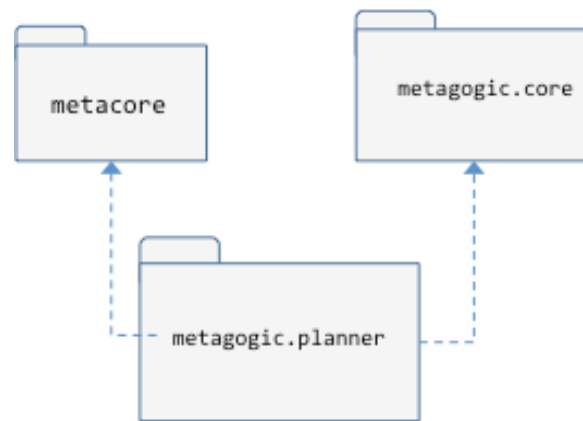


Figure 5.22. Dependency diagram of `mppsm.metagodic.planner` package

The `PedagogicalElement` class is the backbone of the integration between the `mppsm.metagodic.core` and `mppsm.metacore` packages. In Figures 5.21 and 5.23 can be observed that 15 of the 17 classes that compose the package `mppsm.metagodic.planner` inherit functionality from `PedagogicalElement` class.

The integration with the `mppsm.metagodic.core` package is done through classes `Goal` and `Strategy`. Class `LearningGoal` inherits features from class `Goal` and class `PedagogicalStrategy` inherits functions of `Strategy` class. The `PedagogicalStrategy` class is the most important within the package because it represents the pedagogical strategy that is personalized according to each student. The Figure 5.23 shows the class diagram of the `mppsm.metagodic.planner` package in MPPSM.

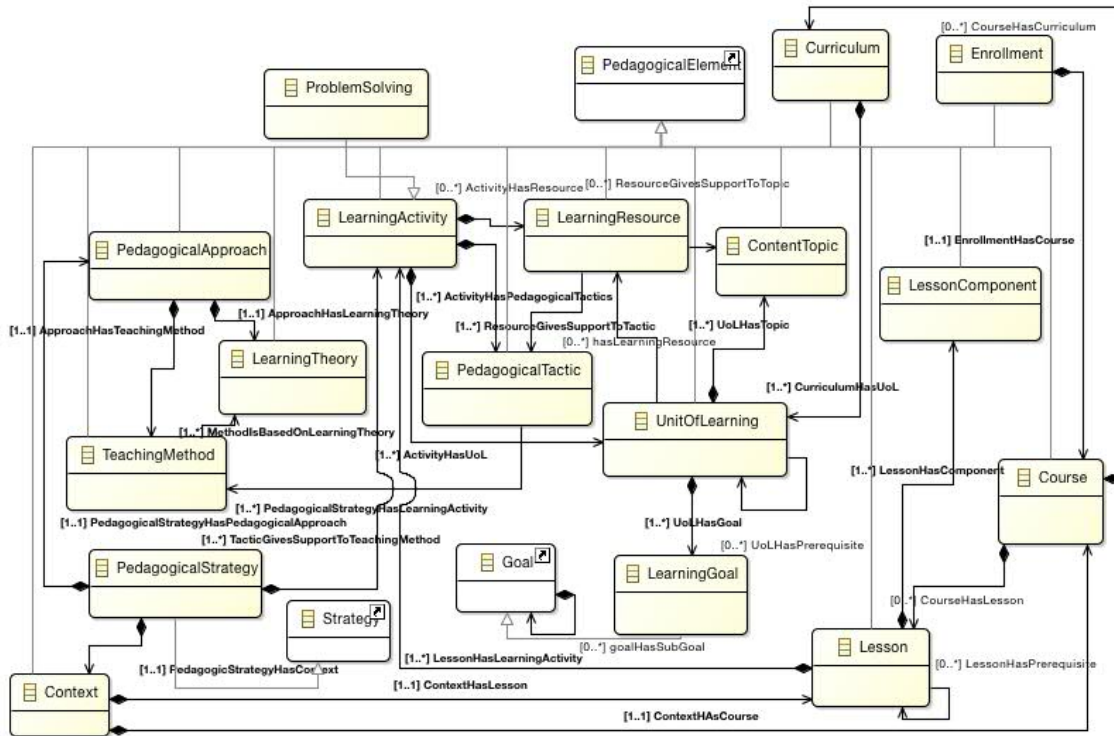


Figure 5.23. Planner package model. Classes imported from other packages in white color; Source: the author.

5.2.2.6.3 The `mppsm.metagolic.advisor` package

The Figure 5.24 shows the classes that constitute the `mppsm.metagolic.advisor` package in MPPSM.

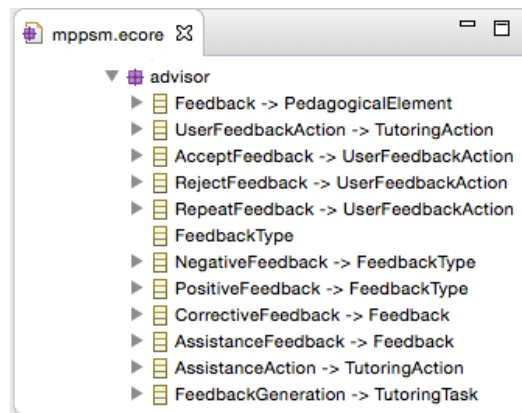


Figure 5.24. Advisor package specification in ECORE

The `advisor` package contains classes that allows ITS to manage the pedagogical assistance to students. This package enables: (i) adapting feedback in learning activities; and (ii) assisting the student in a timely manner in the event of a problem in developing learning sessions is detected.

The `mppsm.metagogic.advisor` package is integrated with the `mppsm.metagogic.core` package. Figure 5.25 shows the dependency diagram between those two package in MPPSM.

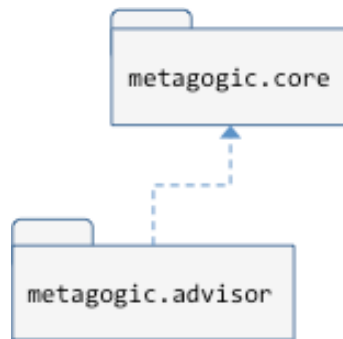


Figure 5.25. Dependency diagram of `mppsm.metagogic.assessment` package

The Figure 5.26 shows the class diagram of `mppsm.metagogic.advisor` package in MPPSM.

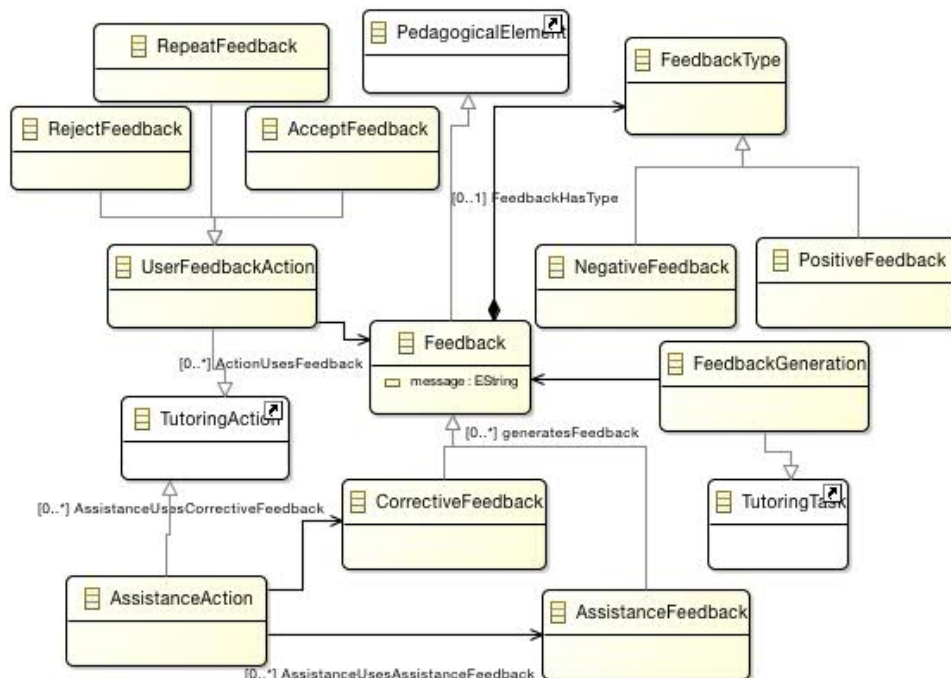


Figure 5.26. Class diagram of `metagogic.advisor` package. Classes imported from other packages in white color; Source: the author.

The `PedagogicalElement`, `TutoringAction` and `TutoringTask` classes facilitate integration with the `mppsm.metagogic.core` package. The `Feedback` class inherits functionalities from `PedagogicalElement` and is the central hub of the package. The `UserFeedbackAction` and `AssistanceAction` classes are of `TutoringAction` type; finally `FeedbackGeneration` class is a `TutoringTask`.

5.2.2.6.4 The `mppsm.metagogic.assessment` package

This package allows a system to monitor and evaluate the academic performance of students. The main functions that support this package are: adaptation of the evaluation tests of the lesson; and monitoring and assessment of student performance. The Figure 5.27 shows the classes that constitute the `mppsm.metagogic.assessment` package in MPPSM.

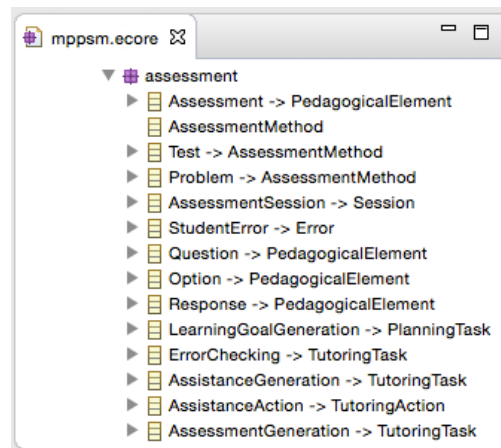


Figure 5.27. Assessment package specification in Ecore

The `mppsm.metagogic.assessment` package is integrated with `mppsm.metacore` and `mppsm.metagogic.core` packages, see Figure 5.28.

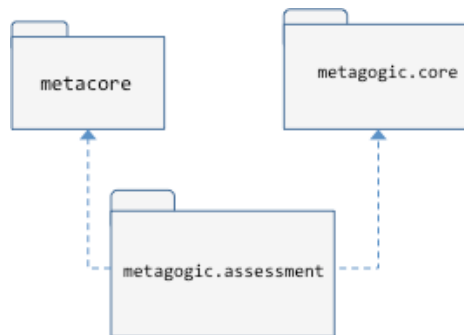


Figure 5.28. Dependency diagram of `mppsm.metagogic.assessment` package

The integration with the `mppsm.metacore` package is done by `Error` class and the integration with `mppsm.metagogic.core` package is made using the classes: `PedagogicalElement`, `PlanningTask`, `TutoringTask`, `TutoringAction` and

Session. The Figure 5.29 shows the class diagram of the `mppsm.metagogic.assessment` package in MPPSM.

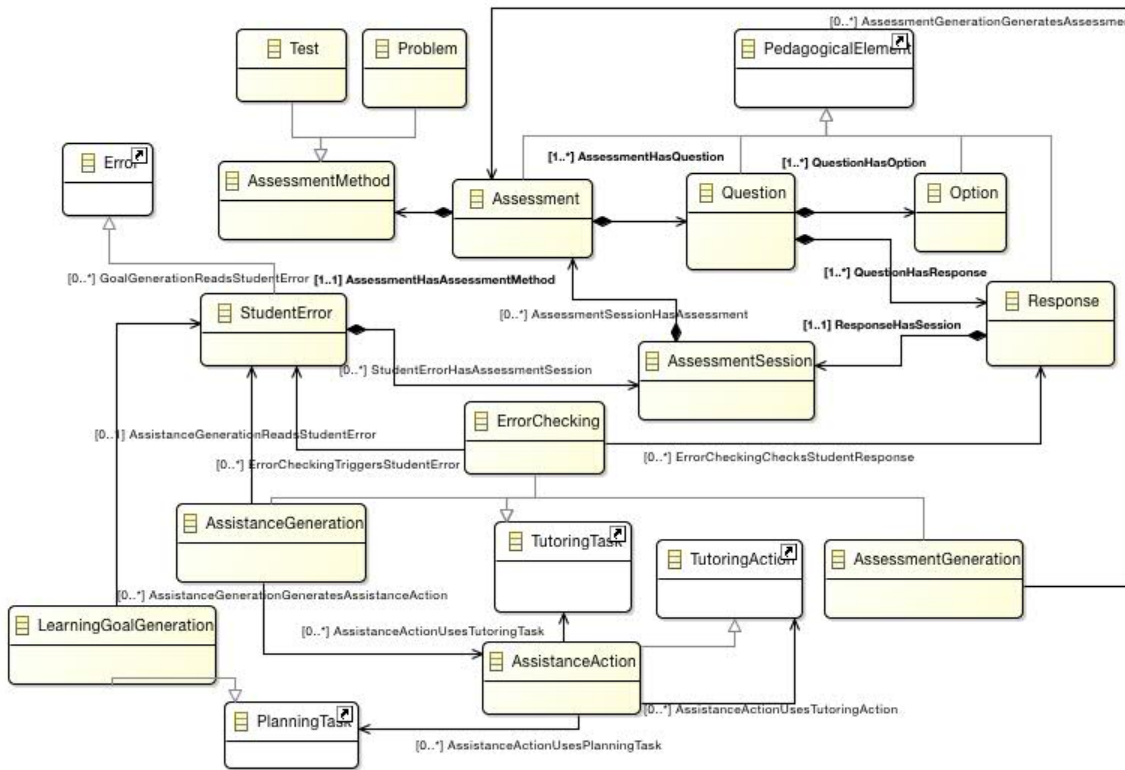


Figure 5.29. Assessment package model. Classes imported from other packages in white color; Source: the author.

5.2.2.6.5 The `mppsm.metagogic.user` package

This package contains the necessary classes for managing system users and the learning sessions of each student. The Figure 5.30 shows the classes that constitute the `mppsm.metagogic.user` package in MPPSM.

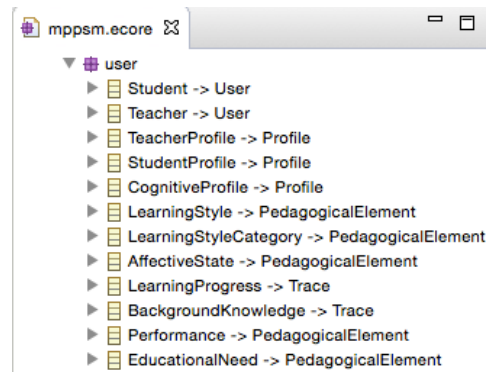


Figure 5.30. User package specification in Ecore

Figure 5.31 shows the integrations of `mppsm.metagogic.user` package with other packages of MPPSM.

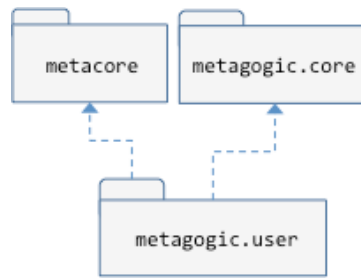


Figure 5.31. Dependency diagram of `mppsm.metagogic.user` package

The Figure 5.32 shows the class diagram of the `mppsm.metagogic.user` package in MPPSM. Profile and Trace classes do integration with the `mppsm.metacore` package. The `mppsm.metagogic.user` package contains three types of profiles: `StudentProfile`, `TeacherProfile` and `CognitiveProfile`. The `LearningProgress` and `BackgroundKnowledge` are Trace-type classes.

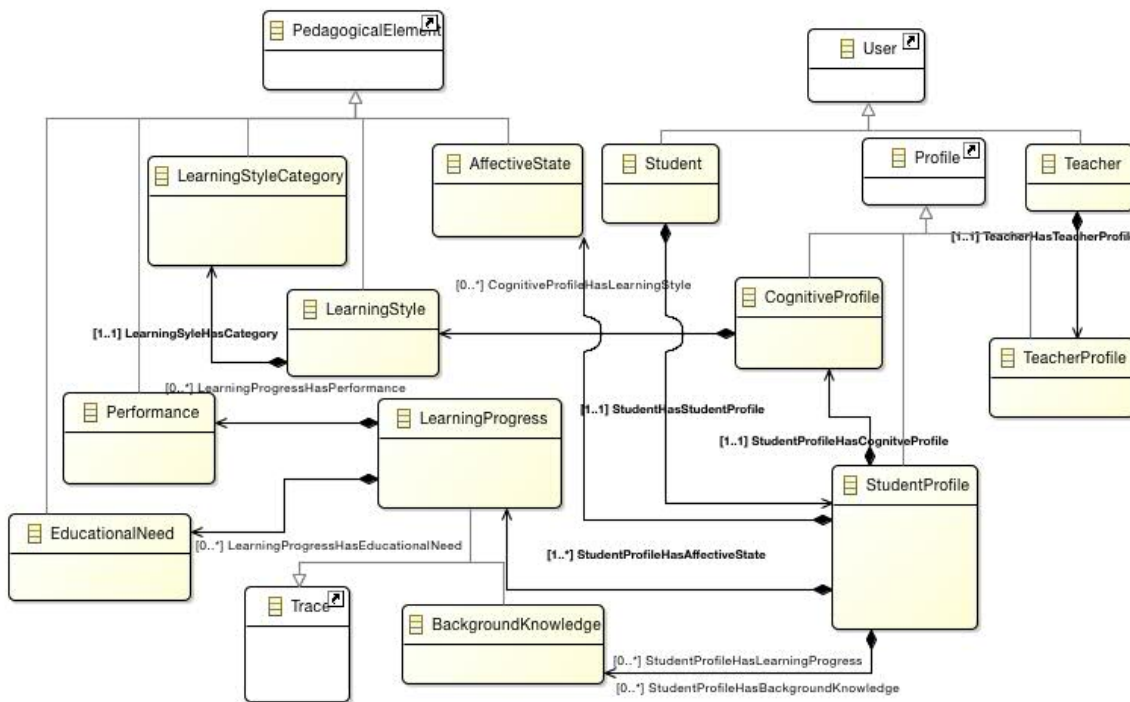


Figure 5.32. User package model. Classes imported from other packages in white color;
Source: the author.

The `PedagogicalElement` and `User` classes make the integration between the `mppsm.metagogic.user` package and the `mppsm.metagogic.core` package. The class `PedagogicalElement` plays an important role in the integration and the reduction on the complexity of the package because it is the super type of `LearningStyle`,

LearningStyleCategory, AffectiveState, Performance and EducationalNeed classes. Moreover, the User class is used to define user types in the system: Student and Teacher.

5.2.3 Semantic definitions for elements in MPPSM

A structural MOF metamodel cannot capture all types of domain-specific constraints, which are relevant for describing a target domain, in this case pedagogical domain and metacognitive domain. Thus, additional constraints are defined by using Object Constraint Language (OCL) (OMG, 2014). OCL was selected for semantic definition because it is easy to write and understand, allowing complex queries over models at a high level of abstraction (Shidqie & Gollmann, 2007). The constraints can identify whether a model of pedagogical strategies (M_0 layer) is legal or illegal, preserving the consistency of models generated from MPPSM.

In this work, OCL invariants are used to define the semantics by encoding MPPSM specific constraints. For the sake of readability, this section only shows some examples of OCL constraints.

Constraint 1. Each ReasoningTask has two attributes called start_time and finish_time; in each instance of ReasoningTask, finish_time has to be greater than start_time:

```
[1] context ReasoningTask
[2]     inv correctTime: self.finish_time > self.start_time
[3]     inv noEmptyGoal: self.ReasoningTaskHasGoal->size()>0
```

Constraint 2. Each ReasoningTask has an attribute of type collection of subtasks called TaskHasSubTask; when an instance of ReasoningTask is created then TaskHasSubTask is initialized with an empty set:

```
1. context ReasoningTask::TaskHasSubTask:Task
2.     init: self.TaskHasSubTask={}
```

Constraint 3. The is_focused and failure_indicator attribute of a Sensor must be initialized with False value:

```
[1] context Sensor::is_focused:boolean
[2]     init: self.is_focused = False

[3] context Sensor::failure_indicator:boolean
[4]     init: self.failure_indicator = False
```

Constraint 4. Each User defines an attribute called name, which is composed of the concatenation of the first_name and the last_name:

```
[1] context User
```

```
[2]     def: name: String = self.first_name.concat('
    ').concat(last_name)
```

Constraint 5. Each FailureDetection has an attribute called `failure_detected` stating with False value:

```
[1] context FailureDetection::failure_detected:boolean
[2]     init: self.failure_detected = False
```

Constraint 6. In a FailureDetection, the `generateFailure` method is executed if a difference between the perception and expectation of a Sensor is found. The `failure_detected` attribute receives the True value after the execution of the `generateFailure` method:

```
[1] context FailureDetection::generateFailure()
[2]     pre:
    self.readsSensor.perception<>self.readsSensor.expectation
[3]     post: self.failure_detected=true
```

Constraint 7. Each Task defines an attribute called `completionTime` calculating the difference between the `finish_time` and `start_time` attributes:

```
[1] context Task
[2]     def: completionTime:Real = self.finish_time -
    self.start_time
```

Constraint 8. Each Profile must be associated with at least one ReasoningTask.

```
[1] context Profile
[2]     inv validProfile: self.isProfileOfReasoningTask->size()>0
```

Constraint 9. Each FailureSolutionPlan has two attributes called `start_time` and `finish_time`; in each instance of FailureSolutionPlan, `finish_time` has to be greater than `start_time`.

```
[1] context FailureSolutionPlan::completionTime:Real
[2]     inv correctTime: self.end_time > self.start_time
```

5.2.4 Mapping Approach for MPPSM

Mapping is the specification of a mechanism for transforming the elements of a model conforming to a particular metamodel into elements of another model that conforms to another (possibly the same) metamodel (*Erche, Wagner, & Hein, 2007; OMG, 2005*). A *mapping* implicitly or explicitly defines a relationship between a source and a target model element (*Jouault & Kurtev, 2006*) and it describes the rules used for the transformations. The *mapping* is used to realize transformation of instances of the mapped models.

The MPPSM metamodel has specifications of endogenous and exogenous mapping.

5.2.4.1 Endogenous mapping

Endogenous mapping in this work consists of a series of rules that allow the generation of models of pedagogical strategies (M_1 layer) based on the MPPSM specifications (M_2 layer) in an automated way. MPPSM uses instantiation semantics based on a one-to-one `instanceOf` relation to map: (i) M_2 elements to M_1 elements; and (ii) M_1 elements to M_0 elements. In this case endogenous mapping is used for the creation of a model in M_1 layer in which each model element of M_1 corresponds to one metamodel element of M_2 layer. Figure 5.33 contains an example of endogenous mapping in MPPSM.

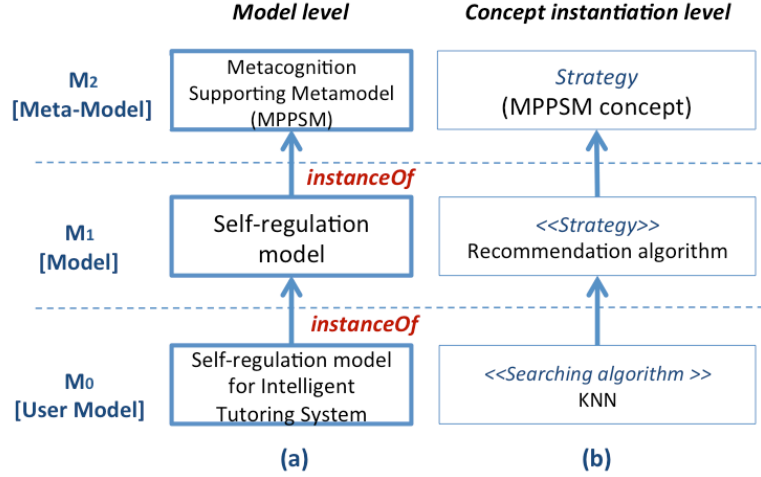


Figure 5.33. Endogenous transformation representation in MPPSM

The list of the translation operations is given in a generic language with operations including the MOF *Reflective* interface.

```
[1] ForAll view  $v_i$  in {ViewSet = View.ref_all_objects (false)} do
[2] domain = ref_create_instance ("Domain",  $v_i$ .name, ...)
[3] M2.ref_add_value("containedConcepts", domain)
[4] ForAll class $_i$  in { ClassSet =  $v_i$ .ref_value ("containedClasses")}
do
[5] concept = ref_create_instance ("Concept" , class.name, ... )
[6] domain.ref_add_value("containedConcepts", concept)
[7] ForAll prop $_i$  in {CollProperties = class $_i$ .ref_value("attribute")}
do
[8] feature = ref_create_instance ("Property ", prop.name,...)
[9] concept.ref_add_value("feature ", feature)
```

The *Reflective* interfaces of MOF allow: *create*, *update*, *access*, *navigate* and *invoke* operations on M_1 -level *Instance objects*. For example in line [5] a *concept* artifact in M_1 layer is created as an instance of (`instanceOf`) a class from M_2 layer with similar name using the MOF *Reflective* interfaces `ref_create_instance`.

5.2.4.2 Exogenous mapping

The exogenous mapping system that has been integrated in this work consists of a series of transformation from MPPSM to a Relational Database Schema (RDBS). The transformations to database schemas were selected because databases are a component widely used in the design of ITS. An ITS stores the domain and pedagogical knowledge in a database.

Exogenous transformations are implemented with a horizontal mapping pattern. Horizontal mapping establish *one-to-one* relations between elements from the source model (MPPSM) to elements of the target model (RDBS).

Exogenous transformations facilitate the design of MPPSM-based systems because the designers could generate the database schema in an automated way. In MPPSM, the horizontal transformations are supported in the language QVT-Relations (Query/View/Transformation) (OMG, 2011; Rensink & Nederpel, 2008). Figure 5.34 shows the QVT-based transformation model implemented for MPPSM.

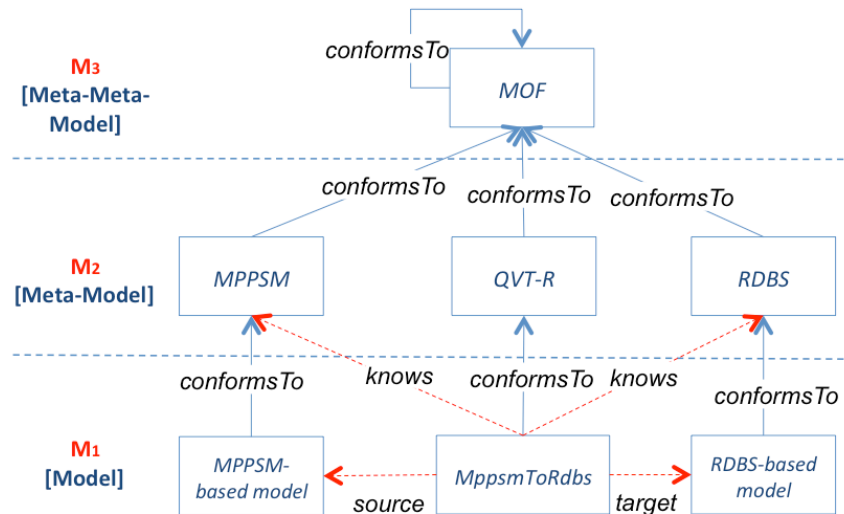


Figure 5.34. Exogenous transformation model in MPPSM, based on (Bezivin et al., 2006)

A transformation between MPPSM and RDBS is specified as a set of relations that must hold for the transformation to be successful. Following the specification of a transformation called `MppsmToRdbs` is shown.

```
[1] transformation MppsmToRdbs (mppsm : MPPSM, rdbs : RDBS) {
[2]   top relation PackageToSchema {...}
[3]   top relation MetaElementToTable {...}
[4]   relation AttributeToColumn {...}
[5] }
```

The transformation `MppsmToRdbms` allows the generation of a RDBS-based model from a model based on MPPSM. The script of `MppsmToRdbms` is based upon the official QVT specification in (OMG, 2011).

The transformation `MppsmToRdbms` is unidirectional in direction to RDBS and maps Packages to Schemas, MetaElements to Tables and Attributes to Columns using the relations `PackageToSchema`, `MetaElementToTable` and `AttributeToColumn`. Executing the transformation in check only mode checks consistency of the RDBS generated models; the transformation returns `True` if the RDBS model is consistent according to the transformation and `False` otherwise, for example see (Line 5 in `PackageToSchema` script). The same transformation is used in enforce mode to attempt to modify one model in order to enforce the consistency of RDBS generated model, see (Line 6).

The `PackageToSchema` relation realizes the transformation of each Package in MPPSM to a Schema of RDBS. The consistency is checked in lines 5 and 6 of the following script.

```
[1] -- map each package to a schema
[2] top relation PackageToSchema
[3] {
[4]     package_name : String;
[5]     checkonly domain mppsm p : Package { name = package_name
[6] };
[7]     enforce domain rdbms s : Schema { name = package_name };
[8] }
```

The `MetaElementToTable` relation maps the transformation of each `MetaElement` in MPPSM to a Table of RDBS. For each `MetaElement` found in source model, a Table with the name of the `MetaElement` is created in target model. Then, in an automated way a primary key is created by using the name of the table and a prefix, see (Line 19).

The `when` clause in Line 25 specifies that the `MetaElementToTable` relation holds only when the `PackageToSchema` relation is maintained between the package containing the `MetaElement` and schema that contains the Table. At line 30, the `-where-` clause specifies the condition that `MetaElementToPkey` and `AttributeToColumn` must satisfy for all model elements that participate in the relationship.

```
[1] -- map each MetaElement to a table
[2] top relation MetaElementToTable
[3] {
[4]     cn : String;
[5]     prefix : String;
[6]
[7]     checkonly domain mppsm m : MetaElement
[8]     {
[9]         _package = p : Package { },
```

```

[10]         name = mn
[11]     };
[12]
[13]     enforce domain rdbms t : Table
[14]     {
[15]         schema = s : Schema {},
[16]         name = mn,
[17]         columns = cl : Column
[18]         {
[19]             name = mn + '_id',
[20]             type = 'NUMBER'
[21]         },
[22]         primaryKeys = k : PrimaryKey {columns = cl :
Column{} }
[23]     };
[24]
[25]     when
[26]     {
[27]         PackageToSchema(p, s);
[28]     }
[29]
[30]     where
[31]     {
[32]         MetaElementToPkey(c, k);
[33]         prefix = mn;
[34]         AttributeToColumn(c, t, prefix);
[35]     }
[36] }

```

The `AttributeToColumn` relation maps the transformation of each `Attribute` of a `MetaElement` in MPPSM to a `Column` of a `Table` in RDBS. Two other relations that mapped attributes are also described in the next script: `MetaElementToPkey` and `SuperAttributeToColumn`.

`MetaElementToPkey` relation allows generating the primary key of a table in the target model. The `SuperAttributeToColumn` relation maps attributes inherited by the `MetaElements` into table columns at the target model.

```

[1]  relation MetaElementToPkey
[2]  {
[3]      cn : String;
[4]      checkonly domain mppsm m : MetaElement {name = mn};
[5]      enforce domain rdbms k : PrimaryKey {name = mn + '_pk'};
[6]  }
[7]
[8]  relation AttributeToColumn
[9]  {

```



```

[10]         checkonly domain mppsm m : MetaElement { };
[11]         enforce      domain rdbms t : Table { };
[12]         primitive domain prefix      : String;
[13]
[14]         where
[15]         {
[16]             SuperAttributeToColumn(m, t, prefix);
[17]         }
[18]     }
[19]
[20]     relation SuperAttributeToColumn
[21]     {
[22]         checkonly domain mppsm m : MetaElement
[23]         {
[24]             general = sm : MetaElement {}
[25]         };
[26]
[27]         enforce domain rdbms t : Table {};
[28]
[29]         primitive domain prefix : String;
[30]
[31]         where
[32]         {
[33]             AttributeToColumn(sm, t, prefix);
[34]         }
[35]     }

```

5.3 Concrete Syntax for the design of metacognitive functions in ITS

Metamodeling has the objective to specify the implementation of a modeling language. In this case a concrete syntax was defined in order to make the MPPSM metamodel more usable. The concrete syntax is composed of a graphic notation called M++. M++ is a Domain-Specific Visual Language (DSVL) for modeling metacognition in an ITS and incorporates two meta-reasoning mechanisms, these are: introspective monitoring and meta-level control. Figure 5.35 includes a list of the elements of the M++ notation.

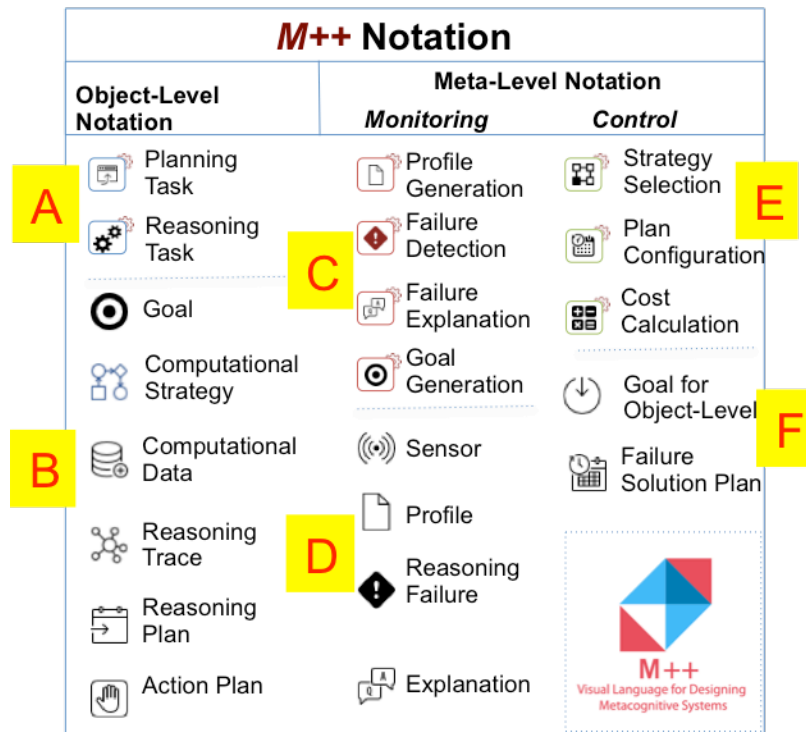


Figure 5.35. Main elements in M++ notation; (Caro, Josyula, Jiménez, Kennedy, & Cox, 2015)

In M++ the abstract syntax is specified with MPPSM metamodel and the concrete syntax is expressed by some mapping of the abstract syntax elements to visual constructs e.g. icons. The main artifacts of M++ are models specified in a visual manner.

The icons were designed bearing in mind their usability when applied by users. The Figure 5.35 in section (A) shows the icons used to represent object-level tasks and section (B) displays icons representing elements that interact with the tasks at object-level. Section (C) contains the notation related to the tasks of monitoring introspective act from the meta-level on the object-level. Section (D) displays icons representing elements that interact with the monitoring tasks. The Figure 5.35 in section (E) displays icons representing the metacognitive control tasks and section (F) displays icons representing elements that interact with the tasks of metacognitive control.

In summary of this section may show that M++ has approximately 20 notation elements for modeling metacognitive systems.

5.3.1 MetaThink tool

The MetaThink tool has been developed with the aim of supporting the modeling of metacognitive functions in ITS commented in previous sections. M++ allows the generation of metacognitive diagrams in a visual editor named MetaThink. MetaThink graphical user

interface is comprised of the following components: *title bar*, *property bar*, *tool bar* and *workspace*. Figure 5.36 shows element distributions in the GUI.

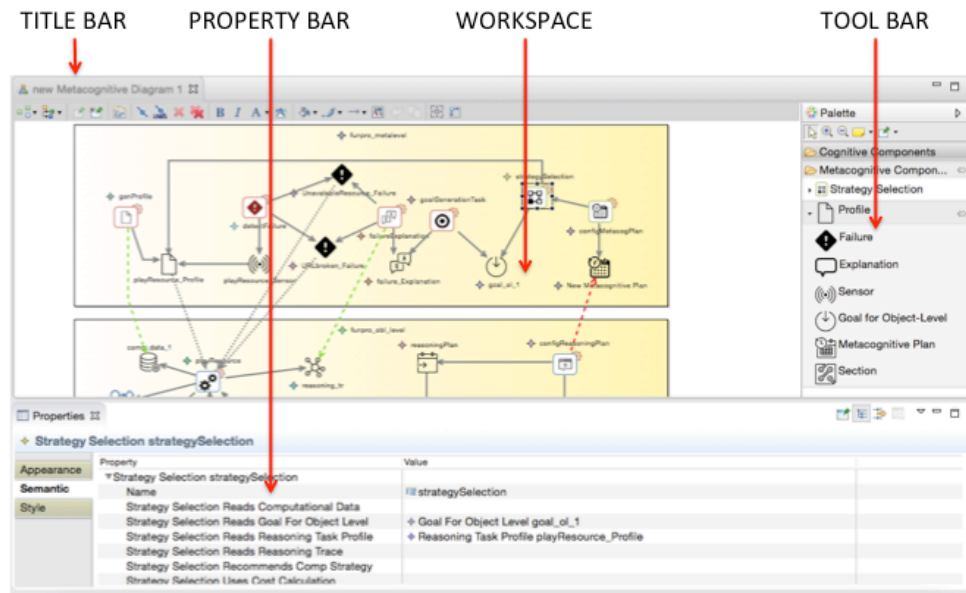


Figure 5.36. Plugin-MetaThink graphical user interface; Source: the author.

MetaThink provides the fundamental infrastructure and components for the generation of metacognitive diagrams in a visual editor based on *MPPSM metamodel*. MetaThink has been developed using the plugins in the Eclipse Modeling Project (B. Moore, Dean, & Gerber, 2004; Steinberg et al., 2008). Specifically, MetaThink tool has been implemented as an Eclipse plug-in (Clayberg & Rubel, 2008) using SIRIUS and ECORE Frameworks.

5.3.1.1 Title bar

On this bar both the application name and the name of the current working file are displayed.

5.3.1.2 Property bar

In this bar the user can view and edit the properties of the selected metacognitive elements on the workspace. The properties vary depending on the selected element; however the following properties are common to all metacognitive elements: *type*, *label*, *width*, *height*, *x* and *y*.

5.3.1.3 Toolbar

Metacognitive elements are organized in four categories in the toolbar: *Cognitive level*, *Meta-level*, *Object-level* and *Associations*.

Figure 5.37 contains a screenshot of the toolbar of MetaThink tool.

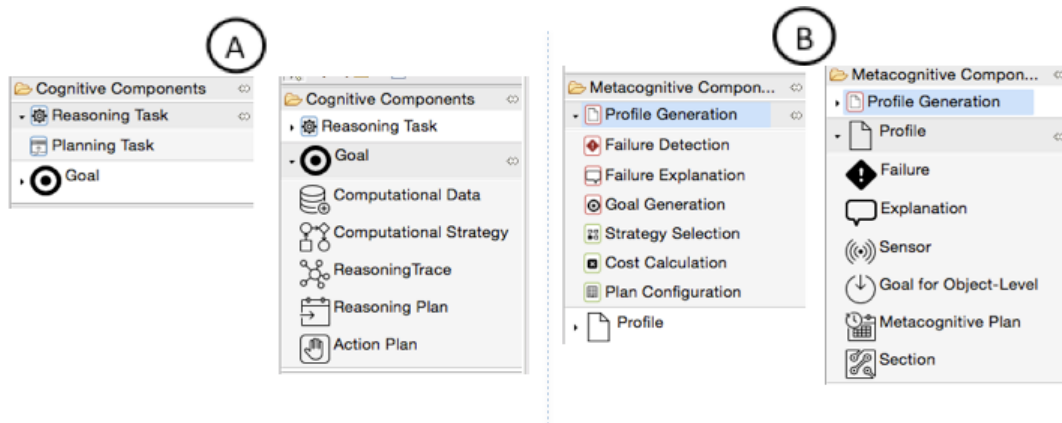


Figure 5.37. MetaThink toolbar

5.4 Example of use: design of a metacognitive model based on MPPSM using M++

A metacognition model (M_1) for an ITS was generated from the MPPSM metamodel; see Figure 5.38. The generated model was used to develop an ITS called FUNPRO (FUNdamentos de PROgramación). FUNPRO is a ITS for teaching *Introduction to Programming in Engineering* and was developed using MODESEC (Caro, Toscazo, Hernández, & David, 2009) methodology.

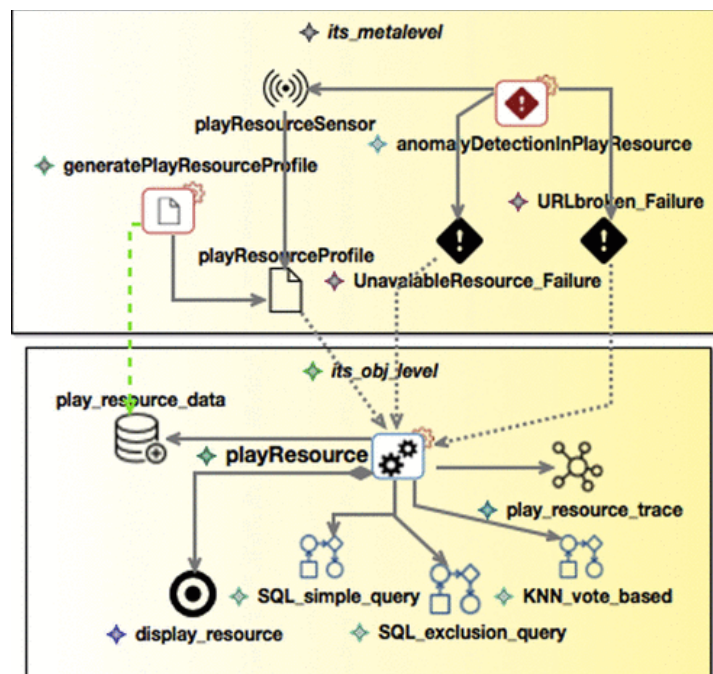



Figure 5.38. Example of a metacognitive model genetrated for FUNPRO - ITS; Source: the author.

FUNPRO will be described in detail in Chapter 6, but in this section a system function is used as example for the description of a metacognitive model using the notation of M++. FUNPRO has a function called `playResource` that is responsible for retrieving the URL of learning resources from the knowledge base and deploying them in the lesson. Figure 5.38 shows a metacognitive model for the `playResource` function of the ITS-FUNPRO.

The `playResource` function is represented with the  icon, which means it is a reasoning task.

The `playResource` function has implemented three types of computational strategies (see  icons): (i) matching simple query, the search query in a simple Structured Query Language (SQL) type; (ii) exclusive search is similar to (i), but excludes some results and; (iii) vote-based search, this strategy is based on the nearest neighbor algorithm



The meta-level intervenes in the `playResource` function in the following cases: (i) *Unavailable resource* ( icons). If a resource for some reason cannot be deployed in the lesson, e.g. resource has the URL broken; (ii) *Unexpected result*. It is given when a recommended resource has received a poor evaluation; (iii) if the student obtains a low performance in the lesson. The green lines represent the flow of information of introspective monitoring; allowing the meta-level keep updated with respect to the object-level state ( icons). The red lines represent the metacognitive control.

Figure 5.39 shows the objects instantiated (M_0) from metacognitive model (M_1) represented in Figure 5.38. The objects are clearly organized into the object-level and the meta-level according to specifications of MPPSM.

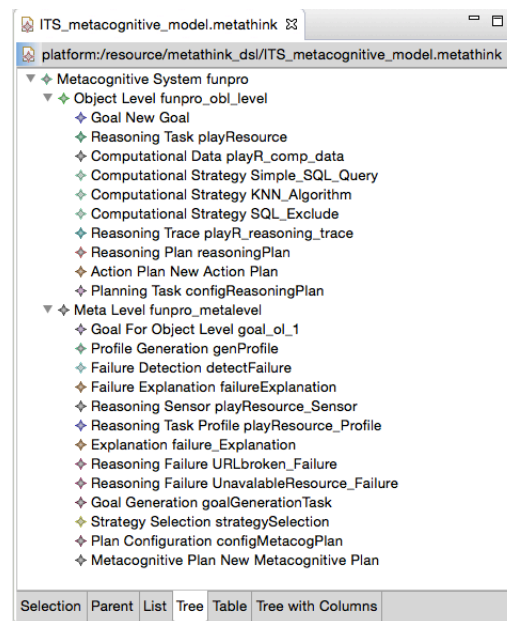


Figure 5.39. Example of a metacognitive model genetrated for an ITS corresponding with Figure 5.38

5.5 Validation

M++ validation was performed on three dimensions: *potential usefulness*, *usability*, *logic of generated models*. For the validation process, the following methods were used: (i) *Empirical study based on user perception*. In empirical study the user perception with regard to the quality of the M++ notation was measured; (ii) *Tracing*. The instantiation of different types of specific concepts in the metamodel are traced (followed) through the model generation process to determine if the model's logic is correct (Sargent, 2005).

5.5.1 Empirical validation of M++

A practical experiment was conducted in order to verify the potential *usefulness* and *usability* of M++ for modeling metacognition in IS. The experimental study was developed based on design parameters of software engineering experiments described in the works of (Molina et al., 2013; Wohlin et al., 2000) and (Sjøberg et al., 2005).

5.5.2 Configuration of the experiment

The goal of the experiment was to evaluate the notation of M++ with regard to the *ease of use*, *usefulness* and *intention to use* of the models in the context of the design of metacognition in IS.

The experiment was conducted with the followings two research questions: (i) RQ#1: "Is M++ perceived as easy to use and useful for modeling metacognition in IS?"; and (ii) RQ#2: "Is there an intention to use M++ in the future for modeling metacognition in IS?".

The experiment was conducted with 28 students enrolled in Educational Informatics Program of Universidad de Córdoba - Colombia.

A second experiment was conducted by way replica to contrast the results obtained in the first experiment. In this case involving 12 professionals who voluntarily participated in the experiment.

In the two experiments the user perception with regard to the quality of the notation was measured.

The variables used to measuring the user perception with regard to the quality of the notation are based on (Abrahão, Insfran, Carsí, & Genero, 2011; Wohlin et al., 2000): (i) *Perceived ease of use*. This variable represents a perceptual judgment of the effort required to use M++; (ii) *Perceived usefulness*. This variable expresses the degree to which a person believes that the use of M++ will achieve its intended objectives regarding the design of metacognition in IS; and (iii) *Intention to use*. The intention to use is defined as the extent to which a person intends to use M++ in the future for designing metacognition in IS.

To complete the profile of the participants, they were asked about their knowledge of other notations that could be used in the design of intelligent systems. In particular, participants were asked about (Unified Modeling Language) UML and use of ontologies.

With respect to UML only 6 participants reported having very low practical/theoretical knowledge, but the rest had some previous knowledge about this notation. Regarding the use of ontologies 11 participants reported having very little knowledge about using it; see Table 5.2. But 100% of participants did not have any previous experience or knowledge on modeling metacognition using the M++ notation.

Table 5.2. Knowledge about UML and ontology notation

Areas	Undergraduate students		Professionals	
	Mean	Std dev	Mean	Std dev
Modeling of software systems using UML notation	2,57	1,20	3,92	1,00
Modeling of software systems using ONTOLOGY notation	2,25	1,35	3,83	1,34

5.5.3 Data analysis

Initially, participants were asked about their preferences regarding the use of textual or graphical representations for specifying software systems. 78.57% of the students preferred the graphical notations, compared to 21.43% that preferred textual notations. In the case of professionals 75.00% of the subjects preferred graphical representations, compared to 25.00% that preferred textual specifications. Regarding this aspect, the percentages are very similar.

The variable '*perceived ease of use*' was measured by opinion of the participants about how easy or difficult they found the modeling of metacognition in intelligent systems using M++. The subjects rated the '*perceived ease of use*' on a scale of 1 (very easy to use) to 5 (very difficult to use) according to their perceived ease of use of M++ in the realization of the modeling exercises. Table 5.3 shows the mean of the scores assigned by participants (students and professionals) to M++.

Table 5.3. Perception of usability

Graphical specification	Undergraduate students		Professionals	
	Mean	Std dev	Mean	Std dev
Usability of M++ for modeling Metacognitive diagrams.	2,46	1,45	2,08	1,31

The participants were asked about how they could describe their perception of the M++ notation as a whole. 78.57% of the students considered the M++ notation a complete one. In the case of professionals the percentage was very similar (75.00%).

Also 78.57% of the students and 91.67% of professionals assessed the homogeneity of the notation positively.

Regarding the usefulness of the notation of M++, the 78.57% of students considered useful the notation as compared to 21.43% who did not consider it, see Figure 5.40. The percentage of professionals who considers useful the notation was 83.33%. This result is consistent with the response data of this group of respondents in relation to their overall perception of the use of conceptual models.

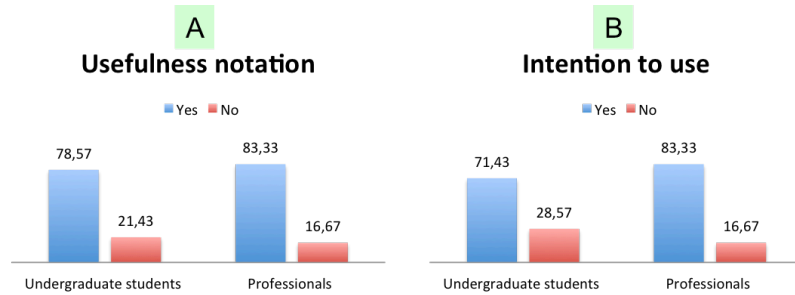


Figure 5.40. Result of the variables: (A) Usefulness notation and (B) Intention to use in the empirical study.

Regarding the intention to use M++ to design intelligent systems with metacognitive components. 78,57% of students expressed the intention to use M++ in the future, compared with 21,43% who responded negatively. For professionals 83,33% indicated their intention to use M++. Again, the answer to this issue is consistent with the subjective perception of this group on utility of conceptual models.

5.5.4 Model validation

In this type of validation, the behavior of different types of specific entities in the model is traced (*followed*) through the model to determine if the logic of the model is correct and if the necessary accuracy is obtained (*Sargent, 2005*). In this section the validation of models generated from MPPSM is described using as reference the FUNPRO system.

The description of MPPSM artifacts shows a situation of how a possible model of metacognition is generated in M_1 from the metamodel at M_2 . The model generation process is followed by the instantiation of a model for application in real life (M_0) from the model layer M_1 . Figure 5.41 in Section A shows the partial view of the MPPSM metamodel (Layer M_2 in MOF) which metacognitive models used in the validation are generated.

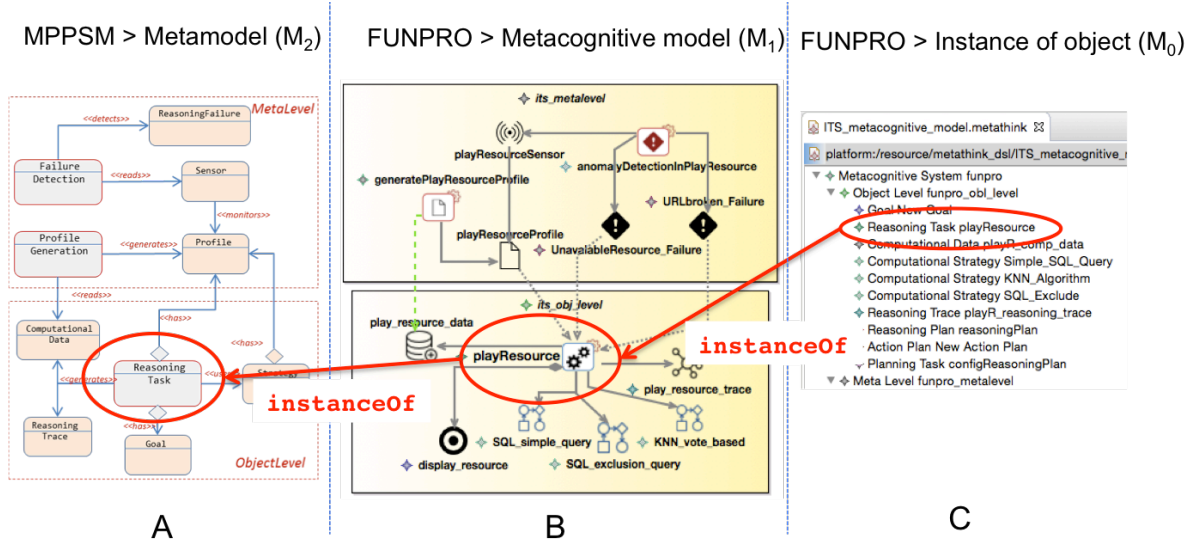


Figure 5.41. (A) Section of MPPSM (Layer M_2 in MOF) with object-level specification a partial view of introspective monitoring process at meta-level; (B) Metacognitive model at M_1 conforms to partial view of MPPSM in section A; (C) User model conforms with the metacognitive model in section B.

Below the basic rules used to verify traceability of the models is presented. The traceability *rule (1)* checks instantiations of artifacts between different layers of the MPPSM metamodel.

$$\begin{aligned}
 M_0(x) &: x \text{ is an instance in } M_0 \text{ layer} \\
 M_1(c) &: c \text{ is a class in } M_1 \text{ layer} \\
 M_2(mc) &: mc \text{ is a meta-class in } M_2 \text{ layer} \\
 In(x, y) &: x \text{ is a model artifact instantiated from } y \\
 In2(x, y) &: x \text{ is a model artifact with instantiation trace from } y \\
 \forall x, c, mc & In(x, c) \wedge In(c, mc) \\
 \Rightarrow In2(x, mc)
 \end{aligned} \tag{1}$$

A partial mapping of example in Figure 5.38 is listed in Table 5.4. The traceability between the artifacts was checked using *rule (1)*.

Table 5.4. ITS-FUNPRO mapping table

MPSSM concept (M_2)	Artifact in FUNPRO Metacognitive model (M_1)
ReasoningTask	<i>playResource</i>
Strategy	<i>SQL_simple_query</i> ; <i>SQL_exclusion_query</i> ; <i>KNN_vote_based</i>
Goal	<i>display_resource</i>
ComputationalData	<i>recommendation_trace</i>

ReasoningTrace	<i>reasoning_trace</i>
Profile	<i>play_resource; simple_query; exclusion_query; votation_query</i>
Sensor	<i>play_resource_sensor</i>
ReasoningFailure	<i>Error_display_resource</i>
FailureDetection	<i>isErrorDisplay</i>
ProfileGeneration	<i>update_profile</i>

Model in Figure 5.39 and the mapping table (Table 5.4) show that the metacognitive model (M2) is consistent with the MPPSM metamodel. The results described in the validation show that the metacognitive models in M++ generated from MPPSM using MetaThink are reliable because they have consistency and are based on an international standard (MOF).

5.6 Conclusion of the chapter

In this chapter a MOF-based metamodel for the generation of personalized adaptation models of pedagogical strategies integrating metamemory and self-regulation in ITS was described. The metamodel is called MPPSM and is located in the M_2 layer of the MOF standard. An implementation of E-MOF called ECORE was used to build MPPSM in the Eclipse Modeling Framework.

MPPSM contains 123 classes organized into three main packages called `mppsm.metacore`, `mppsm.mism` and `mppsm.metagogic`. The `mppsm.mism` package contains the functionality of the meta-level and abstract description of the object-level into a meta-reasoning loop of an intelligent system. The `mppsm.metagogic` package contains the schema of the object-level domain in an ITS.

The `mppsm.metacore`, `mppsm.mism.core` and `mppsm.metagogic.core` packages have dual functionality: (i) allow reducing the complexity of MPPSM because group common classes that are used by other packages; (ii) maintain the integration and reusing classes among the different packages that compose MPPSM.

In MPPSM, OCL invariants are used to define the semantics by encoding MPPSM specific constraints. The MPPSM metamodel has specifications of endogenous and exogenous mapping.

Endogenous mapping in MPPSM consists of a series of rules that allow the generation of models of pedagogical strategies at M_1 layer based on the specifications of M_2 layer in an automated way. MPPSM has an exogenous transformation model implemented with a horizontal mapping pattern. Horizontal mapping establish *one-to-one* relations between elements from the source model (MPPSM) to elements of the target model (RDBS).

Exogenous transformations facilitate the design of MPPSM-based systems because allows to designers the generation of database schema in an automated way.

A DSVL called M++ with a central core based on MPPSM was created. M++ has approximately 20 tools for modeling metacognitive systems supporting introspective monitoring and meta-level control.

Two types of validations were performed to validate M++ notation and the consistency of the generated models using M++. Validation of M++ notation was made by an experiment and the validation of the consistency of the generated models was performed using the technique of tracing.

The results given in the experimental study demonstrate positive perceptions of the proposed DSVL and provide preliminary information concerning the quality of the concrete syntax of M++.

It can be conclude from the results that M++ is a language that has a useful notation to help designers in the process of modeling metacognitive components in intelligent systems.

Tracing validation shown that the concepts of the metamodel are actually usable by designers of intelligent systems with metacognitive support.

6 Intelligent Tutoring System for teaching Introduction to Programming - FUNPRO

FUNPRO (*FUNDamentos de PROgramación*) is a prototype of ITS, which aims to provide personalized instruction in the subject of *Introduction to Programming* (Caro, Josyula, & Jiménez, 2015). FUNPRO was designed based on MPPSM metamodel and it was developed entirely in SWI-Prolog.

The general architecture of FUNPRO is based on two layers called object-level and meta-level, as it is shown in Figure 6.1. The object-level and the meta-level are designed according to MPPSM metamodel. The object-level has architecture consistent with the `mppsm.metagologic` package, while the meta-level is designed with based on the `mppsm.mism` package.

Figure 6.1 shows a double reasoning loop in FUNPRO. The first reasoning loop occurs between the students at ground level and the system at object-level, in this case the ground level represents the environment (e.g. Student's behavior interacting with FUNPRO). The system receives information from the environment (e.g. reasoning about student information) then the information is processed and a pedagogical strategy is generated according to student's profile.

The second reasoning loop is between the object-level and the meta-level. The meta-level receives information related to the process of reasoning at object-level then this information is processed and a recommendation is generated. The recommendation from meta-level to object-level may be: (i) to act and stop the reasoning process or; (ii) to do further reasoning; (e.g. object-level reasoning about the student).

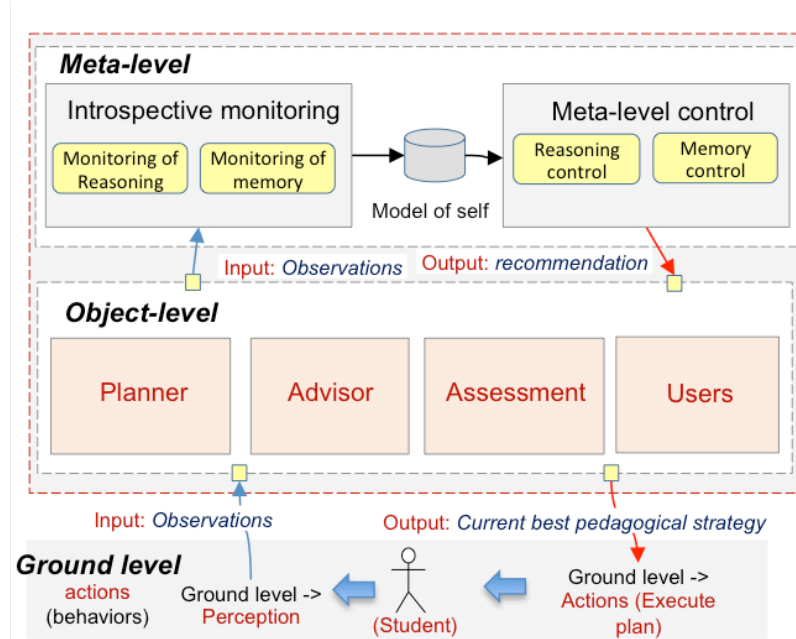


Figure 6.1. Architecture of double-loop of reasoning in FUNPRO; Source: the author.

6.1 Object-level

The object-level is comprised of the following four modules: Planner, Advisor, Assessment and Users.

- The *Planner module* is designed according to the `mppsm.metagogic.planner` package and it is responsible for selecting, organizing and sequentialize learning theories, teaching tactics and content according to student's profile. The set of BLU that conform the course of *Introduction to Programming* in FUNPRO are listed below:
 - Introduction to Algorithms (*Introducción a los algoritmos*).
 - Variables and constants (*Variables y constantes*).
 - Control statements "IF THEN" (*Sentencias de control "SÍ ... ENTONCES"*).
 - Loop "FOR" (*Ciclo "PARA"*).
 - Loop "WHILE" (*Ciclo "MIENTRAS"*).
- The design of the *Advisor module* is based on the specifications of the `mppsm.metagogic.advisor` package. This module is responsible for generating the feedback that the system gives to the student.
- The *Assessment module* aims to manage the performance indicators of a student in a course including test generation and monitoring the student performance. This module is based on the `mppsm.metagogic.assessment` package.
- The *Users module* maintains updated the behavior models of each user of the system and it is configured following the specifications of `mppsm.metagogic.users` package. In particular, the learning style is an important input in the process of personalization of pedagogical strategies in FUNPRO. The term learning styles refers to the concept that individuals differ in regard to what mode of instruction or study is more effective for them (*Pashler, McDaniel, Rohrer, & Bjork, 2009*). The approach used for modeling the student learning style was based on the model developed by Felder (*Felder & Henriques, 1995*), see Table 6.1.

Table 6.1. Learning styles modeled in FUNPRO

Dimension	Learning style
<i>Perception</i>	Sensing/Intuitive
<i>Processing</i>	Active/Reflective
<i>Reception</i>	Visual/Verbal
<i>Understanding</i>	Global/Sequential

Table 6.2 shows the equivalence of functions between the modules of FUNPRO and traditional modules of an ITS.

Table 6.2. Equivalence of functions between the modules

FUNPRO module	Traditional ITS module	Equivalent function in ITS
<i>Planner</i>	Tutor module	Planning; Sequencing
	Expert module	Domain content management
<i>Advisor</i>	Tutor module	Feedback; Scaffolding
<i>Assessment</i>	Tutor module	Learning assessment
	Expert module	Test management
<i>Users</i>	Student module	Student profile

Table 6.2 shows the traditional functions of *Tutor module* were distributed among the FUNPRO modules in order to have a greater degree of specialization. Thus, (i) the planning and sequence of contents are performed in *Planner module*; (ii) the advice and feedback are done in *Advisor module*; and assessment management is done in *Assessment module*.

6.1.1 Multi-level Pedagogical model in FUNPRO

The primary objective of the ITS is to provide personalized instruction (*Rongmei & Lingling, 2009; Z. Wang et al., 2010*). In ITS, the pedagogical model contained in the tutor module is responsible for selecting pedagogical strategies that are the most appropriated to guide the learning process of a particular student (*Barros et al., 2011; Bezerra, 2012; K. S. Cheung et al., 2010; Seridi et al., 2006*).

The pedagogical model in FUNPRO is composed by a multilevel architecture and a set of rules for the enrichment of the possibilities in personalization of pedagogical strategies. The pedagogical model is an ontology that uses components distributed among the four modules of FUNPRO.

The rules are mechanisms to determine the relationship among the components of the model and determine the pedagogical knowledge of FUNPRO. The pedagogical strategy is personalized at each level according to the characteristics of each student. The followings five abstraction levels compose the proposed pedagogic model: *Theory level, Method level, Tactic level, Activity level* and *Resource level*, see Figure 6.2.

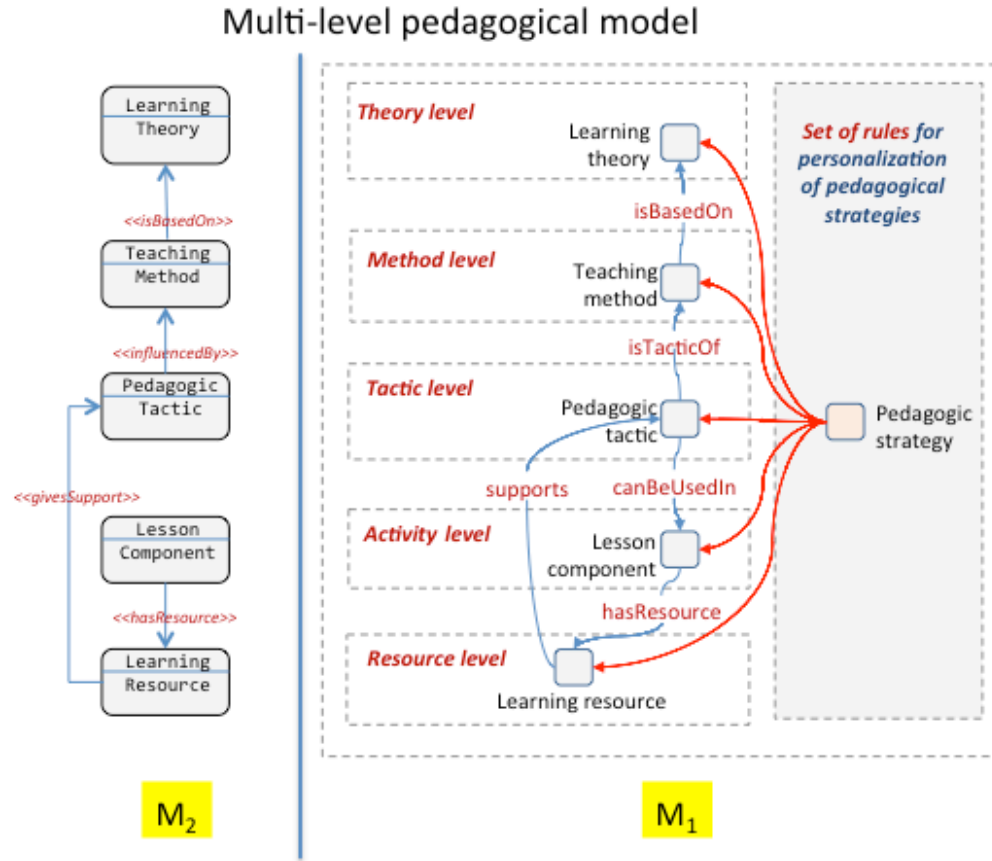


Figure 6.2. Multi-level pedagogical model in layer M_1 according to MPPSM metamodel at M_2 .

Figure 6.2 shows the concordance between the pedagogical model in the M_1 layer and the MPPSM metamodel at layer M_2 . Each level of the pedagogical model is represented by ontologies. For the definition of the terminology of educational domain used for pre-selection of the teaching methods and pedagogical tactics, a literature review was carried out. Then several meetings were held with a group of experts of the Department of Educational Psychology of the university. In these sessions, the terminology was validated and the pedagogical tactics to implement were selected. In this way, the elements of the structure of the pedagogical model, which are described below were defined.

6.1.1.1 Theory level

Learning theories are composed of a diverse set of theoretical frameworks, which try to explain how individuals access knowledge. Many features of pedagogical theories can be partially modeled computationally. This thesis have only included those characteristics that can be modeled computationally, as the type of content sequencing, the type of assistance provided to students and the type of evaluation, see model in Figure 6.3.

The proposed model supports two types of educational theories: behaviorism and constructivism. The characteristics of the behaviorism theory supported by the multilevel

model are: linear navigation between contents; immediate reinforcement and organization of content for levels with prerequisites.

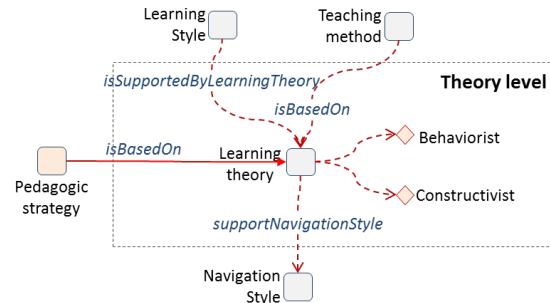


Figure 6.3. Ontology in Theory level; Source: the author.

Moreover, the multilevel model supports the followings constructivist features: free navigation among content, content organization with minimal and necessary prerequisites, formative assessment, and activities for active student participation.

6.1.1.2 Method level

A teaching method comprises the principles that imply an orderly logical arrangement of tactics and activities used in the lessons of a course. The teaching methods are based on pedagogical theories; each method may contain all or part of the pedagogical principles of theory, which is derived see Figure 6.4.

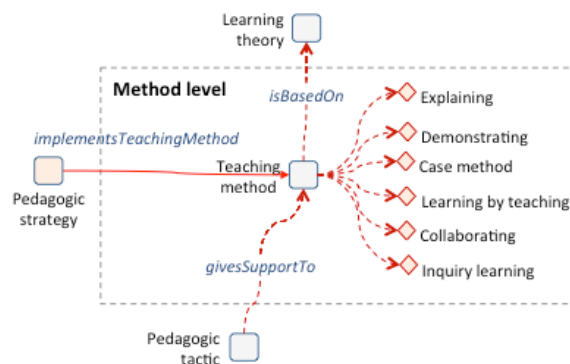


Figure 6.4. Ontology in teaching method level; Source: the author.

Relationship *isBasedOn* determines the learning theory, which is based on a teaching method and determines the organization of the content. While the relationship *givesSupportTo*, allows the association of the pedagogical tactics that will be used in the teaching method.

6.1.1.3 Tactic level

Pedagogical model provides the learning objectives based on student characteristics. In addition, it offers a range of pedagogic tactics for the student to achieve the learning

objectives that have been established for him. Pedagogic Tactics are composed of actions and resources which are used in the interaction with the student (Bezerra, 2012) for providing a personalized teaching. The criterion that was used to select the pedagogical tactics in this work was that such tactics were implementable computationally. (Peña, Marzo, De la Rosa, & Fabregat, 2002; Woo et al., 2006). In this work were implemented 19 pedagogic tactics (see Figure 6.5).

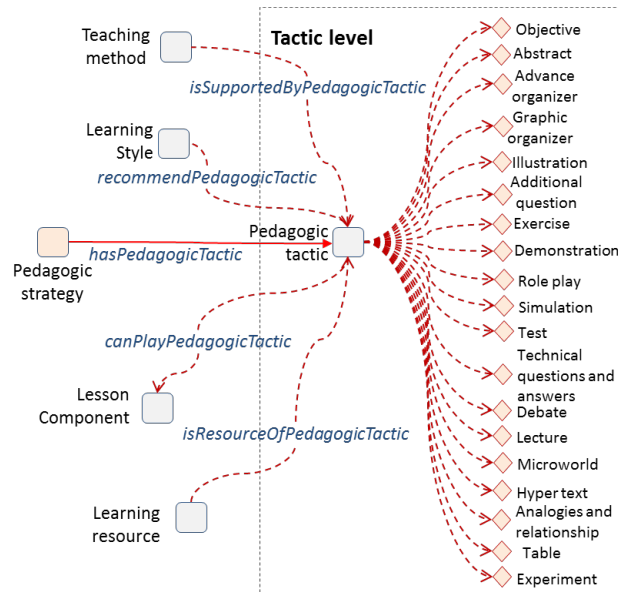


Figure 6.5. Ontology in pedagogical tactic level; Source: the author.

The relationship *canPlaysPedagogicalTactic* allows associating the pedagogical strategy with one or more lesson components. When a new learning resource is added, then a new relationship *isResourceOfPedagogicalTactic* is created.

6.1.1.4 Activity level

The components of the lesson are the sections in which the lesson activities are organized. The division into sections of the lesson has been widely used in the practice of teaching (Amorim et al., 2006; Vesin, Ivanović, Klačnja-Milićević, & Budimac, 2012; Viccari & Jiménez, 2007). Particularly in this model, the lesson components are based on (Vesin et al., 2012), but with some modifications to adjust it to context of education in the participant universities. Therefore, the pedagogical model suggests that a lesson is structured by six sections: introduction, definition, explanation, example, activity and evaluation; as shows Figure 6.6.

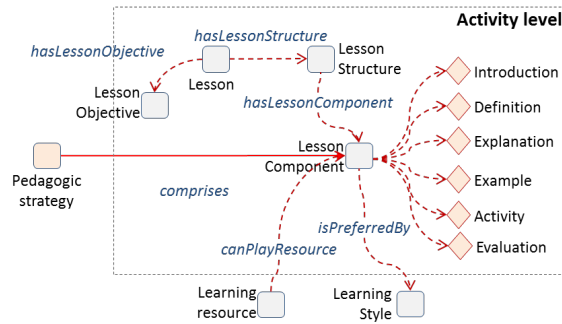


Figure 6.6. Ontology in activity level; Source: the author.

In *Introduction* section the objectives and information about the context of the lesson are presented. In the *Description* the definitions and concepts related to the lesson are displayed. The *Explanation* section delves into concepts and issues relevant to the lesson. In *Example* section are provided examples and demonstrations related to the themes of the lesson. In the *Activity* section active pedagogical tactics, as experiments, simulations and exercises are provided. Finally, in the *Evaluation* section questionnaires and various types of tests to measure student achievement in lesson are presented.

6.1.1.5 Resource level

Learning resources are digital objects such as images, animations, simulations, web pages, and more. Learning resources are the carriers of the content of the lesson and have different formats. Figure 6.7 shows the relationship between the level of resource and other components of the model.

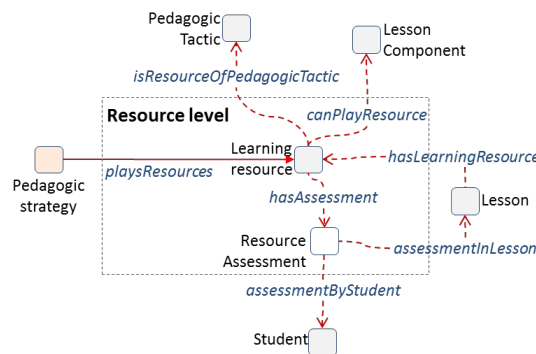


Figure 6.7. Ontology in resource level; Source: the author.

Students have the opportunity to assess learning resources that instructional planner recommends for each component of the lesson. The assessment has a scale of 1-5 as follows: (1) The resource was not useful to learn the subject of the lesson; (2) the resource made a little contribution to learning the lesson topic; (3) the resource partially contributed in learning the lesson; (4) the resource was useful for the lesson; and (5) the resource was very useful for learning the lesson topic.

6.1.2 Personalization of pedagogical strategies

The pedagogical strategy has an internal representation according to MPPSM metamodel. The pedagogical strategy is modeled with an ontology consisting of three sections: context, recommendation and performance, see Figure 6.8.

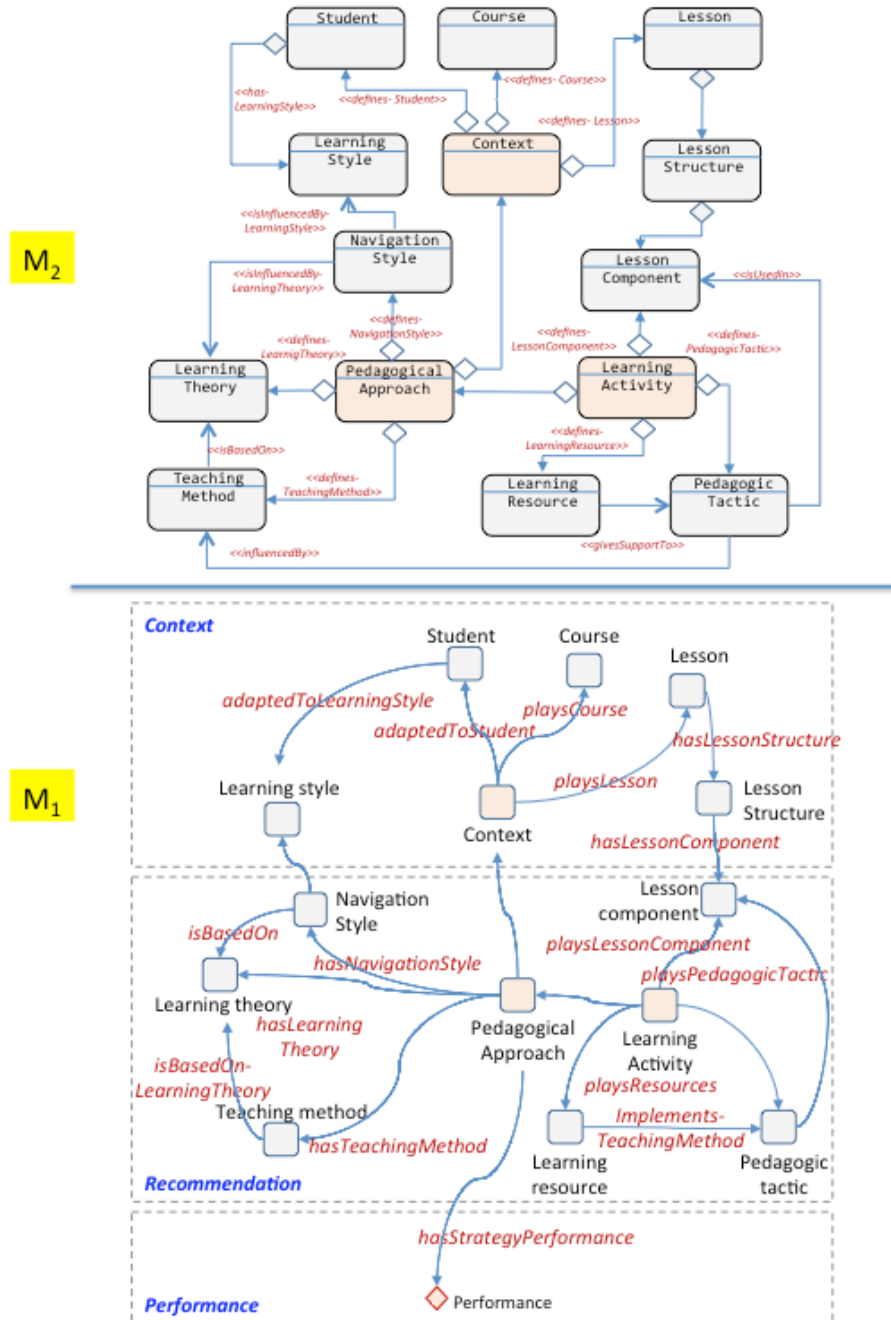


Figure 6.8. Level M_1 contains the ontological representation of pedagogical strategy in FUNPRO; Level M_1 corresponds to MPPSM metamodel at level M_2

In this work, pedagogical rules compose the pedagogical knowledge of FUNPRO and at the same time, are the mechanism used to determine the relationship among the components of the pedagogical strategies. The configuration of the set of pedagogical knowledge rules determines the capabilities of FUNPRO to adapt in a personalized way the pedagogical strategies. The declarative programming language SWI-Prolog (Wielemaker, Schrijvers, Triska, & Lager, 2012) was used for the implementation of the set of rules that form the pedagogical knowledge.

The context section contains the input data used to configure the pedagogical strategy, which are: student information, course and lesson. The recommendation section contains settings for the pedagogical strategy adapted to the student. This section consists of the navigation style, pedagogical theory, the teaching method, the components of the lesson enabled to students and pedagogical tactics for each component. The performance section stores the result of the recommendation of the strategy for a particular student. The performance of the strategy depends on the performance of the student in the lesson.

The scale of student performance is based on the guidelines established in the universities involved in this work. The scales are easily adaptable to other systems because of its general nature. Student performance is represented by qualitative values that are associated with the numerical results obtained in assessments and exercises. Performance values are: *poor* → between 0.0 and 2.0; *low* → between 2.0 and 3.0; *medium* → between 3.0 and 4.0, and *high* → between 4.0 and 5.0.

Following, the personalization process of the pedagogical strategy at each level of the model is described. The personalization process is made by the application of the reasoning rules contained in the pedagogical knowledge of the model.

Table 6.3 presents the main processes that are performed in FUNPRO for customizing pedagogical strategies. Each process consists of several tasks. The tasks are implemented through rules. For better understanding, task names reflect what rules are specified.

Table 6.3. Reasoning tasks in FUNPRO

Process	Id_rule	Reasoning task - Rule
<i>Student profile identification</i>	(1f)	hasLearningStyle(S, Y)
	(2f)	hasLearningStyleDimension(Y, D)
	(3f)	hasLessonPerformance(S, L, P)
<i>Verification of prerequisites and selection of lesson</i>	(4f)	hasLesson(C, L)
	(5f)	hasPrerequisite(L, P)
	(6f)	registeredInCourse(S, C)
	(7f)	hasLessonPerformance(S, P)
	(8f)	playLesson(S, L)
<i>Recommendation of pedagogical tactics</i>	(9f)	playTactic(S, T, I)
	(10f)	recommendPedagogicTactic(S, T)

	(11f)	<code>canPlayPedagogicTactic(T, I)</code>
<i>Recommendation and deployment of learning resources (instructional planning)</i>	(12f)	<code>hasLearningResource(L, R)</code>
	(13f)	<code>isResourceOfPedagogicTactic(R,</code>
	(14f)	<code>T)</code>
	(15f)	<code>isResourceOfLessonComponent(R,</code> <code>I)</code> <code>playResource(L, I, T, U)</code>

In Table I; S: Student - T: Pedagogic tactic - L: Lesson - R: Resource - U: URL - I: Lesson component - Y: Learning style - C: Course - P: Performance

6.1.2.1 Personalization in recommendation of learning theory

The pedagogical strategies are implemented under the criteria of learning theories; otherwise it would be limited to sequence of activities and tasks without clear educational purpose (Ozdamli, 2012). When a new student is registered, a profile is created immediately. The student diligences a form to generate his learning style profiles. When a registered student enters into the ITS, the pedagogical model activates the corresponding student model to adapt the teaching session. Preferences and indicators related to the level and learning style of the student is obtained from the Student model, see Figure 6.9.

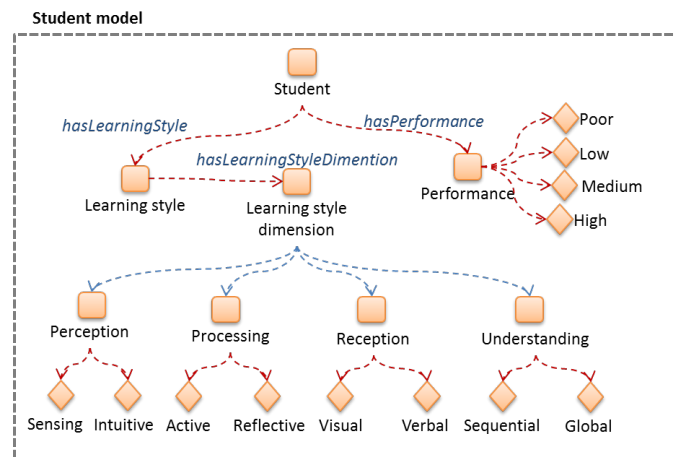


Figure 6.9. Ontology in student model

The approach used in this work for modeling the student learning style was based on the model developed by Felder (Felder & Henriques, 1995). Then the following processes perform the personalization of pedagogical strategies in learning theory level:

Profile-based adaptation: The adaptation has as input parameters the learning style of the student, including the dimension of understanding. For example: If the student has a predisposition to favor the development of content sequentially, then the system will recommend a pedagogical strategy based on behaviorism. Otherwise it will recommend a teaching strategy based on constructivism.

Rule 1:

```
[1] Student (?s) ^ hasLearningStyle (?s, sequential) ^
[2] HaslearningStyleDimension (?ls, understanding)
[3] → playsLearningTheory (?s, behaviorism) (1)
```

Dynamic adaptation: This type of adaptation occurs when the recommended tactic for teaching a lesson is changed by a new one. If the new pedagogical tactic is based on a different pedagogical theory, then the pedagogical strategy and student profile are updated.

Rule 2:

```
[1] TeachingMethod (?tm) ^ TeachingMethod (?new_tm) ^
[2] isbasedOnLearningTheory (?tm , ?lt) ^
[3] isbasedOnLearningTheory (?new_tm, ?lt') ^
[4] ¬( LearningTheory (?lt) = LearningTheory (?lt') )
[5] → playsLearningTheory (?s, ?lt') (2)
```

6.1.2.2 Personalization in recommendation of teaching method

The adaptation at teaching method level is performed in the following ways:

Adaptation based on the theory of learning. Each teaching method is influenced by one or more learning theories. Thus, if constructivist theory of learning is recommended to the student, then constructivist methods will be recommended.

Rule 3:

```
[1] Student (?s) ^ hasLearningStyle (?s, ?ls) ^ LearningStyle(?ls) ^
[2] isSupportedByLearningTheory (?ls, ?lt) ^ LearningTheory (?lt) ^
[3] isBasedOnLearningTheory(?ls, ?tm) ^ TeachingMethod (?tm)
[4] → implementsTeachingMethod (?s, ?tm) (3)
```

Dynamic adaptation: This type of adaptation occurs when the recommended tactic for teaching a lesson is changed by a new one. If the new pedagogical tactic is based on a different teaching method, then the pedagogical strategy and student profile are updated.

Rule 4:

```
[1] PedagogicTactic (?pt) ^
[2] PedagogicTactic (?new_pt) ^
[3] isSupportedByTeachingMethod (?pt , ?tm) ^
[4] isSupportedByTeachingMethod (?new_pt, ?tm') ^
[5] ¬( TeachingMethod (?tm) = TeachingMethod (?tm') )
[6] → implementsTeachingMethod (?s, ?tm')(4)
```

6.1.2.3 Personalization in recommendation of pedagogic tactics

The personalization of pedagogical strategies at pedagogical tactics level is performed in the following ways:

Profile-based adaptation: The recommendation of pedagogical tactics is carried out based on the student profile.

Rule 5:

```
[1] Student (?s) ^ hasLearningStyle (?s, ?ls) ^ LearningStyle(?ls) ^
    implementsTeachingMethod(?s, ?tm) ^ TeachingMethod(?tm) ^
    givesSupportToPedagogicTactic(?tm, ?pt) ^
    isSupportedByPedagogicTactic (?ls, ?pt) ^ PedagogicTactic (?pt)
    ^ canPlayInLessonComponent (?pt, ?lc)
[2] → playsPedagogicTactic (?lc, ?pt) (5)
```

In this work, 19 pedagogic tactics were implemented see Figure 6.5. The selection of the pedagogical tactics was based on (Bezerra, 2012; Peña et al., 2002; Woo et al., 2006).

Dynamic Adaptation: This type of adaptation occurs when a student changes a recommended resource for a lesson with a new one. In this case if the new resource supports a different kind of pedagogical tactics then the system reconfigures the preferences of recommendation for the student and tag the recommendation as inappropriate.

Rule 6:

```
[1] LearningResource (?current_lr) ^ LearningResource (?new_lr) ^
    isResourceOfLessonComponent (?new_lr, ?lc) ^
    isResourceOfPedagogicTactic (?new_lr, ?pt') ^
    isResourceOfPedagogicTactic (?lr, ?pt) ^ ¬( PedagogicTactic
    (?pt) = PedagogicTactic (?pt') )
[2] → playsPedagogicTactic (?lc, ?pt') (6)
```

6.1.2.4 Personalization in recommendation of learning resources

Adapting pedagogical strategies in the level of resources occurs in the following cases:

Profile-based adaptation: Consists of the recommendation of resources according to the characteristics of the student profile.

Rule 7:

```
[1] Student (?s) ^ hasLearningStyle (?s, ?ls) ^ LearningStyle(?ls) ^
    canUseTeachingMethod(?s, ?tm) ^ TeachingMethod(?tm) ^
    givesSupportToPedagogicTactic(?tm, ?pt) ^ PedagogicTactic (?pt)
    ^ isSupportedByPedagogicTactic (?ls, ?pt) ^
    canPlayInLessonComponent (?pt, ?lc) ^ LearningResource (?lr) ^
    isResourceOfLessonComponent (?lr, ?lc) ^
    isResourceOfPedagogicTactic (?lr, ?pt)
[2] → playsLearningResource (?pt, ?lr) (7)
```


Rule 7 is conditioned by both, the context of the pedagogical strategy and the result of the evaluation done by the student to the learning resource after using it in the lesson. The resources are sorted from highest to lowest evaluation result. The model selects the resource with highest performance.

Adaptation by preference: This type of recommendation occurs when a student changes the recommended learning resource to a different resource. In this case, the system updates the student's preferences according to the characteristics of the newly selected resource and the new profile will be used for new recommendations.

Rule 8:

```
[1] LearningResource (?current_lr) ^ LearningResource (?new_lr) ^
    isResourceOfLessonComponent (?new_lr , ?lc) ^
    isResourceOfPedagogicTactic (?new_lr, ?pt) ^
    isResourceOfPedagogicTactic (?lr , ?pt') ^ PedagogicTactic (?pt)
    = PedagogicTactic (?pt')
[2] → playsLearningResource (?pt, ?new_lr) (8)
```

6.1.3 Learning environment

The general process that describes the functioning of FUNPRO is as follows:

- The first page of FUNPRO contains (i) menu section; (ii) login section and; (iii) the workspace with a descriptive message, see Figure 6.10.



Figure 6.10. Welcome page in FUNPRO with menu section, login section and workspace

- The system identifies the student logged. If FUNPRO detects that is the first time the student enters, then a Felder test is displayed to identify the student's learning style, see Figure 6.11.



Figure 6.11. Identification of student's profile in FUNPRO

- The Tutor Module activates the corresponding student model to adapt the teaching session.
- Tutor Module provides the LO as student characteristics.
- Tutor Module offers a range of pedagogic tactics for the student to achieve the LO that have been established for him. From Expert module lessons are obtained to teach and the resources available for such lessons. From Student module is obtained: preferences and indicators related to the level and learning style of the student, see Figure 6.12.

In the help section, the Advisor module shows the feedback according to student behavior in the system.

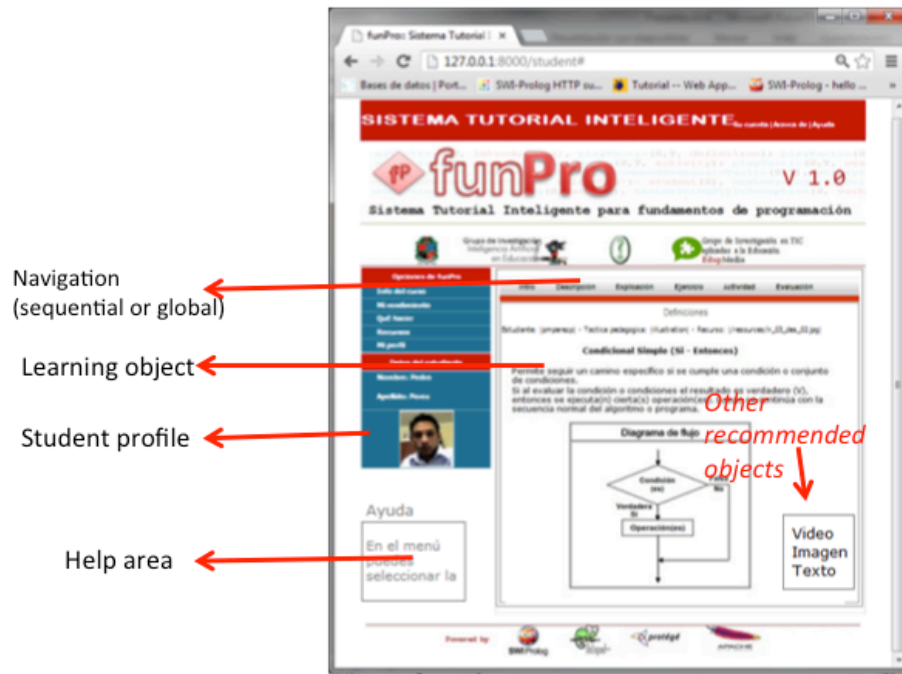


Figure 6.12. Description of the interface components in FUNPRO

- Tutor Module executes the lesson plan designed for the individual student and verifies the student's responses and performance, see Figure 12. If the student's performance is not as expected, makes Tutor Module replanning lesson.

FUNPRO personalizes the pedagogical strategies for a particular student based on the implemented rules. In this way, the system is able to suggest the most appropriated pedagogical tactics for a student according to his learning style and performance on the course or in a particular section of a lesson. Moreover, the system is able to select from the available resources in a lesson, those that are the most suitable for a pedagogical tactic. These capabilities are achieved through a variety of pieces of code that are included in the rules of the GUI module. As an example the following piece of code that allows displaying an educational resource into the Web environment of FUNPRO is presented:

```
[1] display_lesson_content(S,L,I) -->
[2] {
[3]     playTactic(S,T,I), playResource(L,I,T,R)
[4] },
[5] html([
[6]     \display_resource(R)
[7] ]).
```

In this way, the piece of code `display_lesson_content` first calls for rules in object-level and then make a call to other included piece of code `display_resource`, this is

responsible for displaying the educational resource in the web environment. Thus, the result of the call to the piece of code can be appreciated in Figure 6.13.

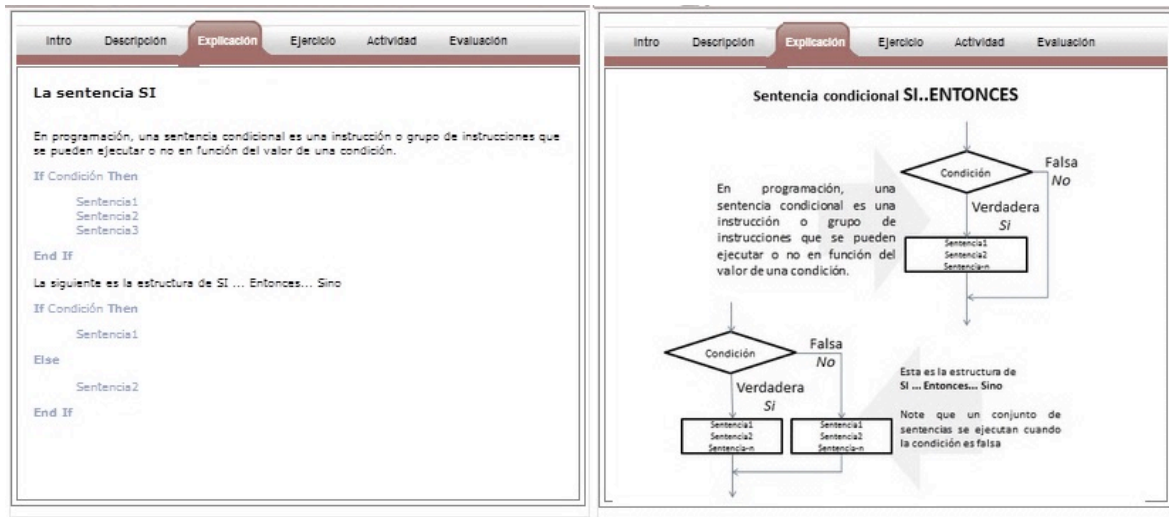


Figure 6.13. Explanation of lesson –“Sentencia SI”-. Left side an explanation for a verbal student. Right side an explanation for a visual student.

Similarly, the implementation of rule-based reasoning allows FUNPRO to adapt the navigation system through the components of the lesson. Rules for the personalization of navigation process are based on the assessment of the state of the student's learning style in the dimension *understanding*.

The following is the behavior of the GUI for configuring the navigation system:

```
[1] div_menu_content(S) -->
[2]   {
[3]     (isNavigationGlobal(S)->
[4] Menu=ul_global_tabs_content;
[5] Menu=ul_sequential_tabs_content)
[6]   },
[7]   html([
[8]     ul([id('ul_tabs'),class('glossymenu')],\Menu)
[9]   ]).
```

The piece of code makes a call to the rule: `isNavigationGlobal(S):-student(S), hasLearningStyleUnderstanding(S, global)`. This rule is true if the student *S* has the learning style *global* in dimension *understanding*. Thus the piece of code selects the global menu in the form of tabs or the sequential menu in the form of buttons of type next-previous. The result of the rule is shown in Figure 6.14.



Figure 6.14. Navigation model: A) Navigation style Tab for global students, B) Navigation style buttons (Next-Previous) for sequential students

The implementation of a multi-level pedagogical model based on MPPSM enables to FUNPRO the achievements of goals in the following items:

- **Environment:** The web environment is configured according to the user logged. The configuration includes the navigation system among lessons and the components of the lesson.
- **Audience:** The personalization of learning experience involves students' learning styles and the performance on each lesson.
- **The principles of learning theories:** In this work the pedagogical theories influence teaching strategies through the rules associated with the student's learning styles.
- **Pedagogical tactics:** The selection of 19 pedagogical tactics was based on the advice of experts and the literature reviewed. These 19 pedagogical tactics are widely used in face education but at the same time can be used within ITS.
- **Learning resources:** There are a number of resources associated with the lesson. The rules used allow the selection of the most appropriated resources to be displayed in each section of the lesson. The presentation of the resources is done in a personalized way, according to the student's learning style. The support that the resource gives some kind of pedagogical tactics and deployment restrictions in each component of the lesson.

6.1.3.1 Meta-level

The meta-level is composed of the following two modules: Self-regulation and Metamemory.

- The *Self-regulation module* aims to monitor and control the processes of reasoning at object-level. This module is based on the `mppsm.mism.selfregulation` package.
- The *Metamemory module* monitors and controls the events related to the search for information stored in Long Term Memory (LTM). Metamemory in FUNPRO consists of a cycle of reasoning about events that occur in the LTM. The reasoning cycle inputs are the memory events that occur in LTM and the output consists of recommendations, which may vary according to the memory events. In particular, this thesis is focused on the reasoning process that allows the

adaptation to constraint changes related to retrieving information from LTM. Metamemory module is designed according to `mppsm.mism.metamemory` package.

The implementation of metacognition in FUNPRO includes the specification of metacognitive reasoning points (MRPs) and the definition of metacognitive mechanisms such as introspective monitoring and meta-level control.

6.1.3.1.1 Defining MRPs

MRPs define the reasoning tasks at object-level that are monitored and controlled using metacognition in meta-level. According to the object-level processes listed in Table 6.3, instructional planning task is selected as a MRP. A failure in the execution of instructional planning task could significantly affect performance of the system, see rule 3. Rule (15f) gets the path U of the resource R selected to be displayed in section I of lesson L.

The meta-level keeps updated an abstract model of each MRP of object-level (e.g. self-model). The meta-level performs reasoning and decision making based on the self-model.

6.1.3.1.2 Metacognitive mechanisms in FUNPRO

Introspective monitoring and meta-level control is the metacognitive mechanisms implemented in the modules of self-regulation and meta-memory.

6.1.3.1.2.1 Self-regulation of reasoning process in FUNPRO

Instructional planning in FUNPRO aims to generate a pedagogical strategy adapted to the profile of each student. The pedagogical strategy takes the form of an instructional plan that contains the necessary actions to select among the following: learning theory, teaching methods, pedagogical tactics, and resources for a course or lesson.

FUNPRO makes replanning to the pedagogical strategy in the following cases: (i) if a resource for some reason cannot be deployed in the lesson, e. g. resource has the URL broken; (ii) if a recommended resource has received a poor evaluation; (iii) and if the student obtains a low performance in the lesson. The *instructional planning* constantly refines the pedagogical strategy for each student using three types of recommendation strategies: (i) matching simple query, the search query in a simple SQL type; (ii) exclusive search is similar to (i), but excludes some results and; (iii) vote-based search, this strategy is based on the nearest neighbor algorithm.

The meta-level intervenes in the process of instructional planning by deciding whether to continue reasoning for a better plan or execute the current plan. When a plan is generated, the meta-level analyzes the possibility of refining the plan (reasoning about the planning process) using as evidence the effectiveness of similar plans in the past (Rule 7). If the meta-level finds that the expected performance of the current plan is sufficient to achieve the planned goals, then it proceeds to execute the plan. But if there are possibilities to improve the plan in a reasonable time, then the meta-level decides to continue planning further. Figure 6.15 shows a user model for the *instructional planning* function of the ITS. The meta-level uses the variables *latency* (λ) (Benjamin, Schmidt, Newman, & Leonard, 2013;

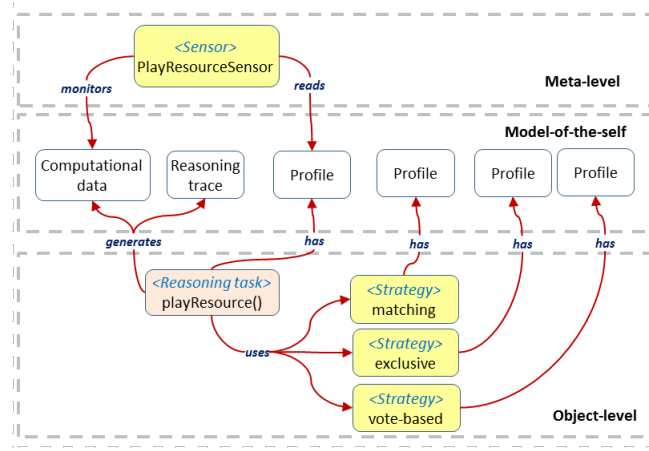


Figure 6.16. Basic flow of information in the introspective monitoring implementation - FUNPRO

The following code snippet in SWI-Prolog shows the implementation of the activation of a sensor at meta-level in FUNPRO.

```
[1] sensorActivation(ID_reasoning_task, Sensor):-
[2]   reasoning_task(ID_reasoning_task),
[3]   newReasoningTaskOutput(ID_reasoning_task, U),
[4]   sensor(ID_sensor, _),
[5]   sensorMonitors(ID_sensor, ID_reasoning_task),
[6]   updateSensorState(D_sensor, active).
```

Reasoning failure detection starts when a sensor is activated. The meta-level gets updated observations from the sensor associated to the MRP (rule 3), using the instructions `showActiveSensor(Action, Sensor)` and `update-SensorObservation(Sensor, Observation)`.

Once the current reading of the active sensor is obtained, then meta-level checks whether the observation is consistent with the expectation of the sensor based on rule (5). In the following piece of code we can see the identification of a reasoning failure.

```
[1] isReasoningFailure(Action, Observation):-
[2]   showActiveSensor(Action, Sensor),
[3]   updateSensorObservation(Sensor, Observation),
[4]   anomalyInExpectation(Sensor),
[5]   generateReasoningFailure(ID_reasoning_task, ReasoningFailure),
[6]   updateFailureCounter(V).
```

After reasoning failure is detected, the meta-level generates an explanation, rule (6). There are three possible explanations for a reasoning failure occurred in `playResource`:

- *Resource available* refers to an unavailable resource when deploying on FUNPRO.
- *Inappropriate resource* refers to a resource that was not adequate to the student profile.
- *URL broke* refers to an available or valid resource that has a broken URL.

Explanations can be generated in two ways: *search for known explanations* and *reasoning trace analysis*.

Strategy *search for known explanations* queries for explanations given to reasoning failures in the past and then evaluates and prioritizes explanations, see the following piece of code.

```
[1] explanationGeneration(ReasoningFailure, Explanation):-
[2]   hasExplanation(ReasoningFailure,Explanation),
[3]   explanationPriorization(Explanation).
```

The *reasoning trace analysis* strategy performs a more complex reasoning because it makes queries on the trace of the structures of reasoning performed by the failed task looking for anomalies, see the following piece of code.

```
[1] explanationGeneration(ReasoningFailure, Explanation):-
[2]   hasReasoningFailure(ReasoningTask,ReasoningFailure),
[3]   anomalyInReasoningTrace (ReasoningTask,Anomaly),
[4]   anomalyExplanation(Anomaly,Explanation),
   explanationPriorization(Explanation).
```

Finally, the meta-level generates a goal based on the explanation in order to solve the reasoning failure.

```
[1] goalGeneration(Explanation, Goal):-
[2]   hasReasoningFailure(ReasoningTask, Explanation),
[3]   reasoningTaskHasProfile(ReasoningTask, Profile),
[4]   reasoningTaskUsesStrategy(ReasoningTask, Strategy),
[5]   reasoning_task_profile(Profile_r,Goal_r,Performance,_,_,_),
   strategy_profile(Profile_s,Goal_s,Performance,_,_,_),
[6]   goalCandidate(Profile_r,Profile_s,Goal).
```

6.1.3.1.2.3 Implementation of Meta-level control in self-regulation

Meta-level control starts after goal generation. The main function of meta-level control is to select the best available strategy to address the reasoning failure. The selection of strategies receives the Goal to be achieved as a parameter and searches through the available strategies those which satisfy the Goal.

```
[1] buildGoal(G,A,T,Gs,Grslt):-
[2]   generateGoal(G),
[3]   assert(hasGoalAction(G,A)),
[4]   assert(hasGoalTarget(G,T)),
[5]   assert(hasGoalState(G,Gs)),
[6]   assert(hasGoalResult(G,Grslt)),
[7]   setCurrentGoal(G).
[8] abstractModel(search_strategy,ss_association).
[9] abstractModel(search_strategy,ss_exclusion).
[10] abstractModel(search_strategy,ss_voting).
[11] abstractModel(search_strategy,ss_neighbors).
```


Meta-level control then evaluates the performance of each strategy by selecting the best, in the following piece of code can be see the general implementation.

```
[1] recommendStrategy(Goal, Strategy):-
    goalAchievedWith(Goal,Strategy, Performance),
    strategyPriorization(Strategy,Performance).
```

6.1.3.1.3 Metamemory in FUNPRO

Metamemory functionalities in the meta-level of FUNPRO are activated when some process from object-level calls a search task to retrieve information from LTM. After call, a memory event is triggered. The meta-level stores traces of all the events that occur in LTM.

The events (E) represent actions that are performed on the memory. $E=\{ID, y, g, d, t\}$ is the set of components that represents the structure of an event, where:

ID is the unique identifier of the event.

y is the type of the event, $y \in Y$ and $Y=\{call, execute, re-configure\}$.

g is the goal of the event.

d is the constraint of the event.

t is the memory task that originated the event.

The events that occur at object-level can be of different types, for this particular research three types of events are processed: i) **call** if the event is a call to a search, acquisition or retention task on memory. This type of event is previous to the execution of the task. This event indicates to the system that a specific task on memory is required; ii) **execute** indicates that a search, acquisition or retention task is running on memory; iii) **re-configure** indicates that a search, acquisition or retention task has failed and it needs to be reconfigured.

Goals (G) are subcomponents of events. Each event can have only one goal. Goals contain relevant meta-data about information to be stored or retrieved from memory. $G=\{ID, a, t, s, r\}$ is the set of components that represents the structure of a goal, where:

ID is the unique identifier of the goal.

a is an action performed on memory, $a \in A$ and $A=\{acquisition, retention, retrieval\}$.

t is the target of the action a .

s is the state of the goal, $s \in S$ and $S=\{starting, waiting, working, finished\}$.

r represents the final result of the goal, $r \in R$ and $R=\{satisfied, unsatisfied\}$.

For illustration in FUNPRO as example: if the system is doing a search of resources for a student's lesson then the type of event memory is **execute**; the goal action is **retrieval**; the goal status is **working** and the goal result will depend on the success or failure of the search.

In FUNPRO, retrieval includes tasks that are associated with the access of stored information from each module at object-level. The information stored in LTM is mainly composed of:

- User profiles
- Records of student's behavior in the system
- Monitoring of student performance
- Course content
- Pedagogical strategies
- Pedagogical tactics
- Teaching methods
- Learning Resources

FUNPRO has implemented three types of search strategies for retrieving information from LTM: (1) *matching simple query*, the search query in a simple SQL type; (2) *exclusive search* is similar to (1), but excludes some results and; (3) *vote-based search*, this strategy is based on the nearest neighbor algorithm. One crucial influence on the outcome of any retrieval process is the knowledge available to that process (Leake, 1995). This knowledge includes search constraints (Kizilirmak, Rösler, & Khader, 2012), parameters and other information related to the target of the search (Unsworth, 2010). The search constraints restrict the information retrieved by influencing the search strategy used to fulfill the goal of the search task (Mecklinger, 2010).

The search constraints (D), in an event (E) refer to the information requirements that must be satisfied so that the **event** fulfills the goals. $D=\{ID, K, X, Q, y\}$ is the set of components that represents the structure of a constraint, where:

ID is the unique identifier of the goal.

$K=\{k1, \dots, kn\}$ represents the set of information requirements needed to retrieve or store the target of the goal.

$X=\{x1, \dots, xn\}$ is the set of information excluded from retrieval.

$Q=\{q1, \dots, qn\}$ is the set of special requirements needed to retrieve or store the target of the goal.

y is the type of the constraint, $y \in Y$ and $Y=\{\text{basic}, \text{complex}\}$.

Changes in search constraints affect the performance of information retrieval (Huet & Mariné, 1997; Kizilirmak et al., 2012). When there are changes in constraints, information retrieval cannot be done effectively by the same search strategy for all cases. Thus the system needs to assess changes in the constraints of the search tasks and select the most appropriate search strategy. For example in FUNPRO, if the system recommends a

resource, which is poorly evaluated by the students, then the resource is excluded of a new search with similar settings. The constraints assessment is made using metacognitive judgments.

Metacognitive judgments (J) represent assessments performed in the meta-level about events that occur in memory. These judgments provide information that the system uses to determine whether it is able to attempt retrieval or storage. The meta-level of FUNPRO has implemented two types of metacognitive judgments, these are:

COP (Certainty of Optimal Performance) measures the degree of certainty that the system has with regard to optimum performances obtained in the past, having constraints similar to the current user.

CSRD (Certainty of Satisfying the Retrieval constraints) measures the degree of certainty that the system has with regard to the level of knowledge that the system possesses to attend the requirements of the retrieval constraints.

Figure 6.17 shows the basic flow of the FUNPRO behavior. Figure 6.17 has been divided into four sections labeled A, B, C and D; representing different cases of object-level information retrieval tasks.

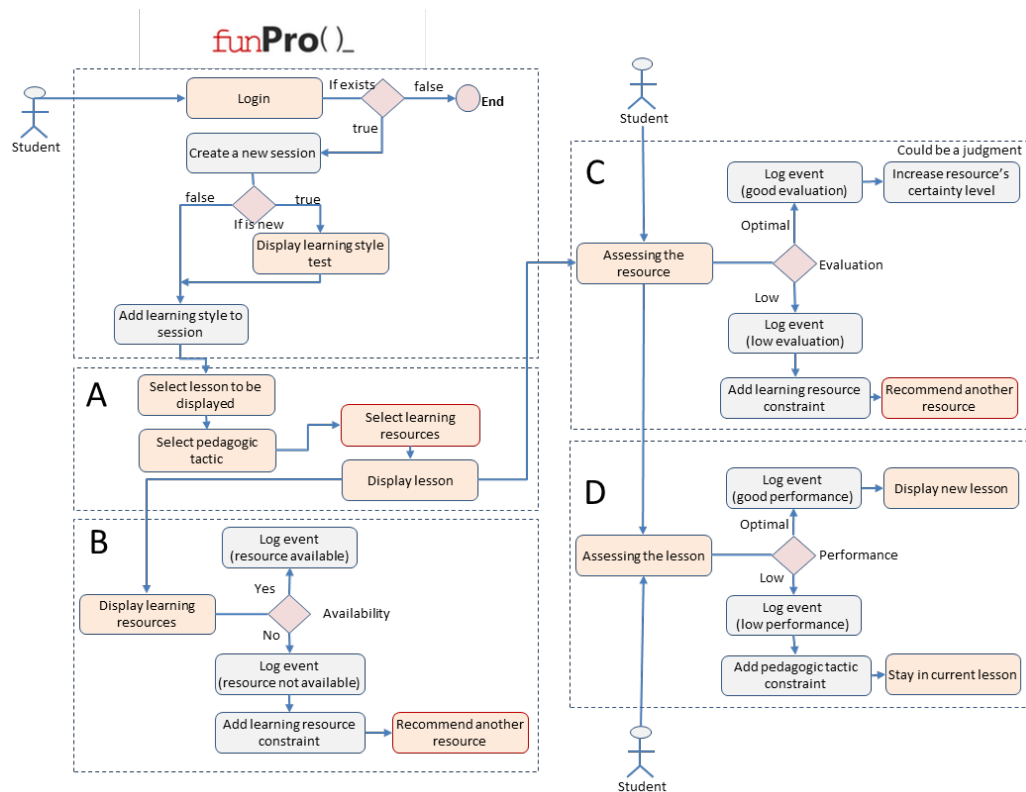
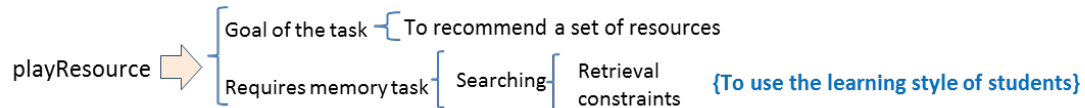


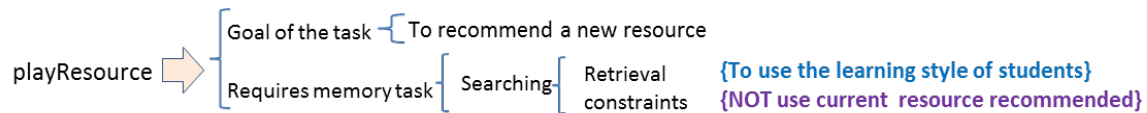
Figure 6.17. Flow diagram of FUNPRO with different sections regarding information retrieval

FUNPRO has a function called `playResource` that is responsible for retrieving the URL of learning resources from the knowledge base, and deploying them in the lesson. However, the constraints that FUNPRO generates to search for resources for the lesson are dynamic according to several criteria described below.

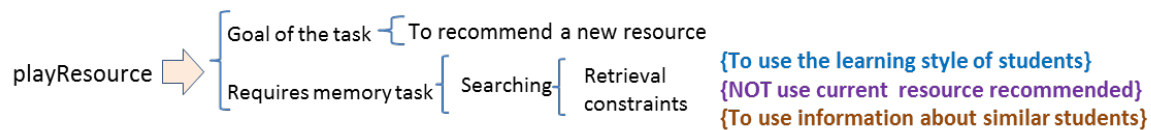
Case A. The student enters the lesson for the first time; therefore FUNPRO has only collected information about the student's learning style to recommends learning resources and teaching strategies for the lesson. Thus `playResource` function receives a single constraint.



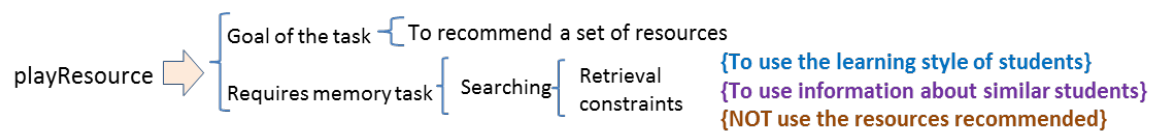
Case B. FUNPRO finds a resource that meets the restrictions of the search, but for some reason cannot be deployed in the lesson, for example: the resource URL is broken. In this case FUNPRO has to retrieve from the knowledge base another resource that supports the student's learning style, but it has to exclude the resource with the URL broken.



Case C. The student evaluates the resource after using it. If the resource has received a poor evaluation, the system recommends new resources that have been well evaluated by students with similar characteristics to the current.



Case D. Case D has two variants. In the first, if the student obtains a low performance in the lesson, then the system remains in the current lesson but reconfigures strategies for teaching and learning resources. In the second variant, if the student obtains a high performance; then the system presents a new lesson.



6.1.3.1.3.1 Implementation of introspective monitoring in metamemory

In the code snippets shown below, the restrictions are called "demands" to match with the common name used in the cognitive sciences.

When a new memory event trace is stored in the meta-level, the monitoring process starts at the meta-level. The meta-level detects and identifies the event in LTM, as it is shown in the following code snippet.

```
[1] inputEvent(E,Y,A,S,T,K,X,Q):-
[2]   buildGoal(G,A,T,started, unsatisfied),
[3]   buildDemand(D,K,X,Q),
[4]   buildEvent(E,Y,G,D,S).
[5]
[6] isCallCurrentEvent(V):-
[7]   metaLevelState(current_memory_event,V),
[8]   hasEventType(V,call),
[9]   newDebugML('{Meta-level}->[isCallCurrentEvent] -> reasoning
    about Event type {Event type}: ',call).
```

If the event detected is a call to a search task (e. g. *a new search for learning resource recommendation*), then the meta-level checks for changes in task constraints of the target of the search.

```
[1] isCallCurrentEvent(V):-
[2]   metaLevelState(current_memory_event,V),
[3]   hasEventType(V,call).
[4]
[5] buildDemand(D,K,X,Q):-
[6]   generateDemand(D),
[7]   assert(needKnowledgeAbout(D,K)),
[8]   assert(excludeKnowledge(D,X)),
[9]   assert(hasSpecialRequirement(D,Q)),
[10]   getDemandType(D,T),
[11]   assert(hasDemandType(D,T)),
[12]   setCurrentDemand(D).
```

The changes in the task constraints occur when there is a failure due to difference between the observation and expectation of target of search. Expectations in FUNPRO can be specified by default in the system configuration or may be self-generated by the system task (e. g. *when a new learning resource is recommended then the system aspects that the resource will be useful for student learning*).

```
[1] isReasoningFailure(Action,Observation):-
[2]   showActiveSensor(Action,Sensor),
[3]   updateSensorObservation(Sensor,Observation),
[4]   anomalyInExpectation(Sensor),
[5]   updateFailureCounter(V),
```

```
[6] newDebugML(' {Meta-level}->[isReasoningFailure] -> Reasoning
    failure {ID}: ',V).
```

If any change in constraints of the search task is detected (*rule 4*), then the meta-level decides to launch a deeper reasoning process about the memory event. The reasoning involves examination and assessment of the performance of the information retrieval task with similar constraints in the past. In the examination and assessment of the performance process the meta-level searches for events that occurred in the past with similar restrictions.

If the events with similar meta-level constraints are located, meta-level then proceeds to obtain the search strategies that have been used to process such events. If the meta-level has at least one event that has been processed successfully, then it makes a COP judgment with high value. This means that the meta-level in FUNPRO has a high level of certainty of knowing the appropriate search strategy to satisfy the request of retrieving information contained in the current event.

```
[1] certaintyKnowingContent(E):-
[2] hasGoal(E,G),hasGoalTarget(G,T),
[3] hasKnowledgeAbout(T,K),
[4] hasLinkBetweenKnowledge(T,K).
```

The meta-level maintains a performance profile of search tasks, which consists of a record of the search strategies that have been used to process information retrieval requests in the past.

6.1.3.1.3.2 Implementation of Meta-level Control in metamemory

The meta-level control is based on the value of the metacognitive judgments. For example, if a COP judgment has a high value, then the meta-level recommends the search strategy that has had better performance in events with similar constraints in the past.

```
[1] triggerJudgment(E,A,J,S):- (A->
    J=high,getOptimalProcessing(E,S);J=low).
[2] recommendStrategy(E,S):-
[3] (isStandardDemand(D)->
[4]     strategyRecommendedForDemand(standard_demand,S),
[5]     newDebugML(' {Meta-level}->[recommendStrategy] ->
        reasoning about Demand type {Demand type}: ',standard_demand),
[6]     newDebugML(' {Meta-level}->[recommendStrategy] ->
        reasoning about Strategy recommendation for demand type
        {Strategy recommended}: ',S)
[7] ;
[8] (isExcludeDemand(D)->
```

```

[9]         strategyRecommendedForDemand(exclude_demand,S),
[10]         newDebugML(' {Meta-level}->[recommendStrategy] ->
reasoning about Demand type {Demand type}: ',exclude_demand),
[11]         newDebugML(' {Meta-level}->[recommendStrategy] ->
reasoning about Strategy recommendation for demand type
{Strategy recommended}: ',S)
[12]         ;
[13]         getOptimalProcessing(E,S),
[14]         newDebugML(' {Meta-level}->[recommendStrategy] ->
reasoning about deep search {Strategy recommended}: ',S)
[15]     )
[16] ).

```

In case the judgment has a low value, and the system has available intelligent complex search strategies, then metamemory offers the possibility for the meta-level to recommend these strategies. For this purpose, the meta-level evaluates the knowledge about the requirements implicit in the constraints of the search.

```

[1] hasKnowledgeAbout(T,K):-knowledge_source(T),knowledge_source(K).
[2] hasLinkBetweenKnowledge(T,K):-knowledge_link(_,K,T).

```

If some knowledge related to constraint is obtained, then the meta-level triggers a CSRD judgment with high value. Otherwise, the meta-level control mechanism recommends to the object-level to stop the information retrieval, because there is not enough knowledge to process the search.

$ST(st)$: st is a search task
 $E(e)$: e is a memory event
 $T(st, e)$: search task st causes event e to be triggered
 $TJK(e, j)$: due to the characteristics of the event e ; CSRD judgment j is triggered
 $JVK(j, low)$: CSRD judgment j has value *Low*
 $STP(st)$: the meta-level recommends stopping search strategy st

$$\forall st \forall e \forall j (ST(st) \wedge T(st, e) \wedge E(e) \wedge TJK(e, j) \wedge JVK(j, low) \rightarrow STP(st))$$

6.2 Validation

Performance evaluation of intelligent or metacognitive systems is a difficult task. In the field of ITS, performance is typically measured in terms of the end-user application metrics. Student performance, usability, precision ratio are some common examples. In the

case of FUNPRO, several tests were conducted in order to evaluate the performance of the object-level and the meta-level.

The metacognitive mechanisms for self-regulation and metamemory were validated using different performance metrics because self-regulation works on the reasoning process and metamemory monitors the memory events.

6.2.1 Validation of self-regulation for monitoring and control of personalization of pedagogical strategies

A practical experiment was conducted in order to verify the performance of the metacognitive mechanism of self-regulation in the process of the personalization of pedagogical strategies. The experiment took into account the students' preferences and profiles using FUNPRO.

The experiment was a comparison between two groups of students. A first group of 22 students who used FUNPRO with metacognitive module enabled (experimental group-EG) in relation to a second group of 22 students who used FUNPRO with metacognitive module disabled (control group-CG). This validation can be classified as a quasi-experiment, because the sample subjects were not chosen randomly (*Haas & Kraft, 1984; Shadish, Cook, & Campbell, 2002*). These students were selected because of their previous contact with our research group and also because of the interest of their teachers.

The course consisted of five lessons with basic level of complexity.

The performance metrics used to measuring the use of self-regulation in the personalization of pedagogical strategies was: (i) the average of changes made over the pedagogical strategies recommended at each level of the pedagogical model and; (ii) the relationship observed between the evaluation made by students to learning resources and the changes made by the system to the pedagogical strategy.

i) Personalization of pedagogical strategies

Changes in pedagogical strategies can be made directly by the student (e.g. when the student changes a learning resource) or dynamically (e. g., when the ITS detects a change in the learning preferences of the student).

Changes made on the components of the pedagogical strategy recommended for each student will be interpreted as inappropriate recommendations. In this sense, if the amount of changes needed to adjust the pedagogical strategy according to the student's profile is high, then the level of personalization of the teaching strategy will be low.

In this sense, the goal of the experiment was to see whether the use of self-regulation could increase the level of personalization of pedagogical strategies by reducing changes to the recommended strategy for each student in each lesson of the course; and observing

if the reduction of inappropriate recommendations had a positive effect on student performance in the lesson.

In the data obtained from the experiment with respect to the behavior of the adaptation of the pedagogical strategies to the student profile at each level of the model, the difference between the mean of the *group-CG* and *group-EG* related to adaptations per student observed at each level of the pedagogic model is statistically significant. The difference observed at learning resource level was 3.49; the difference observed at pedagogical tactic level was 2.62; the difference observed at teaching method level was 1.73; and the difference observed in level of learning resources was 1.16, see Table 6.4.

On the other hand, the average of adaptations in pedagogic tactic level was 0.82; the average of adaptations at the level of teaching method was 0.36 and the average of adaptations at the level of learning theory was 1.73; and the difference observed at learning theory level was 1.16.

Table 6.4. Changes in pedagogical strategies - pretest and posttest mean and standard deviation (sd).

	Mean (group- CG)	Mean (group- EG)	Gain
Learning			
Resource	4,93	1,44	-3,49
Pedagogic Tactic	3,44	0,82	-2,62
Teaching			
Method	2,09	0,36	-1,73
Learning Theory	1,31	0,15	-1,16

The negative value of the *Gain* column means that there was a reduction of incorrect recommendations in the FUNPRO. The negative value occurs because FUNPRO used metacognition for monitoring and controlling the process of personalization of pedagogical strategies.

With respect to the occurrence of adaptation generated by the relationship among the levels, it can be said that 56.9% of resource changes made by the student, generated changes in pedagogical tactics by the application of rule (7). 42.7% of pedagogical tactic changes, generated changes of teaching methods by the application of rule (4); and 42.9% of changes of teaching method, generated changes of Learning Theory for pedagogical strategy as a result of rule (2).

ii) **Relationship between the resource evaluation and the changes made to the pedagogical strategy**

Section (A) in Figure 6.18 shows the inverse relationship found between the average of resources assessment made by the student and the changes on the pedagogical strategy. The students evaluated with a better score the learning resources that were included into the teaching strategy recommended for a new lesson, than those learning resources recommended in the previous lesson. As the evaluation of the students to the learning resources improved, a decrease in the percentage of adaptations required to personalize the pedagogical strategy was observed. This is due to the fact that, the system learns with each adaptation made to the pedagogical strategy, producing better recommendations at each level of the model.

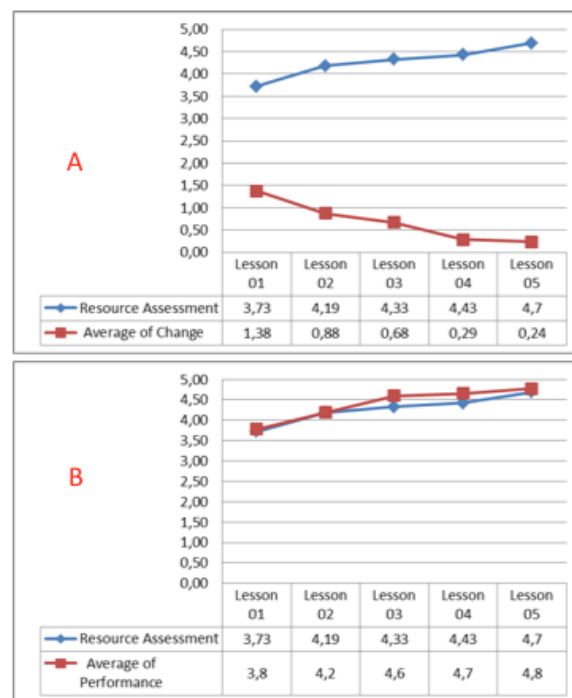


Figure 6.18. Relation between resource assessment and (average of change and performance average); “y” axis corresponds to the score.

Section (B) of Figure 6.18 shows the relationship between evaluations of learning resources and student performance in each lesson. When recommendations at each level of the pedagogical strategy are adapted to the profile of the student, an improvement in the student performance in the next lesson is observed.

The student’s performance was used as metrics in order to evaluate the effectiveness of FUNPRO, see Table 6.5.

Table 6.5. Students' pretest and posttest mean and standard deviation (sd).

	Pretest	Posttest	Gain
<i>Group-CG</i>			
Mean	3.69	4.11	0.42
sd	0.51	0.54	0.37
<i>Group-EG</i>			
Mean	3.68	4.65	0.97
sd	0.55	0.43	0.51

In the experimental group (group-EG) a significant improvement was observed in the pre-test and the post-test, because with a confidence level of 95% ($\alpha=0.05$); the T-value (t) was 3.9213 and the P-Value (p) was 0.000134. The result is significant at $p < \alpha$ ($0.000134 < 0.05$). This implies that the use of metacognition in the system has a significant effect on student performance.

6.2.2 Validation of metamemory in FUNPRO

Since the primary purpose of the metamemory in FUNPRO is to monitor and control failures in information retrieval process, then the reasoning failures dimension was used as performance metrics of the metacognitive capacity of the system. The metric represents retrieval performance (Ghetti, Lyons, Lazzarin, & Cornoldi, 2008) on the number of available resources that were recommended for the lesson. A resource available is one that can be deployed in a lesson, Table 6.6 provides a description of the metric for performance evaluation of ITS with metamemory functions.

Table 6.6. Performance metrics used for metamemory

Metric	Description
ART	% of available resources in retrieval
URT	% of unavailable resources in retrieval

6.2.2.1 Process

For validation, 50 student profiles with random assignment of learning styles were generated. Then 400 educational resources profiles were generated, 20 educational resources for each one of the 20 pedagogical tactics supported by FUNPRO. For each student profile, a recommendation of learning resources is required for the lesson, based on the learning style was generated.

The simulation of the recommendation process was conducted in eight sessions. In each session the number of unavailable resources in the resource base was gradually increased,

see Table 6.7 for details. Finally, each session was repeated five times to observe the behavior of the meta-level.

Table 6.7. Session configuration

Session	# of Students	# of resources	# available	# unavailable
1	50	400	360	40
2	50	400	320	80
3	50	400	280	120
4	50	400	240	160
5	50	400	200	200
6	50	400	160	240
7	50	400	120	280
8	50	400	80	320

6.2.2.2 Data analysis and discussion

Figure 6.19 shows the results of the comparison between the performance of FUNPRO ~~without~~ using metamemory and using metamemory.

Figure 6.19 in Section A shows the results obtained in the 8 sessions without the implementation of metamemory in FUNPRO. In this case the average of ART was 71% and the average of URT was 29%. It can be seen that the performance of FUNPRO decreases when the number of unavailable resources in the resource base increases.

In Figure 6.19, section B shows the results obtained in the 8 sessions with the implementation of metamemory in FUNPRO. In this case the average of ART was 98% and the average of URT was 2%. In the worst scenario depicted in session 8 FUNPRO shows an average yield of 94% with respect to the number of recommendations that contain available resources.

It is noted that when including metamemory, FUNPRO shows a low sensitivity to the progressive increase of unavailable resources in the resource base. This means that FUNPRO can adapt to such situations because it is able to select the appropriate search strategy in the event of failures in the information retrieval. Thus, when performing information retrieval based on the student's learning styles and the available learning resources, then FUNPRO excludes the resources that are not available for future searches. Afterward, FUNPRO presents the excluded resources to the system manager (the teacher) in order to review the cause of the problem. If the system manager solves the problem then the resource becomes available for future searches.

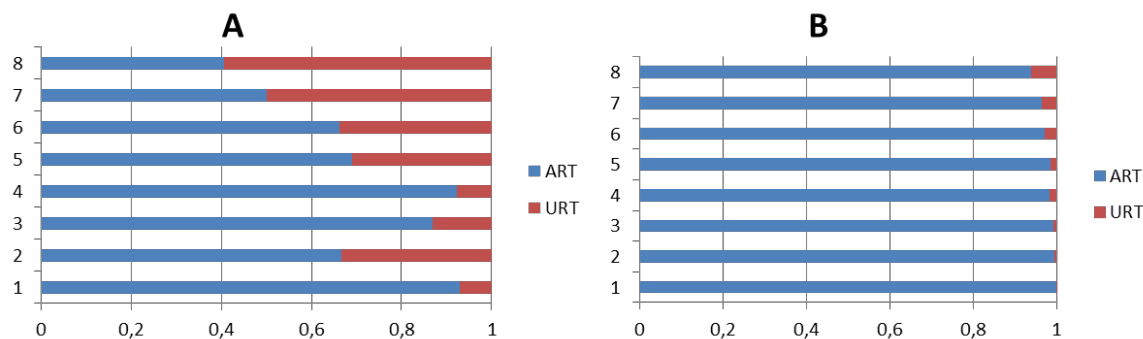


Figure 6.19. Comparison between retrieval rates in FUNPRO. Section A shows the performance of FUNPRO without using metamemory. Section B shows the performance of FUNPRO using metamemory.

The results obtained in the experimental tests show that metamemory in FUNPRO are able to make adaptations in the search strategies. Adaptations are based on changes in the constraints of information retrieval and allow the system to recognize and prevent failures in the recommendations. Therefore, metamemory increases the robustness in terms of failure tolerance in information retrieval from LTM.

6.3 Conclusions of the chapter

An prototype of ITS for teaching introduction to programming (FUNPRO) was presented in this chapter. FUNPRO is the result of the fifth specific objective, which corresponds to building a prototype of ITS and its application in an educational environment. FUNPRO has an architecture based on two layers called object-level and meta-level according to MPPSM metamodel. The object-level contains a multi-level pedagogical model for personalization of pedagogical strategies. The main elements of the multi-level pedagogical model are the learning theories, pedagogic strategies and pedagogic tactics. The meta-level in FUNPRO supports self-regulation and metamemory and contains metacognitive mechanisms such as introspective monitoring and control of object-level. FUNPRO uses self-regulation to monitor and control the processes of reasoning at object-level and metamemory for the adaption to changes in the constraints of information retrieval tasks from LTM.

In FUNPRO, a MRP related to instructional planning was selected for evaluation. The instructional planning task includes a subtask for recommendation of learning resources identified as `playResource`.

The results of the experimental tests showed that multi-level pedagogical model enhanced with metacognition allows a dynamic adaptation of the pedagogical strategy to the profile of each student. Adaptations in each level of the model influence the improvement of student performance in the following lessons. FUNPRO's performance using metamemory was superior in terms of available retrievals in comparison to the

performance of FUNPRO without metamemory. The evidence found in data generated in the tests showed that the implementation of metamemory is a valid tool for improving the process of information retrieval from LTM in ITS.

7 EVALUATION

In this chapter the answers to the research questions formulated in this doctoral thesis likewise the contributions of this research and published articles are presented.

7.1 Answers to research questions

In this section the research questions raised in Chapter 1 are answered. Answers are built based on the contributions and developments made in the course of the thesis.

According to the identified problem, the following research question was formulated:

RQ. How to design a metamodel for personalized adaptation of pedagogical strategies in ITS with integration of self-regulation and metamemory?

The Metamodel for Personalization of Pedagogical Strategies in ITS using Metacognition (MPPSM) was developed using the following steps:

(i) Implementation of metamodeling technique based on FAML for creating a metamodel for personalization of pedagogical strategies in ITS.

A 6-step metamodeling process adapted from FAML (*Beydoun et al., 2009*) was used to create a metamodel for personalization of pedagogical strategies in ITS called METAGOGIC. Metamodeling is a technique promoted by the Object Management Group (OMG) (OMG, 2013) with the goal to automate the process of model generation in software engineering. The adaptations in the methodology of metamodeling with respect to FAML include: (i) addition of step 0 for the collection of pedagogical models; (ii) inclusion of the task, generalization of concepts, in step 5; and (iii) inclusion of validation techniques in step 6.

The 6-step metamodeling process is a guide that contains detailed instructions on the tasks and processes performed at each stage of metamodeling, see (Caro et al., 2014) for more details. The goal of each step is as follows:

- Step 0: Identifying sources of information and collection of pedagogical models in ITS.
- Step 1: Classification (into sets) of pedagogical models according to the type of pedagogical features.
- Step 2: Extraction of concepts related to pedagogical strategies in each set created in step 1.
- Step 3: Selection of the concepts commonly used in the models.
- Step 4: Classification of the concepts selected in step 3.
- Step 5: Identification of relationships between selected concepts.
- Step 6: Creating the metamodel of personalization of pedagogical strategies based on steps 4 and 5.

(ii) Implementation of metamodeling technique based on FAML for creating a metamodel for metacognition support in Intelligent Systems.

Similar to previous step, a metamodeling process adapted from FAML (*Beydoun et al., 2009*) was used to create a metamodel for metacognition support in intelligent systems called MISM.

The 6-step metamodeling process used in this step is as follows:

- Step 0: Identifying sources of information and collection of metacognitive models.
- Step 1: Classification (into sets) of metacognitive models according to the type of metacognition.
- Step 2: Extraction of concepts related to metacognition in each set created in step 1.
- Step 3: Selection of the concepts commonly used in the models.
- Step 4: Classification of the concepts selected in step 3.
- Step 5: Identification of relationships between selected concepts.
- Step 6: Creating the metacognition metamodel based on steps 4 and 5.

(iii) Integration of pedagogical metamodel (METAGOGIC) with metacognitive metamodel (MISM).

The output of integration process is a MOF-based metamodel for personalization of pedagogical strategies using metamemory and self-regulation in ITS called MPPSM, which is the main objective of this thesis.

The MPPSM metamodel consists of the integration of MISM and METAGOGIC metamodels:

- MISM metamodel represents the meta-level and contains all the necessary elements to support metacognitive processes related to self-regulation and metamemory in an intelligent system.
- METAGOGIC metamodel represents the object-level and contains all the necessary elements to model pedagogical strategies in an ITS.

MPPSM is divided into three main packages: `metacore`, `metagogic` and `mism`. A package in MPPSM is a mechanism for grouping related metamodel elements together in order to manage complexity and facilitate the reuse. The `mppsm.metacore` contains fundamental metamodel *classes* needed for the integration of metacognition and pedagogical strategies in MPPSM.

(iv) Creating a graphical tool for generation of metacognitive models based on MPPSM metamodel.

A DSVL called M++ with a central core based on MPPSM was created. M++ has approximately 20 tools for modeling metacognitive systems supporting introspective monitoring and meta-level control. M++ allows the generation of metacognitive diagrams in a visual editor named MetaThink.

The MetaThink tool has been developed with the aim of supporting the modeling of metacognitive functions in ITS commented in previous sections.

MetaThink provides the fundamental infrastructure and components for the generation of metacognitive diagrams in a visual editor based on MPPSM *metamodel*. MetaThink has been developed using the plugins in the Eclipse Modeling Project. Specifically, MetaThink tool has been implemented as an Eclipse plug-in using SIRIUS and ECORE Frameworks.

(v) Validation of MPPSM metamodel by using a prototype of ITS

An ITS called FUNPRO was developed for validation of the MPPSM metamodel. FUNPRO (*FUNdamentos de PROgramación*) is a prototype of ITS that aims to provide personalized instruction in the subject of *Introduction to Programming*.

The general architecture of FUNPRO is based on two layers called object-level and meta-level. The object-level and the meta-level are designed according to MPPSM metamodel. The object-level has architecture consistent with the `mppsm.metagogic` package, while the meta-level is designed based on the `mppsm.mism` package.

A practical experiment was conducted in order to verify the performance of the metacognitive mechanism of self-regulation in the process personalization of pedagogical strategies with respect to the preferences and profiles of students using FUNPRO.

The experiment was a comparison between two groups of students. A first group of 22 students who used FUNPRO with metacognitive module enabled (experimental group-EG) in relation to a second group of 22 students who used FUNPRO with metacognitive module disabled (control group-CG). This validation can be classified as a quasi-experiment, because the sample subjects were not chosen randomly.

The answers to each of the questions that arose from the research question are presented below:

- **SRQ1.** Which should be the specifications of a pedagogical model with properties and methods for improving processes related to personalized adaptation of pedagogical strategies in ITS?

The pedagogical model is multi-level to enrich the possibilities of personalization of pedagogical strategies. The pedagogical strategy is personalized at each level according to the profile of each student. The following five abstraction levels compose the proposed pedagogic model: Theory level, Method level, Tactic level, Activity level and Resource level. Each level of the pedagogical model is represented by ontologies.

- The proposed model supports two types of educational theories: behaviorism and constructivism. The characteristics of the behaviorism theory supported by the multilevel model are: linear navigation among contents; immediate reinforcement and organization of content for levels with prerequisites. Also,

the multilevel model supports the followings constructivist features: free navigation among content, content organization with minimal and necessary prerequisites, formative assessment, and activities for active student participation.

- A teaching method comprises the principles that imply an orderly logical arrangement of tactics and activities used in lessons of a course.
 - Pedagogic Tactics are composed of actions and resources, which are used in the interaction with the student.
 - The components of the lesson are the sections in which the lesson activities are organized. The pedagogical model suggests that a lesson is structured by six sections: introduction, definition, explanation, example, activity and evaluation
 - Learning resources are digital objects such as images, animations, simulations, web pages, and more. Learning resources are the carriers of the content of the lesson and have different formats.
- *SRQ2. What kind of structural properties of meta-cognitive models can be used for integration of metamemory and self-regulation in processes related to personalization of pedagogical strategies in ITS?*

The MISM and METAGOGIC metamodels share a common package called `metacore` but with some differences in the amount and types of concepts according to the nature of each metamodel. The concepts and relationships common to MISM and METAGOGIC were used to create a common package allowing integration of the metamodels. The `mppsm.metacore` contains fundamental metamodel *classes* needed by the other packages. Following the *classes* that constitute the `mppsm.metacore` package in MPPSM are listed.

- Action, BasicElement, CognitiveTask, Error, FunctionalElement, Goal, Level, MetacognitiveTsk, MetaElement, Meta-Level, MetareasoningTask, Object-Level, Plan, Profile, ReasoningTask, Strategy, StructuralElement, Task, Trace.
- *SRQ3. What MDA techniques are necessary for designing a metamodel containing the specifications required for the modeling of personalized adaptation of pedagogical strategies using metacognition in ITS?*
- Metamodeling is the analysis and the development of abstract schemes, rules and restrictions applicable to modeling process of specific types of problems in software engineering.
 - MDA standars used for the development of MPPSM were MOF, OCL and UML.

- Definition of constraints was specified with OCL.
- Mapping approach (Transformations). The mapping is used to realize transformation of instances of the mapped models. The MPPSM metamodel has specifications of endogenous and exogenous mapping.
In this case endogenous mapping is used to the creation of a model in M_1 layer in which each model element of M_1 corresponds to one metamodel element of M_2 layer. The exogenous mapping system that has been integrated in this work consists of a series of transformation from MPPSM to a Relational Database Schema (RDBS). The transformations to database schemas were selected because databases are a component widely used in the design of ITS.
- **SRQ4.** *What are the components and specifications of a MDA-based metamodel that allows the creation of personalized adaptation models of pedagogical strategies by using metacognition in ITS?*

The architecture of the MPPSM metamodel is organized into four levels according to the MOF standard:

- **Meta-MetaModel Level (M_3).** This level comprises meta-metamodel (MOF 2.0) that is used for the design of the MPPSM metamodel (M_2).
- **Metamodel Level (M_2).** The MPPSM metamodel is placed at the M_2 -level in the MOF metamodeling framework. Therefore, a Model that is positioned at the M_1 -level can be modeled by the metamodel. MPPSM Metamodel is specified using MOF standard and implemented in EMF.
- **Model Level (M_1).** This level contains the conceptual models of ITS that are implemented by designers according to the metamodel specified at M_2 level. A MPPSM-based model (M_1 level) is a Metacognitive Model for monitoring and controlling of reasoning failures in ITS.

In the MOF metamodeling framework, the derivation of a model from its metamodel is called a 'conformance.' Through the conformance process, a realization of concept in the MPPSM Metamodel in a new instance (object) in the Model at the M_1 level can be achieved.

- **User Model Level (M_0).** The User Model at the M_0 -level is the target model that is the aim of the MPPSM Metamodel. The derived target model represents an ITS in the real-world. In MOF, the domain concept used in a metamodel is presented as a Class. The data for a Class is presented as an Object. As such, the data for the Object are in turn presented as an Instance in User Model. End-Users manipulate real data using ITS applications generated by a modeling framework from M_1 , i.e. users can create and use models of entities from real world (M_0), using the conceptual model

(M₁).

- **SRQ5.** *Which indicators should be taken into account by a prototype to validate the metamodel designed for generating personalized adaptation models of pedagogical strategies by using metacognition in ITS?*

Since the primary purpose of the metacognition in FUNPRO is to monitor and control failures in reasoning process then the reasoning failures dimension was used as performance metrics of the metacognitive capacity of the system.

- The performance metrics used to measure the use of self-regulation in the personalization of pedagogical strategies were: (i) the average of changes made over the pedagogical strategies recommended at each level of the pedagogical model and; (ii) the relationship observed between the evaluation made by students to learning resources and the changes made by the system to the pedagogical strategy.
- The metrics used to measure the use of metamemory in the personalization of pedagogical strategies was the retrieval performance on the number of available resources that were recommended for a lesson.

7.2 Contributions of the thesis

The main contribution of this thesis was to generate knowledge leading to the construction of a MOF-based metamodel for personalization of pedagogical strategies using computational metacognition in ITS called MPPSM.

The MPPSM metamodel provides the conceptual support necessary to design models for the personalization of pedagogical strategies integrating self-regulation and metamemory in ITS in an consistent way.

MPPSM metamodel avoids the development of specific tools for the design of each new kind of metacognitive capability required for an ITS because it has a visual modeling tool called MethaThink.

The knowledge generated in this thesis has scientific quality, is original and unpublished. The knowledge is structured and based on a rigorous methodology that overcame the borders of current knowledge in designing metacognitive systems applied to education. The metamodel resulting from this research constitutes a significant contribution to advance in the field of AI applied in Education.

Other contributions resulting from the development of this thesis are:

- The METAGOGIC metamodel for pedagogic strategy modeling in ITS. METAGOGIC contains concepts and relationships that are present in the

following tasks related to the design of pedagogical strategies: instructional planning, assessment of instruction and advice on learning activities.

- A comprehensive and general purpose metamodel called MISM, which covers and describes a broad range of commonly referenced concepts in metacognitive models in AI.
- The M++ DSVL for modeling metacognition in ITS incorporating introspective monitoring and meta-level control as meta-reasoning mechanisms.
- MetaThink tool provides the fundamental infrastructure and components for the generation of metacognitive diagrams in a visual editor based on *MPPSM* metamodel.
- The ITS prototype called FUNPRO (*FUNdamentos de PROgramación*) that aims to provide personalized instruction in the subject of *Introduction to Programming*. FUNPRO was designed based on *MPPSM* metamodel and it has a general architecture based on two layers called object-level and meta-level.

7.3 Publications

In this section the intellectual production of the thesis is presented.

7.3.1 Articles published in international journals

Caro, M., Josyula, D., Cox, M., & Jiménez, J. (2014). Design and validation of a metamodel for metacognition support in artificial intelligent systems. *Biologically Inspired Cognitive Architectures (BICA)*, 9 (1), 82. doi:10.1016/j.bica.2014.07.002

Caro, M., Josyula, D., & Jiménez, J. (2014). A Formal model for metacognitive reasoning in intelligent systems. *International Journal of Cognitive Informatics and Natural Intelligence (IJCINI)*, 8(3), 70-86. doi:10.4018/IJCINI.2014070105

Caro, M., Josyula, D., Jiménez, J., Kennedy, C., & Cox, M., (2015). A Domain-Specific Visual Language for Modeling Metacognition in Intelligent Systems. *Biologically Inspired Cognitive Architectures (BICA)* (**In press**)

7.3.2 Articles published in national journals

Caro, M., & Jiménez, J. (2013). Analysis of models and metacognitive architectures in intelligent systems. *Dyna*. 80 (180), 50-59.

Caro, M., Josyula, D., Jiménez, J. (2015). Multi-level pedagogical model for personalization of pedagogical strategies in Intelligent Tutoring Systems. *Dyna* (**In press**)

7.3.3 Papers presented at international events

Caro, M., Josyula, D., Cox, M., & Jiménez, J. (2014). MISM: a metamodel of computational metacognition. *In proceeding of BICA 2014 - Symposium on Neural-Symbolic Networks for Cognitive Capacities*. At Massachusetts Institute of Technology (MIT).

Caro, M., & Josyula, D. (2014). A metamemory model for an Intelligent Tutoring System. *In proceeding of VI International Conference of Adaptive and Accessible Virtual Learning Environment (CAVA -2014)*. Monteria, Colombia.

Caro, M., Jimenez, J. & Josyula, D. (2013). Metamemory for Information Retrieval from Long-term Memory in Artificial Cognitive Systems. *In proceeding of 2013 Annual Conference on Advances in Cognitive Systems: Workshop on Metacognition in Situated Agents*. At University of Maryland.

7.3.4 Papers presented at national events

Caro, M., & Jimenez, J. (2014). MOF-based metamodel for pedagogical strategy modeling in Intelligent Tutoring Systems. *In proceeding of 9th Computing Colombian Conference (9CCC)*. doi: 10.1109/ColumbianCC.2014.6955365.

7.3.5 Book Chapters

Giraldo., G., Jimenez, J. & Caro, M., (2013). Ontology-based semantic model for decision-making in teaching practice process. *In Innovative ways of knowledge representation and management*. Sello editorial Universidad de Medellín.

7.4 Conclusions of the chapter

In this chapter the research questions formulated in this doctoral thesis are responded. Similarly, the contributions and intellectual production are described.

It can be said that the initial hypotheses for each of the questions were verified through the generation of satisfactory results and achieving the objectives. It is noteworthy that the main contribution of the thesis to the generation of knowledge was the presentation of the MPPSM metamodel for personalization of pedagogical strategies using metacognition in ITS. Other contributions of this thesis were: MISM metamodel, METAGOGIC metamodel, M++ DSVL, MetaThink modeling tool and FUNPRO ITS.

Likewise, it is significant the number of scientific articles published for the dissemination of research outcomes in various modalities around this thesis.

8 CONCLUSIONS AND FUTURE WORKS

In this chapter the conclusions of this doctoral thesis in Engineering - Systems and Informatics are presented. Finally, open research points and possible improvements are explained.

8.1 Conclusions

Metacognition has been used in AI to increase the level of autonomy of intelligent systems. However the design of systems with metacognitive capabilities is a difficult task due to the number and complexity of processes involved. In this sense, the main contribution of this doctoral thesis was the design and validation of a MOF-based metamodel for the generation of personalized adaptation models of pedagogical strategies integrating metamemory and self-regulation in ITS. The metamodel called MPPSM is located in the M2 layer of the MOF standard and it was presented in UML format for easy understanding.

MPPSM adds precision to metacognitive concepts used in ITS design because it was synthesized from the analysis of 40 metacognitive models and 45 ITS models that exist in the literature. A 6-step metamodeling process adapted from FAML (*Beydoun et al., 2009*) was used to synthesize MPPSM. Adaptations in the methodology of metamodeling with respect to FAML include: (i) addition of step 0 for the collection of pedagogical models; (ii) inclusion of the task, generalization of concepts, in step 5; and (iii) inclusion of validation techniques in step 6. The 6-step metamodeling process is a guide that contains detailed instructions on the tasks and processes performed at each stage of metamodeling and is structured by steps as follows:

- Step 0: Identifying sources of information and collection of pedagogical models in ITS.
- Step 1: Classification (into sets) of pedagogical models according to the type of pedagogical features.
- Step 2: Extraction of concepts related to pedagogical strategies in each set created in step 1.
- Step 3: Selection of the concepts commonly used in the models.
- Step 4: Classification of the concepts selected in step 3.
- Step 5: Identification of relationships between selected concepts.
- Step 6: Creation of the metamodel of personalization of pedagogical strategies based on steps 4 and 5.

The metamodel is organized into three main packages called `mppsm.metacore`, `mppsm.mism` and `mppsm.metagogic`. The `metacore` package facilitates the reuse of elements in different metacognitive components because: (i) it allows reducing the complexity of MPPSM because it groups common classes that are used by other packages;

(ii) it maintains the integration and the reutilization of classes among the different packages that compose MPPSM. The `mppsm.mism` package contains the functionality of the meta-level and abstract description of the object-level into a meta-reasoning loop of an intelligent system. It contains a comprehensive and general set of classes that cover and describe a broad range of commonly referenced concepts in metacognitive models in the area of AI. The `mppsm.metagodic` package contains the schema of the object-level domain in an ITS and it has a central core based on the following classes: `Context`, `PedagogicalApproach` and `InstructionalActivity`. The `Context` contains the general configuration of the pedagogical strategy. The `PedagogicalApproach` addresses the strategy from learning theories and teaching methods. `InstructionalActivity` defines the most appropriate pedagogic tactics to address the contents of the lesson. The structure of the pedagogical strategy allows generating models with three levels of adaptation.

MPPSM facilitates the design of ITS with metacognitive functions because it acts as a guide with predesigned components that are commonly used in the computational metacognition and ITS scientific community. In this regard, a DSL called M++ with a central core based on MPPSM was developed. M++ has approximately 20 tools for modeling metacognitive systems supporting introspective monitoring and meta-level control. M++ allows the generation of metacognitive diagrams in a visual editor named MetaThink. MetaThink provides the fundamental infrastructure and components for the generation of metacognitive diagrams in a visual editor developed as an Eclipse plugin that supports rapid prototyping of metacognitive architectures by allowing candidate systems to be built, tested and revised in an automated way.

The results given in the experimental study to validate M++ demonstrate that it is a language that has a useful notation to help designers in the process of modeling metacognitive components in intelligent systems.

The implementation of endogenous and exogenous transformations in MPPSM enables the automation of metacognitive-ITS prototyping process. Endogenous transformations allow the generation of pedagogical strategies models at M_1 layer based on the specifications of M_2 layer in an automated way. Exogenous transformation establishes *one-to-one* relations between elements from the source model (MPPSM) to elements of the target model (RDBS). Exogenous transformations facilitate the design of MPPSM-based systems because allows to designers the generation of database schema in an automated way. The tracing validation of consistency of the models generated with M++ shown that the concepts of the metamodel are actually usable by designers of intelligent systems with metacognitive support.

A prototype of ITS called FUNPRO was developed for validation of the performance of metacognitive mechanism of MPPSM in the process personalization of pedagogical strategies with respect to the preferences and profiles of real students. FUNPRO has an architecture based on two layers called object-level and meta-level. The object-level and the meta-level are designed according to MPPSM metamodel. The object-level contains a multi-

level pedagogical model for personalization of pedagogical strategies. The main elements of the multi-level pedagogical model are the learning theories, pedagogic strategies and pedagogic tactics. The meta-level in FUNPRO supports self-regulation and metamemory and contains metacognitive mechanisms such as introspective monitoring and control of object-level. FUNPRO uses self-regulation to monitor and control the processes of reasoning at object-level and metamemory for the adaptation to changes in the constraints of information retrieval tasks from LTM. The results of the experimental tests show that multi-level pedagogical model enhanced with metacognition allows dynamic adaptation of the pedagogical strategy to the profile of each student. Adaptations in each level of the model influence the improvement of student performance in the following lessons.

Finally it can be concluded that the objectives proposed in this thesis were fully achieved generating contributions that extend the frontiers of knowledge (See concluding section of each chapter). Similarly, the results obtained in this thesis were validated in national and international conferences and journals.

8.2 Future works

The proposed work is an excellent starting point for improving the teaching / learning in computer-mediated virtual education. The idea of integrating metacognition to enhance the personalization of pedagogical strategies in ITS is that in the future various research can be done in several aspects described below. These researches may be subjects of master's and doctoral theses.

8.2.1 MPPSM metamodel

- The next step in relation to MPPSM is to create a cognitive architecture with dual cycle of reasoning for designing metacognitive-ITS.

8.2.2 M++ and MetaThink

- The next objective is to adapt M++ and support tools for designing intelligent agents in non-pedagogical domains.
- In future work, the usability and speed of prototyping by users of different backgrounds (e.g. software engineering, cognitive science, psychology) will be investigated.
- To compare M++ with other visual frameworks such as GAIA (*Rugaber, Goel, & Martie, 2013*) for usability and simplicity improvement.
- It is recommended as future work to design a version of MetaThink for mobile devices.

8.2.3 FUNPRO

FUNPRO has a good performance using metacognition, but can be improved in the following aspects:

- A safety mechanism is required because the system can be used on the web, preventing access of unauthorized personnel.
- An important question for future work is to investigate the effect of an ITS with autonomous metacognition (such as FUNPRO) on student learning and on student metacognition.
- To design a version of FUNPRO for mobile devices.

9 REFERENCES

- Aamodt, A. (1994). Knowledge-Intensive Case-Based Reasoning and Intelligent Tutoring. *AI Communications*, Vol.7(1), 35–39.
- Abbasi, Z., & Abbasi, M. A. (2008). Reinforcement Distribution in a Team of Cooperative Q-learning Agents. *2008 Ninth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing*, 154–160. doi:10.1109/SNPD.2008.154
- Abrahão, S., Insfran, E., Carsí, J. A., & Genero, M. (2011). Evaluating requirements modeling methods based on user perceptions: A family of experiments. *Information Sciences*, 181, 3356–3378. doi:10.1016/j.ins.2011.04.005
- Aguilar, R., Muñoz, V., González, E. J., Noda, M., Bruno, a., & Moreno, L. (2011). Fuzzy and MultiAgent Instructional Planner for an Intelligent Tutorial System. *Applied Soft Computing*, 11(2), 2142–2150. doi:10.1016/j.asoc.2010.07.013
- Aleven, V., Kay, J., & Mostow, J. (2010). Intelligent Tutoring Systems. *Intelligent Tutoring Systems*, 228(4698), 1–457. doi:10.1007/978-3-642-13388-6
- Aleven, V., McLaren, B., Koedinger, K., & Roll, I. (2006). Toward Meta-Cognitive Tutoring : A Model of Help-Seeking with a Cognitive Tutor. *International Journal of Artificial Intelligence in Education*, 16(1), 101–128.
- Aleven, V., Roll, I., McLaren, B., Ryu, E. J., & Koedinger, K. (2005). An architecture to combine meta-cognitive and cognitive tutoring : Pilot testing the Help Tutor. *Human-Computer Interaction*, (Aied), 18–22.
- Alonso, J. B., Arnold, K. C., & Havasi, C. (2010). Envisioning a Robust , Scalable Metacognitive Architecture Built on Dimensionality Reduction. *AAAI Workshop: Metacognition for Robust Social Systems 2010: Atlanta, Georgia, USA*.
- Amorim, R., Lama, M., Sánchez, E., Riera, A., & Vila, X. (2006). A Learning Design Ontology based on the IMS Specification. *Educational Technology & Society*, 9(1), 38–57.
- Anderson, M., Fults, S., Josyula, D., Oates, T., Perlis, D., & Schmill, M. (2008). A Self-Help Guide for Autonomous Systems. *AI Magazine*, 29(2), 67–76.
- Anderson, M., Oates, T., Chong, W., & Perlis, D. (2006). The metacognitive loop I: Enhancing reinforcement learning with metacognitive monitoring and control for improved perturbation tolerance. *Journal of Experimental & Theoretical Artificial Intelligence*, 18(3), 387–411. doi:10.1080/09528130600926066
- Anderson, M., & Perlis, D. (2004). Logic, Self-awareness and Self- improvement: the Metacognitive Loop and the Problem of Brittleness. *Journal of Logic Computation*, 14(04), 1–20.
- Arias, F., Jiménez, J., & Ovalle, D. (2009). Instructional planning model in intelligent tutorials systems. *Revista Avances En Sistemas E Informática*, 6(1).

- Ausubel, D. P. (1978). In Defense of Advance Organizers: A Reply to the Critics. *Review of Educational Research*, 48(2), 251–257. doi:10.3102/00346543048002251
- Azevedo, R. (2002). Beyond intelligent tutoring systems: Using computers as METAcognitive tools to enhance learning ? *Instructional Science*, 30(1), 31–45.
- Azevedo, R., Witherspoon, A., Chauncey, A., Burkett, C., & Fike, A. (2009). MetaTutor : A MetaCognitive Tool for Enhancing Self-Regulated Learning MetaCognitive Tools for Enhancing learning, 14–19.
- Barros, H., Silva, A., Costa, E., Ig, B., Holanda, O., & Sales, L. (2011). Engineering Applications of Artificial Intelligence Steps , techniques , and technologies for the development of intelligent applications based on Semantic Web Services: A case study in e-learning systems. *Engineering Applications of Artificial Intelligence*, 24(8), 1355–1367. doi:10.1016/j.engappai.2011.05.007
- Ben Ammar, M., Neji, M., Alimi, A. M., & Gouardères, G. (2010). The Affective Tutoring System. *Expert Systems with Applications*, 37(4), 3013–3023. doi:10.1016/j.eswa.2009.09.031
- Benes, V. (2004). Metacognition in Intelligent Systems. *Intelligence*.
- Benjamin, M. R., Schmidt, H., Newman, P. M., & Leonard, J. J. (2013). Marine Robot Autonomy. In *Marine Robot Autonomy* (pp. 47–90). Springer Verlag. doi:10.1007/978-1-4614-5659-9
- Beydoun, G., Low, G., Henderson-Sellers, B., Mouratidis, H., Gonzalez-perez, C., Gomez-sanz, J. J., & Pavo, J. (2009). FAML : A Generic Metamodel for MAS Development. *IEEE Transactions on Software Engineering*, 35(6), 841–863.
- Bezerra, C. (2012). Pedagogical Model Based on Semantic Web Rule Language. In *2012 12th International Conference on Computational Science and Its Applications* (pp. 125–129). IEEE. doi:10.1109/ICCSA.2012.31
- Bezivin, J., Buttner, F., Gogolla, M., Jouault, F., Kurtev, I., & Lindow, A. (2006). Model transformations? Transformation models! *Model Driven Engineering Languages and Systems, Proceedings*, 4199, 440–453. doi:10.1007/11880240_31
- Bhat, G., & Kolodner, J. (2009). A case-based system to aid cognition and meta-cognition in a design-based learning environment. In *AAAI Fall Symposium* (pp. 26–31). Retrieved from <http://www.aaai.org>
- Bittencourt, I., Costa, E., Almeida, H., Fonseca, B., & Maia, G. (2007). Towards an Ontology-based Framework for Building Multiagent Intelligent Tutoring Systems. *III Workshop on Software Engineering for Agent-Oriented Systems SEAS 2007*, 53–64.
- Bittencourt, I., Costa, E., Silva, M., & Soares, E. (2009). A computational model for developing semantic web-based educational systems. *Knowledge-Based Systems*, 22(4), 302–315. doi:10.1016/j.knosys.2009.02.012
- Bonarini, A., Lazaric, A., Montrone, F., & Restelli, M. (2009). Reinforcement distribution in fuzzy Q-learning. *Fuzzy Sets and Systems*, 160(10), 1420–1443.

doi:10.1016/j.fss.2008.11.026

- Bonnet, A. (1985). *Artificial intelligence: promise and performance* (First Edit.). London: Prentice Hall.
- Bragança, A., & Machado, R. J. (2008). Transformation Patterns for Multi-staged Model Driven Software Development. *2008 12th International Software Product Line Conference*, 329–338. doi:10.1109/SPLC.2008.41
- Bravo, C., Joolingen, W. R. Van, & Jong, T. De. (2009). Computers & Education Using Co-Lab to build System Dynamics models : Students ' actions and on-line tutorial advice. *Computers & Education*, 53(2), 243–251. doi:10.1016/j.compedu.2009.02.005
- Brooks, J., & Brooks, M. (1993). *In Search of Understanding: The Case for Constructivist Classrooms*. Association for Supervision and Curriculum Development. Alexandria, VA. Retrieved from http://books.google.com/books?hl=en&lr=&id=9W_VB5TjxxoC&pgis=1\nhttps://books.google.co.uk/books?hl=en&lr=&id=9W_VB5TjxxoC&oi=fnd&pg=PR7&dq=+In+search+of+understanding:+The+case+for+constructivist+classrooms.+Association+for+Supervision+and+Curriculum+Dev
- Brooks, J., & Brooks, M. (1999). The courage to be constructivist. *Educational Leadership*, 57, 1–10. Retrieved from <http://www.ascd.org/publications/educational-leadership/nov99/vol57/num03/The-Courage-to-Be-Constructivist.aspx>
- Brusilovsky, P. (2003). Developing adaptive educational hypermedia systems: From design models to authoring tools. In *Authoring Tools for Advanced Technology Learning Environments* (pp. 377–409). Springer Netherlands. doi:10.1007/978-94-017-0819-7_13
- Burns, H. L., & Capps, C. G. (1988). Foundations of intelligent tutoring systems : an introduction. In *Foundations of Intelligent Tutoring Systems* (pp. 1–19). Retrieved from <http://hal.archives-ouvertes.fr/hal-00699852>
- Cabada, R. Z., Barrón Estrada, M. L., & Reyes García, C. A. (2011). EDUCA: A web 2.0 authoring tool for developing adaptive and intelligent tutoring systems using a Kohonen network. *Expert Systems with Applications*, 38(8), 9522–9529. doi:10.1016/j.eswa.2011.01.145
- Cannella, V., Chella, A., & Pirrone, R. (2013). A meta-cognitive architecture for planning in uncertain environments. *Biologically Inspired Cognitive Architectures*, 5, 1–9. doi:10.1016/j.bica.2013.06.001
- Capraro, G., Wicks, M., & Schneible, R. (2010). Metacognition in Radar. In *2nd International Workshop on Cognitive Information Processing Metacognition* (pp. 1–6). IEEE.
- Capuano, N., Marsella, M., & Salerno, S. (2000). ABITS: An agent based Intelligent Tutoring System for distance learning. In *International Workshop on Adaptive and Intelligent Web-Based Education Systems, ITS*. Retrieved from http://www.capuano.biz/papers/ITS_2000.pdf
- Caro, M., Jimenez, J., & Paternina, A. (2012). Architectural modeling of metamemory

- judgment in case-based reasoning systems. *2012 XXXVIII Conferencia Latinoamericana En Informatica (CLEI)*, 1–8. doi:10.1109/CLEI.2012.6427152
- Caro, M., Josyula, D., Cox, M., & Jiménez, J. (2014). Design and validation of a metamodel for metacognition support in artificial intelligent systems. *Biologically Inspired Cognitive Architectures*, 9, 82–104. doi:10.1016/j.bica.2014.07.002
- Caro, M., Josyula, D., & Jiménez, J. (2015). Multi-level pedagogical model for the personalization of pedagogical strategies in Intelligent Tutoring Systems. *Dyna*, 82(194), 185–193. doi:http://dx.doi.org/10.15446/dyna.v82n194.49279
- Caro, M., Josyula, D., Jiménez, J., Kennedy, C., & Cox, M. (2015). A domain-specific visual language for modeling metacognition in intelligent systems. *Biologically Inspired Cognitive Architectures*, 13, 75–90. doi:10.1016/j.bica.2015.06.004
- Caro, M., Toscazo, R., Hernández, F., & David, M. (2009). Diseño de software educativo basado en competencias. *Ciencia E Ingenieria Neogranadina. jun2009, Vol. 19 Issue 1, p71-98. 28p. 4 Black and White Photographs, 10 Diagrams, 12 Charts., 19(1), 71–98. Retrieved from <http://search.ebscohost.com/login.aspx?direct=true&db=fua&AN=46984112&lang=es&site=ehost-live>*
- Celiberto, L., Matsuura, J., de Mantaras, R., & Bianchi, R. (2010). Using Transfer Learning to Speed-Up Reinforcement Learning: A Cased-Based Approach. In *2010 Latin American Robotics Symposium and Intelligent Robotics Meeting* (pp. 55–60). Ieee. doi:10.1109/LARS.2010.24
- Chan, T. (1992). Curriculum tree: a knowledge-based architecture for intelligent tutoring systems. In *Intelligent Tutoring Systems. Lecture Notes in Computer Science. Volume 608* (pp. 140–147). Montréal, Canada: Springer Berlin Heidelberg. doi:10.1007/3-540-55606-0_19
- Chang-long, W. (2009). Quality evaluation of universities undergraduate practice teaching work based on artificial neural network, (1), 393–396. doi:10.1109/CINC.2009.24
- Chen, C., & Huang, T. (2012). Learning in a u-Museum: Developing a context-aware ubiquitous learning environment. *Computers & Education*, 59(3), 873–883. doi:10.1016/j.compedu.2012.04.003
- Chen, H. (2009). Personalized E-learning system with self-regulated learning assisted mechanisms for promoting learning performance. *Expert Systems With Applications*, 36(5), 8816–8829. doi:10.1016/j.eswa.2008.11.026
- Chen, Y., & Chen, Y. (2009). An Ontology-Based Distributed Case-Based Reasoning for Virtual Enterprises. In *International Conference on Complex, Intelligent and Software Intensive Systems*. doi:10.1109/CISIS.2009.23
- Chen, Y., Juang, Y., Feng, K., Chou, C., & Chan, T. (2004). Defining Instructional Plan Meta-Data for a Wireless Technology Enhanced Classroom. In *Proceedings of the IEEE International Conference on Advanced Learning Technologies (ICALT'04)* (pp. 693–695). IEEE Comput. Soc. doi:10.1109/ICALT.2004.1357625

- Cheng, P. (2011). Application of Case Based Reasoning in Plane Geometry Intelligent Tutoring System. *Science And Technology*, (2010), 4369–4373.
- Cheung, B., Hui, L., Zhang, J., & Yiu, S. M. (2003). SmartTutor: An intelligent tutoring system in web-based adult education. *Journal of Systems and Software*, 68(1), 11–25. doi:10.1016/S0164-1212(02)00133-4
- Cheung, K. S., Lam, J., Lau, N., & Shim, C. (2010). Instructional Design Practices for Blended Learning. In *Computational Intelligence and Software Engineering (CiSE), 2010 International Conference on* (pp. 1–4). Ieee. doi:10.1109/CISE.2010.5676762
- Chitforoush, F., Yazdandoost, M., & Ramsin, R. (2007). Methodology Support for the Model Driven Architecture. In *14th Asia-Pacific Software Engineering Conference (APSEC'07)* (pp. 454–461). Ieee. doi:10.1109/ASPEC.2007.58
- Christodoulou, E., & Keravnou, E. T. (1998). Metareasoning and meta-level learning in a hybrid knowledge-based architecture. *Artificial Intelligence in Medicine*, 14(1), 53–81.
- Clayberg, E., & Rubel, D. (2008). *Eclipse Plug-ins*. Addison-Wesley.
- Conati, C. (2000). Toward Computer-Based Support of Meta-Cognitive Skills: a Computational Framework to Coach Self-Explanation. *International Journal of Artificial Intelligence in Education*, 398–415.
- Corbett, A., Kauffman, L., Maclaren, B., Wagner, A., & Jones, E. (2010). A Cognitive Tutor for Genetics Problem Solving: Learning Gains and Student Modeling. *Journal of Educational Computing Research*, 42(2), 219–239. doi:10.2190/EC.42.2.e
- Cox, M. (1995). Metacognition in Computation : A selected history. *Memory*.
- Cox, M. (1996). *Introspective multistrategy learning: Constructing a learning strategy under reasoning failure* (Tech. Rep. No. GIT-CC-96-06). Georgia Institute of Technology.
- Cox, M. (1997). An Explicit Representation of Reasoning Failures. In S. B. Heidelberg (Ed.), *Case-Based Reasoning Research and Development* (pp. 211–222).
- Cox, M. (2005, December). Metacognition in computation: A selected research review. *Artificial Intelligence*. doi:10.1016/j.artint.2005.10.009
- Cox, M. (2007). Perpetual Self-Aware Cognitive Agents. *AI Magazine*, (2002), 32–51. Retrieved from <http://www.aaai.org/ojs/index.php/aimagazine/article/view/2027/1920>
- Cox, M., Oates, T., & Perlis, D. (2011). Toward an Integrated Metacognitive Infrastructure. *2011 AAAI Fall Symposium*, 74–81.
- Cox, M., & Raja, A. (2012). Metareasoning: An introduction. In Cox and Raja (Ed.), *In Metareasoning: Thinking about thinking* (pp. 1–23). Cambridge. MA: MIT.
- Cox, M., & Ram, A. (1999). Introspective multistrategy learning: On the construction of learning strategies. *Artificial Intelligence*, 112(1), 1–55. doi:10.1016/S0004-3702(99)00047-8

- Crowley, R. S., & Medvedeva, O. (2006). An intelligent tutoring system for visual classification problem solving. *Artificial Intelligence in Medicine*, 36(1), 85–117. doi:10.1016/j.artmed.2005.01.005
- D'Mello, S., Olney, A., Williams, C., & Hays, P. (2012). Gaze tutor: A gaze-reactive intelligent tutoring system. *International Journal of Human-Computer Studies*, 70(5), 377–398. doi:10.1016/j.ijhcs.2012.01.004
- Dannenbauer, D., Cox, M., Gupta, S., Paisner, M., & Perlis, D. (2014). Toward Meta-level Control of Autonomous Agents. In *Procedia Computer Science. 5th Annual International Conference on Biologically Inspired Cognitive Architectures, 2014 BICA* (Vol. 41, pp. 226–232). Elsevier Masson SAS. doi:10.1016/j.procs.2014.11.107
- de Bruin, A. B. H., Thiede, K. W., Camp, G., & Redford, J. (2011). Generating keywords improves metacomprehension and self-regulation in elementary and middle school children. *Journal of Experimental Child Psychology*, 109(3), 294–310. doi:10.1016/j.jecp.2011.02.005
- Demirbas, O. O., & Demirkan, H. (2007). Learning styles of design students and the relationship of academic performance and gender in design education. *Learning and Instruction*, 17(3), 345–359. doi:10.1016/j.learninstruc.2007.02.007
- Dick, W., Carey, L., & Carey, J. (2005). *The systematic design of instruction*. (Pearson, Ed.) (6th ed.).
- Ding, J., Liu, H., & Deng, A. (2010). Application of Bayesian Network Knowledge Reasoning Based on CBR in ITS. *2010 Third International Joint Conference on Computational Science and Optimization*. doi:10.1109/CSO.2010.113
- Duan, Y., & Ren, H. (2011). Building Students' Models Based on an Enhanced Concept-Effect Relationship in Intelligent Tutoring Systems. *Science And Technology*, 5318–5321.
- Elorriaga, J., & Fernandez-Castro, I. (2000). Evaluation of a hybrid self-improving instructional planner. In *Proceedings - International Workshop on Advanced Learning Technologies: Advanced Learning Technology: Design and Development Issues, IWALT 2000* (pp. 133–136). doi:10.1109/IWALT.2000.890587
- Erche, M., Wagner, M., & Hein, C. (2007). Mapping Visual Notations to MOF Compliant Models with QVT Relations. *Applied Computing 2007, Vol 1 and 2*, 1037–1038. doi:10.1145/1244002.1244228
- Escudero, H., & Fuentes, R. (2010). Exchanging courses between different Intelligent Tutoring Systems: A generic course generation authoring tool. *Knowledge-Based Systems*, 23(8), 864–874. doi:10.1016/j.knosys.2010.05.011
- Espinosa, M. L., Sánchez, N. M., Valdivia, Z. G., & Pérez, R. B. (2007). Concept Maps Combined with Case-Based Reasoning to Elaborate Intelligent Teaching-Learning Systems. *Computer*, 205–210. doi:10.1109/ISDA.2007.33
- Ezechil, L., & Coman, P. (2012). Analysis of didacticians' psycho-pedagogical competences. *Procedia - Social and Behavioral Sciences*, 33, 233–237.

doi:10.1016/j.sbspro.2012.01.118

- Felder, R. M., & Henriques, E. R. (1995). Learning and Teaching Styles In Foreign and Second Language Education. *Foreign Language Annals*, 28(1), 21–31. doi:10.1111/j.1944-9720.1995.tb00767.x
- Feng, Y., Huang, G., Yang, J., & Mei, H. (2006). Traceability between Software Architecture Models, 2–5.
- Flavell, J., & Resnick, L. (1976). Metacognitive Aspects of Problem Solving. In *The Nature of Intelligence* (pp. 231–235).
- Flavell, J., & Wellman, H. (1977). Metamemory. In J. W. H. (Eds. . R. V. Kail, Jr. (Ed.), *Perspectives on the Development of Memory and Cognition* (pp. 3–33). Hillsdale, NJ: Lawrence Erlbaum Associates.
- Fox, S., & Leake, D. (1994). Using Introspective Reasoning to Guide Index Refinement in Case-Based Reasoning. In *In Proceedings of the sixteenth annual conference of the cognitive science society* (pp. 324–329). Lawrence Erlbaum.
- Franklin, S. (2000). Deliberation and Voluntary Action in “ Conscious ” Software Agents. *Neural Network World*, 10, 505–521.
- Gadhiok, M., Amanna, A., Price, M. J., & Reed, J. H. (2011). Metacognition : Enhancing The Performance of a Cognitive Radio. In *2011 IEEE International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support (CogSIMA), Miami Beach, FL* (pp. 198–203).
- Gaeta, M., Mangione, G. R., Orciuoli, F., & Salerno, S. (2011). Metacognitive Learning Environment: a semantic perspective. *Journal of E-Learning and Knowledge Society (English Edition)*, 7(2), 69–80.
- Ganjanasuwan, T., & Sanrach, C. (2006). Multi-agent instructional resource planning. In *2006 IEEE Conference on Cybernetics and Intelligent* (pp. 1–7). Bangkok: IEEE Comput. Soc. doi:10.1109/ICCIS.2006.252338
- Ghetti, S., Lyons, K. E., Lazzarin, F., & Cornoldi, C. (2008). The development of metamemory monitoring during retrieval: the case of memory strength and memory absence. *Journal of Experimental Child Psychology*, 99(3), 157–181. doi:10.1016/j.jecp.2007.11.001
- Gong, H., Zhao, H., Wang, Y., & Sun, G. (2009). Cultivation of Metacognition in the Web-Based Autonomous Learning Environment. *2009 International Conference on Computational Intelligence and Software Engineering*, 1–5. doi:10.1109/CISE.2009.5364683
- Gordon, A. S., Hobbs, J. R., & Cox, M. (2007). Anthropomorphic Self-Models for Metareasoning Agents Self-models in Metareasoning Anthropomorphic Self-Models.
- Graesser, A., Chipman, P., Haynes, B., & Olney, A. (2005). AutoTutor: An Intelligent Tutoring System With Mixed-Initiative Dialogue. *IEEE Transactions on Education*, 48(4), 612–618. doi:10.1109/TE.2005.856149

- Graesser, A., Wiemer-Hastings, K., Wiemer-Hastings, P., & Kreuz, R. (1999). AutoTutor: A simulation of a human tutor. *Cognitive Systems Research*, 1(1), 35–51. doi:10.1016/S1389-0417(99)00005-4
- Graham, C. R. (2011). Theoretical considerations for understanding technological pedagogical content knowledge (TPACK). *Computers & Education*, 57(3), 1953–1960. doi:10.1016/j.compedu.2011.04.010
- Gui-mei, Y. I. N., & Guang-Xing, G. (2010). An Affective Recognition-Based Architecture for Intelligent Learning Environments. *2010 International Conference on Computer Application and System Modeling (ICCA SM 2010)*, (Iccasm), 237–239.
- Gulz, A., & Haake, M. (2006). Design of animated pedagogical agents—A look at their look. *International Journal of Human-Computer Studies*, 64(4), 322–339. doi:10.1016/j.ijhcs.2005.08.006
- Haas, D. F., & Kraft, D. H. (1984). Experimental and quasi-experimental designs for research in information science. *Information Processing & Management*. doi:10.1016/0306-4573(84)90053-0
- Haidarian, H., Dinalankara, W., Fults, S., Wilson, S., Perlis, D., Schmill, M., ... Anderson, M. (2010). The Metacognitive Loop: An Architecture for Building Robust Intelligent Systems. In *AAAI Fall Symposium* (pp. 33–39).
- Haitao, O., Weidong, Z., Wenyan, Z., & Xiaoming, X. (2000). A novel multi-agent Q-learning algorithm in cooperative multi-agent system. *Proceedings of the 3rd World Congress on Intelligent Control and Automation (Cat. No.00EX393)*, 1, 272–276. doi:10.1109/WCICA.2000.859964
- Hayes-Roth, F. (1982). *The handbook of artificial intelligence Volume I. Artificial Intelligence* (Vol. 18). doi:10.1016/0004-3702(82)90027-3
- Heift, T. (2010). Developing an intelligent language tutor. *CALICO Journal*, 27(3), 443–459. Retrieved from <http://journals.sfu.ca/CALICO/index.php/calico/article/view/865>
- Hsu, C.-H., & Juang, C.-F. (2011). Self-Organizing Interval Type-2 Fuzzy Q-learning for reinforcement fuzzy control. *2011 IEEE International Conference on Systems, Man, and Cybernetics*, (1), 2033–2038. doi:10.1109/ICSMC.2011.6083971
- Huang, C.-Y., Chung, W.-C., Chang, C.-J., & Ren, F.-C. (2009). Fuzzy Q-Learning-Based Hybrid ARQ for High Speed Downlink Packet Access. *2009 IEEE 70th Vehicular Technology Conference Fall*, 1–4. doi:10.1109/VETECF.2009.5378870
- Hudlicka, E. (2005). Modeling Interaction Between Metacognition and Emotion in a Cognitive Architecture. In *AAAI Spring Symposium on Metacognition in Computation. Technical Report SS-05-04*. Menlo Park, CA: AAAI Press. 2005. (p. 7).
- Huet, N., & Mariné, C. (1997). Memory strategies and metamemory knowledge under memory demands change in waiters learners. *European Journal of Psychology of Education*, XII(1), 23–35.
- Hwang, K.-S., Lin, H.-Y., Hsu, Y.-P., & Yu, H.-H. (2011). Self-organizing state aggregation

- for architecture design of Q-learning. *Information Sciences*, 181(13), 2813–2822. doi:10.1016/j.ins.2011.02.017
- Iglesias, A., Martínez, P., Aler, R., & Fernández, F. (2009). Reinforcement learning of pedagogical policies in adaptive and intelligent educational systems. *Knowledge-Based Systems*, 22(4), 266–270. doi:10.1016/j.knosys.2009.01.007
- International, O. T. (2003). Eclipse Platform Technical Overview. Retrieved on November, 2003(July 2001), 1–20.
- Irfan, R., & Shaikh, M. U. (2008). Framework for Embedding Tacit Knowledge in Pedagogical Model to Enhance E-Learning. *2008 New Technologies, Mobility and Security*, 1–5. doi:10.1109/NTMS.2008.ECP.48
- Jeremić, Z., Jovanović, J., & Gašević, D. (2012). Student modeling and assessment in intelligent tutoring of software patterns. *Expert Systems with Applications*, 39(1), 210–222. doi:10.1016/j.eswa.2011.07.010
- Jones, J. (1992). Intelligent tutoring systems: the first component of integrated information systems. [Proceedings] *1992 IEEE International Conference on Systems, Man, and Cybernetics*, 531–536. doi:10.1109/ICSMC.1992.271719
- Jones, J. K., & Goel, A. K. (2012). Perceptually grounded self-diagnosis and self-repair of domain knowledge. *Knowledge-Based Systems*, 27, 281–301. doi:http://dx.doi.org/10.1016/j.knosys.2011.09.012
- Josyula, D., Hughes, F., Vadali, H., & Donahue, B. (2009). Modeling emotions for choosing between deliberation and action. *2009 World Congress on Nature and Biologically Inspired Computing, NABIC 2009 - Proceedings*, 782–787. doi:10.1109/NABIC.2009.5393730
- Josyula, D., Vadali, H., Donahue, B., & Hughes, F. (2009). Modeling metacognition for learning in artificial systems. In *2009 World Congress on Nature & Biologically Inspired Computing (NaBIC)* (pp. 1419–1424). Ieee. doi:10.1109/NABIC.2009.5393706
- Jouault, F., Allilaire, F., Bézivin, J., & Kurtev, I. (2008). ATL: A model transformation tool. *Science of Computer Programming*, 72(1-2), 31–39. doi:10.1016/j.scico.2007.08.002
- Jouault, F., & Kurtev, I. (2006). Transforming models with ATL. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (Vol. 3844 LNCS, pp. 128–138). doi:10.1007/11663430_14
- Jozefowicz, J., Staddon, J. E. R., & Cerutti, D. T. (2009). Reinforcement and Metacognition. *Comparative Cognition & Behavior Reviews*, 4, 58 –60. doi:10.3819/ccbr.2009.40007
- Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996). Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research*, 4, 237–285.
- Karampiperis, P., & Sampson, D. (2004). Adaptive instructional planning using ontologies. In *Proceedings - IEEE International Conference on Advanced Learning Technologies, ICALT 2004* (pp. 126–130). doi:10.1109/ICALT.2004.1357388

- Kazi, H., Haddawy, P., & Suebnukarn, S. (2012). Employing UMLS for generating hints in a tutoring system for medical problem-based learning. *Journal of Biomedical Informatics*, 45(3), 557–565. doi:10.1016/j.jbi.2012.02.010
- Keener, M. (2011). Integration of Comprehension and Metacomprehension. *Educational Psychology*, (August).
- Keleş, A., Ocak, R., Keleş, A., & Gülcü, A. (2009). ZOSMAT: Web-based intelligent tutoring system for teaching–learning process. *Expert Systems with Applications*, 36(2), 1229–1239. doi:10.1016/j.eswa.2007.11.064
- Kennedy, C. (2010). Decentralised metacognition in context-aware autonomic systems: Some key challenges. In *In Metacognition for Robust Social Systems. AAAI Workshops Workshops at the Twenty-Fourth AAAI Conference on Artificial Intelligence* (pp. 34–41). Atlanta, Georgia: AAAI Publications. Retrieved from <http://aaai.org/ocs/index.php/WS/AAAIW10/paper/view/1994>
- Kennedy, C., & Sloman, A. (2003). Autonomous Recovery from Hostile Code Insertion using Distributed Reflection. *Journal of Cognitive Systems Research*, 4(2), 89–117.
- Kim, J., Gil, Y., & Rey, M. (2008). Developing a Meta-Level Problem Solver for Integrated Learners. In *AAAI Workshop on Metareasoning: Thinking about thinking*. AAAI Technical Report (Vol. 8, p. 07).
- Kim, J., & Shinn, Y. (2010). An Instructional Strategy Selection Model Based on Agent and Ontology for an Intelligent Tutoring System. doi:10.1109/WAINA.2010.147
- Kinnebrew, J. S., Biswas, G., Sulcer, B., Taylor, R. S., & Sta, B. (2010). Investigating Self-Regulated Learning in Teachable Agent Environments. *International Handbook of Metacognition and Learning Technologies*. Berlin, Germany: Springer., 1–29.
- Kizilirmak, J., Rösler, F., & Khader, P. (2012). Control processes during selective long-term memory retrieval. *NeuroImage*, 59(2), 1830–41. doi:10.1016/j.neuroimage.2011.08.041
- Kleppe, A., Warmer, J., & Bast, W. (2003). *MDA Explained: The Model Driven Architecture: Practice and Promise*. AddisonWesley Professional (Vol. 83). doi:10.1016/S0031-9406(05)65759-8
- Koch, N., & GmbH, F. (2006). Transformation Techniques in the Model-Driven Development Process of UWE. In *Workshop Proceedings of the Sixth International Conference on Web Engineering (ICWE'06)*. ACM (p. 3). doi:10.1145/1149993.1149997
- Koedinger, K., Aleven, V., Roll, I., & Baker, R. (2009). In vivo experiments on whether supporting metacognition in intelligent tutoring systems yields robust learning. *Handbook of Metacognition in Education*, 42(1983), 647–51. doi:10.1002/lsm.20954
- Kolodner, J. (1992). An introduction to case-based reasoning. *Artificial Intelligence Review*, 6(1), 3–34. doi:10.1007/BF00155578
- Kolodner, J., Camp, P., Crismond, D., Fasse, B., Gray, J., Holbrook, J., ... Ryan, M. (2003). Problem-Based Learning Meets Case-Based Reasoning in the Middle-School Science Classroom: Putting Learning by Design(tm) Into Practice. *Journal of the Learning*

- Kolodner, J., Cox, M., & Gonzalez-Perez, C. (2005). Case-based reasoning-inspired approaches to education. *The Knowledge Engineering Review*. doi:10.1017/S0269888906000634
- Kolodner, J., Owensby, J. N., & Guzdial, M. (2004). Case-Based Learning Aids. *Cognition*, 2, 829–862.
- Koper, R. (2001). *Modeling Units of Study from a Pedagogical Perspective: The Pedagogical Meta-Model Behind EML. ...of the Netherlands*. <http://eml.ou.nl/introduction/docs/....> Heeren, The Netherlands. Retrieved from <http://lnx-hrl-075v.web.pwo.ou.nl/bitstream/1820/36/1/Pedagogical>
- Krause, E., Schermerhorn, P., & Scheutz, M. (2012). Crossing Boundaries: Multi-Level Introspection in a Complex Robotic Architecture for Automatic Performance Improvements. In *Proceedings of the Twenty-Sixth Conference on Artificial Intelligence*. Palo Alto, CA: AAAI Press.
- Laird, J. (2008). Extending the Soar Cognitive Architecture. In *Proceedings of the Conference on Artificial General Intelligence* (Vol. 171, pp. 224–235). doi:/10.1016/j.cogsys.2006.07.004
- Landowska, A. (2010). Student Model Representation for Pedagogical Virtual Mentors. *Learning*, (June), 61–64.
- Leake, D. B. (1995). Representing Self-knowledge for Introspection about Memory Search A Planful Framework for Internal Reasoning. In *AAAI Spring Symposium on Representing Mental States and Mechanisms*. Stanford, CA.
- Lee, M., & Baylor, A. L. (2006). Designing Metacognitive Maps for Web-Based Learning Disorientation and Metacognition in Web-Based Learning Environments The Underlying Metacognitive Principles of a Metacognitive Map. *Educational Technology & Society*, 9, 344–348.
- Leelawong, K., Biswas, G., & Isis, E. (2008). Designing Learning by Teaching Agents The Betty 's Brain System. *International Journal of Artificial Intelligence in Education*, 18(3), 181–208.
- Legaspi, R., Sison, R., & Numao, M. (2004a). A Category-Based Self-Improving Planning Module. In J. C. Lester, R. M. Vicari, & F. Paraguaçu (Eds.), *Intelligent Tutoring Systems Lecture Notes in Computer Science Volume 3220* (pp. 554–563). Springer Berlin Heidelberg. doi:10.1007/978-3-540-30139-4_52
- Legaspi, R., Sison, R., & Numao, M. (2004b). MSIP : Agents Embodying a Category-Based Learning Process for the ITS Tutor to Self-improve Its Instructional Plans 2 The MSIP : An Agent-Based Planning Module, 114–123.
- Legaspi, R., Sison, R., & Numao, M. (2004c). Self-improving instructional plans on the level of student categories. In S. Looi, Chee-Kit, Sutinen, Erkki (Ed.), *IEEE International Conference on Advanced Learning Technologies* (pp. 475–479). Joensuu, Finland: IEEE

- Li, J., Sheng, Z., & Ng, K. (2011). Multi-goal Q-learning of cooperative teams. *Expert Systems with Applications*, 38(3), 1565–1574. doi:10.1016/j.eswa.2010.07.071
- Li, L. (2011). Based on the Agent model to study the intelligent teaching system, 2843–2846.
- Lian, Y. (2011). An Online Adaptive Tutoring System for Design- Centric Courses. *Computer Engineering*, 1191–1194.
- Linn, J., Segedy, J., Jeong, H., Podgursky, B., & Biswas, G. (2009). A Reconfigurable Architecture for Building Intelligent Learning Environments. In V. Dimitrova, R. Mizoguchi, B. DuBulay, & A. Graesser (Eds.), *Proceedings of the 14th Intl. Conf. on Artificial Intelligence in Education (AIED 2009)* (pp. 115–122). Amsterdam.
- Liu, J. (1988). The use of fuzzy reasoning in intelligent computer aided instructional systems. In *Multiple-Valued Logic, 1988., Proceedings of the Eighteenth International Symposium on*.
- Livingston, J. A. (2003). *Metacognition: An Overview*. U.S.A Deparmet of Education. Educational Resources Information Center (ERIC). New York.
- Lopes, R. D. S., & Fernandes, M. A. (2009). Adaptative Instructional Planning Using Workflow and Genetic Algorithms. In *2009 Eighth IEEE/ACIS International Conference on Computer and Information Science* (pp. 87–92). Shanghai: IEEE. doi:10.1109/ICIS.2009.197
- Maeda, Y., & Hanaka, S. (2008). Differential Reinforcement-type Shaping Q-Learning Method, 2066–2071.
- Magnisalis, I., Demetriadis, S., & Karakostas, A. (2011). Adaptive and Intelligent Systems for Collaborative Learning Support: A Review of the Field. *IEEE Transactions on Learning Technologies*, 4(1), 5–20. doi:10.1109/TLT.2011.2
- Makgato, M. (2012). Identifying Constructivist Methodologies and Pedagogic Content Knowledge in the Teaching and Learning of Technology. *Procedia - Social and Behavioral Sciences*, 47, 1398–1402. doi:10.1016/j.sbspro.2012.06.832
- Mandl, H., & Lesgold, A. (1988). *Learning Issues for Intelligent Tutoring Systems*. *Learning Issues for Intelligent Tutoring Systems*. doi:10.1007/978-1-4684-6350-7
- Mclaren, B. M., Deleeuw, K. E., & Mayer, R. E. (2011). Computers & Education Polite web-based intelligent tutors: Can they improve learning in classrooms? *Computers & Education*, 56(3), 574–584. doi:10.1016/j.compedu.2010.09.019
- Mecklinger, A. (2010). The control of long-term memory: brain systems and cognitive processes. *Neuroscience and Biobehavioral Reviews*, 34(7), 1055–65. doi:10.1016/j.neubiorev.2009.11.020
- Melis, E., & Siekmann, J. (2004). Activemath: An intelligent tutoring system for mathematics. In L. Rutkowski, J. H. Siekmann, R. Tadeusiewicz, & L. Zadeh (Eds.),

- Artificial Intelligence and Soft Computing-ICAISC 2004* (pp. 91–101). Berlin: Springer Berlin Heidelberg. doi:10.1007/978-3-540-24844-6_12
- Mens, T., & Van Gorp, P. (2006). A taxonomy of model transformation. *Electronic Notes in Theoretical Computer Science*, 152(1-2), 125–142. doi:10.1016/j.entcs.2005.10.021
- Merks, E., Eliersick, R., & Grose, T. (2004). The Eclipse Modeling Framework. In *Sun's 2004 Worldwide Java Developer Conference* (pp. 1–37). doi:10.1108/02641610810878585
- Metcalfe, J., & Dunlosky, J. (2008). Metamemory. In H. L. Roediger III (ed.), *Cognitive Psychology of Memory. Vol. [2] of Learning and Memory: A Comprehensive Reference, 4 vols.* (J).
- Mikic-Fonte, F. (2010). A BDI-based intelligent tutoring module for the e-learning platform INES. In *Frontiers in Education Conference (FIE), 2010 IEEE* (pp. 1–6). Washington, DC: IEEE Comput. Soc. doi:10.1109/FIE.2010.5673365
- Mizoguchi, R., Hayashi, Y., & Bourdeau, J. (2010). Ontology-Based Formal Modeling of the Pedagogical World: Tutor Modeling. *Studies in Computational Intelligence, 2010, Volume 308, Advances in Intelligent Tutoring Systems, Pages 229–247*, 229–247.
- Molina, A. I., Gallardo, J., Redondo, M. A., Ortega, M., & Giraldo, W. J. (2013). Metamodel-driven definition of a visual modeling language for specifying interactive groupware applications: An empirical study. *Journal of Systems and Software*, 86, 1772–1789. doi:10.1016/j.jss.2012.07.049
- Moore, A., Macarthur, V., & Conlan, O. (2011). Core Aspects of Affective Metacognitive User Models 2 Metacognitive / Affective Systems. In *International Workshop at UMAP 2011 on Augmenting User Models with Real World Experiences to Enhance Personalization and Adaptation*.
- Moore, B., Dean, D., & Gerber, A. (2004). *Eclipse Development using the Graphical Editing Framework and the Eclipse Modeling Framework*. IBM Redbooks (Vol. 1). doi:10.1147/JRD.2010.2041693
- Morbini, F., & Schubert, L. (2008). Metareasoning as an integral part of commonsense and autocognitive reasoning. In A. Raja & M. Cox (Eds.), *AAAI-08 Workshop on Metareasoning*. Chicago, Illinois: AAAI Press.
- Moura, I., & Sarma, V. (2005). Multiagent System and Imitative Consciousness. In *IICAI - 2nd Indian International Conference on Artificial Intelligence*. Pune, India.
- Muldner, K., & Conati, C. (2007). Refining Tailored Scaffolding for Meta- Cognitive Skills during Analogical Problem Solving. *Interface*, 3–12.
- Muñoz-Merino, P. J., Fernández Molina, M., Muñoz-Organero, M., & Delgado Kloos, C. (2012). An adaptive and innovative question-driven competition-based intelligent tutoring system for learning. *Expert Systems with Applications*, 39(8), 6932–6948. doi:10.1016/j.eswa.2012.01.020
- Murdock, W., & Goel, A. (2001). Meta-case-Based Reasoning : Using Functional Models to Adapt Case-Based Agents. In *Proceedings of the 4th. International Conference on Case-*

- Based Reasoning (ICCBR'01)* (pp. 407–421). Vancouver, Canada: Springer-Verlag Lecture Notes in Computer Science series.
- Murdock, W., & Goel, A. (2008). Meta-case-based reasoning: self-improvement through self-understanding. *Journal of Experimental & Theoretical Artificial Intelligence*, 20(1), 1–36. doi:10.1080/09528130701472416
- Murray, T. (1998). Authoring Knowledge-Based Tutors: Tools for Content, Instructional Strategy, Student Model, and Interface Design. *Journal of the Learning Sciences*, 7(1), 5–64. doi:10.1207/s15327809jls0701_2
- Nelson, T., & Narens, L. (1990). Metamemory: A Theoretical Framework and New Findings. In *Psychology of Learning and Motivation* (Vol. 26, pp. 125–173). doi:10.1016/S0079-7421(08)60053-5
- Nelson, T., Narens, L., & Dunlosky, J. (2004). A revised methodology for research on metamemory: Pre-judgment Recall and Monitoring (PRAM). *Psychological Methods*, 9(1), 53–69. doi:10.1037/1082-989X.9.1.53
- Norman, D., & Shallice, T. (1986). *Attention to action* (pp. 1-18). Springer US.
- Noy, N. F., & McGuinness, D. L. (2000). *Ontology Development 101 : A Guide to Creating Your First Ontology*. Stanford. Press, 1–25.
- Nwana, H. S. (1990). Intelligent Tutoring Systems : an overview. *Artificial Intelligence*, 251–277.
- Oehlmann, R. (1995). Metacognitive Adaptation : Regulating the Plan Transformation Process. In *Proceedings of the Fall Symposium on Adaptation of Knowledge for Reuse*.
- Oehlmann, R., Edwards, P., & Sleeman, D. (1995). Introspection Planning : Representing Metacognitive Experience. In *AAAI Spring Symposium 1995 on Representing Mental States and Mechanisms*.
- Oentaryo, R., & Pasquier, M. (2008). Towards a novel integrated neuro-cognitive architecture (INCA). In IEEE (Ed.), *IJCNN , IEEE International Joint Conference on Neural Networks* (pp. 1902–1909).
- OMG. (2005). *Meta Object Facility (MOF) 2 . 0 Core Specification*.
- OMG. (2011). *Meta Object Facility (MOF) 2.0 Query/View/Transformation, V1.1*. Retrieved from <http://www.omg.org/spec/QVT/1.1/>
- OMG. (2013). *OMG Meta Object Facility (MOF) Core Specification. OMG - Technical Report*.
- OMG. (2014). *Object Constraint Language*. doi:10.1167/7.9.852
- Omg, Q. (2008). *Meta Object Facility (MOF) 2 . 0 Query / View / Transformation Specification. Transformation, (January), 1–230*. Retrieved from <http://www.omg.org/spec/QVT/1.0/PDF/>
- Opdenakker, M.-C., & Van Damme, J. (2006). Teacher characteristics and teaching styles as

- effectiveness enhancing factors of classroom practice. *Teaching and Teacher Education*, 22(1), 1–21. doi:10.1016/j.tate.2005.07.008
- Ovalle, D., & Jiménez, J. (2004). Entorno Integrado de Enseñanza / Aprendizaje basado en Sistemas Tutoriales Inteligentes & Ambientes Colaborativos. *Iberoamericana de Sistemas, Cibernética E Informática Vol. 1, No 1*.
- Ozdamli, F. (2012). Pedagogical framework of m-learning. *Procedia - Social and Behavioral Sciences*, 31(2011), 927–931. doi:10.1016/j.sbspro.2011.12.171
- Ozuru, Y., Kurby, C. A., & McNamara, D. S. (2012). The effect of metacomprehension judgment task on comprehension monitoring and metacognitive accuracy. *Metacognition and Learning*, 7(2), 113–131. doi:10.1007/s11409-012-9087-y
- Pachoulakis, I., Profit, A. N., & Kapetanakis, K. (2012). The Question Manager and Tutoring Module for the EViE-m platform. In *2012 International Conference on Telecommunications and Multimedia (TEMU)* (pp. 196–201). Chania: IEEE Comput. Soc. doi:10.1109/TEMU.2012.6294716
- Palmer, D. (2005). A Motivational View of Constructivist- informed Teaching, 27(15), 1853–1881. doi:10.1080/09500690500339654
- Pashler, H., McDaniel, M., Rohrer, D., & Bjork, R. (2009). Learning styles concepts and evidence. *Psychological Science in the Public Interest, Supplement*, 9(3), 105–119. doi:10.1111/j.1539-6053.2009.01038.x
- Pasquali, A., Timmermans, B., & Cleeremans, A. (2010). Know thyself: metacognitive networks and measures of consciousness. *Cognition*, 117(2), 182–90. doi:10.1016/j.cognition.2010.08.010
- Payne, J., & Israel, N. (2010). Beyond teaching practice: Exploring individual determinants of student performance on a research skills module. *Learning and Individual Differences*, 20(3), 260–264. doi:10.1016/j.lindif.2010.02.005
- Payne, V., Medvedeva, O., Legowski, E., Castine, M., Tseytlin, E., Jukic, D., & Crowley, R. S. (2009). Effect of a limited-enforcement intelligent tutoring system in dermatopathology on student errors , goals and solution paths. *Artificial Intelligence in Medicine*, 47, 175–197. doi:10.1016/j.artmed.2009.07.002
- Pěchouček, M., Štěpánková, O., Marik, V., & Jaroslav, B. (2003). Abstract architecture for meta-reasoning in multi-agent systems. In *Multi-agent systems and applications III* (LNAI, pp. 84–99). Berlin: Springer Verlag.
- Peña, C. I., Marzo, J., De la Rosa, J. L., & Fabregat, R. (2002). Un Sistema de Tutoría Inteligente Adaptativo Considerando Estilos de Aprendizaje. *VI Congreso Iberoamericano de Informática Educativa*, 1 – 12.
- Peterson, E. R., Rayner, S. G., & Armstrong, S. J. (2009). Researching the psychology of cognitive style and learning style: Is there really a future? *Learning and Individual Differences*, 19(4), 518–523. doi:10.1016/j.lindif.2009.06.003
- Phobun, P., & Vicheanpanya, J. (2010). Adaptive intelligent tutoring systems for e-learning

- systems. *Procedia - Social and Behavioral Sciences*, 2(2), 4064–4069. doi:10.1016/j.sbspro.2010.03.641
- Prentzas, J., Hatzilygeroudis, I., & Garofalakis, J. (2002). A Web-Based Intelligent Tutoring System Using Hybrid Rules as Its Representational Basis. *Knowledge Creation Diffusion Utilization*, 119–128.
- Priya, S. S., Subhashini, R., & Akilandeswari, J. (2012). Learning Agent based Knowledge Management in Intelligent Tutoring System. In *2012 International Conference on Computer Communication and Informatics (ICCCI -2012)*, Jan. 10 – 12, 2012, Coimbatore, INDIA.
- Pule, M., & Anderson, T. (2009). A Model to Understanding Metacomprehension. *Reading*, 1–6.
- Qadoori, O. (2010). Design a framework for intelligent differentiated tutoring system. In *Distance Learning and Education (ICDLE)*, 2010 4th International Conference on (pp. 174–177). doi:10.1109/ICDLE.2010.5606012
- Qiang, W., & Zhongli, Z. (2011). Reinforcement learning model, algorithms and its application. *2011 International Conference on Mechatronic Science, Electric Engineering and Computer (MEC)*, (1), 1143–1146. doi:10.1109/MEC.2011.6025669
- Raghupathi, W., & Umar, A. (2008). Exploring a model-driven architecture (MDA) approach to health care information systems development. *International Journal of Medical Informatics*, 77(5), 305–14. doi:10.1016/j.ijmedinf.2007.04.009
- Rahman, S. A., & Farag, I. (2011). An Auto-Recommender Based Intelligent E-Learning System. *International Journal of Computer Science and Network Security*, 11(1), 67–70.
- Raja, A., Alexander, G., & Mappillai, V. (2006). Leveraging Problem Classification in Online Meta-Cognition. In *AAAI Spring Symposium: Distributed Plan and Schedule Management* (pp. 97–104).
- Raja, A., & Lesser, V. (2007). A framework for meta-level control in multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 15(2), 147–196. doi:10.1007/s10458-006-9008-z
- Rensink, A., & Nederpel, R. (2008). Graph Transformation Semantics for a QVT Language. *Electronic Notes in Theoretical Computer Science*, 211(C), 51–62. doi:10.1016/j.entcs.2008.04.029
- Richardson, J. T. E. (2011). Approaches to studying, conceptions of learning and learning styles in higher education. *Learning and Individual Differences*, 21(3), 288–293. doi:10.1016/j.lindif.2010.11.015
- Rishi, O. P., & Chaplot, N. (2010). Predictive Role of Case Based Reasoning for Astrological Predictions about Profession : System Modeling Approach. *Computational Intelligence*, 313–317.
- Rishi, O. P., Govil, R., & Sinha, M. (2007). Distributed Case Based Reasoning for Intelligent Tutoring System : An Agent Based Student Modeling Paradigm. *Engineering and*

Technology, 273–276.

- Roll, I., Aleven, V., McLaren, B., & Koedinger, K. (2011a). Improving students' help-seeking skills using metacognitive feedback in an intelligent tutoring system. *Learning and Instruction*, 21(2), 267–280. doi:10.1016/j.learninstruc.2010.07.004
- Roll, I., Aleven, V., McLaren, B., & Koedinger, K. (2011b). Metacognitive Practice Makes Perfect: Improving Students' Self-Assessment Skills with an Intelligent Tutoring System. In *Artificial Intelligence in Education* (pp. 288–295). Springer. Retrieved from <http://www.springerlink.com/index/L606583G54VK9433.pdf>
- Roll, I., Aleven, V., McLaren, B. M., Ryu, E., Baker, R. S. J., & Koedinger, K. (2006). The Help Tutor : Does Metacognitive Feedback Improve Students' Help-Seeking Actions, Skills and Learning ?, 360–369.
- Roll, I., Ryu, E., Sewall, J., Leber, B., McLaren, B. M., Aleven, V., & Koedinger, K. (2006). Towards Teaching Metacognition: Supporting Spontaneous Self-Assessment. *Artificial Intelligence*, 738–740.
- Rollande, R., & Grundspenkis, J. (2012). Representation of study program as a part of graph based framework for tutoring module of intelligent tutoring system. In *2012 Second International Conference on Digital Information Processing and Communications (ICDIPC)* (pp. 108–113). Klaipeda City: IEEE Comput. Soc. doi:10.1109/ICDIPC.2012.6257276
- Rongmei, Z., & Lingling, L. (2009). Research on Internet Intelligent Tutoring System Based on MAS and CBR. In *2009 International Forum on Information Technology and Applications* (Vol. 3, pp. 681–684). doi:10.1109/IFITA.2009.511
- Rugaber, S., Goel, A., & Martie, L. (2013). GAIA: A CAD Environment for Model-Based Adaptation of Game-Playing Software Agents. *Procedia Computer Science: Conference on Systems Engineering Research*, 16(CSER'13), 29–38. doi:10.1016/j.procs.2013.01.004
- Russell, S., & Wefald, E. (1989). On Optimal Game-Tree Search using Rational Meta-Reasoning. In M. Kaufma (Ed.), *11 International Joint Conference on Artificial Intelligence* (pp. 334–340). Detroit, MI, USA: AAAI Press.
- Saberi, S., & Mohammad, E. (2008). A case-based planning approach to design and plan ITMAS. In *2008 4th International IEEE Conference on Intelligent Systems*.
- Samsonovich, A. (2009). The Constructor Metacognitive Architecture. In A. Samsonovich (Ed.), *Biologically Inspired Cognitive Architectures II: Papers from the AAAI Fall Symposium (FS-09-01)*. AAAI Technical Report FS-09-01 (pp. 124–134). AAAI Press. ISBN 978-1-57735-435-2.
- Samsonovich, A. (2010). A Human-Inspired Cognitive Architecture Supporting Self Regulated Learning in Problem Solving. In A. Raja & D. Josylula (Eds.), *Metacognition for Robust Social Systems: Papers from the 2010 AAAI Workshop*, AAAI Technical Report WS-10-07 (pp. 50–53). Menlo Park, CA: AAAI Press.
- Samsonovich, A., & Ascoli, G. (2006). Integrated hybrid cognitive architecture for a virtual

- roboscout. In M. T. Beetz, K. Rajan (Ed.), *Cognitive Robotics: Papers from the AAAI Workshop, AAAI Technical Reports, volume WS-06-03* (pp. 129–134). Menlo Park, CA: AAAI Press.
- Samsonovich, A., & Jong, K. (2005). Designing a self-aware neuromorphic hybrid. In S. M. Thorisson, H. Vilhjalmsen (Ed.), *AAAI-05 Workshop on Modular Construction of Human-Like Intelligence: AAAI Technical Report, volume WS-05-08* (pp. 71–78). Menlo Park, CA: AAAI Press.
- Sargent, R. (2005). Verification and validation of simulation models. In *Proceedings of the 37th conference on winter simulation*. Orlando, Florida.
- Sateesh, G., & Suresh, S. (2012). Meta-cognitive Neural Network for classification problems in a sequential learning framework. *Neurocomputing*, 81, 86–96. doi:10.1016/j.neucom.2011.12.001
- Schiaffino, S., Garcia, P., & Amandi, A. (2008). eTeacher: Providing personalized assistance to e-learning students. *Computers & Education*, 51(4), 1744–1754. doi:10.1016/j.compedu.2008.05.008
- Schmill, M., Anderson, M., Fults, S., Josyula, D., Oates, T., Perlis, D., ... Wright, D. (2011). The Metacognitive Loop and Reasoning about Anomalies. In M. Cox & A. Raja (Eds.), *Metareasoning: Thinking about thinking* (pp. 183–198). Cambridge, MA: The MIT Press.
- Schmill, M., Josyula, D., Anderson, M., Wilson, S., Oates, T., Perlis, D., ... Fults, S. (2007). Ontologies for Reasoning about Failures in AI Systems. In *in Proceedings from the Workshop on Metareasoning in Agent Based Systems at the Sixth International Joint Conference on Autonomous Agents and Multiagent Systems*.
- Segal, L. (2001). *The dream of reality: Heinz von Foerster's constructivism*. Springer Science & Business Media.
- Self, J. (1994). *Dormobile: a vehicle for metacognition*. Lancaster University. (TechnicalReport).Lancaster LA1 4YR - England.
- Seridi, H., Sari, T., Khadir, T., & Sellami, M. (2006). Adaptive Instructional Planning in Intelligent Learning Systems. *Sixth IEEE International Conference on Advanced Learning Technologies (ICALT'06)*, 133–135. doi:10.1109/ICALT.2006.1652386
- Shadish, W., Cook, T., & Campbell, D. (2002). Experimental and Quasi-Experimental Designs. In *Experimental and Quasi-Experimental Designs for Generalized Causal Inference*. (pp. 171–206). doi:10.1093/obo/9780195389678-0053
- Shapiro, S. C., Rapaport, W. J., Kandefer, M., Johnson, F. L., & Goldfain, A. (2007). Metacognition in SNePS. *AI Magazine*, 28, 17–31.
- Shidqie, A., & Gollmann, D. (2007). Compilation of OCL into Java for the Eclipse OCL Implementation. *Media*, (May). Retrieved from <http://www.sts.tu-harburg.de/pw-and-m-theses/2007/jibr07.pdf>
- Silva, L. A. L., Buxton, B. F., & Campbell, J. A. (2003). Enhanced Case-Based Reasoning through Use of Argumentation and Numerical Taxonomy. *Computer*, 423–428.

- Singh, P. (2005). EM-ONE: An Architecture for Reflective Commonsense Thinking. *Dissertation of Doctor of Philosophy in Computer Science and Engineering - Massachusetts Institute of Technology*.
- Sjøberg, D. I. K., Hannay, J. E., Hansen, O., Kampenes, V. B., Karahasanović, A., Liborg, N. K., & Rekdal, A. C. (2005). A survey of controlled experiments in software engineering. *IEEE Transactions on Software Engineering*, 31, 733–753. doi:10.1109/TSE.2005.97
- Skinner, B. (1950). Are theories of learning necessary? *Psychological Review*, 57(4), 193–216. doi:10.1037/h0054367
- Skinner, B. (1954). The Science of Learning and the Art of Teaching. *Harvard Educational Review* 1, 86–97. doi:10.1111/j.1467-8535.2007.00682_9.x
- Sloman, A., & Chrisley, R. (2003). Virtual machines and consciousness. *Journal of Consciousness Studies*, 10(4).
- Snaider, J., McCall, R., & Franklin, S. (2011). The LIDA Framework as a General Tool for AGI. *Artificial General Intelligence: 4th International Conference, Agi*.
- Soh, L. (2007). Integrated introspective case-based reasoning for intelligent tutoring systems. *Proceedings of the National Conference on Artificial Intelligence*, 2(Kolodner), 1566–1571. Retrieved from <http://www.scopus.com/inward/record.url?eid=2-s2.0-36348967670&partnerID=40&md5=2b97df805018e5232a2509e43f1d7dd8>
- Soh, L., & Blank, T. (2008). Integrating case-based reasoning and meta-learning for a self-improving intelligent tutoring system. *International Journal of Artificial Intelligence in Education*. Retrieved from <http://ovidsp.ovid.com/ovidweb.cgi?T=JS&PAGE=reference&D=psyc6&NEWS=N&AN=2008-03820-003>
- Solomonidou, C. (2009). Constructivist design and evaluation of interactive educational software: a research-based approach and examples. *Open Education*, 5(1).
- Specht, M., & Augustin, D.-S. (1998). ATS-adaptive teaching system a WWW-based ITS. *Web-Based Knowledge Servers (Digest No. 1998/307), IEE Colloquium on*.
- Steinberg, D., Budinsky, F., Paternostro, M., & Merks, E. (2008). EMF: Eclipse Modeling Framework. *Engineering*. doi:10.1108/02641610810878585
- Stroulia, E., & Goel, A. (1995). Functional Representation and Reasoning for Reflective Systems. *Journal of Applied Artificial Intelligence. Special Issue on Functional Reasoning*, 9(1), 101–124.
- Sun, R., Zhang, X., & Mathews, R. (2006). Modeling meta-cognition in a cognitive architecture. *Cognitive Systems Research*, 7(4), 327–338. doi:10.1016/j.cogsys.2005.09.001
- Tan, S.-T. (1996). Architecture of a generic instructional planner. *Journal of Network and Computer Applications*, 19(3), 265–274. doi:10.1006/jnca.1996.0018

- Thompson, B., Cohen, M., & Freeman, J. (1995). Metacognitive behavior in adaptive agents. In *Proceedings of the World Congress on Neural Networks* (pp. 266–273). Hillsdale, NJ: IEEE, Lawrence Erlbaum.
- Tulbure, C. (2012). Learning styles, teaching strategies and academic achievement in higher education: A cross-sectional investigation. *Procedia - Social and Behavioral Sciences*, 33, 398–402. doi:10.1016/j.sbspro.2012.01.151
- Unsworth, N. (2010). On the division of working memory and long-term memory and their relation to intelligence: A latent variable approach. *Acta Psychologica*, 134(1), 16–28. doi:10.1016/j.actpsy.2009.11.010
- Vanlehn, K., Jordan, P. W., Ros, C. P., Bhembé, D., Michael, B., Gaydos, A., ... Srivastava, R. (2002). The Architecture of Why2-Atlas: A Coach for Qualitative Physics Essay Writing. In S. Cerri, G. Gouardères, & F. Paraguaçu (Eds.), *Intelligent Tutoring Systems* (pp. 158–167). Berlin: Springer Berlin Heidelberg. doi:10.1007/3-540-47987-2_20
- Vassiliades, V., Cleanthous, A., & Christodoulou, C. (2011). Multiagent reinforcement learning: spiking and nonspiking agents in the iterated Prisoner's Dilemma. *IEEE Transactions on Neural Networks / a Publication of the IEEE Neural Networks Council*, 22(4), 639–653. doi:10.1109/TNN.2011.2111384
- Veenman, M. V. J., Hout-Wolters, B. H. a. M., & Afflerbach, P. (2006). Metacognition and learning: conceptual and methodological considerations. *Metacognition and Learning*, 1(1), 3–14. doi:10.1007/s11409-006-6893-0
- Vesin, B., Ivanović, M., Klačnja-Milićević, A., & Budimac, Z. (2012). Protus 2.0: Ontology-based semantic recommendation in programming tutoring system. *Expert Systems with Applications*, 39(15), 12229–12246. doi:10.1016/j.eswa.2012.04.052
- Vicari, R., & Jiménez, J. (2007). ALLEGRO: Teaching / Learning Multi-Agent Environment using Instructional Planning and Cases- Based Reasoning (CBR). *CLEI Electronic Journal*, 10(1), 1–20.
- Vidal-castro, C., Sicilia, M.-ángel, & Prieto, M. (2012). Systems Representing instructional design methods using ontologies and rules. *Knowledge-Based Systems*, 33, 180–194. doi:10.1016/j.knosys.2012.04.005
- Vinokurov, Y., Lebiere, C., Herd, S., & Reilly, R. O. (2011). A Metacognitive Classifier Using a Hybrid ACT-R / Leabra Architecture. In *AAAI Workshop (WS-11-15)* (pp. 50–55).
- Vockell, E. (2004). *Educational Psychology: A Practical Approach*.
- Von Foerster, H., & Poerksen, B. (2002). *Understanding systems: Conversations on epistemology and ethics*. New York: Kluwer Academic/Plenum Publishers.
- von Glasersfeld, E. (1984). An introduction to radical constructivism. In P. Watzlawik (Ed.), *The Invented Reality*.
- Von Glasersfeld, E. (1996). *Introduction: Aspects of constructivism. Constructivism: Theory, perspectives, and practice*.

- Vygotsky. (1978). Tool and Symbol in Child Development. In *Mind in Society* (p. chap 1).
- Walker, E., Koedinger, K., McLaren, B., & Rummel, N. (2006). Cognitive tutors as research platforms: Extending an established tutoring system for collaborative and metacognitive experimentation. In *8th International Conference on Intelligent Tutoring Systems*. Jhongli, Taiwan.
- Wang, E., & Kim, Y. S. (2009). A Two-Layer Reasoning Framework for a Teaching Strategies Engine using SWRL. *Proceedings of the 17th International Conference on Computers in Education*, 3–10.
- Wang, H. (2011). Research on the model of knowledge representation ontology based on framework in intelligent learning system. *Electrical and Control Engineering (ICECE)*, 6757–6760. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6056799
- Wang, H., Zhou, X., Zhou, X., Liu, W., & Li, W. (2010). Adaptive and Dynamic Service Composition Using Q-Learning. *2010 22nd IEEE International Conference on Tools with Artificial Intelligence*, 145–152. doi:10.1109/ICTAI.2010.28
- Wang, Z., Gu, X., He, J., Zheng, S., & Wang, W. (2010). Design and implementation of an intelligent tutoring system for English instruction. *Intelligent Computing and Intelligent Systems (ICIS)*, 2010 IEEE International Conference on. doi:10.1109/ICICISYS.2010.5658777
- Watkins, C. J. C. H., & Dayan, P. (1992). Q-learning. *Machine Learning*, 8(3-4), 279–292. doi:10.1007/BF00992698
- Watson, J. (1913). Psychology as the behaviorist views it. *Psychological Review*, 20(2), 158–177. doi:10.1037/h0074428
- Watson, J. (1930). *Behaviorism (Rev. ed.)*. *Behaviorism (Rev. ed.)*. Retrieved from <http://search.ebscohost.com/login.aspx?direct=true&db=psyh&AN=1931-00040-000&site=ehost-live>
- Wen, M. (2004). Cognitive?metacognitive and content-technical aspects of constructivist Internet-based learning environments: a LISREL analysis. *Computers & Education*, 43(3), 237–248. doi:10.1016/j.compedu.2003.10.006
- Wielemaker, J., Schrijvers, T., Triska, M., & Lager, T. (2012). SWI-Prolog. *Theory and Practice of Logic Programming*. doi:10.1017/S1471068411000494
- Wiemer-hastings, P., & Glasswell, K. (2003). StoryStation: Agent-based scaffolding of metacognitive processes for writing. In *AIED2003 - Supplementary Proceedings of the 11th International Conference on Artificial Intelligence in Education*, University of Sydney, Sydney, 200, pp. 534 – 541.
- Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., & Wesslén, A. (2000). *Experimentation in software engineering: an introduction*. Springer Netherlands (Vol. 15).
- Woo, C., Evens, M., Freedman, R., Glass, M., Shim, L. S., Zhang, Y., ... Michael, J. (2006). An intelligent tutoring system that generates a natural language dialogue using

- dynamic multi-level planning. *Artificial Intelligence in Medicine*, 38(1), 25–46. doi:10.1016/j.artmed.2005.10.004
- Woo, C., Evens, M., Michael, J., & Rovick, A. (1991). Dynamic instructional planning for an intelligent physiology tutoring system. In *Computer-Based Medical Systems - Proceedings of the Fourth Annual IEEE Symposium* (pp. 226–233). IEEE Comput. Soc. Press. doi:10.1109/CBMS.1991.128971
- Xiao, L., & Greer, D. (2009). Adaptive Agent Model: Software Adaptivity using an Agent-oriented Model-Driven Architecture. *Information and Software Technology*, 51(1), 109–137. doi:10.1016/j.infsof.2008.02.002
- Yong, Z., & Zhijing, L. (2003). A Model of Web Oriented Intelligent Tutoring System for Distance Education. In *Fifth IEEE International Conference on Computational Intelligence and Multimedia Applications*.
- Yonglin, L., Weiping, W., Qun, L., & Yifan, Z. (2009). A transformation model from DEVS to SMP2 based on MDA. *Simulation Modelling Practice and Theory*, 17(10), 1690–1709. doi:10.1016/j.simpat.2009.08.003
- Yu, S., & Zhiping, L. (2008). Intelligent Pedagogical Agents for Intelligent Tutoring Systems. *2008 International Conference on Computer Science and Software Engineering*, 516–519. doi:10.1109/CSSE.2008.414
- Yu-Liang Ting, R. (2012). Using mobile technologies to create interwoven learning interactions: An intuitive design and its evaluation. *Computers & Education*, 60(1), 1–13. doi:10.1016/j.compedu.2012.07.004
- Zhang, L., VanLehn, K., Girard, S., Burleson, W., Chavez-Echeagaray, M. E., Gonzalez-Sanchez, J., & Hidalgo-Pontet, Y. (2014). Evaluation of a meta-tutor for constructing models of dynamic systems. *Computers & Education*, 75, 196–217. doi:10.1016/j.compedu.2014.02.015
- Zhang, Z., Franklin, S., & Dasgupta, D. (1998). Metacognition in software agents using classifier systems. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence* (pp. 83–88). Madison, Wisconsin: MIT press.
- Zhang, Z., Geng, X., Jiang, Y., & Yang, Y. (2009). An Intelligent Tutoring System (ITS) for Tactical Training based on Ontology. In *Information Engineering and Computer Science, 2009. ICIECS 2009. International Conference on*.
- Zhiping, L. I. (2009). A Formal Model of Knowledge Base Systems in Intelligent Tutoring Systems, (2004), 0–3.
- Zhiping, L. I., Yu, S. U. N., & Tianwei, X. U. (2011). A Formal Model of Personalized Recommendation Systems in Intelligent Tutoring Systems *. *The 6th International Conference on Computer Science & Education (ICCSE 2011)*, (210210), 1006–1009.
- Zouhair, A., En-naimi, E. M., Boukachour, H., Person, P., & Bertelle, C. (2010). MultiAgent Case-Based Reasoning and Individualized Follow-up of Learner in Remote Learning. *Architecture*.

