
Diploma Project

Kyoto Mobile Exergaming Project

Version 1.0

Author:	Laurent PREVOST
Department / Orientation:	TIC – IL
Responsible Professor:	Pr. Olivier LIECHTI & Pr. Michael LYONS
Mandatory:	HEIG-VD – Ritsumeikan University
Realization year:	2008
Submission date:	23 December 2008

Abstract

Japan offers an excellent location for the development of mobile phone applications. Japanese mobile communications technologies are in advance of European ones, with mobile networks offering higher bandwidth and speed. Mobile phone handsets provide rich functionality with providers offering a large variety of interesting services.

This offers the developer a very interesting situation for creating pervasive games with embedded geo-localization services. Moreover, the popularity of video game platforms such as the Wii and Playstation is widespread amongst the Japanese population, and, in general, it may be said that most Japanese people enjoy new games which employ novel technologies.

This project explored the opportunity to build a reusable platform for the creation of pervasive games with geo-localization services. The main feature of the game is that it offers mini games, or quests, to the player. These quests have to be solved to discover new quests, to visit new locations, and to meet new characters.

The platform is oriented around three major mechanisms to allow the users playing to the game. In addition, of quest, they are the items that are collectable by the players and the non-player characters to add the background to the game.

With the platform, this is possible to create a pervasive game in the context of the Kyoto city. Kyoto is an excellent place to build such a game because of its rich historical heritage and numerous sites of interest.

Finally, the possibility to integrate the Swiss inTrack platform developed at the HEIG-VD is a great challenge in the Japanese context. The collaboration between this project and the inTrack team can be profitable to improve both platforms during the development.

Acknowledgement

I want to thank especially the Hasler Foundation and HEIG-VD for their financial support that allows me to do my diploma project in Japan. Without this support, it would not be possible for me to undertake this adventure.

The Professor Olivier Liechti has my special thanks for his involvement in organizing the project in Sw. and Japan. The project owes its birth to his preparation work. He was always available to help me when it was necessary. His knowledge was very helpful to improve some parts of this project. His visit during my project in Tokyo was especially interesting to learn more about the person rather than the Professor.

The Professor Michael Lyons, his homologue, did a great work to allow me coming and doing my work in Ritsumeikan University in Kyoto. He has all my special thanks too. I want to underline the work he did for me when I arrived in Japan. He explained me many things about the day life and helped me for my installation. His ideas were always interesting and great to investigate in detail.

I would like to thank Professor Koichi Hosoi, director of the Game Archive Project at Ritsumeikan University for his role in enabling my visit to Japan and for the possibility to join the Ritsumeikan Global COE (Center of Excellence) Digital Humanities project as a visiting researcher. Further thanks are due to Mr. Tomokazu Takagi, of the Kinugasa Campus Research Office, who aided greatly my visa application procedure as well as other preparations for my visit to Ritsumeikan University.

During the project, the help of the professors' assistants was very enjoyable for me. Masahiro Kato, assistant of Professor Lyons, was able to help in my day life during my stay. He always helped me to have the best stay that I can imagine. David Johanot, assistant at the IICT and developer on inTrack platform, helped me with his support for the inTrack platform. The technical discussions were especially appreciated.

Jean-Philippe Steullet has my special consideration due to the same conditions of work. He also does his work in a foreign country. With this situation, our friendship has strengthened. We always helped the other in delicate situations of work or for the day life. The adventure would not be so funny without our discussions.

The list of the persons I want to thanks is too long for this document. All the persons that help me for my stay or my work, I wan to tell us "many thanks" and more. Without the moral support, this stay would be probably more difficult than it was.

To all, I say: Arigato

Laurent PREVOST

<http://japan.prevole.ch>

Table of Contents

ABSTRACT	1
ACKNOWLEDGEMENT	1
INTRODUCTION	1
1.1. Introduction	5
1.2. Project orientation	5
1.3. Definitions	5
1.3.1. "Exergaming"	5
1.3.2. Pervasive games	6
1.3.3. Geo-localization services	6
1.3.4. Ubiquitous	6
1.4. Technologic context	7
1.4.1. inTrack	7
1.4.2. Internet on mobile phones	7
1.5. Contributions	8
1.5.1. Game platform	8
1.5.2. Social aspect	9
1.5.3. inTrack evaluation and refactoring	9
1.6. Document structure	10
PROJECT SCOPE	11
2.1. Introduction	15
2.2. Game platform	15
2.2.1. Kmep Entities	15
2.2.2. Quests	16
2.2.3. Items	16
2.2.4. Dialogs	17
2.2.5. Messages	17
2.2.6. Summary	17
2.3. Services	17
2.3.1. User services	17
2.3.2. Administration services	18
2.4. Quests	19
2.4.1. Reach a place	19
2.4.2. Collect items	19
2.4.3. Solve picture puzzles	19
2.5. Configuration	19
2.6. inTrack	20
2.7. KMEP	20
2.8. Conclusion	21
USER EXPERIENCE	23

3.1. Introduction	27
3.2. Game play	27
3.2.1. First Connection – The beginning	27
3.2.2. First Quest – Reach a place	29
3.2.3. Second Quest – Carry an item	30
3.2.4. Third Quest – Collect items	32
3.2.5. Another quest – Puzzle quest	38
3.3. GPS Tracker	41
3.4. Conclusion	43
ARCHITECTURE	45
4.1. Introduction	51
4.2. Global Architecture	51
4.3. System architecture	52
4.3.1. Mobile	52
4.3.2. inTrack Server	54
4.3.3. Game Server (MEP)	54
4.4. Concrete Server Architecture	56
4.5. MEP Game Mechanisms	57
4.5.1. Dialogs	57
4.5.2. Dialogs specificities	60
4.5.3. Items	62
4.5.4. Quests	63
4.5.5. Messages	66
4.5.6. Message flows	69
4.6. Major quest types	70
4.6.1. Find/Reach a NPC	71
4.6.2. Collect items	71
4.6.3. Solve a picture puzzle	72
4.6.4. Summary	72
4.7. Conclusion	73
IMPLEMENTATION	75
5.1. Introduction	85
5.2. KMEP-Common	85
5.3. KMEP-EJB Generalities	87
5.3.1. inTrack error bindings	87
5.3.2. Formatter Factories Interfaces	87
5.3.3. Formatter Interfaces	89
5.3.4. Server configuration	89
5.3.5. Identity	90
5.4. KMEP-EJB Configuration	91
5.4.1. XML Configuration	91
5.4.2. XML Configuration Infrastructure	94
5.4.3. XML Configuration Location	96

5.4.4. XML Configuration MEP Entities	96
5.4.5. XML Configuration Behavior	97
5.4.6. XML Configuration Dialog	99
5.4.7. XML Configuration Dialog Answer	99
5.4.8. XML Configuration Dialog Condition	100
5.4.9. XML Configuration Dialog State	100
5.4.10. XML Configuration Item	100
5.4.11. XML Configuration Message	101
5.4.12. XML Configuration Quest	103
5.4.13. XML Configuration Quest Material	104
5.4.14. XML Configuration Quest Condition	104
5.4.15. XML Configuration Quest Reward	105
5.4.16. XML Configuration Quest State	105
5.4.17. XML Configuration Quest Condition State	105
5.4.18. XML Configuration Statistic	106
5.5. KMEP-EJB Model	108
5.5.1. Model base	108
5.5.2. Model Behaviors	109
5.5.3. Model Dialog	111
5.5.4. Model Dialog Answer	113
5.5.5. Model Dialog Condition	114
5.5.6. Model Item	114
5.5.7. Model Message	115
5.5.8. Model Quest	117
5.5.9. Model Quest Materials	117
5.5.10. Model Quest Conditions	118
5.5.11. Model Quest Rewards	118
5.5.12. Model Quest State	119
5.5.13. Model Statistic	120
5.5.14. Model Utils	123
5.6. KMEP-EJB Services	123
5.6.1. Services	123
5.6.2. Administrative Services	125
5.7. KMEP Web Application Global Architecture	128
5.8. KMEP-WAR Configuration & Infrastructure	130
5.8.1. Configuration	130
5.8.2. Configuration mappings	131
5.8.3. Factories	133
5.8.4. Utilities	133
5.8.5. Servlets	134
5.9. KMEP-WAR Transfer Objects	134
5.9.1. TO General	135
5.9.2. TO Item	135
5.9.3. TO Map	136

5.9.4. TO Message	136
5.9.5. TO NPC Dialog	136
5.9.6. TO Player	136
5.9.7. TO Quest	137
5.9.8. TO Statistic	138
5.10. KMEP-WAR Actions	138
5.10.1.Actions General	139
5.10.2.Actions Item	140
5.10.3.Actions Map	141
5.10.4.Actions Message	142
5.10.5.Actions NPC Dialog	142
5.10.6.Actions Player	143
5.10.7.Actions Quest	144
5.10.8.Actions Statistic	145
5.11. KMEP-WAR iMode application	146
5.11.1.iMode Configuration	146
5.11.2.iMode Localization	146
5.11.3.iMode Factories	146
5.11.4.iMode Servlets	147
5.11.5.iMode HTML tags	150
5.11.6.iMode Views Formatter	156
5.11.7.iMode Views Helpers	159
5.11.8.iMode Views Utils	159
5.12. inTrack in use	160
5.12.1.Installation	160
5.12.2.Usage	165
5.12.3.Summary	166
5.13. Conclusion	166
6.1. Introduction	171
6.2. Contributions	171
6.2.1. Game platform	171
6.2.2. Social aspect	172
6.2.3. inTrack evaluation and refactoring	172
6.3. Project review	172
6.3.1. Mobile phone development	172
6.3.2. Model View Controller	175
6.3.3. inTrack	176
6.3.4. Double click	178
6.3.5. Development platform	178
6.3.6. Limits	179
6.3.7. Planning	180
6.4. Future work	185
6.4.1. Configuration tool	185
6.4.2. New quest types	185

6.4.3. Items	185
6.4.4. NPC	186
6.4.5. KMEP	186
6.5. Conclusion	187
APPENDIX	191
A. Game Rules	193
A.1. Introduction	199
A.2. Definitions	199
A.3. Sort of players	199
A.4. Avatar	200
A.5. Geo-localization	200
A.6. Base game	201
A.7. Game mechanisms	203
A.8. Quests	216
A.9. Rewards system	220
A.10. Map	220
A.11. Alerting	222
A.12. Conclusion	222
B. Specifications Review	223
B.1. Introduction	229
B.2. Use Cases	229
B.3. Conclusion	319
C. Configuration	321
C.1. Introduction	327
C.2. Properties files	328
C.3. XML Configuration files	337
C.4. Conclusion	377
D. MEP Installation	379
D.1. Introduction	385
D.2. Prerequisites	385
D.3. Database configuration	385
D.4. Project configuration	386
D.5. MEP configuration	388
D.6. Conclusion	388
E. Error Codes	389
E.1. Introduction	393
E.2. Principles	393
E.3. Categories	393
E.4. Conclusion	398
F. Mobile Phone Research	399
F.1. Introduction	403
F.2. Criteria	403
F.3. Mobile phone environment	404
F.4. Operators	404

F.5.	Final choice	405
G.	DoJa 5.1 Quick Start	407
G.1.	Introduction	413
G.2.	Download	413
G.3.	Installation	413
G.4.	Configuration	418
G.5.	NetBeans Module Installation	420
G.6.	First project in DoJa 5.1 with NetBeans 6.1	422
G.7.	Emulator	426
G.8.	Screen	431
G.9.	Conclusion	432
H.	Code Standards	433
H.1.	Introduction	437
H.2.	Natural language	437
H.3.	Code standards	437
H.4.	Conclusion	440
I.	Tools	441
I.1.	Introduction	445
I.2.	Communication	445
I.3.	Storage	445
I.4.	Integrated Development Environment	446
I.5.	Working diary	447
I.6.	Trac	447
I.7.	Others	447
I.8.	Tools and versions	448
I.9.	Conclusion	449
BIBLIOGRAPHY		453
GLOSSARY		457

Table of Figures

FIGURE 1 : GPS TRACKER – PART 1.....	41
FIGURE 2 : GPS TRACKER – PART 2.....	41
FIGURE 3 : GPS TRACKER – PART 4.....	42
FIGURE 4 : GPS TRACKER – PART 3.....	42
FIGURE 5 : GLOBAL ARCHITECTURE	51
FIGURE 6 : SYSTEM ARCHITECTURE	52
FIGURE 7 : CELL PHONE ARCHITECTURE	53
FIGURE 8 : MEP GLOBAL ARCHITECTURE	55
FIGURE 9 : CONCRETE MEP ARCHITECTURE	57
FIGURE 10 : DIALOGS MECHANISM.....	58
FIGURE 11 : NORMAL DIALOG MECHANISM	58
FIGURE 12 : QUEST PROPOSAL DIALOG MECHANISM	59
FIGURE 13 : QUESTION DIALOG MECHANISM	59
FIGURE 14 : PUZZLE ANSWER DIALOG MECHANISM.....	59
FIGURE 15 : ITEM PROPOSAL DIALOG MECHANISM.....	60
FIGURE 16 : QUEST ENDING MECHANISMS	60
FIGURE 17 : DIALOG WITHOUT STATE MEMORIZATION	60
FIGURE 18 : DIALOG WITH STATE MEMORIZATION	61
FIGURE 19 : MULTIPLE DIALOGS MECHANISM	61
FIGURE 20 : ITEM TYPE DESIGN	62
FIGURE 21 : ITEM SOURCE DESIGN	62
FIGURE 22 : MULTI ITEM TYPE DESIGN	63
FIGURE 23 : QUEST PARTS	64
FIGURE 24 : QUEST STATES	64
FIGURE 25 : MESSAGE ITEM GIFT STRUCTURE	67
FIGURE 26 : MESSAGE ITEM GIFT CONFIRMATION STRUCTURE	67
FIGURE 27 : MESSAGE ITEM EXCHANGE STRUCTURE	67
FIGURE 28 : MESSAGE ITEM EXCHANGE CONFIRMATION STRUCTURE	68
FIGURE 29 : MESSAGE QUEST GIFT STRUCTURE	68
FIGURE 30 : MESSAGE QUEST GIFT CONFIRMATION STRUCTURE	69
FIGURE 31 : ITEM GIFT MESSAGE FLOW	69
FIGURE 32 : ITEM EXCHANGE MESSAGE FLOW.....	70
FIGURE 33 : QUEST GIFT MESSAGE FLOW.....	70
FIGURE 34 : FIND/REACH A NPC QUEST DESIGN	71

FIGURE 35 : COLLECT ITEMS QUEST DESIGN	71
FIGURE 36 : SOLVE A PICTURE PUZZLE QUEST DESIGN	72
FIGURE 37 : UML – COMMON CLASSES	85
FIGURE 38 : UML – INTRACK / KEMP ERROR CODE BINDINGS	87
FIGURE 39 : UML – FORMATTER FACTORIES INTERFACES	88
FIGURE 40 : UML – FORMATTER INTERFACES	89
FIGURE 41 : UML – SERVER CONFIGURATION CLASSES.....	90
FIGURE 42 : UML – IDENTITY INTERFACES.....	91
FIGURE 43 : UML – XML CONFIGURATION CLASSES	92
FIGURE 44 : UML – XML CONFIGURATION INFRASTRUCTURE CLASSES	95
FIGURE 45 : UML – XML LOCATION CLASS	96
FIGURE 46 : UML – XML MEP ENTITIES CLASSES	96
FIGURE 47 : UML – XML BEHAVIOUR CLASSES.....	98
FIGURE 48 : UML – DIALOG XML CLASSES	99
FIGURE 49 : UML – DIALOG ANSWER XML CLASSES.....	99
FIGURE 50 : UML – DIALOG CONDITION XML CLASSES	100
FIGURE 51 : UML – DIALOG STATE XML CLASSES	100
FIGURE 52 : UML – ITEM XML CLASSES.....	101
FIGURE 53 : UML – MESSAGES XML CLASSES	102
FIGURE 54 : UML – QUEST XML CLASSES.....	103
FIGURE 55 : UML – QUEST MATERIAL XML CLASSES.....	104
FIGURE 56 : UML – QUEST CONDITION XML CLASSES	104
FIGURE 57 : UML – QUEST REWARD XML CLASSES.....	105
FIGURE 58 : UML – QUEST STATE XML CLASSES.....	105
FIGURE 59 : UML – QUEST CONDITION STATE XML CLASSES	105
FIGURE 60 : SOLUTION TO HAVE A UNIQUE STATISTIC NAME BY PLAYER.....	106
FIGURE 61 : UML – STATISTIC XML CLASSES.....	107
FIGURE 62 : UML – MODEL BASE CLASSES	108
FIGURE 63 : UML – MODEL BEHAVIOR CLASSES	110
FIGURE 64 : UML – MODEL DIALOG CLASSES.....	112
FIGURE 65 : UML – MODEL DIALOG ANSWER CLASSES	114
FIGURE 66 : UML – MODE DIALOG CONDITION CLASSES	114
FIGURE 67 : UML – MODEL ITEM CLASSES.....	114
FIGURE 68 : UML – MODEL MESSAGE CLASSES	116
FIGURE 69 : UML – MODEL QUEST CLASSES	117
FIGURE 70 : UML – MODEL QUEST MATERIALS CLASSES	117
FIGURE 71 : UML – MODEL QUEST CONDITIONS CLASSES.....	118

FIGURE 72 : UML – MODEL QUEST REWARDS CLASSES	119
FIGURE 73 : UML – MODEL DIALOG STATE CLASSES	120
FIGURE 74 : UML – MODEL STATISTICS CLASSES	121
FIGURE 75 : UML – MODEL UTILS CLASSES	123
FIGURE 76 : UML – SERVICES INTERFACES	124
FIGURE 77 : UML – ADMINISTRATIVE SERVICES INTERFACES.....	126
FIGURE 78 : KMEP WEB APPLICATION ARCHITECTURE	129
FIGURE 79 : UML – CONFIGURATION CLASSES	130
FIGURE 80 : UML – CONFIGURATION MAPPINGS CLASSES.....	131
FIGURE 81 : UML – IURLFACTORY INTERFACE.....	133
FIGURE 82 : UML – UTILITIES CLASSES	133
FIGURE 83 : UML – SERVLETS CLASSES	134
FIGURE 84 : UML – TO CLASSES	134
FIGURE 85 : UML – GENERAL TO CLASSES.....	135
FIGURE 86 : UML – ITEM TO CLASSES.....	135
FIGURE 87 : UML – MAP TO CLASS.....	136
FIGURE 88 : UML – MESSAGE TO CLASSES	136
FIGURE 89 : UML – NPC DIALOG TO CLASS	136
FIGURE 90 : UML – PLAYER TO CLASSES	137
FIGURE 91 : UML – QUEST TO CLASSES	137
FIGURE 92 : UML – STATISTIC TO CLASS	138
FIGURE 93 : UML – ACTIONS CLASSES.....	138
FIGURE 94 : UML – ACTIONS GENERAL CLASSES	139
FIGURE 95 : UML – ACTIONS ITEM COMMON BASE CLASSES.....	140
FIGURE 96 : UML – OTHER ACTIONS ITEM CLASSES	141
FIGURE 97 : UML – ACTIONS MAP CLASSES	141
FIGURE 98 : UML – ACTIONS MESSAGE CLASSES.....	142
FIGURE 99 : UML – ACTIONS NPC DIALOG CLASSES.....	143
FIGURE 100 : UML – ACTIONS PLAYER CLASSES	143
FIGURE 101 : UML – ACTIONS QUEST COMMON BASE CLASSES	144
FIGURE 102 : UML – OTHER ACTIONS QUEST CLASSES.....	145
FIGURE 103 : UML – ACTIONS STATISTIC CLASS	145
FIGURE 104 : UML – IMODE CONFIGURATION CLASSES.....	146
FIGURE 105 : UML – IMODE LOCATION CLASS	146
FIGURE 106 : UML – IMODE FACTORY CLASSES	147
FIGURE 107 : UML – IMODE CONTROLLERSERVLET CLASS	148
FIGURE 108 : UML – IMODE TAGS ABSTRACT CLASSES.....	151

FIGURE 109 : UML – TAGS BLOCK CLASSES.....	151
FIGURE 110 : UML – TAGS DOCUMENT CLASSES.....	151
FIGURE 111 : UML – TAGS FORM CLASSES	152
FIGURE 112 : UML – TAGS LINK CLASSES.....	153
FIGURE 113 : UML – TAGS LIST CLASSES	153
FIGURE 114 : UML – TAGS MEDIA CLASSES	154
FIGURE 115 : UML – TAGS STYLE CLASS	154
FIGURE 116 : UML – TAGS TABLE CLASSES	155
FIGURE 117 : UML – TAGS TEXT CLASSES	155
FIGURE 118 : UML – VIEWS FORMATTER ABSTRACTFORMATER CLASS.....	156
FIGURE 119 : UML – VIEWS FORMATTER DIALOG CLASSES	156
FIGURE 120 : UML – VIEWS FORMATTER ITEM CLASSES	157
FIGURE 121 : UML – VIEWS FORMATTER MESSAGE CLASSES.....	157
FIGURE 122 : UML – VIEWS FORMATTER QUEST CLASSES.....	158
FIGURE 123 : UML – VIEWS FORMATTER STATISTIC CLASSES.....	158
FIGURE 124 : UML – VIEWS HELPERS CLASSES	159
FIGURE 125 : UML – VIEWS UTILS CLASSES	159
FIGURE 126 : ORIGINAL PLANNING – PART ONE	181
FIGURE 127 : ORIGINAL PLANNING – PART TWO	182
FIGURE 128 : FINAL PLANNING – PART ONE	183
FIGURE 129 : FINAL PLANNING – PART TWO.....	184
FIGURE 130 : SIMPLE GAME MODE MAP SAMPLE	202
FIGURE 131 : QUEST GAME MODE MAP SAMPLE	203
FIGURE 132 : ATTACK FLOW	211
FIGURE 133 : THEFT FLOW	212
FIGURE 134 : ESCAPE FLOW.....	212
FIGURE 135 : ITEM FLOW	213
FIGURE 136 : COMBAT FLOW.....	213
FIGURE 137 : FOG OF WAR	220
FIGURE 138 : ABSTRACTBEHAVIOUR XML SCHEMA	338
FIGURE 139 : DIALOGBEHAVIOUR XML SCHEMA.....	338
FIGURE 140 : DIALOGSTATEBEHAVIOUR XML SCHEMA	339
FIGURE 141 : ITEMBEHAVIOUR XML SCHEMA	339
FIGURE 142 : ITEMINVENTORYBEHAVIOUR XML SCHEMA.....	339
FIGURE 143 : ITEMSOURCEBEHAVIOUR XML SCHEMA	340
FIGURE 144 : MESSENGERBEHAVIOUR XML SCHEMA	340
FIGURE 145 : NPCBEHAVIOUR XML SCHEMA	340

FIGURE 146 : PLAYERBEHAVIOUR XML SCHEMA	341
FIGURE 147 : QUESTDIARYBEHAVIOUR XML SCHEMA	341
FIGURE 148 : STATISTICBEHAVIOUR XML SCHEMA	341
FIGURE 149 : KMEPEntity XML SCHEMA.....	344
FIGURE 150 : LOADERS XML SCHEMA	346
FIGURE 151 : ABSTRACTDIALOG IN DIALOGS	347
FIGURE 152 : ABSTRACTCONDITION IN DIALOGS	347
FIGURE 153 : DIALOGTEXT XML SCHEMA.....	348
FIGURE 154 : DIALOGQUEST XML SCHEMA	348
FIGURE 155 : DIALOGQUESTFINISH XML SCHEMA	348
FIGURE 156 : DIALOGQUESTION XML SCHEMA	349
FIGURE 157 : DIALOGQUESTIONPUZZLE XML SCHEMA	349
FIGURE 158 : DIALOGITEM XML SCHEMA.....	349
FIGURE 159 : DIALOGS XML SCHEMA.....	350
FIGURE 160 : DIALOGSTATES XML SCHEMA.....	353
FIGURE 161 : FIRSTQUESTS XML SCHEMA.....	354
FIGURE 162 : ITEMS XML SCHEMA	355
FIGURE 163 : ITEMSOURCES XML SCHEMA	355
FIGURE 164 : ITEMTYPES XML SCHEMA.....	356
FIGURE 165 : ABSTRACTMESSAGE XML SCHEMA	357
FIGURE 166 : ITEMIDS XML SCHEMA	357
FIGURE 167 : MESSAGEQUESTGIFT XML SCHEMA.....	357
FIGURE 168 : MESSAGEITEMEXCHANGE XML SCHEMA.....	358
FIGURE 169 : MESSAGEITEMEXCHANGECONFIRM XML SCHEMA	358
FIGURE 170 : MESSAGEITEMGIFT XML SCHEMA	359
FIGURE 171 : MESSAGEITEMGIFTCONFIRM XML SCHEMA	359
FIGURE 172 : MESSAGEQUESTGIFTCONFIRM XML SCHEMA	360
FIGURE 173 : MESSAGES XML SCHEMA	360
FIGURE 174 : MULTIITEMTYPES XML SCHEMA	363
FIGURE 175 : NPCS XML SCHEMA	364
FIGURE 176 : PLAYERS XML SCHEMA	365
FIGURE 177 : QUESTS XML SCHEMA.....	366
FIGURE 178 : ABSTRACTQUESTMATERIAL XML SCHEMA.....	367
FIGURE 179 : MATERIALS XML SCHEMA.....	368
FIGURE 180 : ABSTRACTQUESTCONDITION XML SCHEMA	368
FIGURE 181 : CONDITIONS XML SCHEMA	369
FIGURE 182 : ABSTRACTQUESTREWARD XML SCHEMA	370

FIGURE 183 : REWARDS XML SCHEMA.....	370
FIGURE 184 : ABSTRACTQUESTIONSTATE XML SCHEMA	371
FIGURE 185 : QUESTCONDITIONSTATES XML SCHEMA.....	371
FIGURE 186 : QUESTSTATES XML SCHEMA	371
FIGURE 187 : ABSTRACTSTATISTIC XML SCHEMA	373
FIGURE 188 : COUNTABLE XML SCHEMA	373
FIGURE 189 : STATISTICS XML SCHEMA	373
FIGURE 190 : MAPPINGS ROOT XML SCHEMA.....	375
FIGURE 191 : MAPPINGS ACTION XML SCHEMA	375
FIGURE 192 : CONNECTION POOL CREATION	386
FIGURE 193 : CONNECTION POOL ADVANCE PROPERTIES	386
FIGURE 194 : JDBC RESOURCES CONFIGURATION	386
FIGURE 195 : PERSISTENCE CONFIGURATION	388
FIGURE 196 : LANGUAGE CHOICE.....	413
FIGURE 197 : INSTALLATION WELCOME SCREEN	414
FIGURE 198 : AGREEMENT LICENSE	414
FIGURE 199 : PATH CHOICE	415
FIGURE 200 : TYPE OF INSTALLATION.....	415
FIGURE 201 : COMPONENTS TO BE INSTALLED.....	416
FIGURE 202 : PATH OF NETBEANS & ECLIPSE MODULE	416
FIGURE 203 : ECLIPSE PATH CHOICE.....	417
FIGURE 204 : SUMMARY OF THE INSTALLATION	417
FIGURE 205 : INSTALLATION	418
FIGURE 206 : .JAM FILES ASSOCIATION	418
FIGURE 207 : FILES IN LIB/118N DIRECTORY	419
FIGURE 208 : DoJA 5.1 SDK WITH EMULATOR.....	419
FIGURE 209 : DOWNLOADED TAB FROM PLUGINS	420
FIGURE 210 : PLUGIN CHOICE	420
FIGURE 211 : DoJA 5.1 PLUGIN CHOICE.....	421
FIGURE 212 : INSTALLATION OF THE PLUGIN.....	421
FIGURE 213 : SIGNED WARNING	422
FIGURE 214 : NEW DOJA 5.1 PROJECT	422
FIGURE 215 : PROJECT CONFIGURATION.....	423
FIGURE 216 : HELLOWORLD CLASS CREATION	423
FIGURE 217 : DOJA PROPERTIES FILE	424
FIGURE 218 : DoJA 5.1 PROJECT CONFIGURATION (ADF CONFIGURATION).....	425
FIGURE 219 : EMULATOR VIEW FROM THE HELLOWORLD PROJECT.....	426

FIGURE 220 : CONSOLE OUTPUT OF THE HELLOWORLD PROJECT.....	426
FIGURE 221 : DEVICE CONFIGURATION TAB.....	427
FIGURE 222 : DEVICE CONFIGURATION CREATION	428
FIGURE 223 : DEVICE SELECTION.....	428
FIGURE 224 : NATIVE DATA EDIT SCREEN	429
FIGURE 225 : TRUSTED APPLICATION	430
FIGURE 226 : TRUSTED CONFIGURATION	430
FIGURE 227 : RUN SETUP CONFIGURATION.....	431
FIGURE 228 : SCREEN RESOLUTION CONFIGURATION	431

Table of Tables

TABLE 1 : PLAYER CHARACTERISTICS	204
TABLE 2 : USE CASES STATES.....	230
TABLE 3 : INTRACKCONFIG PROPERTIES	329
TABLE 4 : SERVERCONFIG PROPERTIES	330
TABLE 5 : APPLICATIONS PROPERTIES.....	331
TABLE 6 : IMODECONFIG PROPERTIES	334
TABLE 7 : IMODETEXT PROPERTIES.....	335
TABLE 8 : SERVICELOOKUP PROPERTIES.....	336
TABLE 9 : BEHAVIOURS TAGS	344
TABLE 10 : KMEPENTITY TAGS.....	345
TABLE 11 : LOADERS TAGS	347
TABLE 12 : DIALOGS TAGS.....	353
TABLE 13 : DIALOGSTATES TAGS.....	354
TABLE 14 : FIRSTQUESTS TAGS	355
TABLE 15 : ITEMS TAGS	355
TABLE 16 : ITEMSOURCES TAGS.....	356
TABLE 17 : ITEMTYPES TAGS.....	357
TABLE 18 : MESSAGES TAGS	362
TABLE 19 : MULTIITEMTYPES TAGS	364
TABLE 20 : NPCS TAGS	364
TABLE 21 : PLAYERS TAGS	366
TABLE 22 : QUESTS TAGS	367
TABLE 23 : QUESTMATERIALS TAGS.....	368
TABLE 24 : QUESTCONDITIONS TAGS.....	369
TABLE 25 : QUESTREWARDS TAGS	370
TABLE 26 : QUESTSTATES TAGS	372
TABLE 27 : STATISTICS TAGS	374
TABLE 28 : MAPPINGS TAGS	377
TABLE 29 : ERROR CATEGORIES.....	394
TABLE 30 : TOOLS UNDER WINDOWS XP.....	448
TABLE 31 : TOOLS UNDER MAC OS X	449

Tables of codes

CODE 1 : FORMATTER CREATION	89
CODE 2 : XML LOADER – LOAD METHOD.....	93
CODE 3 : ID COLLECTION ANNOTATION CONFIGURATION.....	98
CODE 4 : MESSAGES ANNOTATION CONFIGURATION	101
CODE 5 : QUEST ELEMENT ID CONFIGURATION ANNOTATION.....	103
CODE 6 : STATISTIC ANNOTATION DEMONSTRATION.....	106
CODE 7 : STATISTIC ANNOTATION CONFIGURATION SAMPLE.....	106
CODE 8 : DATA MANAGER IMPLEMENTATION.....	128
CODE 9 : ABSTRACTITEMACTION PROCESS METHOD	141
CODE 10 : ABSTRACTQUESTACTION PROCESS METHOD.....	144
CODE 11 : PROCESSREQUEST CODE PART	147
CODE 12 : GETACTION CODE PART	148
CODE 13 : UPDATELOCATION METHOD CODE	149
CODE 14 : CHECKACCESS METHOD CODE	149
CODE 15 : FORWARDTOVIEW METHOD CODE.....	150
CODE 16 : INTRACK DEPENDENCY	160
CODE 17 : MAIN PROJECT NBACTIONS.XML.....	161
CODE 18 : EJB POM.XML	164
CODE 19 : EJB NBACTIONS.XML	164
CODE 20 : EJB PERSISTENCE.XML.....	165
CODE 21 : PROFILE IN EAR PROJECT.....	387
CODE 22 : DEPLOY ACTION	387
CODE 23 : INTRACK CONFIGURATION.....	388
CODE 24 : CODE OF THE CLASS HELLOWORLD	424
CODE 25 : CLASS HEADER JAVADOC.....	439
CODE 26 : CLASS HEADER JAVADOC SAMPLE.....	439
CODE 27 : LONG CODE PRESENTATION	439

heig-vd

Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

K-MEP

Introduction

Laurent Prévost

The logo for RITSUMEIKAN, featuring a large white letter 'R' on the left and the word 'RITSUMEIKAN' in a smaller, white, sans-serif font to its right, all set against a dark red rectangular background.

R RITSUMEIKAN

Table of Contents

1.1. INTRODUCTION	5
1.2. PROJECT ORIENTATION	5
1.3. DEFINITIONS	5
1.3.1. "Exergaming"	5
1.3.2. Pervasive games	6
1.3.3. Geo-localization services	6
1.3.4. Ubiquitous	6
1.4. TECHNOLOGIC CONTEXT	7
1.4.1. inTrack	7
1.4.2. Internet on mobile phones	7
1.5. CONTRIBUTIONS	8
1.5.1. Game platform	8
1.5.1.1. Quests	8
1.5.1.2. Items	8
1.5.1.3. Dialogs	9
1.5.2. Social aspect	9
1.5.2.1. Culture	9
1.5.3. inTrack evaluation and refactoring	9
1.6. DOCUMENT STRUCTURE	10

1.1. Introduction

Kyoto Mobile Exergaming Project (KMEP) tries to build a platform to develop games based on geo localization and exergaming games. With the usage of the geo localization, the games become pervasive games and mix the real and virtual worlds into the same game. It is important to understand the basic definitions that concern the game.

1.2. Project orientation

KMEP offers a new way of entertainment in real life with mobile device. Idea is to offer to players a lot of possibilities to discover a city – Kyoto in this project – under another kind of regards. Joining entertainment and physical exercises in the same game produces an exergaming.

Application on mobile device offers the possibility to do a game in the real world with a persistent and virtual universe. Players can interact in the game by their actions in the real life. Localization will be used to add the real dimension to the game and help the player to discover old buildings and historical places or other interesting places.

Exercises in real life are important to conserve a good fitness. Use of bicycle in the game is a way we can explore to force players moving and doing exercises. This goal must to be natural and not a constraint. It must to be pleasant. Walking is not so bad too. The principal goal to reach is to invite the players moving without idea of constraining task.

Cultural aspects are important to offer a way to discover historical information about the city with a pleasant approach. The game can be used by indigenous or not. Kyoto is especially a good place for that. The city count about 2'000 shrine and temples. Temples are usually with an entrance fee and shrine not.

1.3. Definitions

This section will try to bring some answers about games and relatives terms about the project.

1.3.1. “Exergaming”

What is “exergaming”? This is the contraction of the word “exercise” and “game”. Globally, the meaning is “doing some exercise during practice of a game”.

Due to our developed societies, we prefer to save time to leisure rather than sports or exercise. The consequence is that we have some obesity problems in the developed societies. Idea of “exergaming” is to produce some game with the obligation to practice some exercise.

In this case, the game must provide possibility to practice exercise without any constraint. It means that participants do not realize that they do some exercises. Activities have to be pleasant and easy to do by ourselves. This definition is based on [7]

1.3.2. Pervasive games

The goal of pervasive game is to join a virtual world with our real world. There are many manners to combine real world and virtual worlds. One of them is to juxtapose player's positions in real world into a virtual world.

Others possibilities could be weather detection and application in virtual world based on real world. When players play their game, the weather reflects what players see in real world.

Another approach could be to move a real object and same move in the virtual world. In facts, that is really near the player's catching position. These explanations are based on [2], [3] and [4]

1.3.3. Geo-localization services

One of the most known services in this domain is the GPS – Global Positional System. This service helps us to find our localization in real time. The localization uses satellites and peripherals to find localization. Advantage of this system is that it is available all around the world but when you are in a building, signal does not pass through walls. In this situation, peripherals have to estimate the position based on the past data. The precision is quite good. The other problem can be the reflection in the very big city between the buildings.

GPS is not the only way to find our localization in space. Some other systems exist but they are not available all around the world. WiFi can be a solution but a system based on WiFi must to know more information about WiFi Access Points to determine the user position.

Geo-localization is an expression for many applications. Very different services can be imagined with the use of geo-localization. One of the most known services is car positioning with GPS. It is very usual to find cars with this functionality. We can imagine a lot of application in many domains for applications like touristic industry, shop guide, and games. These definitions are based on [11], [12] and [13]

1.3.4. Ubiquitous

Ubiquitous computing is general definition for pervasive game and many others mechanisms (pervasive computing is a synonym) that imply everyday electronically systems. Idea is to offer more interaction with ordinary objects like refrigerators, lamps, sensors... to end users.

The core concept is to offer a new way to use information with objects of everyday. In example, you can manage the content of a refrigerator which it can remembers quantity of labeled products. The fridge can warn you when some products are near to be empty or it can propose a meal with current products. This one example of Ubiquitous computing but we can imagine an infinite number of solutions. This definition is based on [1]

1.4. Technologic context

Japan offers the best place to do a game based on mobile phone devices. The technology is largely in advance that the European countries but it seems there are many limitations with the usage of the available technologies.

Each mobile phone provider offers its own services not compatible with the others. Idea is to get the maximum of customer and keep them with the services. When a customer has the habits to use a certain service, it is difficult for him to change the provider if the service is not available in the other provider.

1.4.1. inTrack

Actually, the Japanese market seems to offer more mobile phones with embedded GPS. It is especially appreciable to build applications with localization services. To use the localization in this project, idea is to use the inTrack middleware to manage the localization data.

They are two ways to integrate the localization service with inTrack. The first is to use directly some services of the middleware directly in the application. This integration method is transparent for the application on the mobile phone because the inTrack integration is done on the server.

The second manner is to integrate two modules of inTrack. The first one is only dedicated to update the locations and can be directly integrated with the mobile phone application. The second one is integrated with the server application and is used to get the localized data.

The way to use inTrack middleware depends essentially on the technology possibilities offered by the mobile phone used for the project.

The middleware itself is independent of the final technology used to develop the application. Some communication mechanism must to be set but this is possible with a large range of technologies.

1.4.2. Internet on mobile phones

The Japanese people use a lot their mobile for the internet connection. First, they use mail rather than SMS. It seems that some mobile do not support the SMS functionality. These one, do not apply to be used outside of Japan.

Each provider has many different offers for the packet charge rates included in the monthly fees. The different offers are relatively complicated. It seems that each person can have a personal offer to suit his needs. The best thing to do is to go in mobile phone shop room of the preferred provider to get the information. This research of information takes a long time to do and a good knowledge in Japanese is needed.

There are many different services based on internet services. Some of them use the embedded GPS, others use the camera... Each provider tries to offer the services that their concurrent does not have.

1.5. Contributions

The contributions focus on the principal aspects of the project. They are three principal axes for that. The game platform is the base system to support the project. The social aspect is the non-technical aspect and the inTrack part is the external part. All this part allows building the entire project.

1.5.1. Game platform

To allow the creation of different games for specific context in different cities, it is necessary to build a game platform with a high degree of configuration. The flexibility to offer with the configuration is important for that.

Basically, the game platform must be independent from the context where it is deployed and which data it uses. It is necessary to reduce the constraints to the minimum. It has to offer sufficient mechanism for the diversity of the tasks that a player can do in the game.

The platform has to offer the mechanism to do some quests, collect items, meet and discuss with non-player characters and notify the players about some events concerning the actions done in the game.

1.5.1.1. Quests

The quests allow building some mini stories to immerse the player inside the universe of the game. In this condition, the quests are not sufficient by themselves. It is necessary to build also some characters inside the game to interact with the player.

The characters can discuss with the player. With this mechanism in addition of quest, it is very possible to build some scenarios to immerse the player inside the game. Another way to explore is to build mini stories related to the places to visit and explore. The stories can take the historical view or modern view. It depends only of the imagination of the writer.

1.5.1.2. Items

To add some possibilities to the quests, virtual items can be a great idea. They take place on real geographic points. The players can collect them and give or exchange them with the other nearest players. This is particularly useful to build some quests with large needs of players.

For example, a quest can ask a player to collect some items near some points that are relatively far from the others. The player can share his quest with another player and each one can go to a distinct point to collect the items for him and the other player. This

cooperation does not need any technical concept rather than the possibility to share the quests and exchange/give the items.

1.5.1.3. Dialogs

The game platform needs to offer non-player characters to help the player and offer a context to the game. The characters are like human players but controlled by the computer. In this case, the non-player characters are only static and have only a role to help the player with quest proposals and so on.

The player must to be able to communicate with the non-player characters and for that a dialog mechanism is required. The mechanism must to be sufficiently flexible to bring great and interesting dialogs. The most important thing is that the dialog must not to be linear. This is important. A non-linear dialog is more interesting than a linear one.

1.5.2. Social aspect

For this project, an interesting way to study is the social aspect and how to do the interaction between the players. The first ideas are to use the peripherals of the mobile device to check the proximity of the players. Some functionality can be available only if the players can bring the proof of their proximity via the mobile device peripherals.

Next ideas are to build some game mechanism to invite the player to cooperate between them. For example, they can solve a rebus together. Each player receives a part of the rebus and must to find the other players with other parts. After this finding part, they have to find the solution by assembling their mobile device together to find the solution.

The quests are very important to bring the social interaction in the game and to allow the player visiting many places but simple quests without player interactions can be a great start to create the simplest quests.

1.5.2.1. Culture

Another thing to notice is that the game has specially the culture factor implicitly. This fact is due of the geo localization. To play to the game, the player must to walk or simply move in the city where the game is deployed. In this situation, the player can visit many places that he does not visit usually.

Usually, people do not know their own town as well as they can. Sometimes, they have a better knowledge about the culture of other cities than their own city. For that, this project could be a great opportunity to discover or rediscover a city.

1.5.3. inTrack evaluation and refactoring

To do the localization, the inTrack platform could be used. This is a very great opportunity to use a platform located in Switzerland during the development and tests in Japan. The

principal goal of that is to validate the inTrack model and allows refactoring the platform for the needs of the customer.

KMEP is considered as a customer for this platform. It uses only the services offered by inTrack platform to add the localized services in the project without doing the work of localization inside the project.

1.6. Document structure

This document is structured in six chapters that covered the entire project. There are also some appendixes to complete the main documentation.

Introduction

The introduction contains some data to be introduced in the document. Some subjects are just approached and will be more detailed later in this document.

Project scope

The project scope chapter introduces the mechanism of the game and the technical approach to build the game platform. It introduces the technical aspect for the game management too.

User Experience

This chapter brings an overview of the game with user explanation and technical overview for specifics interesting points.

Architecture

The architecture chapter provides the view of the technical architecture for the different subsystems of the platform developed.

Implementation

After the view of the global architecture, this chapter goes further in details and brings to the reader the view of many implementation details with some samples.

Conclusion

Finally, all the aspect covered in the project will be discussed in the final chapter of the document. The project review will do there too.

heig-vd

Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

K-MEP

Project Scope

Laurent Prévost

The logo for RITSUMEIKAN, featuring a large white letter 'R' on the left and the word 'RITSUMEIKAN' in a smaller, white, uppercase sans-serif font to its right, all set against a dark red rectangular background.

R RITSUMEIKAN

Table of Contents

2.1. INTRODUCTION	15
2.2. GAME PLATFORM	15
2.2.1. Kmep Entities	15
2.2.1.1. Behaviors	15
2.2.2. Quests	16
2.2.3. Items	16
2.2.3.1. Item sources	16
2.2.4. Dialogs	17
2.2.5. Messages	17
2.2.6. Summary	17
2.3. SERVICES	17
2.3.1. User services	17
2.3.1.1. Player service	17
2.3.1.2. Kmep Entity service	18
2.3.1.3. Item service	18
2.3.1.4. Message service	18
2.3.1.5. Quest services	18
2.3.1.6. Dialog service	18
2.3.1.7. Tracking service	18
2.3.2. Administration services	18
2.3.2.1. Data service	18
2.3.2.2. inTrack service	19
2.4. QUESTS	19
2.4.1. Reach a place	19
2.4.2. Collect items	19
2.4.3. Solve picture puzzles	19
2.5. CONFIGURATION	19
2.6. INTRACK	20
2.7. KMEP	20
2.8. CONCLUSION	21

2.1. Introduction

In this chapter, the scope of the project is described largely with the possible solutions to build the game platform and the implementation of a real game. The game management is described too.

2.2. Game platform

To build the game platform, the EJB technology was naturally imposed by the context. During the pre-project, a game called “inTrackSower” was developed especially for the “Summer School”. This game uses inTrack directly embedded inside the platform and is based on the EJB technology like inTrack.

In this context, the fastest and easiest way to develop the MEP platform is to use the same base architecture to start the development of the platform.

Before the start of the project in Japan, the tools were configured to use the same layout project as “inTrackSower”. NetBeans was installed with maven and SVN. With this configuration, the project can be build locally or directly on the Swiss server where inTrack is.

The reason of this choice is also that the Switzerland server is especially usable with Java Applications. On the server, a GlassFish server is installed and can host some applications.

2.2.1. Kmep Entities

The game platform has to manage different types of entities. They are the items, items sources, players and non-player characters. To manage easily all these entity, the possibility to have only one class is planned.

The structure for that is the entities own some behaviors that allow building the specific or shared mechanisms like item inventory, quest diary, dialog... In addition, a field to define the type can be used to find quickly and easily the type of the entity. This approach avoids building behavior rules system to find the entity type.

2.2.1.1. Behaviors

The behaviors bring the relation between the entities and the game mechanism of the platform. An entity without any behavior is useless for the platform. The behaviors bring all the possible interaction.

The behaviors are needed to manage the items, dialogs, quests... They are very important; they are the kernel of the platform with the entities.

2.2.2. Quests

Quests are the short stories and the goals that a player can do to play to the game. In fact, they are two things to see for the implementation. The first thing is the need to persist the quest type. The quest type is the factory to offer the quest to the player.

The second thing is the quest state. This is important to remember for each distinct player the state of the quest. More than one player can own the same quest but the state must to be unique for each player otherwise when a player finish the quest all the players finished the quest too.

With the quests and their corresponding quest states, this is possible to build a quest system to offer mini stories and tasks to do to the players.

The quests need some elements to work. Sometimes, a quest wants to offer some materials to the player to help him in his quest. For that, the quests need to have a mechanism to offer these materials.

The same thing is true for the conditions to reach to finish a quest. Idea is to build the quest around some quest conditions that the players must to reach before to finish a quest. Theses conditions can be configured to create specific and different quests.

Finally, at the end of the quest, the player can be rewarded or not. The rewards can be configured like the conditions or the materials. This is not very important but it can be useful to give an item in the perspective of a future quest for example.

2.2.3. Items

The items are important to add some functionality in the game. To implement these items, it is necessary to distinguish some element. First, the item is from a type. The item type allows optimizing the storage of the data of the items.

Items are entities. For that, it is necessary to add an item behavior to define from which item type they are. In addition, the players can store them in their item inventory. Some mechanism to store the items are necessary too.

2.2.3.1. Item sources

To allow the creation of items regularly in the game for the quests for example, it is necessary to create a special entity that produce the items. A delay can be configured to allow the reappearance of a new item. A timer service is required to allow the reappearance of these items after they are collected.

2.2.4. Dialogs

The dialogs are necessary to help the player to immerse in the game and to get the quests. This is the principal mechanism to get new quests. To add the possibility to discuss to an entity, a dialog behavior must to be created.

The dialogs have to be build in a sort of tree (more a graph than a tree) to allow a better interaction with the player. A dialog behavior has also the possibility to have a main dialog and some conditional dialogs only available when some conditions are reached.

To avoid replay of dialog, a mechanism to save the state of the dialog is needed. It allows storing the state for a specific dialog and specific entity that use the dialog.

It is necessary to create some different types of dialog to offer text, quest, quest ending, question and puzzle question dialogs. Each of these dialogs have different requirement depending on the situation of the player and other parameters.

2.2.5. Messages

A message system is required. This system is a mixed of notification and message system. It has to manage the interaction of asynchronous actions did by the players (for example a player that sends an item to another player).

This system must allow sending messages between players depending on the situation. The players must not have idea that a message is sent to the other player directly by writing a message. The message system must to built-in in the appropriate functionalities.

2.2.6. Summary

With these mechanisms, the bases for the MEP are offered. The other mechanism to offers will be described in the based services. The business work of the MEP is shared between the MEP components and its services.

2.3. Services

A set of services have to be offered to access of the MEP. These services contain a part of the business work to do for the game mechanism. The both part composed the MEP.

2.3.1. User services

The user services are dedicated for the game actions. They are especially used by the players during the game.

2.3.1.1. Player service

The player service allows managing the players. It offers the login/logout, register, avatar management and this kind of actions. It is necessary also to find the players by some criteria.

2.3.1.2. Kmep Entity service

This service allows managing the entities (include player management but only for the creation part, the player service has to delegate some management tasks to this service). It allows finding entities and behaviors relative to the entities.

2.3.1.3. Item service

The item service allows managing the items of the game but also the inventory of the players. It offers the mechanisms to manipulate the different inventories and the different actions between the players like exchange/give items.

2.3.1.4. Message service

This service allows sending message between player and managing them. This is possible to transfer a message from a player to another player. Like a mail client, there is the possibility to get the messages and change the status of “read” from a message.

2.3.1.5. Quest services

The quest service manages the quests and the states relative to the quests. It allows finding the different quests and quest states. It manipulates the quest diary from the different players to add and modify the quest states depending on the situation. It offers a way to evaluate all the quest states at each moment of the game.

2.3.1.6. Dialog service

This service allows managing the dialog states between the players and the others entities in discussion. It offers the different mechanism for the different type of dialogs. It allows finding the starting dialog for a dialog flow and the other dialogs parts.

2.3.1.7. Tracking service

The tracking service is façade of the inTrack service. It allows some consolidate action for the location services. It allows the binding between local and remote entities. It offers the location update possibility too.

2.3.2. Administration services

These services are dedicated for the administrative tasks and a non-public access. It will be accessed under certain conditions.

2.3.2.1. Data service

The data service allows the loading of the data at the deployment phase. This the way to populate the database with the real data for the quests, items and so on.

2.3.2.2. inTrack service

The inTrack service is provided by the inTrack team to manipulate the inTrack data (local and remote). It offers all the methods needed to bind/unbind the entities between the two platforms.

2.4. Quests

To build the quests, the materials, conditions and rewards have to allow multiple configurations. These different configurations offer to create different kind of quest.

2.4.1. Reach a place

This type of quest is the simplest one. It does not require specific configuration in addition of the dialogs for the non-player characters. This type of quest allows validating some mechanism like quest ending, quest proposal...

They are not especially interesting to do but it is a good start to invite the player to move in another place to get other quests.

2.4.2. Collect items

This kind of quest is more interesting for the player because he has to collect items on his path. He has to manipulate the game through his device.

It is also a good way to check the validity of the mechanism around the items like the item inventory, collecting items, moving items...

2.4.3. Solve picture puzzles

This third type of quest is particularly interesting for the player. He has to solve a picture puzzle (also called rebus) but he receives only one piece of the puzzle. He has to find other players with the same quest to solve. Each player receives a different piece of the puzzle. The addition of all pieces allows finding the solution.

The players have to find the other players and interact with them in the real world. It is a good approach to add some social aspect to the game. It is not enough just to play the game, the players must to discuss with the other players and find together the solution.

2.5. Configuration

With the complexity and diversity of the data, it is not possible to use built-in functionality to load the game data. There is a real need to build a mechanism to allow loading the data when the server is deployed.

The first idea is to use JAXB – a built-in technology from the Java platform that allows creating classes to load or save XML documents. With this technology, it is possible to write XML Schema document to constraints the data that could be loaded by the loading process.

It is also possible to validate the XML documents before loading the document. They are some limitations to the checks like the cross references between multiple documents but that is probably the best way to build the loading process.

The loading mechanism allows using a central file that describes the files to load. The others files could be write by everyone that knows the XML language and the specifications of the different format used in the project. In this situation, the data creating process could be delegate to someone external from the project.

2.6. inTrack

There are some ways to use inTrack in the project. It is possible to split the usage of inTrack into two separate pieces. The first one allows updating the location and the second one allows managing them.

In this project, it is not possible to separate the both element because the mobile phone cannot become the tracking module and update its position alone. The reason is that it is not possible to use the GPS directly from an iAppli on the mobile phone for non-trusted organization.

In this context, the best solution is to merge the two parts into the same part for the MEP. The locations and management will be done in the same place and under the responsibility of MEP. MEP uses the different mechanisms offered by inTrack to manage and update the locations.

This approach is not so bad in regard of the first one. With a complete integration, the MEP can be used independently from the technology used for the client. The performances are probably better too because it is possible to group some communication in one time rather than communicate from the client to inTrack and MEP.

2.7. KMEP

Kyoto Mobile Exergaming Project wants to offer an implementation of the MEP in a real context. The city of Kyoto is especially good for that. There are many places to visit and explore. The people use often their bicycle to move in the city.

For a first deployment, the data to create will be especially oriented around some places like the Ritsumeikan University, Sanjo Street and the Kamo River. The reason is these places are easy to use for testing the application.

The MEP will be deployed on the same server than inTrack in Switzerland. It is interesting to notice the latency of the communication between Japan and Switzerland. The communication between inTrack and MEP will be shorter with this architecture too.

2.8. Conclusion

With all the mechanisms described in this chapter, it is possible to build the MEP and its implementation with KMEP. All the necessary is covered to offer a flexible platform to create new games for a specific context.

heig-vd

Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

K-MEP

User Experience

Laurent Prévost



R RITSUMEIKAN

Table of Content

3.1. INTRODUCTION	27
3.2. GAME PLAY	27
3.2.1. First Connection – The beginning	27
3.2.2. First Quest – Reach a place	29
3.2.3. Second Quest – Carry an item	30
3.2.4. Third Quest – Collect items	32
3.2.5. Another quest – Puzzle quest	38
3.3. GPS TRACKER	41
3.4. CONCLUSION	43

Table of Figures

FIGURE 1 : GPS TRACKER – PART 1.....	41
FIGURE 2 : GPS TRACKER – PART 2.....	41
FIGURE 3 : GPS TRACKER – PART 4.....	42
FIGURE 4 : GPS TRACKER – PART 3.....	42

3.1. Introduction

This part of document shows a real application of the platform MEP with the game KMEP. Some quests are shown step-by-step with many explanations.

The next paragraphs in these sections include many pictures taken in real life by a camera directly on the phone. It was especially difficult to get the phone with one hand and take the pictures with the other. The result is that the pictures are not so good. The quality is quite poor.

There is also a lot of description with NPC in the text. The “it” form will be used to differentiate real player from the NPCs that take place only in the virtual world.

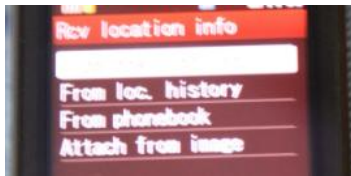
3.2. Game play

The major game play is demonstrated with four different quests. Each quest uses the principal game mechanism as Dialogs, Quests and Items.

3.2.1. First Connection – The beginning



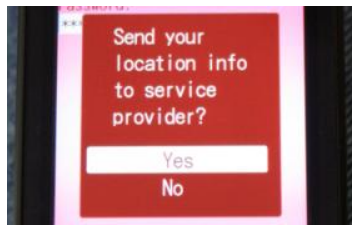
The first connection is a little different from the others connection because of the introduction to the game. There is an introduction dialog to help the player to begin the game and understand the basic mechanism of the quests.



However, for each connection to the system, a location update is required. For that, the built-in system asks the player to provide the location choice. The player can choose the location he wants to send to the system. He has the choice between Current Location, From History, From Phonebook and From Image. Usually, the player has to choose current location.



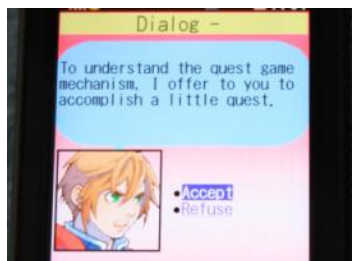
In the case of “use current location”, the player sees the get location process. He can push “use” button when the location is available or wait the end of the process. The number of stars indicates the accuracy of the location. Remember that the provider use A-GPS for Augmented GPS. It is especially dedicated to the mobile phones to help the location by the service network provider.



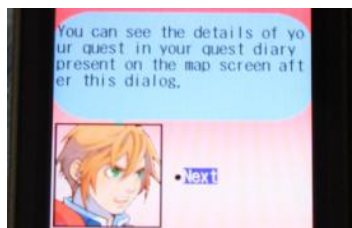
Finally, the system asks the player if he wants to send his location. The player must choose “Yes”. If he refuses, the connection action does not work. The system does not send any data if the “No” action is selected.



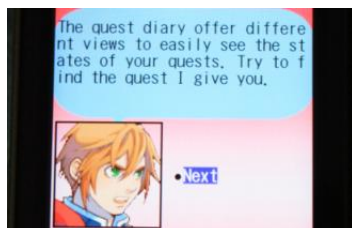
After the login process with localization and in the case of the first connection, the player sees the first dialog. His avatar is used as the speaker. The dialog begins with a welcome message. The player can go to the next dialog with the “Next” link.



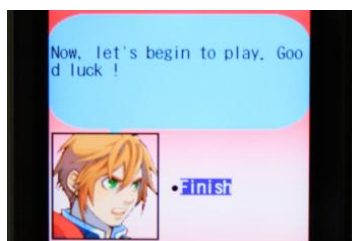
To understand the quest mechanism, the system offers a first quest to the player. There are no details about the quest to force the player to use the mechanism like quest diary to see his quests. To accept the quest, he can click on “Accept” link.



The dialog explains to the player where to go and how to find the information about his first quest. He can continue the dialog by clicking the “Next” button.

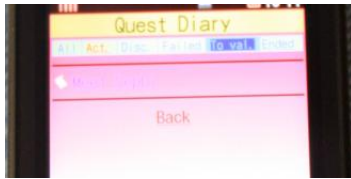


There is other explanation about the Quest Diary available on the principal screen of the game. The player can continue the dialog with “Next” link.

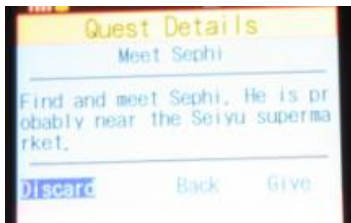


Finally, the player is invited to begin to play. He can finish the dialog with “Finish” link. This first dialog is only shown at the first connection.

3.2.2. First Quest – Reach a place



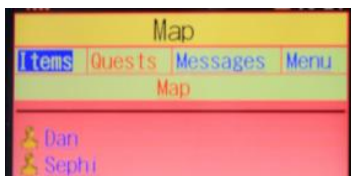
The Quest Diary show to the players the different quests he has in the different state. In this situation, the player has only one quest in the state “To Validate”.



The Quest Details show to the player the information of the quest. This quest proposes to the player to find the NPC called “Sephi”. Some information is added to help the player to find it. The supposed position is indicated.



The player receives the map of the Sanjo Street with the different elements only for tests purposes. He also carries a GPS Tracker to get the exact path he does during the tests. The GPS Tracker and its data will be described later in this part of document.



Now the player is on the road with his bicycle. He has reached the supposed place where “Sephi” seems to be. By luck, “Sephi” is there. There is also another NPC called “Dan”.



The place is the Seiyu Supermarket in the Sanjo Street in Kyoto.



The player tries to discuss with "Sephi". "Sephi" recognize that the player found it. But...

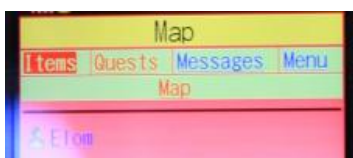


It seems this NPC is not convenient. It has nothing to do with the player. The dialog is short and useless actually. However, remember that there is another NPC near it. The player has to discuss with it too.

3.2.3. Second Quest – Carry an item



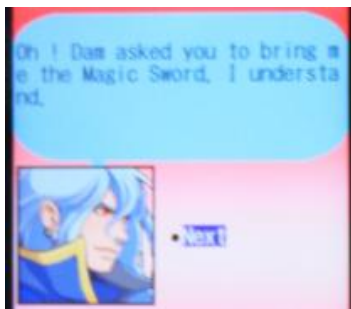
This NPC called "Dan" seems to be more convenient with the player. It has a task for the player. It asks the player to carry a sword to another the NPC called "Elom". The player can accept by clicking "Accept" link. He can also see the details by the corresponding link. There is also geographic information to help the player to find "Elom".



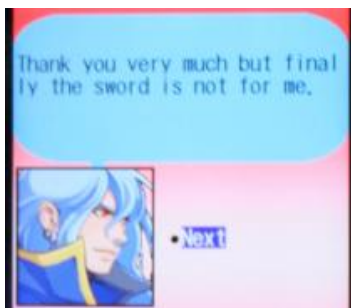
The player found "Elom". It seems to be alone at this place. Probably it is waiting for the sword.



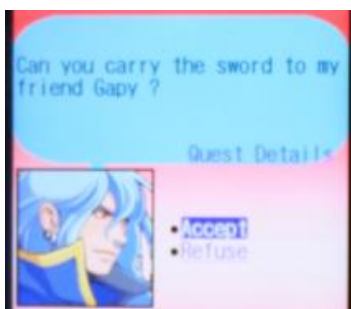
“Elom” is at the East entrance of the Sanjo Street. During the search of “Elom”, the localization has some trouble. The player has to do the localization three times before retrieving “Elom” on his device. The accuracy of the device was not enough during this part of the adventure.



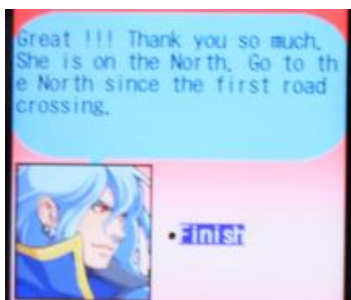
The dialog with “Elom” begins with the keys introduced in the quest. The sword is indicated and “Dan” also. The player can continue the dialog. With this dialog, the quest is ended. The processing for the quest ending will be done after the player click on “Next” link.



A normal dialog prepares to introduce a new quest. “Elom” says that the sword is not for it but.



“Elom” asks the player to carry the sword to another NPC. For the player, it seems to be a continuous quest but for the system, this is two different quests. The construction of dialogs and quests can introduce many subtleties cannot directly build in the system. The multi-part quests cannot exist in the system but with this kind of construction, it is possible to do.



The player has accepted the quest. The quest is added to the Quest Diary as a new quest. This is the problem for this kind of multi-part quest construction. Each part of quest is considered as a new and independent quest. “Elom” tells to the player the place to reach to find the NPC called “Gapy”.



The player reached the place but the dialog with “Gapy” was not the one expected. There is no capture about the dialog.

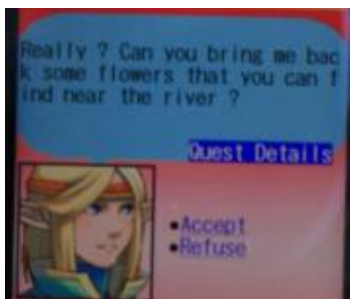
The configuration of the quest was not done correctly and the sword was not given again to the player from “Elom”. Therefore, when the player reaches “Gapy”, he does not have any sword to give and the quest cannot be conclude.

The place reached by the player is the crossing street of Horikawa and Oike.

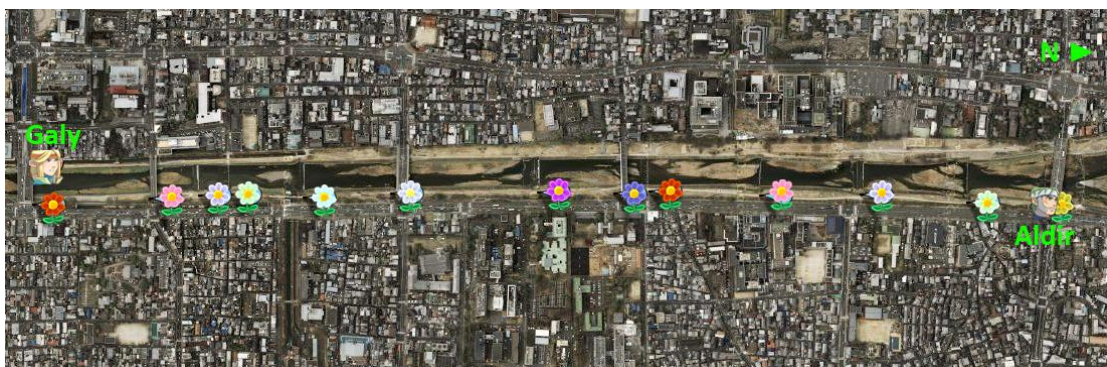
3.2.4. Third Quest – Collect items



The player continues on the road of Oike and finds a new NPC called “Galy” (this is the sister of “Gapy”). It stays near the river.



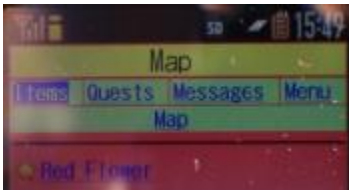
A new quest is proposed to the player. As usual, he can click on “Accept” link to get the quest. This asks the player to collect some flower near the river (East side) in the direction of North. The player has no other information and must to discover the flowers. For this case, the player knows that the flowers are near bridges, ford and waterfalls.



The player disposes the map where the different flowers are. Normally, the players do not have this kind of help. This is the case only to help the game tests. There are two fords, five bridges and six waterfalls.



Reach the first bridge (the other side where “Galy” is). There is the first flower here.



The first flower is here. This is a “Red Flower” like on the map. The player can click on the “Red Flower” link to see the details and actions available for this element.



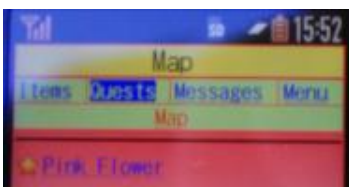
He can see the details of the flower and choose the action to do. He has to choose “Take” button to collect the item. It is what he does here.



After taking the flower, the player comes back to the map screen. He can see there is no more “Red Flower” on the map. This is very normal. The flowers in this quest are configured to be replaced only after 30 seconds. During this time, the flowers would not be available.



The next flower is placed near the second bridge. The player moves there to collect it.



He sees the “Pink Flower” on the map and can collect it with the same manner than the “Red Flower”.

Remark: For a better reading, the number of pictures will be reduced to the minimum for the next flower collecting.



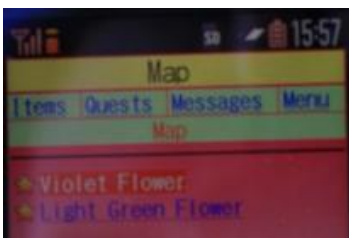
The next flower to collect takes place near the first waterfalls.



This time, the flower to collect is the “Violet Flower”. Actually, the player owns three different flowers but it is not enough to finish the quest.



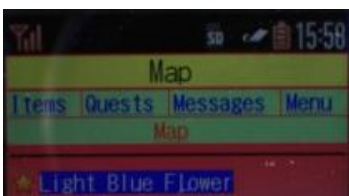
For the next flower, the player can find it near the first ford. This ford is very close to the first waterfalls.



The player can collect the “Light Green Flower” and “Violet Flower”. The reason of the presence of the “Violet Flower” is that the player is in radius of 50 meters from the “Violet Flower”. The “Violet Flower” is available again because 30 seconds have passed since the player collected it.



Next flower is near the second waterfalls. With this quest, the player can see many interesting and beautiful things near the river that not take place in the game.



This time, the player can collect the “Light Blue Flower”.



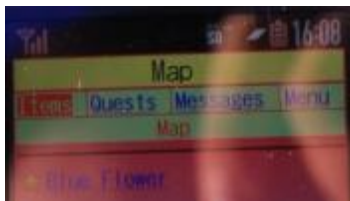
The next flower takes place near the third bridge. For this flower, the player encounters some difficulties to find it. The localization seems to be difficult to do near the bridge. With three or four tries, the location finally worked and the player collected the flower.



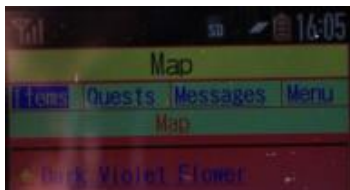
The flower collected by the player is the “White Flower”. This is the sixth flower on the map.



The third waterfalls allow finding the next flower. Notice the birds on the picture. Sometimes, the nature provides some beautiful things to see. This kind of things cannot be programmed or planned in the quest. These very great things can only happen in the real world.



The “blue Flower” can be collected here. The next flower can be found near the second ford but there is no picture this time.



The flower to collect is the “Dark Violet Flower”. It remains five flowers to collect to finish the quest.



The fourth bridge is reached and the flower can be found and collected.



This time, the flower to collect is the “Red Flower” again. Notice that the player could collect the first “Red Flower” more than one time with the same result. However, the quest is configured that the player cannot avoid going to the north because the last flower can only be collected near the last bridge.



The fourth waterfalls allows to the player finding the next flower.



The flower collected is the “Pink Flower”. The same rule can be applied for this flower than the previous one.



The player takes a little break and checks the conditions of the quest. The number and sort of flowers needed to finish the quest are indicated.

Some flowers are requested more than one time. The total of requested flowers is thirteen.



Now, the player checks his inventory to see the number of collected flowers. The player actually owns ten flowers. He seems to be near the end of the quest.



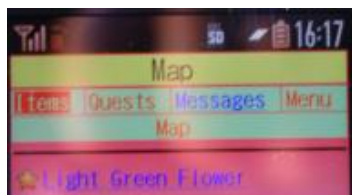
He continues to the North and reaches the fifth waterfalls.



The flower to take here is the “Violet Flower”



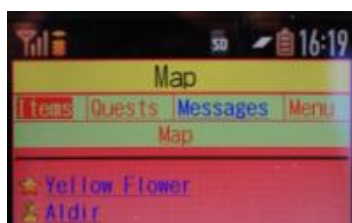
The last waterfalls are reached by the player. He can collect a flower here.



The flower to collect here is the “Light Green Flower”. There is only one more flower to collect and the quest is finished. Really? No, there is also the return path to do to validate the quest.



The last bridge is there. The player can see the last flower to collect on his map.



He can see the “Yellow Flower” and the NPC called “Aldir”. He collects the “Yellow Flower”.



He discusses with “Aldir” but the dialog was not the one expected. Normally, when the player reaches this point with this quest, “Aldir” must say that there are no more flowers to the North of its position. However, in this case, the quest is considered as “PreFinished” because all flowers are collected and the conditions to run the expected dialog indicate that the quest must to be in the state of “InProgress”. This is the normal behavior for the running mechanisms.



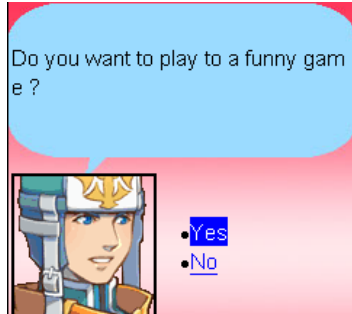
The player comes back to “Galy” and gives to it the flowers with the mechanism for the quest management. The quest began at 15h46 and finished at 16h34. It takes about 45 minutes to do the quest and taking the pictures. This is a long and pleasant quest near the river.



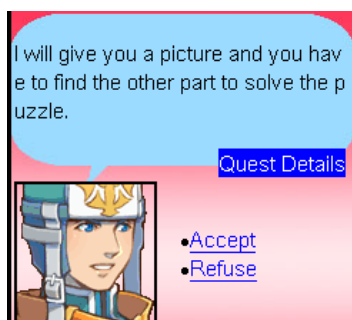
After the dialog, the player notices that there were no more items in his inventory. All the flowers were given to “Galy”.

3.2.5. Another quest – Puzzle quest

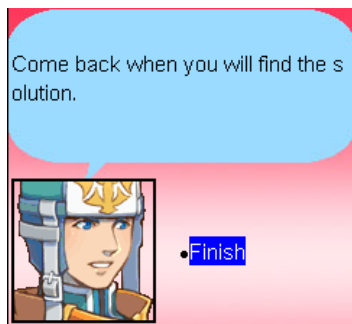
Due to the situation, it was not possible to proceed to a real test for this kind of quest. The next pictures are taken from the emulator.



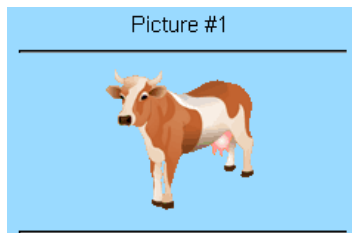
This non-player characters offers to play to a funny game. What kind of funny game is this?



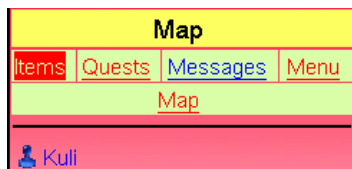
Ok, this is the puzzle quest. The player receives one part of a picture puzzle and he has to solve the puzzle by finding other players with the same quest and different part.



The non-player character tells to the player to come back with the solution. The player has to find other players and solve the puzzle.



The player received this picture. He has to find the other picture.



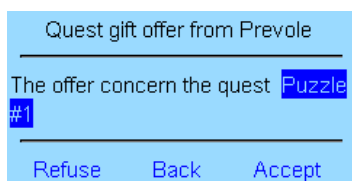
The player "Prevole" sees another player on the map. He sees the player "Kuli". He goes and speaks with him and discovers that "Kuli" has not the same quest as him.



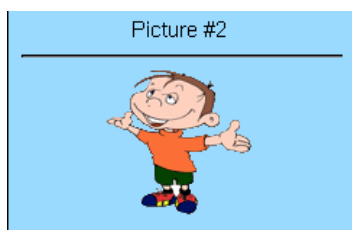
"Prevole" offers to "Kuli" the same quest. "Kuli" has to accept or refuse the offer.



"Kuli" sees he has a pending message in his inbox.



"Kuli" has to accept the offer from "Prevole" to have the same quest. After that, the both players are able to solve the quest together.



The player "Kuli" received this picture for the quest. They know there are no more pictures. They can solve the puzzle together.

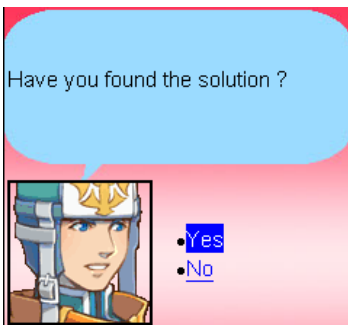


First try. They put together their mobile phones. They do not find any solution. One picture is a boy and the second is a cow.

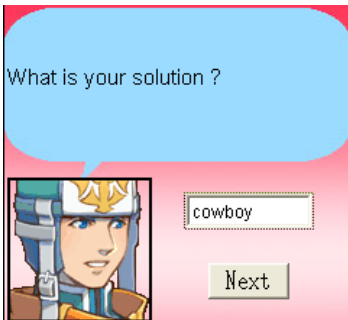


This time, the two pictures compose the word "Cowboy". "Prevole" and "Kuli" think that they have found the solution.

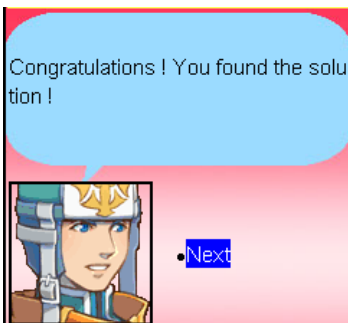
To solve this kind of quest, the players have to put their mobile phones together to find the solution.



Each one goes to talk to "Milo" the non-player character that offered the quest to give the solution.



"Milo" asks for the solution. An input box is used to get the solution from the player. "Prevole" fills the input box and submits his possible solution.



Great, "Prevole" and "Kuli" have found the solution. "Kuli" has to do the same dialog to finish his quest. The dialog after this one is just an invitation to come back later for a new game of this kind.

3.3. GPS Tracker

During the previous tests, a GPS Tracker was used to record the path that the player did during the game.

The Figure 1 shows the path that the player did in the first part of the test. It corresponds to the first three quests (Meet “Sephi”, Carry the sword from “Dan” to “Elom” and Carry the sword from “Elom” to “Gapy”). The path is quite good.



Figure 1 : GPS Tracker – Part 1

The problems described in previous paragraphs are shown by the GPS tracker. In the Figure 1, the green rectangle shown the GPS problems encountered.

In addition, the Sanjo Street is a street with a root on it. With this situation, the localization can encountered a lot of trouble. The mobile phone has a very good localization in this street. This is probably because the phone provider offers directly the location to the phone in addition of the GPS technology.



Figure 2 : GPS Tracker – Part 2

The Figure 2 shows the path for the quest “River of Flowers”. The localization was very good during the quest. There is only one problem to notice. The green circle indicates where the problem was encountered. The GPS Tracker seems to do not have the problem that the mobile phone had. This is probably due to the depression near the bridge. The location was successfully got after the bridge.

The return path is not so much interesting because there is nothing special that happens for the game mechanisms. However, it is interesting to notice they were some trouble with the location shown in the Figure 3. The green rectangles signal the problems. For the Figure 4, there is nothing special.

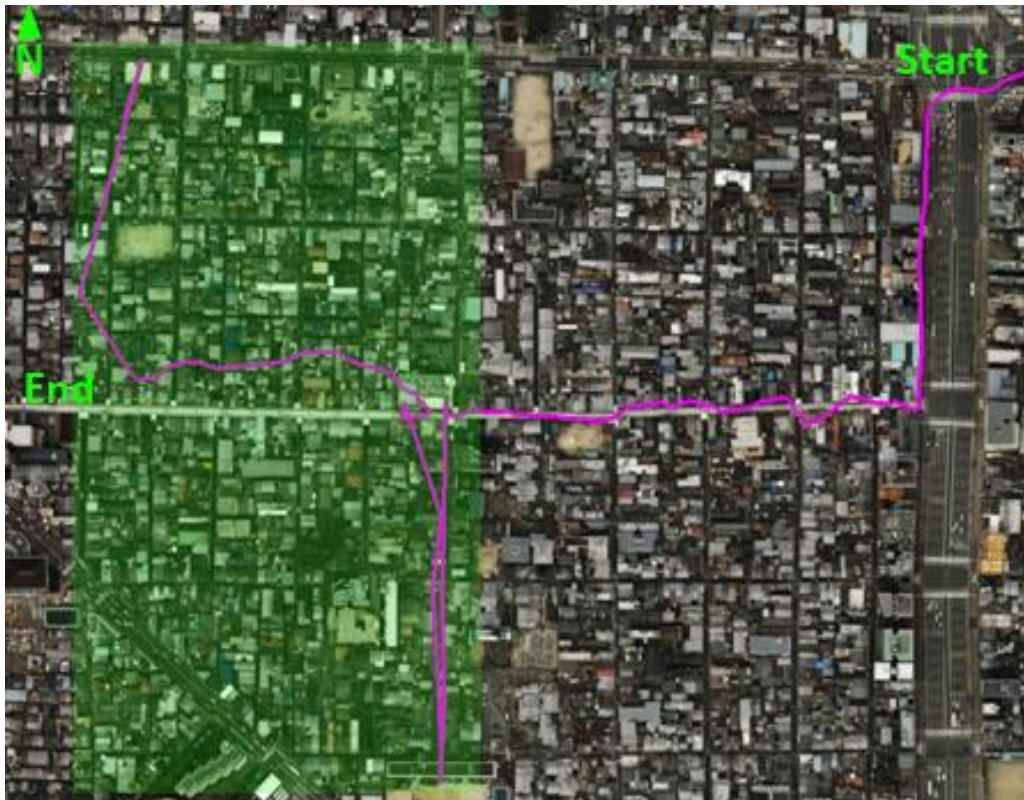


Figure 3 : GPS Tracker – Part 4



Figure 4 : GPS Tracker – Part 3

3.4. Conclusion

Finally, the tests realized demonstrate that the games work pretty well with the location. It seems to be some problems of precision some times. It is very difficult to tune the radius that the player can see the entities on the map. Actually, this value is set to 50 meters and seems to be enough.

The GPS Tracker used to validate the path that the player did during the tests confirm that the localization is enough in the majority cases.

The different quests used here demonstrate the possibility for the game configuration. There are a lot of possible combination to configure the dialogs and quests to create specific quests.

heig-vd

Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

K-MEP

Architecture

Laurent Prévost

R RITSUMEIKAN

Table of Content

4.1. INTRODUCTION	51
4.2. GLOBAL ARCHITECTURE	51
4.3. SYSTEM ARCHITECTURE	52
4.3.1. Mobile	52
4.3.2. inTrack Server	54
4.3.3. Game Server (MEP)	54
4.4. CONCRETE SERVER ARCHITECTURE	56
4.5. MEP GAME MECHANISMS	57
4.5.1. Dialogs	57
4.5.1.1. Normal Dialog	58
4.5.1.2. Quest Proposal	58
4.5.1.3. Question	59
4.5.1.4. Puzzle Answer	59
4.5.1.5. Item Proposal	59
4.5.1.6. Quest Ending	60
4.5.2. Dialogs specificities	60
4.5.2.1. Dialog read	60
4.5.2.2. Multiple dialog	61
4.5.3. Items	62
4.5.3.1. Item type	62
4.5.3.2. Item source	62
4.5.3.3. Multi item type	63
4.5.3.4. Summary	63
4.5.4. Quests	63
4.5.4.1. Quest State	64
4.5.4.2. Materials	65
4.5.4.3. Conditions	65
4.5.4.4. Rewards	66
4.5.4.5. Dialogs and quests	66
4.5.4.6. Summary	66
4.5.5. Messages	66
4.5.5.1. Item gift	66
4.5.5.2. Item gift confirmation	67
4.5.5.3. Item exchange	67
4.5.5.4. Item exchange confirmation	68
4.5.5.5. Quest gift	68
4.5.5.6. Quest gift confirmation	68
4.5.6. Message flows	69
4.5.6.1. Item gift flow	69

4.5.6.2. Item exchange flow	69
4.5.6.3. Quest gift flow	70
4.6. MAJOR QUEST TYPES	70
4.6.1. Find/Reach a NPC	71
4.6.2. Collect items	71
4.6.3. Solve a picture puzzle	72
4.6.4. Summary	72
4.7. CONCLUSION	73

Table of Figures

FIGURE 5 : GLOBAL ARCHITECTURE	51
FIGURE 6 : SYSTEM ARCHITECTURE	52
FIGURE 7 : CELL PHONE ARCHITECTURE	53
FIGURE 8 : MEP GLOBAL ARCHITECTURE	55
FIGURE 9 : CONCRETE MEP ARCHITECTURE	57
FIGURE 10 : DIALOGS MECHANISM.....	58
FIGURE 11 : NORMAL DIALOG MECHANISM	58
FIGURE 12 : QUEST PROPOSAL DIALOG MECHANISM	59
FIGURE 13 : QUESTION DIALOG MECHANISM	59
FIGURE 14 : PUZZLE ANSWER DIALOG MECHANISM	59
FIGURE 15 : ITEM PROPOSAL DIALOG MECHANISM.....	60
FIGURE 16 : QUEST ENDING MECHANISMS	60
FIGURE 17 : DIALOG WITHOUT STATE MEMORIZATION	60
FIGURE 18 : DIALOG WITH STATE MEMORIZATION	61
FIGURE 19 : MULTIPLE DIALOGS MECHANISM	61
FIGURE 20 : ITEM TYPE DESIGN	62
FIGURE 21 : ITEM SOURCE DESIGN	62
FIGURE 22 : MULTI ITEM TYPE DESIGN	63
FIGURE 23 : QUEST PARTS.....	64
FIGURE 24 : QUEST STATES	64
FIGURE 25 : MESSAGE ITEM GIFT STRUCTURE	67
FIGURE 26 : MESSAGE ITEM GIFT CONFIRMATION STRUCTURE	67
FIGURE 27 : MESSAGE ITEM EXCHANGE STRUCTURE	67
FIGURE 28 : MESSAGE ITEM EXCHANGE CONFIRMATION STRUCTURE	68
FIGURE 29 : MESSAGE QUEST GIFT STRUCTURE.....	68
FIGURE 30 : MESSAGE QUEST GIFT CONFIRMATION STRUCTURE	69
FIGURE 31 : ITEM GIFT MESSAGE FLOW	69
FIGURE 32 : ITEM EXCHANGE MESSAGE FLOW.....	70
FIGURE 33 : QUEST GIFT MESSAGE FLOW.....	70
FIGURE 34 : FIND/REACH A NPC QUEST DESIGN	71
FIGURE 35 : COLLECT ITEMS QUEST DESIGN	71
FIGURE 36 : SOLVE A PICTURE PUZZLE QUEST DESIGN	72

4.1. Introduction

This part of document will introduce the architecture of the Mobile Exergaming Platform. This platform as we already seen, allow building pervasive games. The platform has to use another platform to delegate the localization tracking tasks.

The first part of this chapter will concentrate the attention on the technical architecture used for the platform. The second part will focused on the quest part for the games.

4.2. Global Architecture

The Figure 5 shows a first architecture that describes different components of the K-MEP application. This is not the real implementation of the different components. For example, the inTrack platform and MEP are separated with their own data system.

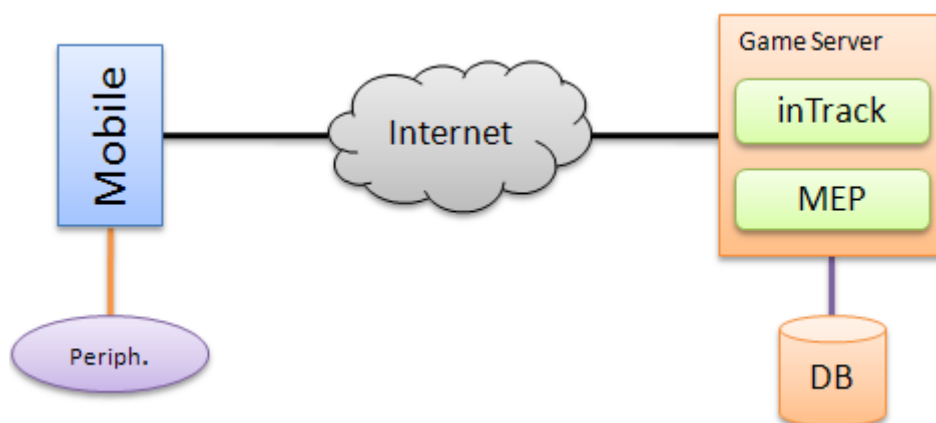


Figure 5 : Global architecture

Database (DB)

The database is used to store the persistence data for the game and the tracking system. This is not necessary the same database.

Internet

Internet allows the communication between the different elements of the system. In this case, the part between the mobile phone and Internet is a private network owned by NTT Docomo. This document will return on this point later.

inTrack

The inTrack part is the necessary to do and use the localization in the MEP. In this project, this is an independent module with its own architecture. This is describing a little later in this document part.

Mobile Exergaming Platform (MEP)

The Mobile Exergaming Platform, called MEP in this document, is the base to implement pervasive games with localization. This is the central mechanisms for the game.

Mobile

Mobile part represents the mobile devices. In this project, the mobile devices are cell phones and especially NTT Docomo cell phones with iMode technologies.

Peripherals (Periph.)

Normally, the project would use some peripherals of the mobile device in addition of the GPS. Finally, only the GPS is used for the game. There are some limitations discovered during the project that drives the project in the use of GPS only.

4.3. System architecture

The Figure 6 shows the detailed architecture of MEP. There are three important parts to build the system. The first is the Game Server part, the second is the inTrack Server and the third is the mobile devices.

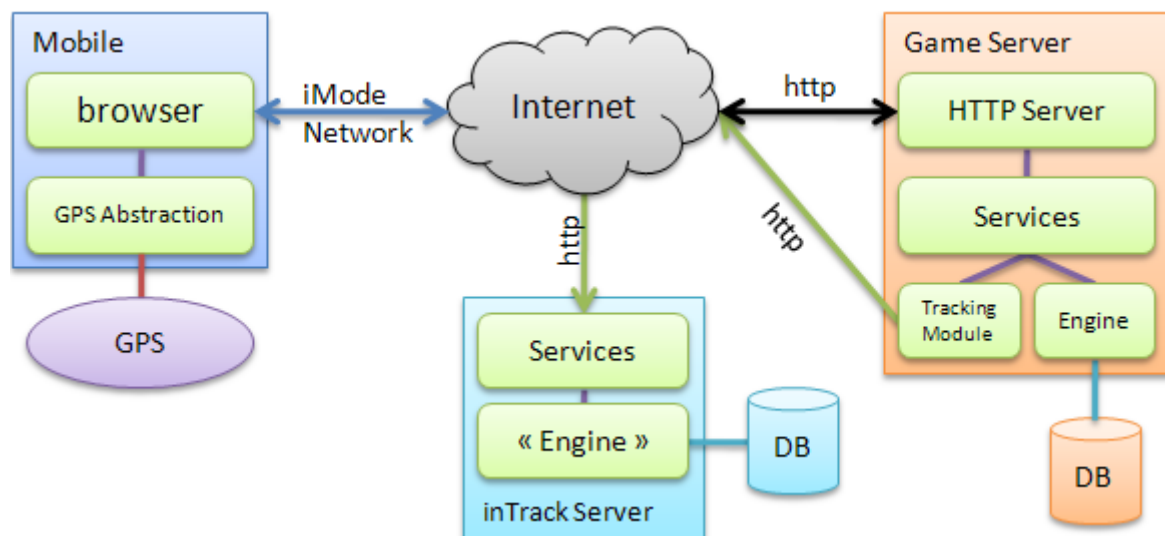


Figure 6 : System architecture

4.3.1. Mobile

Mobile devices must have a GPS peripheral in use. The game is only available for people with a GPS. If this is not the case, they are not able to access to the data correctly.

For this project, the mobile devices are NTT Docomo phones. They are two possibilities to use the GPS from these phones. The first is integration directly in tags of i(X)HTML codes for web pages and the second, directly from an iAppli. This point will be explained more in details later in this document.

The KMEP use web pages for the iMode browser rather than iAppli due to a certain limitation. In this situation, the browser will automatically add the location parameters in the HTTP requests. There is a limitation for the communication of GPS information. The user has to choose a location and confirm that he agrees to send the data to the provider, in this case, the game server.

The communication between the mobile phone and the Internet pass through the iMode infrastructure. NTT Docomo provides an infrastructure called iMode also for the Internet communication. So, the communication is secured from the phone to the Internet. The last step between NTT Docomo infrastructure and the game server is normal HTTP communication.

The Figure 7 shows cell phone architecture for the part corresponding to the project. It shows the technologies used to present the game interface to the players.

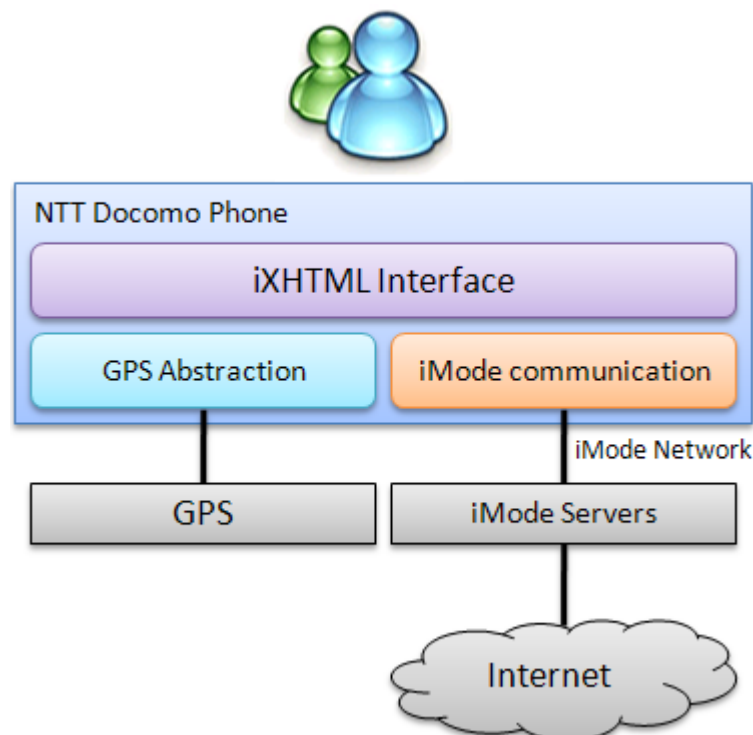


Figure 7 : Cell phone architecture

iXHTML Interface

The iXHTML interface is the subset of XHTML tags from to build web pages for the iMode browser present on the NTT Docomo phones. This subsets contains a very small range of tags but sufficient for simple presentation.

The principle limitation is that there is no JavaScript on this minimal browser. Another limitation is there is no CSS too. There is only inline style. With these limitations, this is difficult to create a great and user-friendly user interface.

GPS Abstraction

The GPS abstraction is the layer from NTT Docomo to access the same manner to the GPS. NTT Docomo offer a lot of different mobile phones and all have the same interface. This is not possible to access directly to the GPS. This layer is not clearly defined.

GPS

The game requires a GPS active on the mobile phone. Due to the type of user interface, the GPS has to be embedded in the device. A special attribute for <a> and <form> to allow the localization processing.

iMode Communication

This is the NTT Docomo technology to allow the communications for Internet. NTT Docomo provides their own protocol between mobiles and iMode Servers.

iMode Servers

The iMode Servers are gateways to reach Internet servers. NTT Docomo offers the possibility to add some services directly on their servers or some filtering.

4.3.2. inTrack Server

The inTrack platform offers the possibility to create rich application with localization services. It offers the mechanism to store the location data and retrieves them. The platform is decomposed in two parts.

Main Server

The main server is the location storing and managing system. It could be on the same system as the game or not. It could be shared with other users or not. Some scenarios are possible.

The main system is composed by the services offered to the public and an engine to process the necessary task to save or manage the data.

Tracking Module

The tracking module is the part that allows the communication from a client application (in this case, the game server) and inTrack platform. Normally, there is a separation with the location updates and the data usage. In this project, the both are include in the same part of the application and this is the MEP tracking module.

4.3.3. Game Server (MEP)

The Figure 8 shows the mechanisms present in the MEP. We can see that the tracking module is included directly in the platform and remains transparent for the users of the MEP.

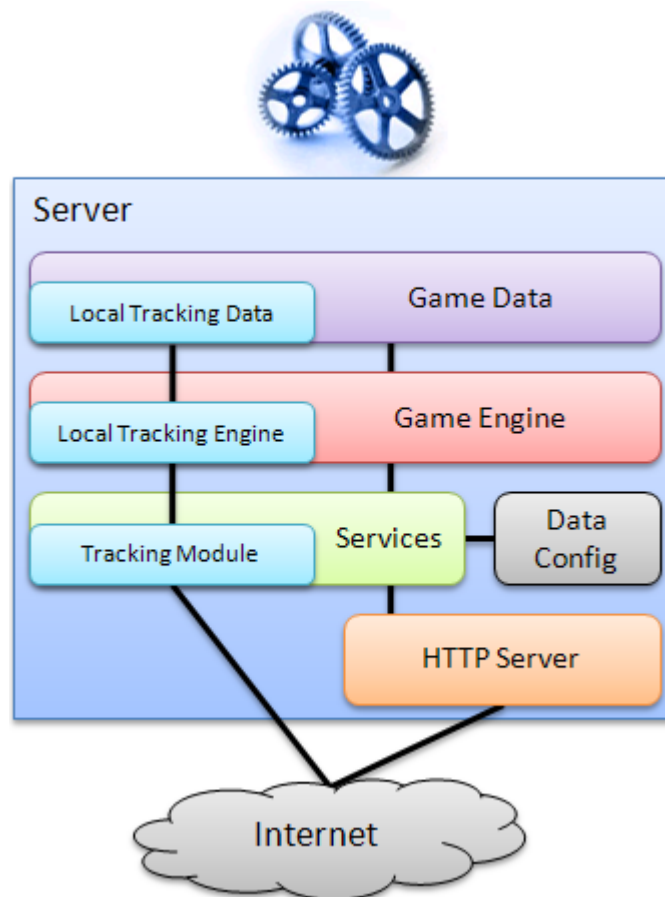


Figure 8 : MEP Global Architecture

The MEP is decomposed in some layers. The Figure 8 shows also the HTTP Server use in the implement of K-MEP. This part is not a part of MEP.

HTTP Server

The HTTP Server is only here to add a comprehensive step for the diagram. This is the client access for the players to reach the MEP in this project. The possibility to create rich user interface other than web user interface exists.

Services

Services layer is the access point to all functionalities of the MEP. This the only way to get the access to the platform. The services offer the mechanisms to manage and the game engine.

During the deployment phase, the services can use the Data Config system to load the initial data of the game. The data can differenciate some type of games or some place of games. For this project, the game data set contains only data placed in the city of Kyoto.

In the services, we can found also some game rules mechanisms that take place appropriately in this part of the architecture.

Data Config

Data Config system allows loading the data to the game server. This mechanism could be automated for the deploying phase with the HTTP Application (in this project). However, at least, there is only one auto data loading allowing per deployment. This is not possible to load the same data twice times.

This part is especially used to configure the MEP data to create a specific game. In this project, the game take place in Kyoto and the data are in relation with the city for the locations for example. This is a high configurable level explained later in this document.

Game Engine

Game Engine integrates the model of the MEP. They are all the mechanism to prepare the storage of data. There are also some game rules in this part. The game rules are in the best and efficiency place for the running platform.

Game Data

This layer contains the data. Concretely, this is the database used by the engine (persistence part). This part is automated by the EJB technologies (JPA).

Tracking Module

The tracking module is included in the Services layer and offers the mechanisms to communicate with the inTrack platform. It allows updating location of players or getting some data from the inTrack.

Local Tracking Engine

Same as the Game Engine, the Local Tracking Engine allows the management of the inTrack data. This is a local copy of the most useful data from inTrack to optimize the processing and management of inTrack entities binding with MEP entities.

Local Tracking Data

This is the data layer for the data of inTrack local engine. The database is used to store the data used locally as explanations just before.

4.4. Concrete Server Architecture

The Figure 9 shows the concrete architecture for the MEP with inTrack. This architecture shows that the two platforms are on the same server. This is not necessary the case for all implementations. During the development phase, the inTrack Server was in the production server and MEP was on local computer.

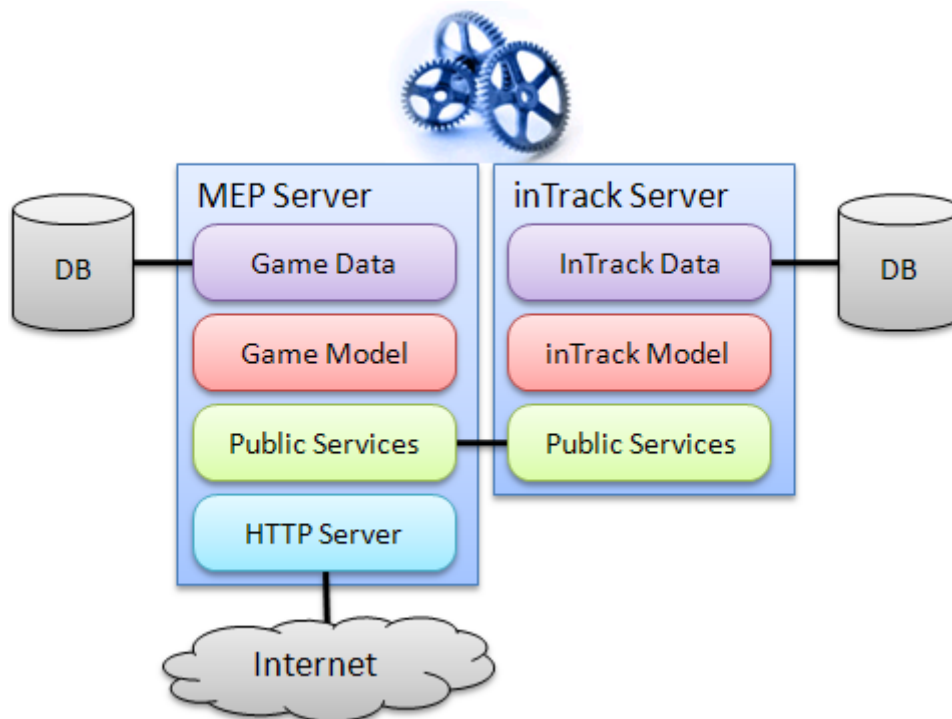


Figure 9 : Concrete MEP Architecture

The layers shown in the Figure 9 are the same as described in the Figure 8 for the MEP part without all details. For the inTrack architecture, the principle is the same MEP but with the specificities of inTrack. This document does not cover all the aspect of inTrack.

The most important thing to remember with this schema is that the inTrack data are synchronized locally with the remote data for a part of the application. This is in the goal of optimization.

4.5. MEP Game Mechanisms

In the MEP, there are three important mechanisms to discuss. These mechanisms are the base of the platform to develop some games.

- Dialogs
- Quests
- Messages

Each mechanism has its importance in the MEP. They offer the base to build a complete game based on the MEP.

4.5.1. Dialogs

The dialog mechanism offers the possibility to the player to communicate with the NPC in the game. The dialogs constructions are organized like a decision tree. The Figure 10 shows the principle of the mechanism. The mechanism is more like a graph than a tree because some dialogs can go back to another.

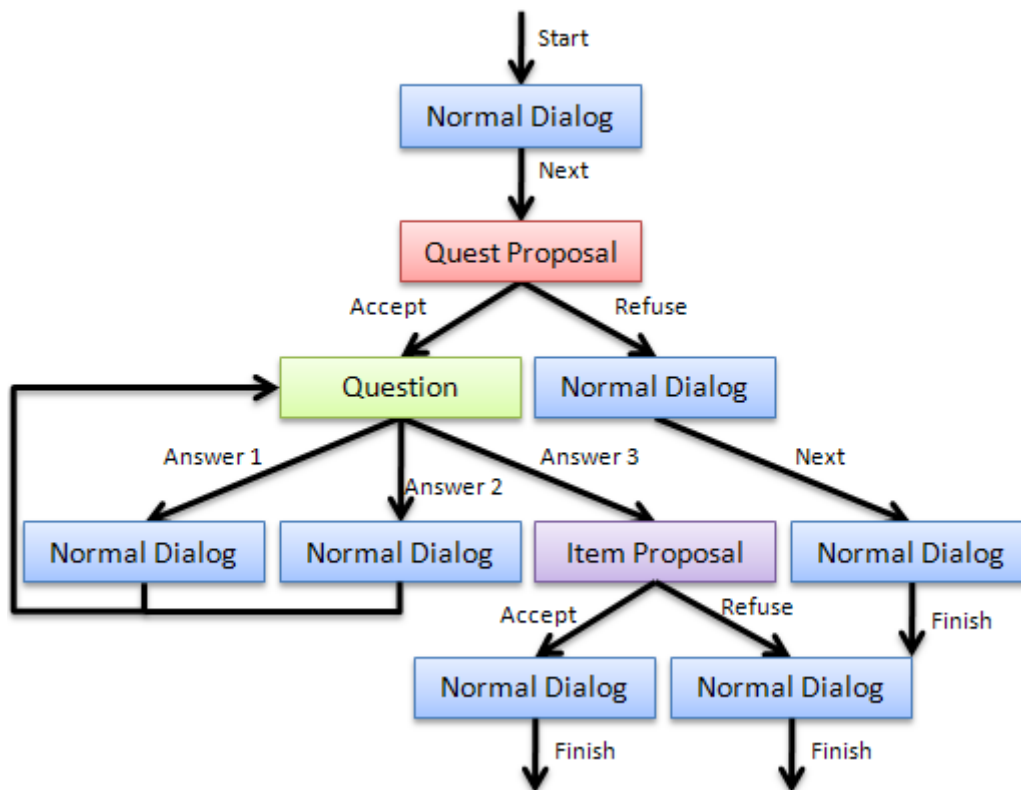


Figure 10 : Dialogs mechanism

4.5.1.1. Normal Dialog

The normal dialog is the simplest one. This is only a piece of text without any special interaction. This is particularly useful during some explanation or something like that. For this dialog type, there is only the next dialog to configure. No dialog means this is the end of dialog. The Figure 11 shows the possible use of the normal dialog.

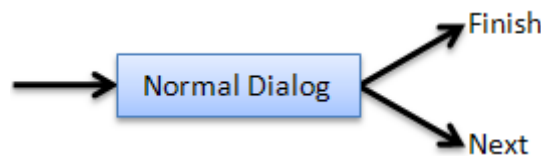


Figure 11 : Normal Dialog mechanism

4.5.1.2. Quest Proposal

The quest proposal offers the possibility to give to the player a quest. The quest will be describe in depth just after this part of document. A player can accept or refuse a quest but he can also see the details if available. There is a special case with the first quest. These quests are only available at the beginning of the game for new player. The mechanism knows that the quest proposal is used for a first quest proposal because there is no configuration about the quest. The Figure 12 shows the mechanism.

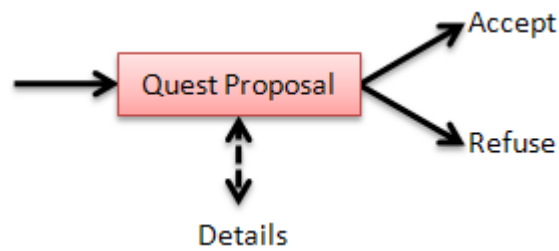


Figure 12 : Quest Proposal Dialog mechanism

4.5.1.3. Question

A question dialog allows a new interaction with the player. The player can choose the answer and with this choice, the dialog flow could be very different. This is especially interesting to add some non-linear dialogs. There is no limitation for the number of answer but it will be better to take care about the number of answers. Each answer corresponds to another dialog. The Figure 13 shows the question mechanism.

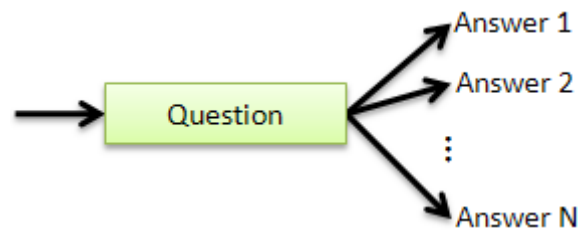


Figure 13 : Question Dialog mechanism

4.5.1.4. Puzzle Answer

The puzzle answer dialog allows a player to answer a puzzle question. This dialog is specific to the quest puzzle type describe later. A text input is necessary to collect the answer from the player. After validation, there are two possibilities for the next dialog part. The answer could be correct or not. Depending of the correctness of the answer, the players follow the right way in dialog flow. The Figure 14 shows the dialog mechanism.

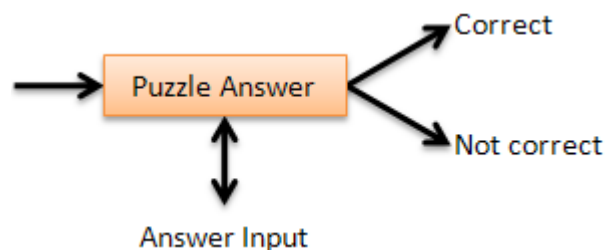


Figure 14 : Puzzle Answer Dialog mechanism

4.5.1.5. Item Proposal

This dialog allows offering items to the player or requesting some. The particularity is the possibility to ask the player to give some items but if there are not enough items to give, there are two possibilities in the flow of the dialog. The first is to configure a “not enough” dialog specifically and the second is to use by default the refuse dialog. The Figure 15 shows the dialog mechanisms.

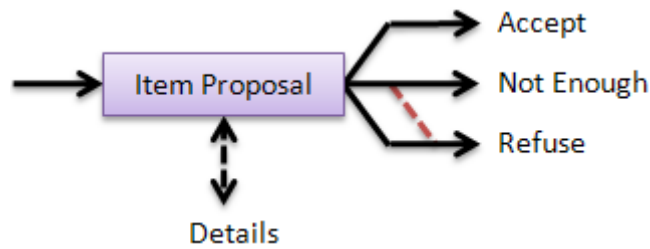


Figure 15 : Item Proposal Dialog mechanism

4.5.1.6. Quest Ending

The quest-ending dialog is a specific dialog to validate a quest in the state “to validate”; this is specifically from the quests. The quests will be describe a little further in this document. This dialog type is only here to allow the processing of quest ending like giving item for reward and other things like that. The Figure 16 shows the dialog mechanisms

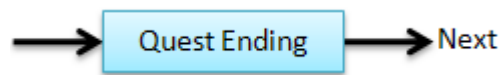


Figure 16 : Quest Ending mechanisms

4.5.2. Dialogs specificities

There are also two others mechanism to discuss about the dialogs. The first one is a mechanism to allow memorizing the state of dialog depending of the player and the second allows configuring multiple dialogs depending on the state of the player.

4.5.2.1. Dialog read

This mechanism is important to memorize if a player has already read a dialog or not. It is very useful to avoid dialog repetition. Imagine this situation shown on the Figure 17.

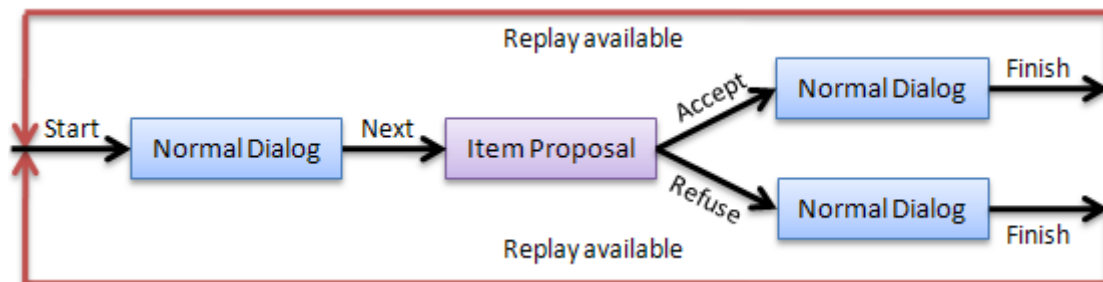


Figure 17 : Dialog without state memorization

In this situation, if there is no mechanism to remember the dialog, the next time the player runs the dialog, he can obtain other items. This is not the goal to reach with the global dialog mechanism. In the Figure 18, this is a remembering mechanism saves the state of the dialog.

The new situation offer the solution to do not run the dialog again but now there is not more dialog to show. For this, there is a need to another mechanism. The next paragraph will explore this second mechanism.

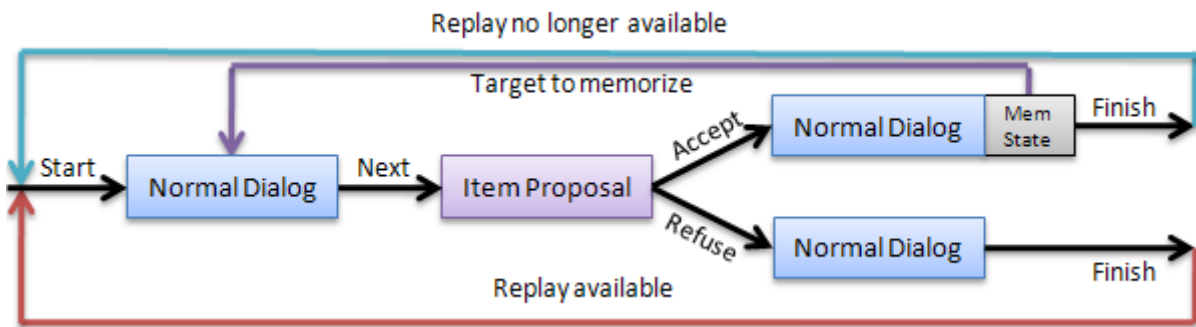


Figure 18 : Dialog with state memorization

This mechanism allows choosing when the state could be memorized. With this functionality, this is possible to memorize all the dialog flow in one time. There is actually a big limitation; it is not possible to choose other dialog than the starting one.

4.5.2.2. Multiple dialog

To solve the problem with no dialog when a dialog state is created for a player, the multiple dialogs allow creating a principal dialog to run again and other dialogs chosen depending of the state of the player.

Idea is to add some conditional dialogs that can be activated only if the player reaches the conditions. When there is more than one possible dialog, the first one is chosen. This is a limitation of the system. The configuration must to take care that all dialogs could be run or some of them will never appear to players. The Figure 19 shows the mechanism.

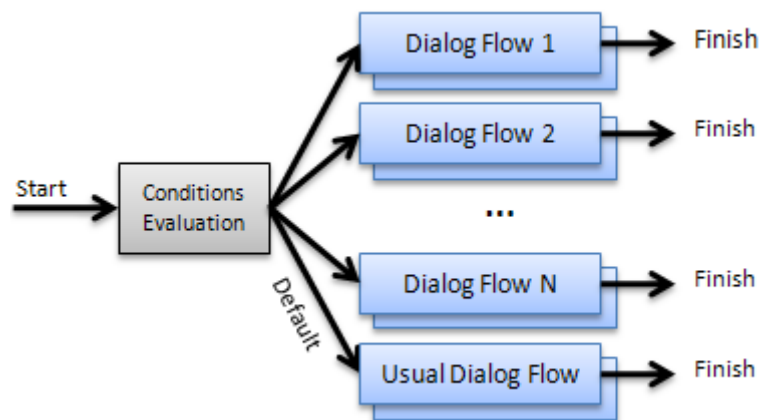


Figure 19 : Multiple Dialogs mechanism

This mechanism is only available for the choice of the first dialog step. The conditions are evaluated and the first dialog that reaches the conditions is chosen. The default dialog is returned for all the other cases.

With the previous mechanism, this is possible to configure a dialog with a state to be run only one times and the default dialog to be run repeatedly.

4.5.3. Items

Items are elements that the player can find, receive, collect and carry with them. They are necessary for the game to add the interaction between the quests, players and NPC. They allow adding a lot of functionality to the MEP.

4.5.3.1. Item type

Item type represents the category for an item. Each item is from a specific type. For example, the player can have three items of the type “Sword of Kyoto”. It means that the player owns three “instance” of the item type. Each item shares the characteristic of its item type like name and description.

These types are not directly represented in the game. There is only a representation of items and item sources. These representations allow the localization that the item type cannot.

The Figure 20 shows the representation of an item type with its name, description and image id.



Figure 20 : Item Type design

4.5.3.2. Item source

The item sources allow placing items on the map with the possibility with creating a new item after a certain time after a players tool the item. That is very useful to auto-create items on the map.

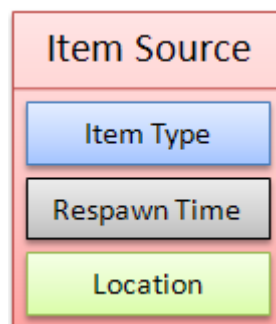


Figure 21 : Item Source design

These sources are configuring by the item type that they will create, the time to wait before a new item is created and the location they take. The Figure 21 shows these elements.

4.5.3.3. Multi item type

Multi item type is a special concept to allow the configuration of item for quest rewards or conditions or dialogs. This is very useful to define a pattern to give some item. With this concept, this is possible to give more than one item on the same time.

The interesting part of this mechanism is that allows creating a circle givable item. It means that the entire item configured in the multi item type are given one after the other in a round order. When the list is finished, the list is beginning again. This mechanism is very great to distribute some items uniformly to the players.

The Figure 22 shows the design of the multi item type. There is also the possibility to configure the quantity and if the items will be added or removed from the player.

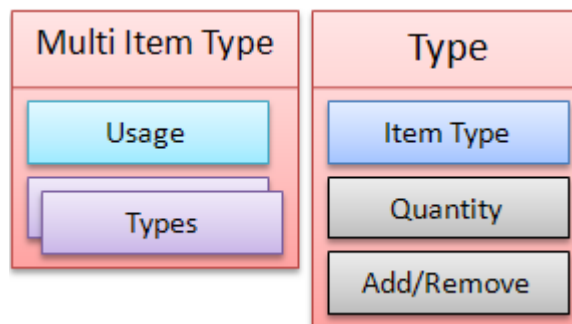


Figure 22 : Multi item type design

With the multi item type, there is a lot of possibility offered to configure the quests and dialogs.

4.5.3.4. Summary

The items are not directly usable in the MEP, they are some supports for the others mechanisms. They offer also a possibility to build some extra functionality. When an item is created in the MEP, it will be permanent since it is destroyed by a certain action like given to a NPC.

Items are localizable only when they do not have any owner. It means that the players can only take items when they are free.

4.5.4. Quests

The quests are the principal things to do in MEP platform. They define the activities that players could do. They are composed from three optional parts like the Figure 23 shows.

The quests are also composed by their name and description. This part is only textual and informational for the players. They allow building enjoyable stories with scenario guideline.

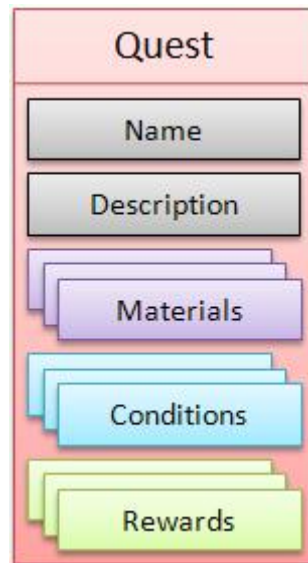


Figure 23 : Quest parts

4.5.4.1. Quest State

Before explanations of these three parts, it is important to understand what the process is when a player accepts a quest. The process creates a quest state that represents the quest and this is not directly the quest. The logic is that if the quest is updated the other player can see the updates, with specific and personal states attached for each player that runs the quest; there is no more this problem.

The quest state contains also the conditions states to maintain the relation between the quest conditions and the state of the player. These mechanisms allow managing a quest for multiple players independently.

The Figure 24 shows the possible flow for the quest states. There are five states possible for a quest:

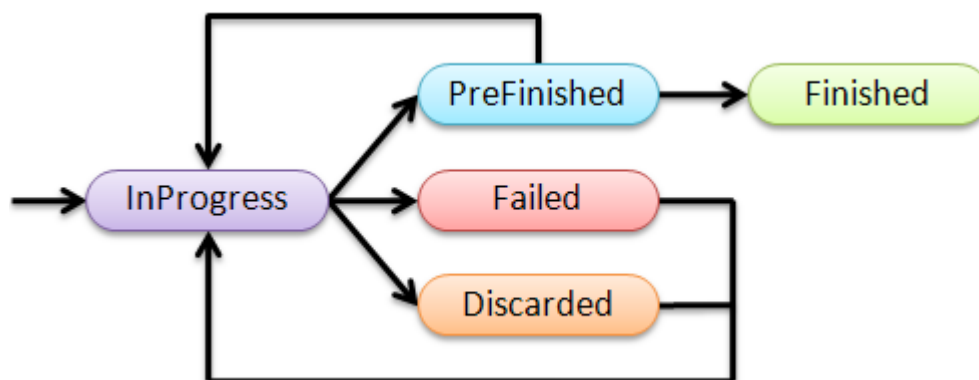


Figure 24 : Quest States

The return path shown on the figure allows a player to redo a quest in the Failed or Discarded state. In this case, the quest state is reinitialized. In special situation, the quest can be changed from PreFinished to InProgress. Some conditions can be changed during the game before the quest is total finished.

InProgress

This state is used to define a quest that a player is currently doing. This is the first step for all new quest states.

Discarded

The player can cancel a quest. When the player does that, the quest use the state of discarded.

Failed

Depending on the conditions, a quest could be failed if the conditions are not reached correctly.

PreFinished

To distinguish the quest finished from the other quest in “finished” state but without doing the processing of rewards, this step is necessary. It allows a finer configuration of dialogs and their conditions.

Finished

This is the final step when the rewards are applied and there is nothing else to do for the quest.

4.5.4.2. Materials

The materials are the prerequisites to give to the player when he accepts the quest. For example, it could be some items. The materials are always applied when the quest state is created.

Actually, there is only one type of materials. This is multiple items. It allows creating a special type of item as the previous description of multi item type.

4.5.4.3. Conditions

The conditions define the necessary to evaluate if a quest is finished or not. They are evaluated with regularity during the player game session. The processing to apply the conditions is done when the quest goes from PreFinished to Finished state.

For example, if a quest conditions ask to have a certain number of items, these items are removed from the game when the quest is finished.

The MEP offers only two types of conditions. The first one is the item condition to manage a certain quantity of item to own. The second one is the puzzle answer condition. This second one allows remembering the answer of a puzzle question. It will be described a little later in this document.

4.5.4.4. Rewards

The rewards are what the player receives when he finished a quest. He can receive new items for example. These rewards are processing just after the conditions at the end of a quest. That could be used to give some special item for a future quest for example.

Actually, they are only one type of rewards. This item reward allows giving item to the players.

4.5.4.5. Dialogs and quests

The problem in the MEP is that there is no representation of quests directly. To obtain a new quest to do, the player has to go and speak with NPCs. The NPCs are one way to obtain quests to do. They are also there to finish or helped to do some quests.

The interaction between NPC and Quest is very important. This is one of the central aspects of MEP. It allows some story building with a larger liberty of action. It is possible to force a player to move from a point A to a point B to get some information and after that to move to the point C to bring something there.

4.5.4.6. Summary

In summary, the three types of element define the quests. The quest is only a container for these elements. With the combination of these elements, it is possible to create a lot of different quest but these quests remains on the same principle offered by the MEP.

4.5.5. Messages

The messaging system is used to help the player in the actions between them. For example, when a player wants to give an item to another player, a message is sent to this player to inform him that he received an offer.

This system does not allow sending textual messages between players like a chat. It is only used to add some augmented functionalities for the interaction of players.

Actually, there are six types of messages. They are described in the next paragraphs. After that, the messages flows will be described too.

4.5.5.1. Item gift

The Figure 25 shows the structure for a message "Item Gift". This message allows sending a gift from a player to another to give one or more items of a unique type. The message stores the collection of items offered by the player.



Figure 25 : Message Item Gift structure

4.5.5.2. Item gift confirmation

This message is used after a player has accepted or refused the gift offer from another player. The message memorized the item type offered, the quantity, the player that accepts or refuses the offer and the answer of the player. The Figure 26 shows the structure.



Figure 26 : Message Item Gift Confirmation structure

4.5.5.3. Item exchange

The Figure 27 shows the structure of an item exchange message. This message is composed by the both entity concerned in the trade. Each entity can become the sender of an offer/counter-offer. The lists of items that composed the trade are stored too. The list of ItemsB can store items only if the player that receives the offer makes a counter-offer.

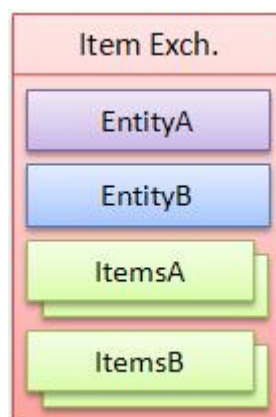


Figure 27 : Message Item Exchange structure

4.5.5.4. Item exchange confirmation

This message allows storing the data for the different confirmation step during an exchange item process. Item types (A and B) and quantities (A and B) correspond to the item types and quantities from the players of the trade. The entity stored is the last sender of the message depending on the current step of the trade. Finally, the answer of this last sender is also stored in the message. The Figure 28 shows the structure.



Figure 28 : Message Item Exchange Confirmation structure

4.5.5.5. Quest gift

The Figure 29 shows the structure of a quest gift message. This is similar to an item gift message. The difference is only about the quest rather than item list. In this message, the quest is stored to know which one is offered to the other player.

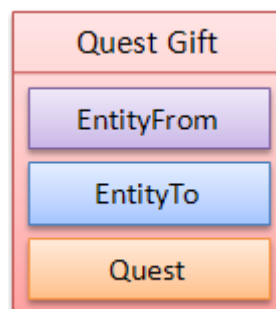


Figure 29 : Message Quest Gift structure

4.5.5.6. Quest gift confirmation

This message is composed by the entity that answers the quest proposal from the other player. The quest proposed is stored too. The answer is a little different from the other messages with this kind of field. The answer has three states: “accept”, “refuse” or “already”. The two firsts state are easy to understand and the third is simple. It represents the state when the player that has to answer the proposition has already the quest in his quest diary.



Figure 30 : Message Quest Gift Confirmation structure

4.5.6. Message flows

The messages flows show the message processing for the different messages types. Generally, the flows group two message types in one process. There are usually the principal message and its confirmation.

4.5.6.1. Item gift flow

The Figure 31 shows the flow used during an item gift process. The “Red” player sends an offer to the “Green” player. The “Green” player can accept or refuse the offer. The message sent back to the “Red” player is an item gift confirmation message.

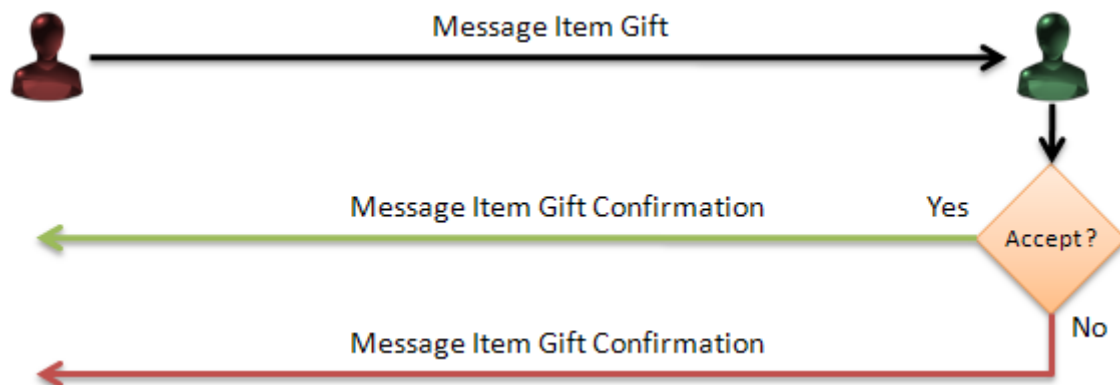


Figure 31 : Item Gift Message flow

4.5.6.2. Item exchange flow

The Figure 32 shows the flow during an item exchange process. The “Red” player sends an offer to the “Green” player. The “Green” player can accept or refuse the offer.

If the “Green” player refuses the offer from “Red” player, an item exchange confirmation message is sent to the “Red” player. In the other case, a counter offer is sent to the “Red” player.

The “Red” player can accept or refuse the counter offer. In the both cases, an item exchange confirmation message is sent to the “Green” player.

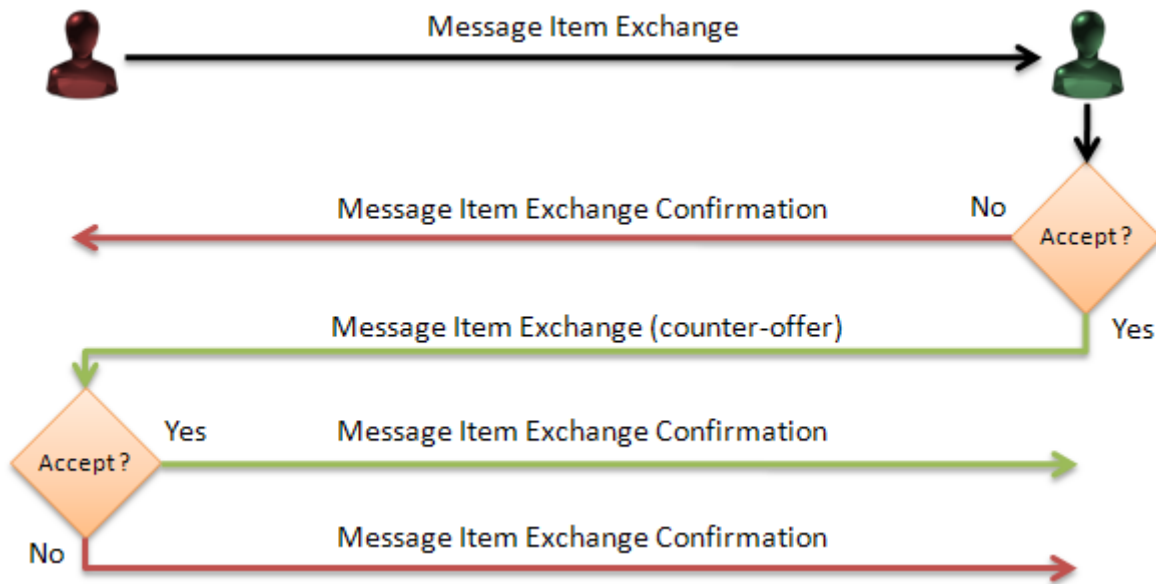


Figure 32 : Item Exchange Message flow

4.5.6.3. Quest gift flow

The Figure 33 shows the flow during a quest gift process. The “Red” player sends an offer to the “Green” player. A check is done to verify if the quest is already in the “Green” player’s quest diary. If this is the case, a quest gift confirmation is sent with the answer “already”. In other cases, the “Green” player can accept or refuse the offer. Independent of his answer, a quest gift confirmation message with the correct answer is sent to the “Red” player.

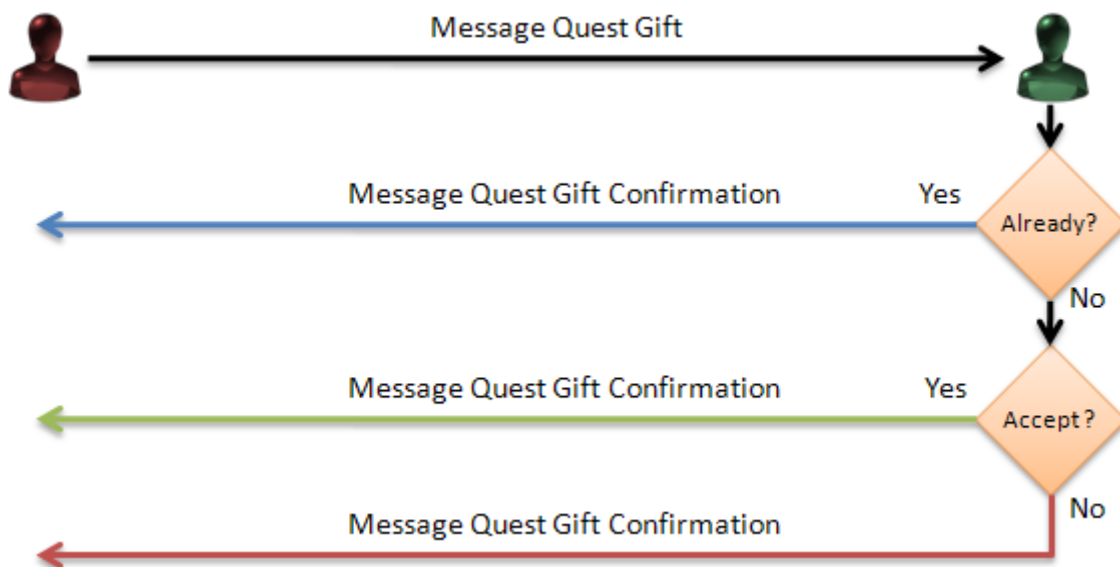


Figure 33 : Quest Gift Message flow

4.6. Major quest types

With the quest mechanisms describe before, it is possible to create, at least, three types of quests. Idea is to create three different difficulties of quests. These types are:

- Find/Reach a NPC
- Collect items
- Solve a picture puzzle

For simplifications reason, the next paragraphs omit the rewards part for the quest types. This part of quest is optional for the understanding of these three major quest types.

4.6.1. Find/Reach a NPC

Idea of this quest is just to offer the simplest quest to the player. He has only the task to move from a certain place to another one. To do that, he gets a quest near a NPC and goes to find another NPC.

The configuration of the quest is very simple because there is no need of special configuration. Only the name and description are required. The dialog part is also relatively simple. There is just the necessity to configure a conditional dialog relative to the quest.



Figure 34 : Find/Reach a NPC quest design

4.6.2. Collect items

This quest type is a little more complicated. It needs to be configured with some quest conditions. The conditions needed are the item condition that indicates which and how many items the player needs to have to complete the quest.

In addition, there is the configuration of the dialog to do. This is always the same for all quest and interaction with NPC. The starting and ending points for a quest could be the same NPC or not. The both approaches are available.

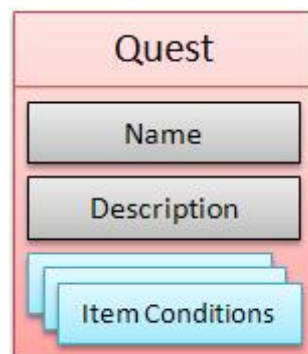


Figure 35 : Collect items quest design

4.6.3. Solve a picture puzzle

This is the most complicated quest type of MEP. It needs a little more configuration than the two previous types. In this one, it is necessary to configure the condition with a puzzle condition that indicates the textual answer waited. The other configuration to do is the material.

The material is used to give to the player a picture of a global puzzle enigma. After that, the player has to find the other parts near the other player. With the help of the other players that have also a part of the puzzle, they can solve the enigma and finish the quest.

A quest of this type can be composed from one to unlimited number of picture piece. Nevertheless, it is recommended to configure reasonable quests.

The puzzle condition contains the answer of the puzzle enigma. This is a textual solution to invite players to find a solution on the base of pictures. To give the answer, the players must go to speak to a NPC. The relative dialog must be configured properly with a puzzle answer part.



Figure 36 : Solve a picture puzzle quest design

This quest type is very interesting for the new dimension it offers. Normally, the game is not so much cooperative but with this kind of quest, it begins to be very cooperative ones. Players have no choice to finish this type of quest that to find other players and interacts socially with them.

Finally, this quest type covers a lot of aspect for the game. It covers the social aspects and the cooperation mode. The others aspects like moves and other ones are also covered like the other types.

4.6.4. Summary

These three quest types offer the base to create the quest for the MEP platform but MEP is not limited to these three quest types. With the combination of the materials, conditions and rewards, it is possible to create mixed quests from the three explained types.

The limitation is actually on the different elements to compose the quests and the imagination to create mini stories. There is also a fact to take care. All the quests are based on the same mechanisms. Therefore, many quests can become a little repetitive.

4.7. Conclusion

This chapter allowed understanding the principal abstract designs for the MEP and its global technical architecture. There is also the link between inTrack and MEP. All the explanations cover only the technical architecture.

The MEP mechanisms are explained with the necessary details for the level of abstraction. The implementation chapter will cover additional details about the realization of MEP.

heig-vd

Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

K-MEP

Implementation

Laurent Prévost

The logo for RITSUMEIKAN, featuring a large white letter 'R' on a dark red background, followed by the word 'RITSUMEIKAN' in white capital letters.

R RITSUMEIKAN

Table of Contents

5.1. INTRODUCTION	85
5.2. KMEP-COMMON	85
5.3. KMEP-EJB GENERALITIES	87
5.3.1. inTrack error bindings	87
5.3.2. Formatter Factories Interfaces	87
5.3.3. Formatter Interfaces	89
5.3.4. Server configuration	89
5.3.5. Identity	90
5.4. KMEP-EJB CONFIGURATION	91
5.4.1. XML Configuration	91
5.4.2. XML Configuration Infrastructure	94
5.4.3. XML Configuration Location	96
5.4.4. XML Configuration MEP Entities	96
5.4.5. XML Configuration Behavior	97
5.4.6. XML Configuration Dialog	99
5.4.7. XML Configuration Dialog Answer	99
5.4.8. XML Configuration Dialog Condition	100
5.4.9. XML Configuration Dialog State	100
5.4.10. XML Configuration Item	100
5.4.11. XML Configuration Message	101
5.4.12. XML Configuration Quest	103
5.4.13. XML Configuration Quest Material	104
5.4.14. XML Configuration Quest Condition	104
5.4.15. XML Configuration Quest Reward	105
5.4.16. XML Configuration Quest State	105
5.4.17. XML Configuration Quest Condition State	105
5.4.18. XML Configuration Statistic	106
5.5. KMEP-EJB MODEL	108
5.5.1. Model base	108
5.5.2. Model Behaviors	109
5.5.3. Model Dialog	111
5.5.4. Model Dialog Answer	113
5.5.5. Model Dialog Condition	114
5.5.6. Model Item	114
5.5.7. Model Message	115
5.5.8. Model Quest	117
5.5.9. Model Quest Materials	117
5.5.10. Model Quest Conditions	118
5.5.11. Model Quest Rewards	118

5.5.12.Model Quest State	119
5.5.13.Model Statistic	120
5.5.14.Model Utils	123
5.6. KMPE-EJB SERVICES	123
5.6.1. Services	123
5.6.2. Administrative Services	125
5.7. KMPE WEB APPLICATION GLOBAL ARCHITECTURE	128
5.8. KMPE-WAR CONFIGURATION & INFRASTRUCTURE	130
5.8.1. Configuration	130
5.8.2. Configuration mappings	131
5.8.3. Factories	133
5.8.4. Utilities	133
5.8.5. Servlets	134
5.9. KMPE-WAR TRANSFER OBJECTS	134
5.9.1. TO General	135
5.9.2. TO Item	135
5.9.3. TO Map	136
5.9.4. TO Message	136
5.9.5. TO NPC Dialog	136
5.9.6. TO Player	136
5.9.7. TO Quest	137
5.9.8. TO Statistic	138
5.10. KMPE-WAR ACTIONS	138
5.10.1.Actions General	139
5.10.2.Actions Item	140
5.10.3.Actions Map	141
5.10.4.Actions Message	142
5.10.5.Actions NPC Dialog	142
5.10.6.Actions Player	143
5.10.7.Actions Quest	144
5.10.8.Actions Statistic	145
5.11. KMPE-WAR IMODE APPLICATION	146
5.11.1.iMode Configuration	146
5.11.2.iMode Localization	146
5.11.3.iMode Factories	146
5.11.4.iMode Servlets	147
5.11.5.iMode HTML tags	150
5.11.5.1. Tags Block	151
5.11.5.2. Tags Document	151
5.11.5.3. Tags Form	152
5.11.5.4. Tags Link	153

5.11.5.5. Tags List	153
5.11.5.6. Tags Media	154
5.11.5.7. Tags Style	154
5.11.5.8. Tags Table	154
5.11.5.9. Tags Text	155
5.11.6.iMode Views Formatter	156
5.11.6.1. iMode Views Formatter Dialog	156
5.11.6.2. iMode Views Formatter Entity	157
5.11.6.3. iMode Views Formatter Message	157
5.11.6.4. iMode Views Formatter Quest	158
5.11.6.5. iMode Views Formatter Statistic	158
5.11.7.iMode Views Helpers	159
5.11.8.iMode Views Utils	159
5.12. INTRACK IN USE	160
5.12.1.Installation	160
5.12.2.Usage	165
5.12.3.Summary	166
5.13. CONCLUSION	166

Table of Figures

FIGURE 37 : UML – COMMON CLASSES	85
FIGURE 38 : UML – INTRACK / KEMP ERROR CODE BINDINGS	87
FIGURE 39 : UML – FORMATTER FACTORIES INTERFACES	88
FIGURE 40 : UML – FORMATTER INTERFACES	89
FIGURE 41 : UML – SERVER CONFIGURATION CLASSES.....	90
FIGURE 42 : UML – IDENTITY INTERFACES	91
FIGURE 43 : UML – XML CONFIGURATION CLASSES	92
FIGURE 44 : UML – XML CONFIGURATION INFRASTRUCTURE CLASSES	95
FIGURE 45 : UML – XML LOCATION CLASS	96
FIGURE 46 : UML – XML MEP ENTITIES CLASSES	96
FIGURE 47 : UML – XML BEHAVIOUR CLASSES	98
FIGURE 48 : UML – DIALOG XML CLASSES	99
FIGURE 49 : UML – DIALOG ANSWER XML CLASSES.....	99
FIGURE 50 : UML – DIALOG CONDITION XML CLASSES	100
FIGURE 51 : UML – DIALOG STATE XML CLASSES	100
FIGURE 52 : UML – ITEM XML CLASSES	101
FIGURE 53 : UML – MESSAGES XML CLASSES	102
FIGURE 54 : UML – QUEST XML CLASSES.....	103
FIGURE 55 : UML – QUEST MATERIAL XML CLASSES.....	104
FIGURE 56 : UML – QUEST CONDITION XML CLASSES	104
FIGURE 57 : UML – QUEST REWARD XML CLASSES.....	105
FIGURE 58 : UML – QUEST STATE XML CLASSES.....	105
FIGURE 59 : UML – QUEST CONDITION STATE XML CLASSES	105
FIGURE 60 : SOLUTION TO HAVE A UNIQUE STATISTIC NAME BY PLAYER.....	106
FIGURE 61 : UML – STATISTIC XML CLASSES.....	107
FIGURE 62 : UML – MODEL BASE CLASSES	108
FIGURE 63 : UML – MODEL BEHAVIOR CLASSES	110
FIGURE 64 : UML – MODEL DIALOG CLASSES.....	112
FIGURE 65 : UML – MODEL DIALOG ANSWER CLASSES	114
FIGURE 66 : UML – MODE DIALOG CONDITION CLASSES	114
FIGURE 67 : UML – MODEL ITEM CLASSES.....	114
FIGURE 68 : UML – MODEL MESSAGE CLASSES	116
FIGURE 69 : UML – MODEL QUEST CLASSES	117
FIGURE 70 : UML – MODEL QUEST MATERIALS CLASSES	117

FIGURE 71 : UML – MODEL QUEST CONDITIONS CLASSES.....	118
FIGURE 72 : UML – MODEL QUEST REWARDS CLASSES	119
FIGURE 73 : UML – MODEL DIALOG STATE CLASSES	120
FIGURE 74 : UML – MODEL STATISTICS CLASSES	121
FIGURE 75 : UML – MODEL UTILS CLASSES	123
FIGURE 76 : UML – SERVICES INTERFACES	124
FIGURE 77 : UML – ADMINISTRATIVE SERVICES INTERFACES.....	126
FIGURE 78 : KMEP WEB APPLICATION ARCHITECTURE	129
FIGURE 79 : UML – CONFIGURATION CLASSES	130
FIGURE 80 : UML – CONFIGURATION MAPPINGS CLASSES.....	131
FIGURE 81 : UML – IURLFACTORY INTERFACE.....	133
FIGURE 82 : UML – UTILITIES CLASSES	133
FIGURE 83 : UML – SERVLETS CLASSES	134
FIGURE 84 : UML – TO CLASSES	134
FIGURE 85 : UML – GENERAL TO CLASSES.....	135
FIGURE 86 : UML – ITEM TO CLASSES.....	135
FIGURE 87 : UML – MAP TO CLASS.....	136
FIGURE 88 : UML – MESSAGE TO CLASSES	136
FIGURE 89 : UML – NPC DIALOG TO CLASS	136
FIGURE 90 : UML – PLAYER TO CLASSES	137
FIGURE 91 : UML – QUEST TO CLASSES	137
FIGURE 92 : UML – STATISTIC TO CLASS	138
FIGURE 93 : UML – ACTIONS CLASSES.....	138
FIGURE 94 : UML – ACTIONS GENERAL CLASSES	139
FIGURE 95 : UML – ACTIONS ITEM COMMON BASE CLASSES.....	140
FIGURE 96 : UML – OTHER ACTIONS ITEM CLASSES	141
FIGURE 97 : UML – ACTIONS MAP CLASSES	141
FIGURE 98 : UML – ACTIONS MESSAGE CLASSES.....	142
FIGURE 99 : UML – ACTIONS NPC DIALOG CLASSES	143
FIGURE 100 : UML – ACTIONS PLAYER CLASSES	143
FIGURE 101 : UML – ACTIONS QUEST COMMON BASE CLASSES	144
FIGURE 102 : UML – OTHER ACTIONS QUEST CLASSES.....	145
FIGURE 103 : UML – ACTIONS STATISTIC CLASS	145
FIGURE 104 : UML – IMODE CONFIGURATION CLASSES.....	146
FIGURE 105 : UML – IMODE LOCATION CLASS	146
FIGURE 106 : UML – IMODE FACTORY CLASSES	147
FIGURE 107 : UML – IMODE CONTROLLERSERVLET CLASS	148

FIGURE 108 : UML – I MODE TAGS ABSTRACT CLASSES	151
FIGURE 109 : UML – TAGS BLOCK CLASSES	151
FIGURE 110 : UML – TAGS DOCUMENT CLASSES.....	151
FIGURE 111 : UML – TAGS FORM CLASSES	152
FIGURE 112 : UML – TAGS LINK CLASSES.....	153
FIGURE 113 : UML – TAGS LIST CLASSES	153
FIGURE 114 : UML – TAGS MEDIA CLASSES	154
FIGURE 115 : UML – TAGS STYLE CLASS	154
FIGURE 116 : UML – TAGS TABLE CLASSES	155
FIGURE 117 : UML – TAGS TEXT CLASSES	155
FIGURE 118 : UML – VIEWS FORMATTER ABSTRACT FORMATER CLASS	156
FIGURE 119 : UML – VIEWS FORMATTER DIALOG CLASSES	156
FIGURE 120 : UML – VIEWS FORMATTER ITEM CLASSES	157
FIGURE 121 : UML – VIEWS FORMATTER MESSAGE CLASSES.....	157
FIGURE 122 : UML – VIEWS FORMATTER QUEST CLASSES.....	158
FIGURE 123 : UML – VIEWS FORMATTER STATISTIC CLASSES.....	158
FIGURE 124 : UML – VIEWS HELPERS CLASSES	159
FIGURE 125 : UML – VIEWS UTILS CLASSES	159

Table of Codes

CODE 1 : FORMATTER CREATION	89
CODE 2 : XML LOADER – LOAD METHOD.....	93
CODE 3 : ID COLLECTION ANNOTATION CONFIGURATION.....	98
CODE 4 : MESSAGES ANNOTATION CONFIGURATION	101
CODE 5 : QUEST ELEMENT ID CONFIGURATION ANNOTATION.....	103
CODE 6 : STATISTIC ANNOTATION DEMONSTRATION.....	106
CODE 7 : STATISTIC ANNOTATION CONFIGURATION SAMPLE.....	106
CODE 8 : DATA MANAGER IMPLEMENTATION.....	128
CODE 9 : ABSTRACTITEMACTION PROCESS METHOD	141
CODE 10 : ABSTRACTQUESTACTION PROCESS METHOD.....	144
CODE 11 : PROCESSREQUEST CODE PART	147
CODE 12 : GETACTION CODE PART	148
CODE 13 : UPDATELOCATION METHOD CODE	149
CODE 14 : CHECKACCESS METHOD CODE	149
CODE 15 : FORWARDTOVIEW METHOD CODE.....	150
CODE 16 : INTRACK DEPENDENCY	160
CODE 17 : MAIN PROJECT NBACTIONS.XML.....	161
CODE 18 : EJB POM.XML	164
CODE 19 : EJB NBACTIONS.XML	164
CODE 20 : EJB PERSISTENCE.XML.....	165

5.1. Introduction

This part covers the details of the MEP implementations and the KMEP application. The technical details will be exposed.

The project is subdivided into five maven projects. There is one main project (parent project) and four sub projects. There is the KMEP-Common that contains the common classes for other projects. KMEP-EAR contains the persistence definition (database access). The KMEP-EJB contains the data model and configuration data loader classes. The last project, KMEP-WAR, is the implementation of the Kyoto Mobile Exergaming Project. This is the Web User Interface for the game.

Some project like KMEP-EJB and KMEP-EAR are in reality the construction of MEP. For historical reason, the name remains KMEP for the development. Sometimes, it can be some misunderstood about the naming.

5.2. KMEP-Common

The KMEP-Common project regroups utility classes used by other projects. The Figure 37 shows the different classes used in KMEP projects.

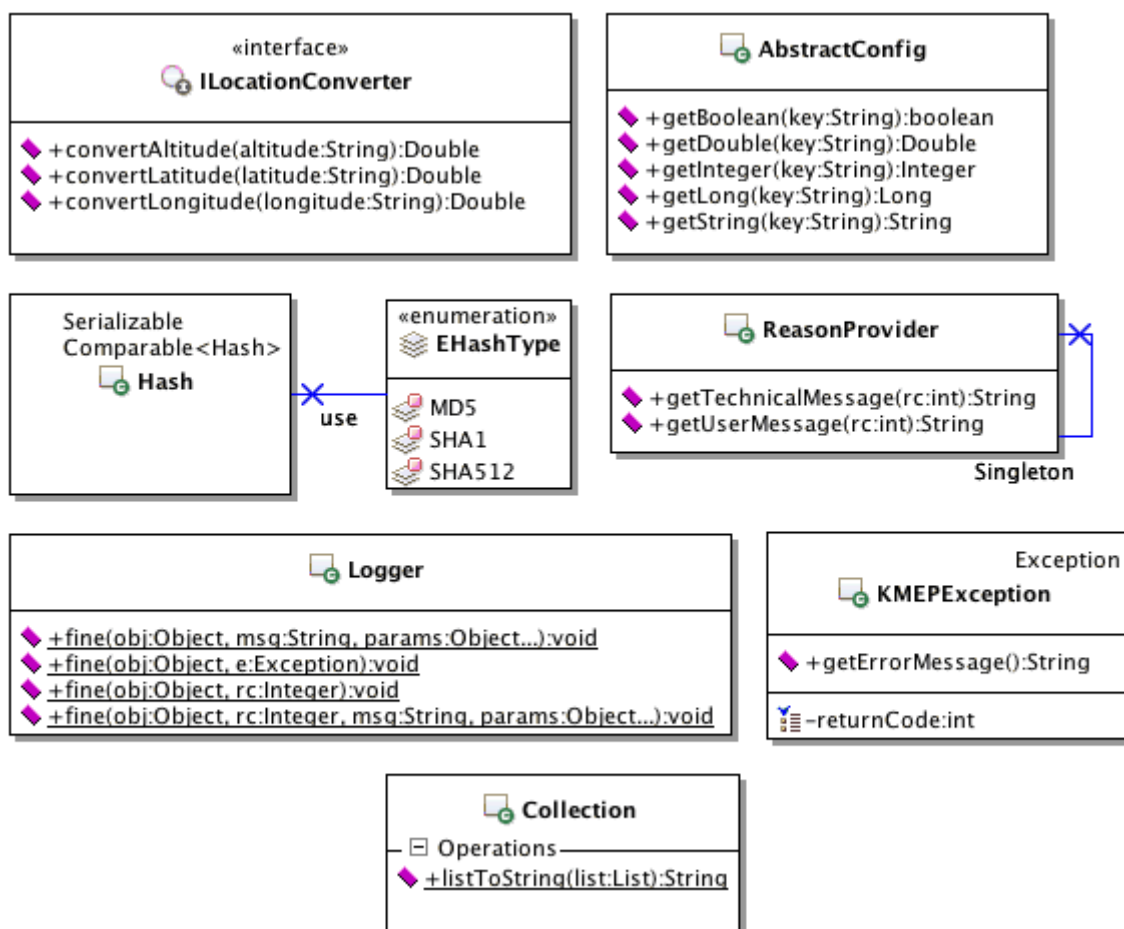


Figure 37 : UML – Common classes

ILocationConverter

This interface allows the creation of classes to convert from a non-inTrack Geo Localization Coordinates System to the inTrack Coordinates System.

AbstractConfig

The AbstractConfig allows creation a of configuration classes with a specific resources properties file. There are some methods to get native data type.

Hash & EHashType

This class and enumeration are especially useful to manipulate some text to hash. This is an implementation of the Java tools to hash. The enumeration contains the algorithms offered to the user of the class but if the JVM does not support the algorithm, an exception will be thrown.

ReasonProvider

The ReasonProvider contains all the Return Codes constants for KMEP. This is very useful for the code maintenance. This class works in addition of two property files (userMessages.properties and technicalMessages.properties) that contains textual messages for the error codes.

KMEPException

KMEPException is based on Return Code system. It contains the integer return code value. This is designed to be used everywhere in the application to have the view of what happens during code execution. Idea is to catch exceptions in methods and throw the KMEPException to explain what happens.

Logger

This utility class is a rewriting shortcut to the Java logger class. It adds some useful methods to log directly some exception or return code. There is also the possibility to format some text messages with an easy way to do it. The format system is based on “%n” notation that means the “n” is a numerical value replaced by an element in an Object table passed in parameters. The class diagram shows only the method for the fine level but there are the same methods for all the levels available in Java logger.

Collection

This class is a small utility to get a “toString” representation from a list of objects. Each element from the list passed in parameter of the unique method has its “toString” method called and prepared with the others.

5.3. KMEP-EJB Generalities

The EJB project and especially generalities approach this part of project with the classes that can be categorized as utility classes.

5.3.1. inTrack error bindings

The Figure 38 shows the class that configures the bindings between inTrack return codes and KMEP return codes. If the return code asked for binding cannot be resolve, there is a specific error code to alert for the unknown error.

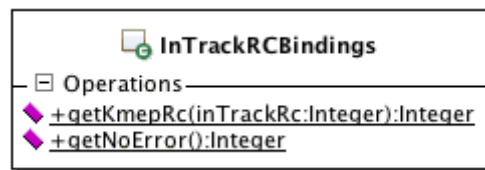


Figure 38 : UML – inTrack / KEMP error code bindings

5.3.2. Formatter Factories Interfaces

The Figure 39 shows the formatter factories that designed to create the formatter for some model classes. The name of the methods and parameters describe correctly the type used to create the formatters. Later in this part of document, the model will be described with UML diagrams but the “getFormater()” methods does not appear on them for simplification reasons.

IKmepEntityFormaterFactory

This factory interface allows creating formatter for the KmepEntitiy class (depending on the EKmepEntityType).

IDialogFormaterFactory

This factory interface allows creating formatters for the different NPC dialog classes.

IQuestFormaterFactory

This factory interface allows creating formatters for Quest class. There are also the methods to create formatters for the Material, Conditions and Rewards classes.

IMessengerFormaterFactory

This factory interface allows creating formatters for Message classes.

IStatisticFormaterFactory

This factory interface allows creating formatters for Statistic classes.

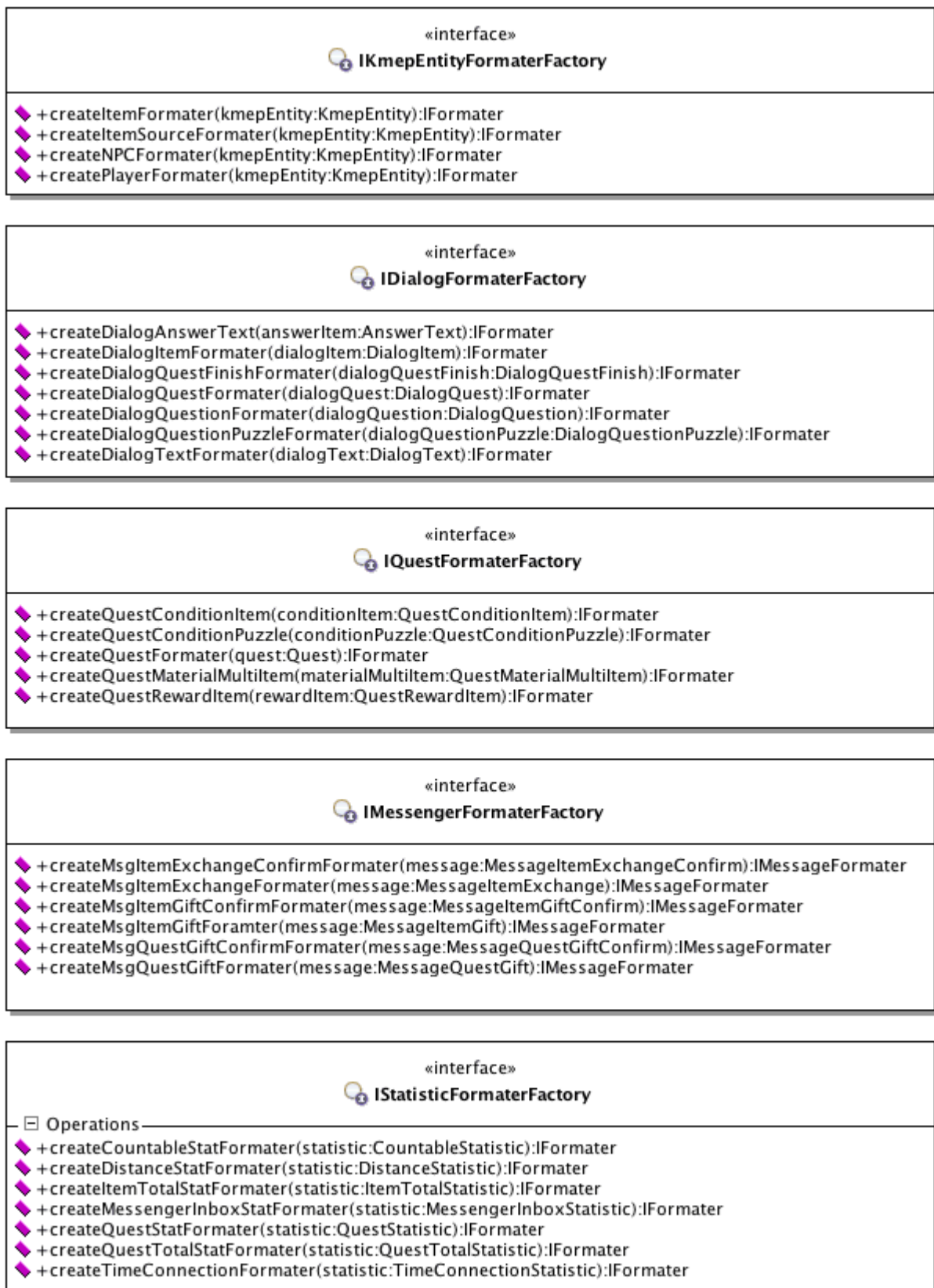


Figure 39 : UML – Formatter Factories interfaces

5.3.3. Formatter Interfaces

The Figure 40 shows the interfaces structure for used to create graphical representation of some data model classes.

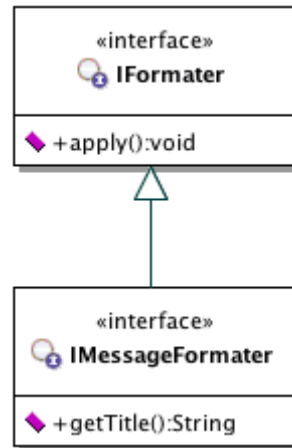


Figure 40 : UML – Formatter interfaces

IFormatter

This interface allows rendering a standard content for the MEP model classes. Idea is that the model classes can return an IFormatter instance build on the base of a factory passed in parameters. Generally, the model classes have this kind of code shown in Code 1.

```

public IFormatter getFormatter(IFactory... factory) {
    return factory.create...(this);
}
  
```

Code 1 : Formatter creation

The dots are replaced by the correct name for the creation of IFormatter. This important point to pay attention is that the object called to get the formatter is used to create the formatter.

IMessageFormatter

This interface has the same goal than the previous due to the inheritance. It adds the possibility to get a message title and is especially dedicated to the message model classes. To get the formatter, the code is similar to Code 1.

5.3.4. Server configuration

To handle the server configuration for the EJB part, some classes are required. There are two classes used for that. There is also a model class to store the Config values directly in the database. The Figure 41 shows these three classes.

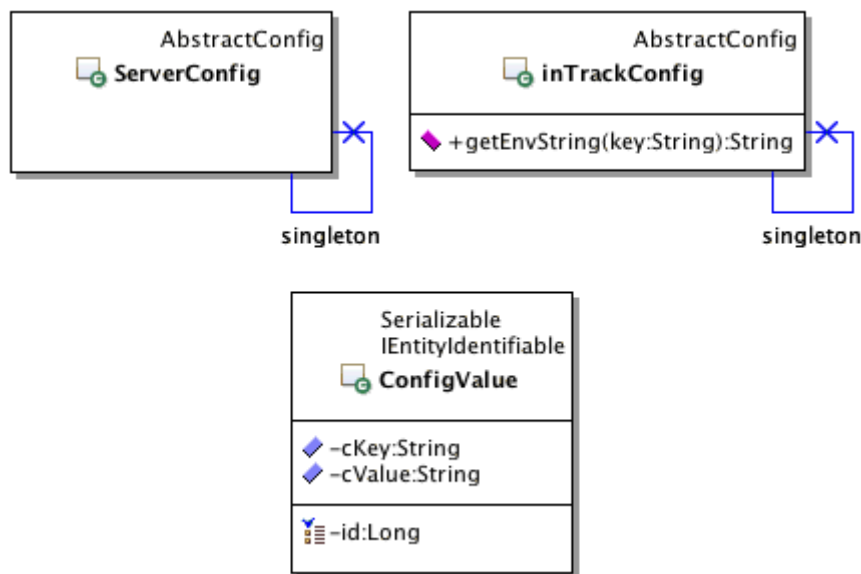


Figure 41 : UML – Server configuration classes

ServerConfig

The ServerConfig class is used to configure the MEP. For example, in the serverConfig property file, there is a value to configure the session duration.

inTrackConfig

The inTrackConfig class is used to configure the link between MEP and inTrack. In this implementation with the relative property file (inTrackConfig.properties), there is the possibility to switch between Dev/Prod environments for the data domain. It is especially used during the development phase.

With the appropriate configuration value and the use of “getEnvString()” method, the code can be the same with the key strings to find in the configuration file but they are automatically redirected to the right environment.

ConfigValue

This class is an Entity Bean that represents the association between a key and a value. The key is configured to be unique. This entity allows storing some configuration values that must to be persistent and can change during the execution of the application.

5.3.5. Identity

To allow managing data loading and storing, the MEP needs a system to link XML ids and EJB ids. The configuration of the EJB persistence layer does not allow creating manual ids. The XML configuration files required ids to link the data during the load phase.

In these conditions, it is necessary to have a mechanism to allow auto creation of ids in EJB layer with a link to XML id stored in files and managed manually.

The Figure 42 shows the `IIdentifiable` interface that helps to force the identification of XML objects and EJB objects.

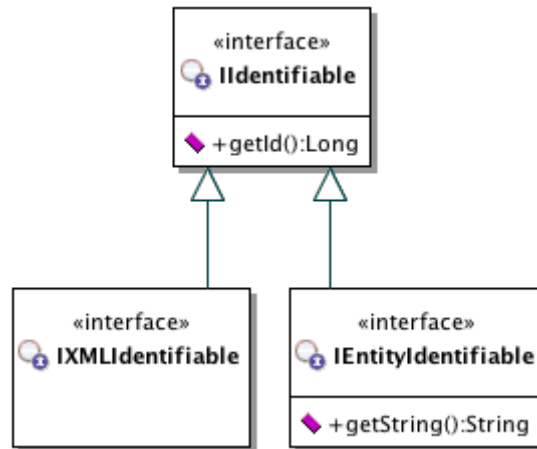


Figure 42 : UML – Identity interfaces

IIdentifiable

The `IIdentifiable` interface defines only one method. This is the method “`getId()`” that allows to retrieve the identifier from an objects that implements this interface. In the projects, all identifiers are in Long values.

IXMLIdentifiable

This interface allows distinguishing the XML objects to identify than the others. This particularly useful to force to have only this kind of objects in the XML loading process.

IEntityIdentifiable

Similar to `IXMLIdentifiable`, this interface defines a “`getString()`” method that allows getting the String representation of the object that implements the interface. This method is used in the “`toString()`” process. More details will be given later about the `toString` process for debugging.

5.4. KMEP-EJB Configuration

The EJB configuration part is the necessary to load and to persist the data when the server is deployed or restarted. It allows creating a game state with a complete data set.

5.4.1. XML Configuration

The XML Configuration classes and interfaces allow processing the load of persistent data and save them into the persistence layer. The persistence is organized in three phase:

1. Load the XML files
2. Persist the data values
3. Persist the references

In addition, of this process, there are some utility classes to manage the link between XML ids and database ids. The Figure 43 shows all the classes for the loading process.

It is important to notice that the XML files are loading in the destination classes through the JAXB technology. This allows annotating the classes with `@XmlRootElement`, `@XmlElement`, `@XmlElements...` These annotations provide the binding between XML file element and the attributes and classes in Java.

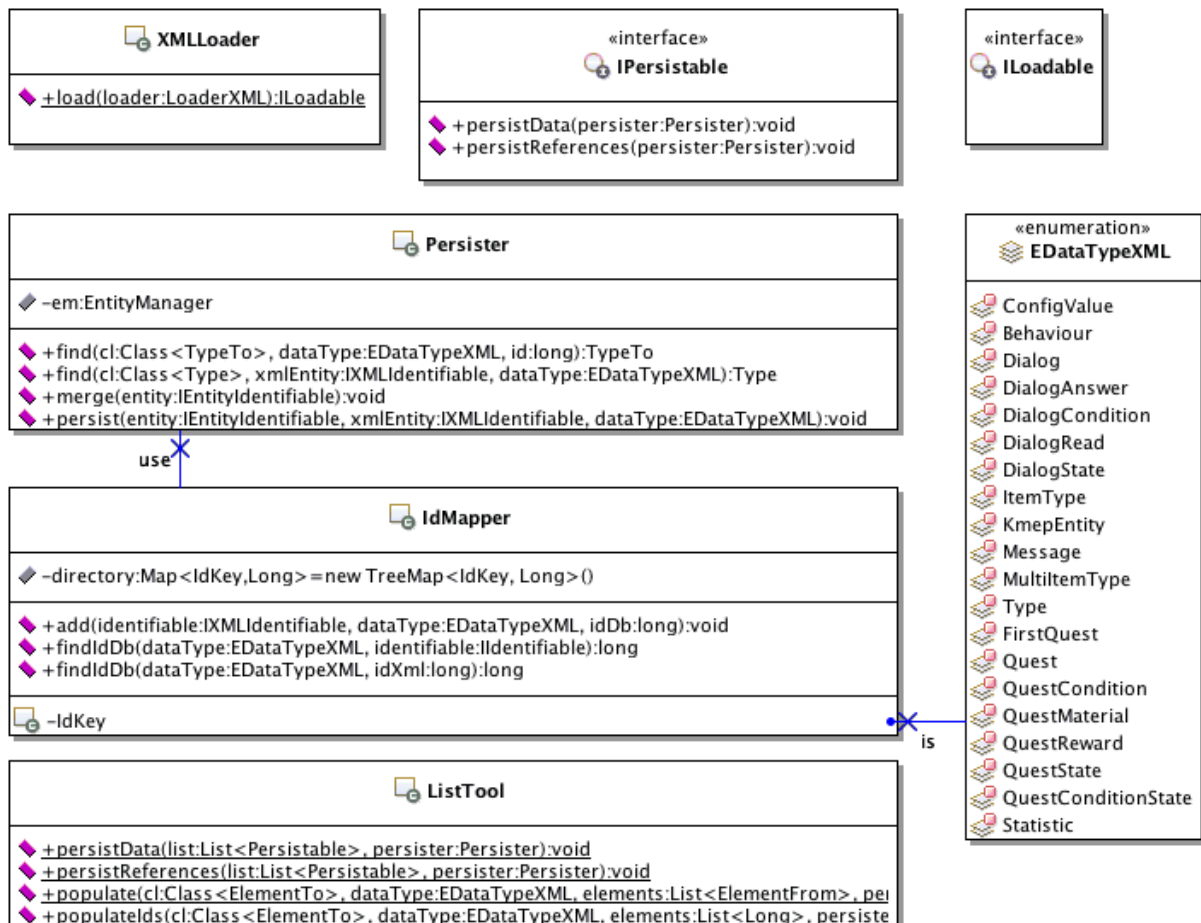


Figure 43 : UML – XML Configuration classes

XMLLoader

The XMLLoader class allows loading and validating the XML data contained in the files described in the XML Configuration Infrastructure.

```

/**
 * Load the data describe in the loader XML
 * @param loader The loader to retrieve the data to load
 * @return The loaded data
 * @exception KMEPEException See error doc. for more details
 */
public static ILoadable load(LoaderXML loader)
    throws KMEPEException {

    try {
    
```

```
// Create an instance of the loadable data
Iloadable loadable = (Iloadable)
    Class.forName(loader.getClassName()).newInstance();

// Prepare the JAXB context
JAXBContext context =
    JAXBContext.newInstance(loadable.getClass());

// Prepare the loader with the correct XML schema for the
// validation
Unmarshaller unmarshaller = context.createUnmarshaller();
SchemaFactory sf = SchemaFactory.newInstance(
    javax.xml.XMLConstants.W3C_XML_SCHEMA_NS_URI);
unmarshaller.setSchema(sf.newSchema(loader.getXsdFile()));

// Load and return the data
return (Iloadable)unmarshaller.unmarshal(loader.getXmlFile());
}
catch (Exception e) {
    throw new KMEPEException(e,
        ReasonProvider.ERR_GENERAL_XML_LOADING_ERROR);
}
}
```

Code 2 : XML Loader – load method

The Code 2 is interesting to see how JAXB works to load the data. In a first time, a context must to be loaded and after that, the XML Schema is configured for the validation. The context takes a class for reference of root element. Finally, the “unmarshal” method is called to load the data.

Iloadable

This interface allows a class to become loadable. It means that the class can be used with the XML Configuration Loading mechanism with XMLLoader class. It is important to notice that the Iloadable classes are always the classes that contain @XmlRootElement. This is the starting point for the XML document and the class structure.

The Code 2 shows also an interesting point to respect for the creation of an Iloadable class. These classes must have a default constructor without parameter. In general, that is always true for all the classes used by the JAXB technology. When a default constructor without parameters exists in class structure, an exception is thrown.

IPersistable

To be persisted, a class must implement the IPersistable interface. This interface provides the methods to persist the data values and references.

In this implementation, an Iloadable class is generally always an IPersistable class. A good correction for a future version is to IPersistable from Iloadable.

Persister

The Persister class is a utility class to regroup some mechanism and attributes used in the persistence (values or references) process. It does the work to persist the data and references with the use of the directory id management links.

It maintains the state between persisted data and their database id with the loaded data and their XML id. For that, the class uses the IdMapper class.

The internal class associate EDataTypeXML and the Long id into a single object with “compareTo” method based on the data type and in case of equality, on XML id.

IdMapper

IdMapper class consists into the id directory to manage the relation between XML ids and database ids. To add a larger usage facility, the class allows to use non-unique XML id but unique by data type categories.

This is especially useful to manage the XML files manually but there is some element to pay attention. For example, the KmepeEntity and Behaviors ids are used into more than one file. It results into some difficulties to manage the uniqueness of ids. A good idea to avoid duplicates ids is to attribute some id ranges for each file.

EDataTypeXML

The data type enumeration is used to distinguish the XML ids. These values are used when a data is persisted and when there is the references building process too. The enumeration values correspond to the category of data that exists in MEP.

ListTool

The ListTool utility allows manipulating list used in the XML configuration part. Due to the fact, the list items are IPersistable, this is easy to build generic method to save the data and/or the references. The reusability of code is very big with this utility class.

5.4.2. XML Configuration Infrastructure

The XML Configuration Infrastructure allows loading the data from XML files to be persisting in the persistence layer of the MEP. The classes shown on Figure 44 are divided into two groups. The first one consists on the data storage of the different XML files with their relative XSD. There is also the binding to the Java class for the data loading. This structure allows preparing the data load for the MEP.

The second part with the ConfigValue XML is dedicated to the load of ConfigValue entities described earlier.

LoaderCollectionXML

The LoaderCollectionXML is the root class to load a collection of LoaderXML. It contains also a specific LoaderXML to handle separately the persistence of ConfigValueXML data. This separation is needed because during the process of loading, there is no way to know which data are loaded. All the process manipulates only ILoadable or IPersistable objects.

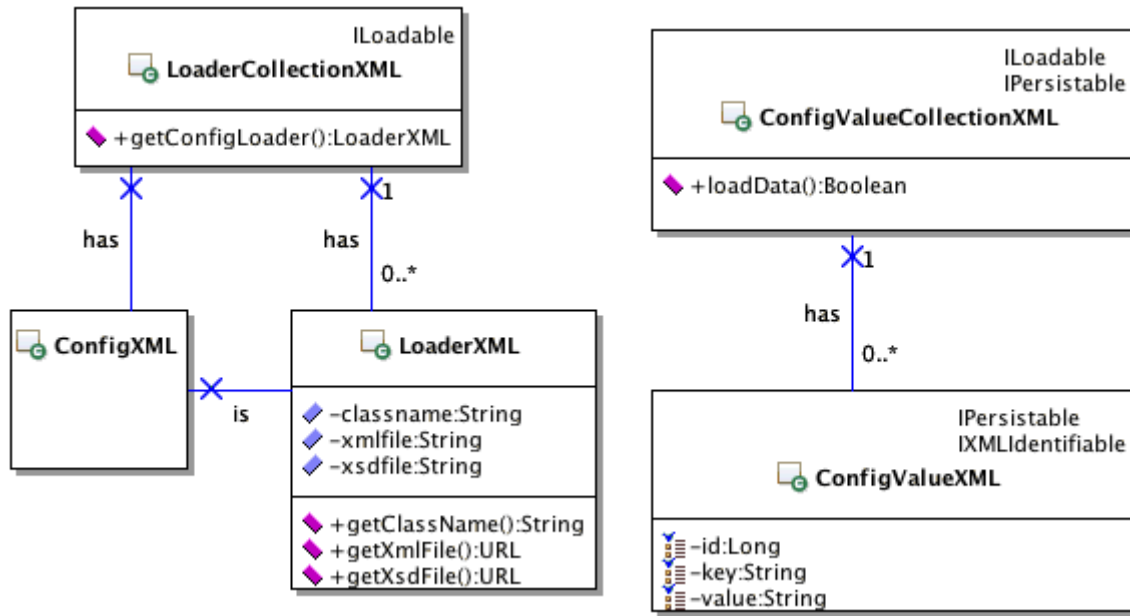


Figure 44 : UML – XML Configuration Infrastructure classes

LoaderXML

LoaderXML contains the definition of a XML file to load. The definition consists in the XML file to load and its XSD for the validation. There is the class name to create the instance of the root element to load. The class name is the fully qualified name.

ConfigXML

The ConfigXML is only here to do the difference between all ILoadable/IPersistable data than the ConfigValueXML but the definition of the file is the same than the others.

ConfigValueCollectionXML

This is the collection of ConfigValueXML to load. The method “loadData()” allows to know if the other data have to be load or not. This particularly useful to override the database values if necessary but it is important to notice that the loading process is done at each server start. If the “loadData” is true in the file, at each restart of the server, the data will be loaded and erased the present data.

ConfigValueXML

The ConfigValueXML handle the values from the XML files.

5.4.3. XML Configuration Location

The locations data are stored directly into the inTrack format. The location is used for the entities and first quest in the actual state of the project. The Figure 45 shows the class with the latitude, longitude and altitude double attributes.

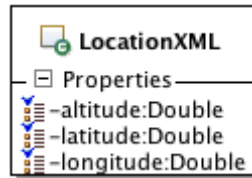


Figure 45 : UML – XML Location class

5.4.4. XML Configuration MEP Entities

The MEP entities are Players, Items, ItemSources and NPCs. Each of these entities shares the same structure. The Figure 46 shows the structure of the entities.

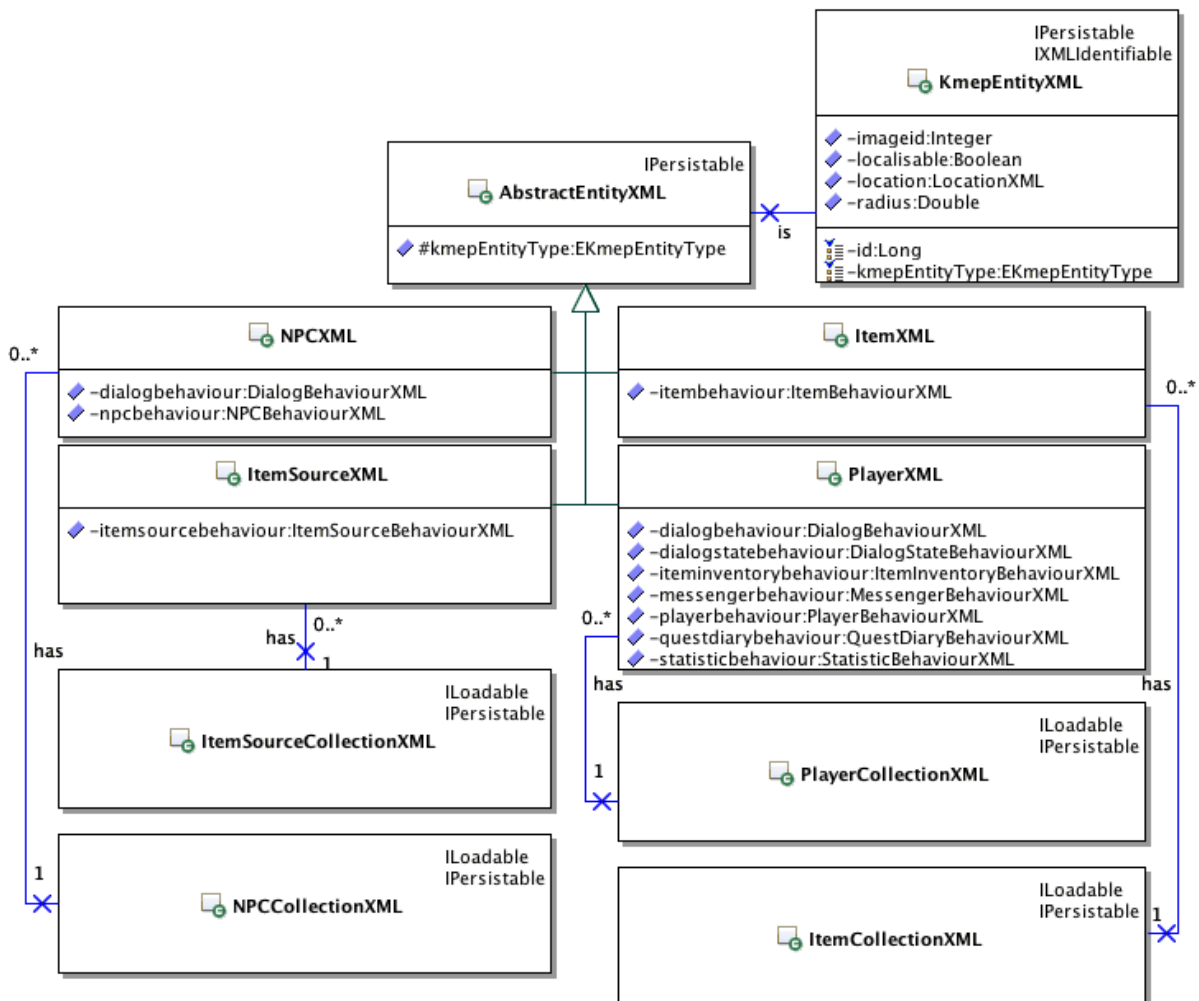


Figure 46 : UML – XML MEP Entities classes

KmepEntityXML

The KmepEntityXML contains the shared data between the entity types. There is also the location for the geo localization system.

AbstractEntityXML

This abstract class regroups all the elements that the entities have in common. There is specificity for the loading phase with EKmepEntityType. The XML files do not contain the type in their XSD. To recognize the different entities, there is a need to know which entity has which type. For that, a field is used and configured by the subclasses. During the persistence phase, the KmepEntity are created with this type.

PlayerXML, ItemXML, ItemSourceXML and NPCXML

These entities are characterized by the behaviors loaded. There are no other specificities.

PlayerXML, ItemXML, ItemSourceXML and NPCXML collections

The collections allow loading the entities. They are the entry points for the @XmlRootElement.

5.4.5. XML Configuration Behavior

This configuration part is dedicated to the behavior that a KmepEntity has. The behaviors contains in general list of objects. The loading phase is most intensive on the lookup for the references build. The Figure 47 shows the structure of behaviors XML.

AbstractBehaviourXML

The abstract behavior is the main class for all the behaviors. The shared attribute is the XML id. The advantage to have an abstract class is to store in the same collection type and constraint to only behavior type and not to ILoadable/IPersistable types.

Behaviors

The different behaviors types are exposed in the Figure 47. There is nothing special to discuss about them. The collection used to store the references id is just discussed after this paragraph.

IdCollectionBehaviourXML

This collection allows storing XML id of different elements independent from the data type that reference. This particularly useful to reduce the number of classes that handle the same type of collection.

To load the data correctly, there is some configuration to do in the collection class. The Code 3 shows the configuration part with the annotation.

```

// The collection of ids
@XmlElements ( {
  @XmlElement(name = "dialogid", type = Long.class),
  @XmlElement(name = "dialogstateid", type = Long.class),
  @XmlElement(name = "itemid", type = Long.class),
  @XmlElement(name = "messageid", type = Long.class),
  @XmlElement(name = "queststateid", type = Long.class),
  @XmlElement(name = "statisticid", type = Long.class)
})
private List<Long> ids = new ArrayList<Long>();
  
```

Code 3 : Id Collection annotation configuration

With this configuration, all the ids collections could be loaded depending on the name they have. There is a risk of mixed data with incorrect loading but the XML Schema is there to prevent this kind of problems.

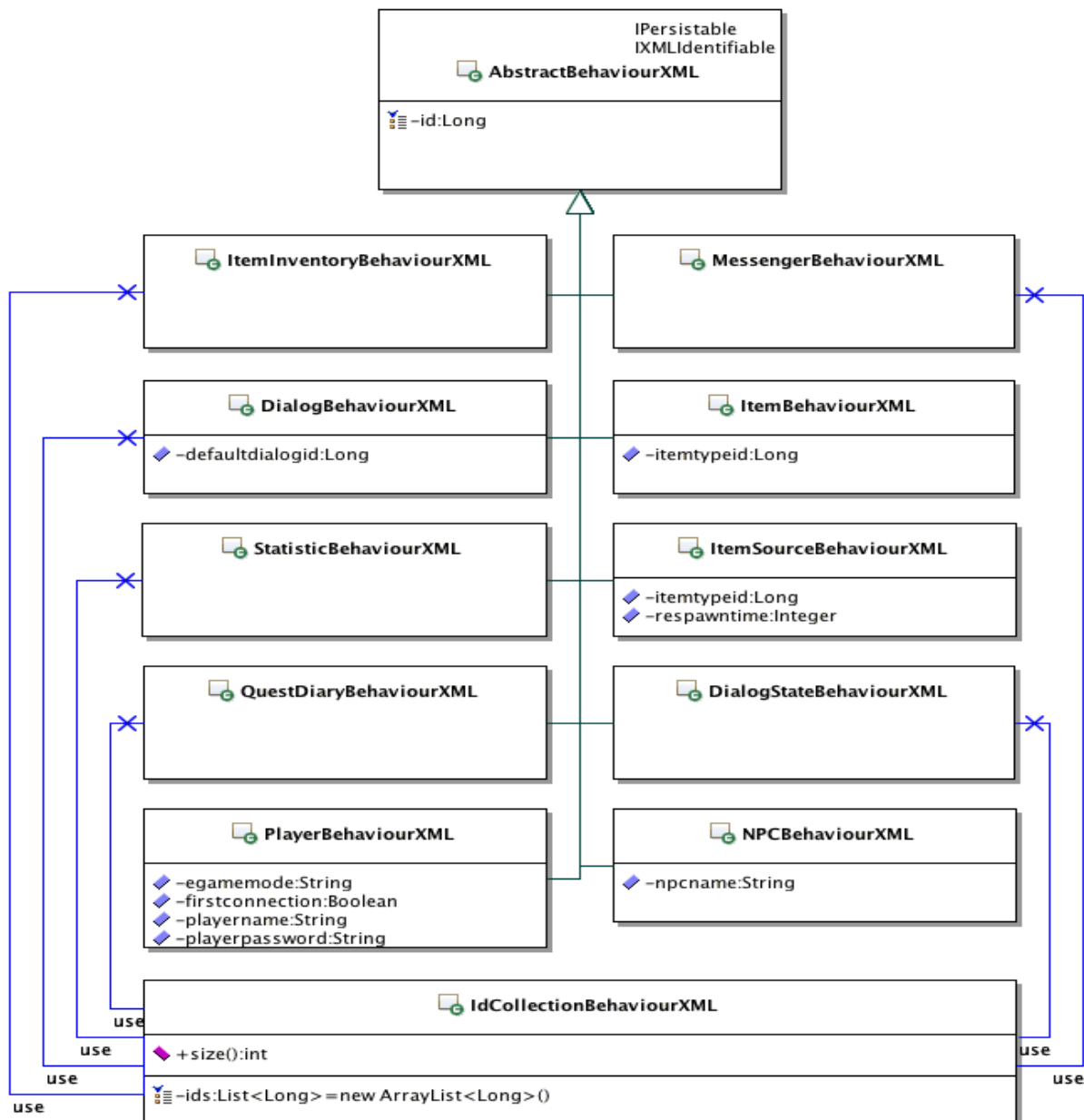


Figure 47 : UML – XML Behaviour classes

5.4.6. XML Configuration Dialog

The Figure 48 shows the structure of the dialog configuration classes. There is a similar structure of inheritance as the KmemEntities.

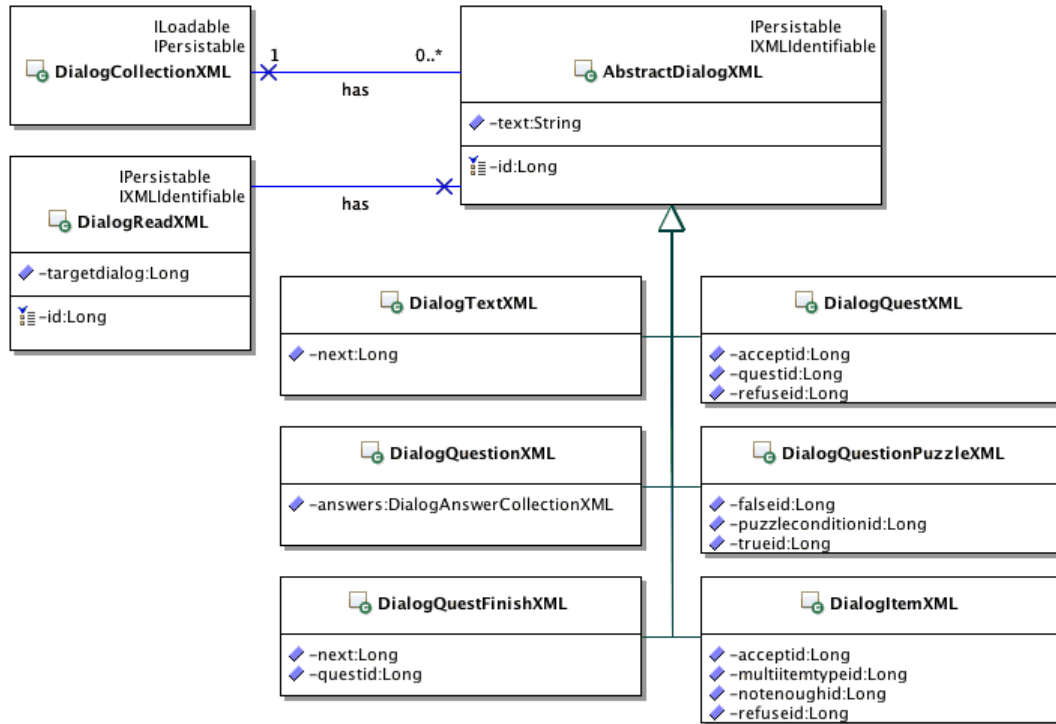


Figure 48 : UML – Dialog XML classes

DialogReadXML

The DialogReadXML is the specific class to load the data that allows creating the dialog state.

5.4.7. XML Configuration Dialog Answer

The Figure 49 shows the structure for the dialog answer construction. The principle of the abstract class is always the same for the storage in collection in addition of other reasons like common attributes.

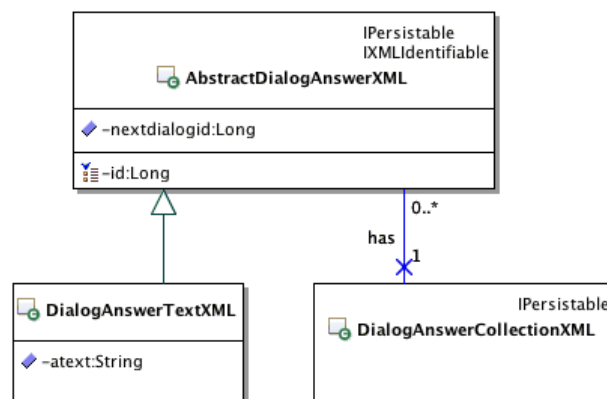


Figure 49 : UML – Dialog Answer XML classes

5.4.8. XML Configuration Dialog Condition

The Figure 50 shows the structure for the dialog conditions construction. There are no specificities to discuss in this structure.

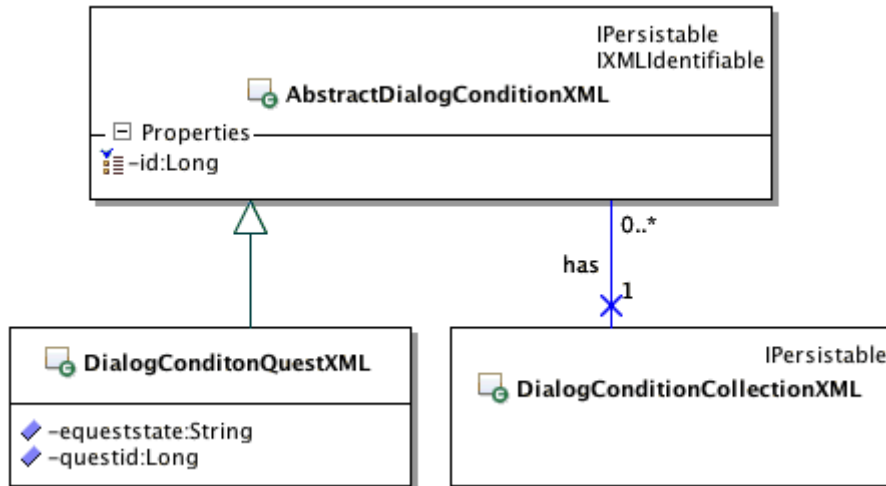


Figure 50 : UML – Dialog condition XML classes

5.4.9. XML Configuration Dialog State

The Figure 51 shows the structure for the dialog state construction. There are no specificities to discuss in this structure.

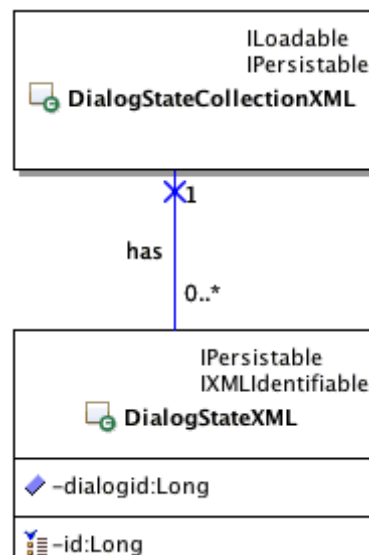


Figure 51 : UML – Dialog State XML classes

5.4.10. XML Configuration Item

The item XML configuration regroups two kind of configuration. The first is dedicated to the item type configuration and the second for the multi item type configuration. The Figure 52 shows the structure of these two kinds of classes.

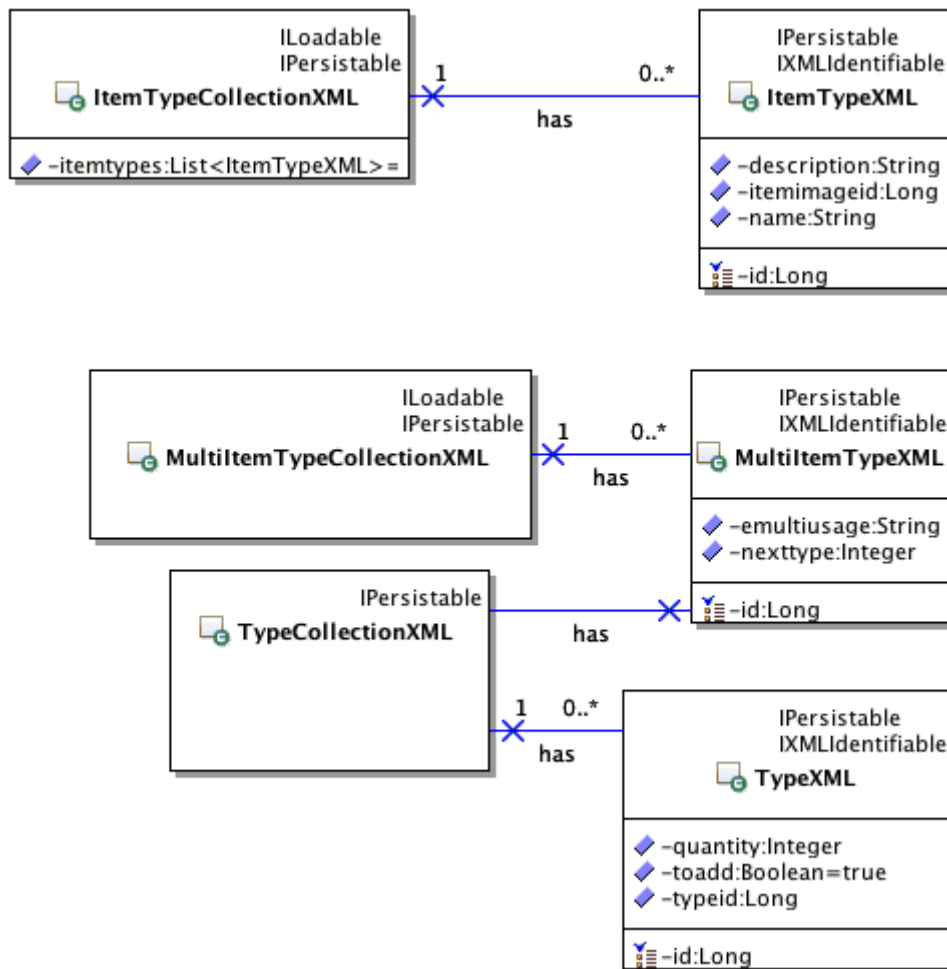


Figure 52 : UML – Item XML classes

5.4.11. XML Configuration Message

The Figure 53 shows the structure for the messages construction. There are no specificities to discuss in this structure.

```

// The collection of messages
@XmlElements ( {
    @XmlElement(name = "messageitemexchange",
        type = MessageItemExchangeXML.class),
    @XmlElement(name = "messageitemexchangeconfirm",
        type = MessageItemExchangeConfirmXML.class),
    @XmlElement(name = "messageitemgift",
        type = MessageItemGiftXML.class),
    @XmlElement(name = "messageitemgiftconfirm",
        type = MessageItemGiftConfirmXML.class),
    @XmlElement(name = "messagequestgift",
        type = MessageQuestGiftXML.class),
    @XmlElement(name = "messagequestgiftconfir",
        type = MessageQuestGiftConfirmXML.class)
})
private List<AbstractMessageXML> messages =
    new ArrayList<AbstractMessageXML>();
  
```

Code 4 : Messages annotation configuration

The Code 4 shows the configuration for the collection of messages. The interesting point is that the list is restricted to the AbstractMessageXML super type and the elements accepted for the list are the children's types. The name's values correspond to the name's fields in the XML document.



Figure 53 : UML – Messages XML classes

MessageItemIdCollectionXML

The MessageItemIdCollectionXML represent the references to the items corresponding of the messages. This is a “generic” list of long values corresponding to the XML ids.

5.4.12. XML Configuration Quest

The quest XML configuration regroups two kind of configuration. The first is dedicated to the first quest configuration and the second for the quest configuration. The Figure 54 shows the structure of these two kinds of classes.

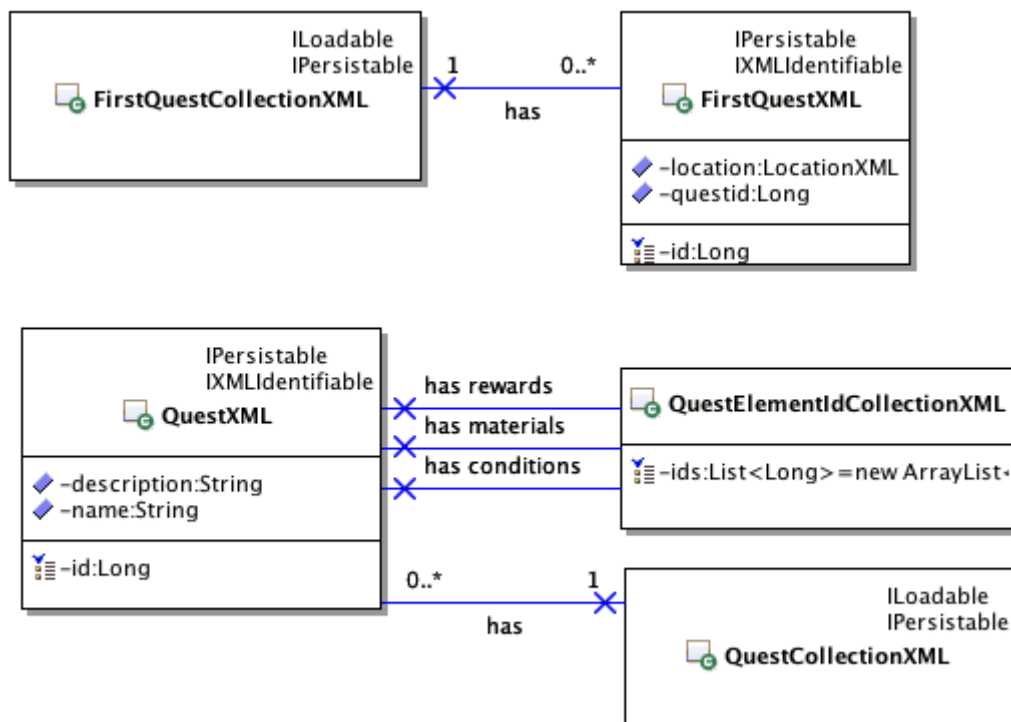


Figure 54 : UML – Quest XML classes

QuestElementIdCollectionXML

The QuestElementIdCollectionXML allows storing long id values in a constraint list. The three different collections used in QuestXML are constraint by the XML Schema used. Each list contains only one type of ids. The Code 5 shows the configuration done to load in the same list different XML ids.

```

// The collection of quest
@XmlElements({
    @XmlElement(name = "conditionid", type = Long.class),
    @XmlElement(name = "materialid", type = Long.class),
    @XmlElement(name = "rewardid", type = Long.class)
})
private List<Long> ids = new ArrayList<Long>();
    
```

Code 5 : Quest Element Id configuration annotation

5.4.13. XML Configuration Quest Material

The Figure 55 shows the structure for the quest materials construction. There are no specificities to discuss in this structure.

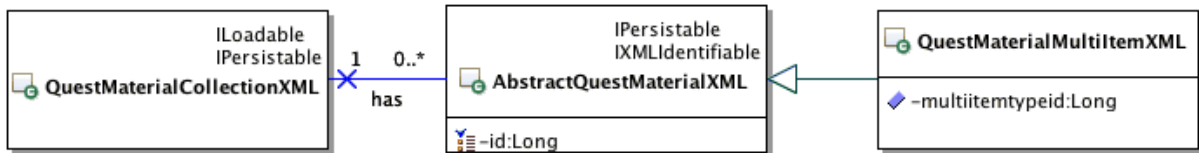


Figure 55 : UML – Quest Material XML classes

5.4.14. XML Configuration Quest Condition

The Figure 56 shows the structure for the quest conditions construction. There are no specificities to discuss in this structure. The configuration for the collection loaded has the same principle already shown before in this section.

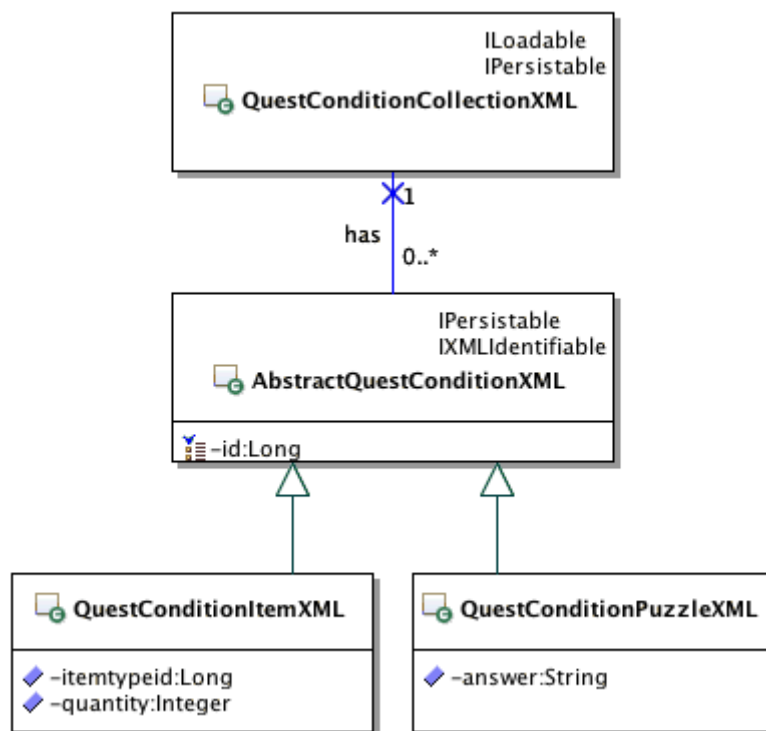


Figure 56 : UML – Quest Condition XML classes

5.4.15. XML Configuration Quest Reward

The Figure 57 shows the structure for the quest rewards construction. There are no specificities to discuss in this structure.

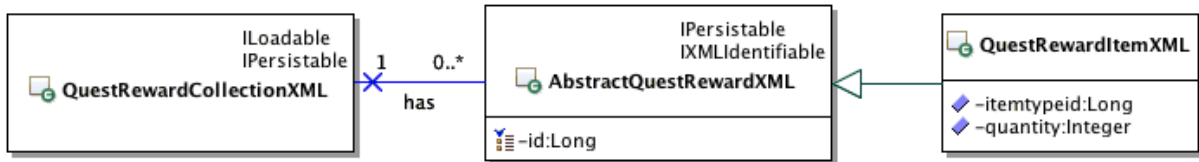


Figure 57 : UML – Quest Reward XML classes

5.4.16. XML Configuration Quest State

The Figure 58 shows the structure for the quest states construction. There are no specificities to discuss in this structure.

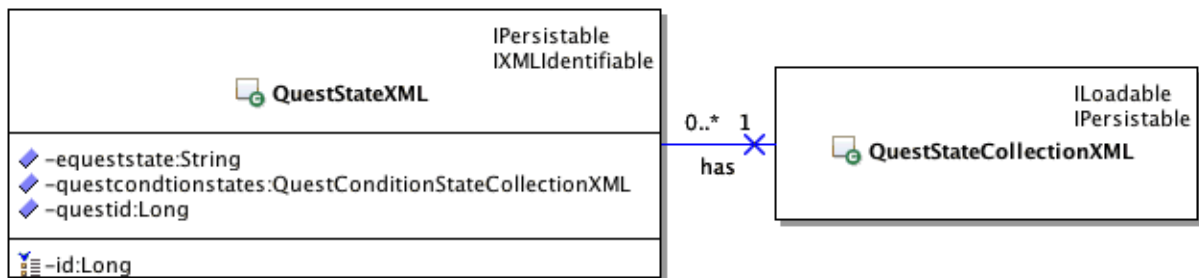


Figure 58 : UML – Quest State XML classes

5.4.17. XML Configuration Quest Condition State

The Figure 59 shows the structure for the quest condition states construction. There are no specificities to discuss in this structure.

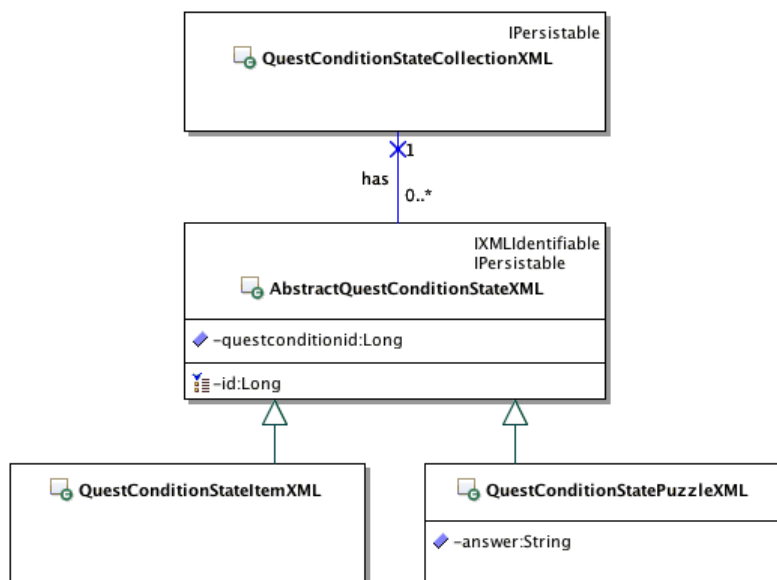


Figure 59 : UML – Quest Condition State XML classes

5.4.18. XML Configuration Statistic

The Figure 61 shows the structure for the statistic construction. There are many classes present in the diagram. One reason for the profusion of classes is that there is no way to know which statistic is loaded with the code presented in Code 6. The hypothesis is that the abstract class `AbstractCountableStatisticXML` is not abstract. The problem is this sample code is there is no way to distinguish a “connection” statistic from “distance” statistic.

A possibility to make the difference is to add an element in the XML structure to get the statistic type and convert to the `EStatisticName` enumeration. However, with the proposed solution, there is another problem. How to constraint the presence of one and only one statistic name. This is not possible with an XML Schema, the dependence between the data cannot be done in XML directly without external tool.

```
// The collection of statistics
@XmlElements({
    @XmlElement(name = "connection", type = CountableStatXML.class),
    @XmlElement(name = "distance", type = CountableStatXML.class),
    ...
})
private List<CountStatisticXML> statistics =
    new ArrayList<CountStatisticXML>();
```

Code 6 : Statistic annotation demonstration

With the solution proposed actually, it is possible to make the difference between elements because each statistic name has its own XML element. With the appropriate configuration like shown on Code 7, this is possible to load known statistic and build correct ones.

```
// The collection of statistics
@XmlElements({
    @XmlElement(name = "connection",
        type = ConnectionStatisticXML.class),
    @XmlElement(name = "distance",
        type = DistanceStatisticXML.class),
    ...
})
private List<AbstractCountableStatisticXML> statistics =
    new ArrayList<AbstractCountableStatisticXML>();
```

Code 7 : Statistic annotation configuration sample

Actually, the solution is not completely implemented. There is the possibility to create more than one statistic of the same name for one player. The solution to avoid the problem is to add the group notion for the statistic. The Figure 60 shows idea.

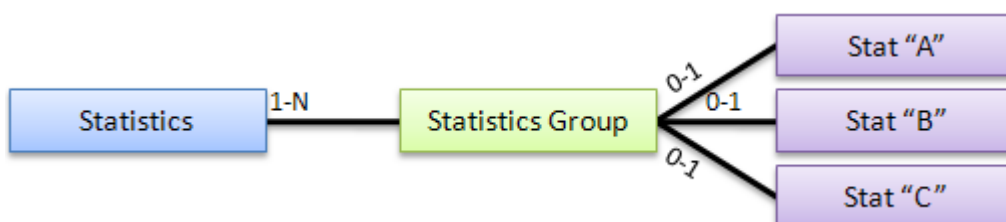


Figure 60 : Solution to have a unique statistic name by player

With the structure XML has shown, this possible to constraint with a XML Schema to have only one of each statistic type for a same player. There is also a modification to do in the statistic behavior to have a reference to the Statistic Group rather than directly to the statistics. After that, there is also the lookup problem to retrieve the Statistic Group that is not saved in the persistence layer.

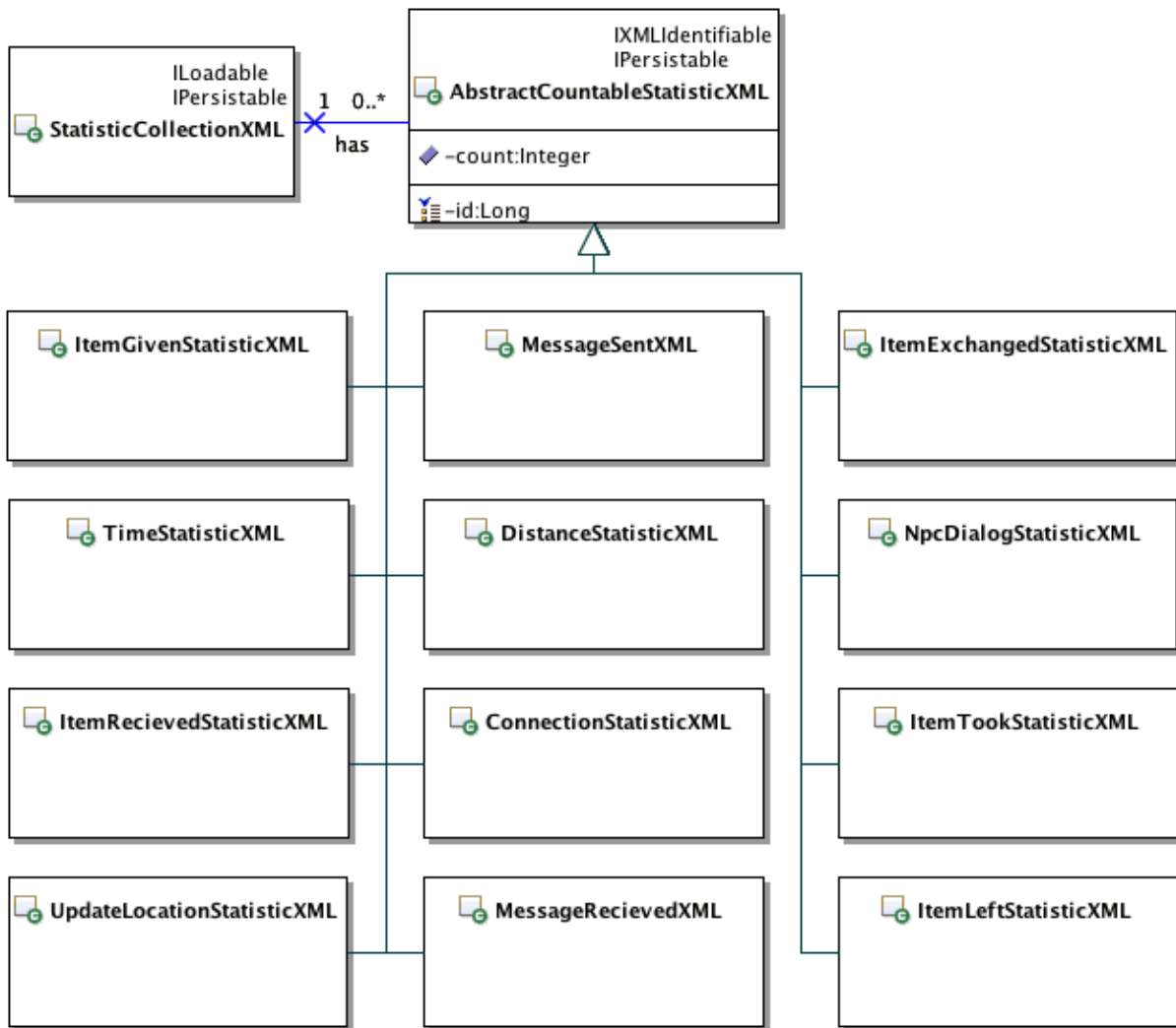


Figure 61 : UML – Statistic XML classes

The solution proposed and implemented is quite enough actually. The reason is that there are no needs to configure this part of the application when the server is deployed. There is only development or backup reason yet. This part of the configuration is not so important.

5.5. KMEP-EJB Model

The EJB model part contains the data model of the application with the persistence. This application layer maintains the state of the MEP.

5.5.1. Model base

The base of the model consists into the principal model classes shown on Figure 62. The KmepEntity is the fundamental class for the model.

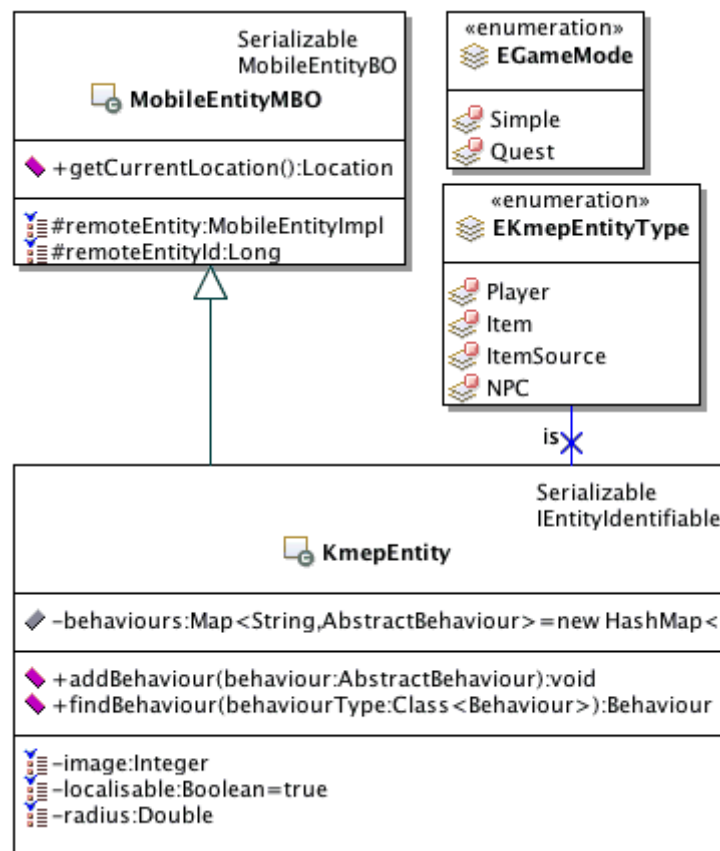


Figure 62 : UML – Model base classes

MobileEntityMBO

The MobileEntityMBO is provided by inTrack platform to extend the inTrack model to this project. The inheritance with the KmepEntity allows it to become a localizable entity. This is not enough to use inTrack completely but this is the first step to do it.

KmepEntity

This is the main class of the MEP. It represents all the different entities. The KmepEntity stores a Map of behaviors. The behaviors allow extending the functionality of the entity. The methods add and find allows managing the behaviors. There is also a private “get/set” for behaviors but only present in the code for the EJB annotations.

The `KmepEntity` has a flag to allow the location or not. This is particularly useful with the manner of using `inTrack`. This flag avoid manipulating `inTrack` to remove and add again the location system for an entity. When a player carries an item for example, there is no need to locate it because the player and item move together. However, the player leave the item on the map, the location must to be reactivated.

The `inTrack` platform offers the method to do that but there is a price of performance to pay when they are used. With a simple flag, there is no more need to do something with `inTrack`. This is an easy and good solution.

`EKmepEntityType`

To make the difference easily between entities, there is an enumeration of entity types. This is a convenient reason for its existence. To differentiate the entities, a more complicated can be used. For example, a rule-based system on behavior can be used but it complicates a lot the implementation and the advantages is not enough to do that.

`EGameMode`

For historical reason, the `EGameMode` type remains in the implementation but is not used actually. The game rules planned a game with two different game play based on the choice of the game mode but actually, there is only one gameplay implemented. The Simple game mode is the implemented game.

5.5.2. Model Behaviors

The model behaviors implement a part of the business methods to manage the game rules. The other part is directly implemented in the services discussed later in this document part. The Figure 63 shows the behavior classes and their most important methods.

`AbstractBehaviour`

The `AbstractBehaviour` is the super class for all the behaviors. The attribute `behaviourName` is used to lookup the behaviors in the method `find` from `KmepEntity` class. `KmepEntity` attribute is a reference of the behavior's owner for an easy navigation.

`DialogBehaviour`

`DialogBehaviour` allows managing a dialog for a NPC in general. The dialog behavior is composed by a default dialog always shown if there is no other option and optional dialogs. The method "`getDialog()`" allows retrieving the best dialog to the corresponding `KmepEntity` state.

`DialogStateBehaviour`

This behavior permit to manage the dialog states for a player. With this management, the dialog already read can be memorized. The method "`addState()`" provide the necessary to add new dialogs state and the method "`isRead()`" allows finding a dialog already read or not.

ItemBehaviour

ItemBehaviour class corresponds to an instance of an ItemType. The KmepeEntity is used to locate the item on the map or to store it into the players' inventory and the behavior is used to know which item it is.

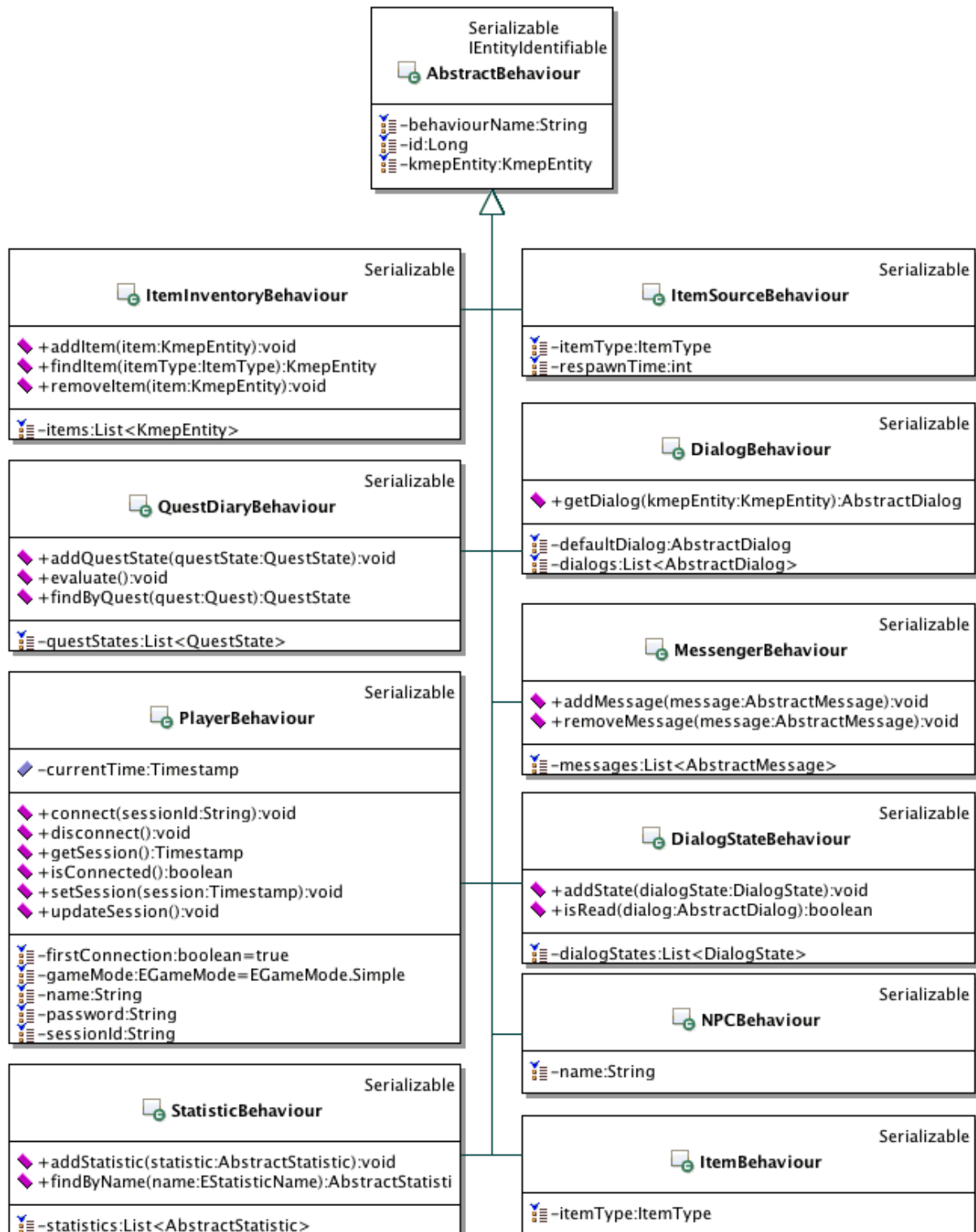


Figure 63 : UML – Model behavior classes

ItemInventoryBehaviour

This behavior is used to manage the inventory of a KmepeEntity. The methods “add” and “remove” allows manipulating the inventory and the “find” method to retrieve an Item.

ItemSourceBehaviour

ItemSourceBehaviour represent a source of item. This behavior is a sort of item factory with a delay between two item drops. The item is not created from the item source, it is created in a service.

MessengerBehaviour

The messenger behavior allows managing the notification system between players. This simple system provides the necessary to store and remove messages with the corresponding methods.

NPCBehaviour

NPCBehaviour is not special. This behavior is only here to store the NPC name actually.

PlayerBehaviour

This behavior is used to manage the players’ data and the connection state to the game. The methods shown allow maintaining the session for a player. The password stored in this class is in String type but the service that manage the player use the Hash class to prepare the password hashed string.

QuestDiaryBehaviour

The quest diary behavior allows managing the quest states for a player. The method “evaluate()” is used to update the quest states. The “findByQuest()” allows finding a quest state corresponding to a quest.

StatisticBehaviour

StatisticBehaviour is used to store the statistic for a player. The methods allow adding new statistics and finding the statistics by their name.

5.5.3. Model Dialog

The Figure 64 shows the dialog model architecture. The different relations allow building a tree (more a graph than a tree) for the dialog flow.

AbstractDialog

The abstract dialog is the main class for the dialogs. It stores the dialog conditions if necessary and the flag to know if the dialog is already read or not. There are two interesting

methods. The first one is use to create the dialog state and the second to know if it is possible.

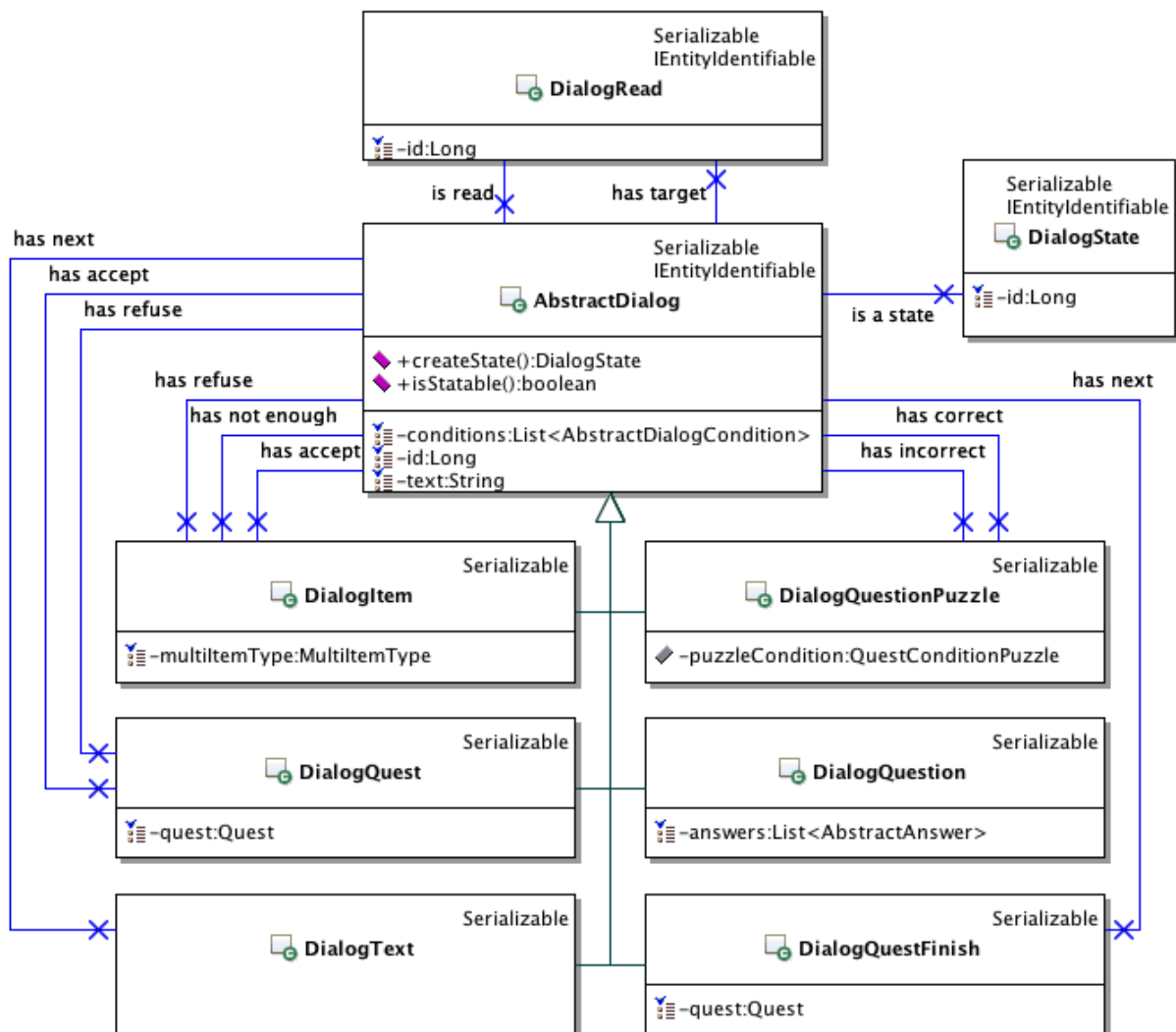


Figure 64 : UML – Model dialog classes

DialogRead

The dialog read can configure the dialog flow to memorize which starting point dialog is considered as read. The “target” relation means this is the starting point of the dialog flow and the “is read” relation can be considered as the ending point. In the facts, when the ending point is reach, the dialog mechanism gets the starting point of the flow to create the dialog sate. These operations are done in the services.

DialogState

DialogState class is used to memorize that the dialog from the relation “is a state” is already read. The dialog states are stored in the DialogStateBehaviour for the players.

DialogItem

A DialogItem represent a dialog that the player can receive or give some items. When it is configured as receive items, the “has not enough” relation could be null and is not used. The relations “has accept” is used when the player accept the proposition and “has refuse” when he refuses the offer.

In the case of give items, the “has not enough” is encouraged to be used because the player can do not have sufficient amount of items. In this case, it can be interesting to have a specific dialog for this case. If the “has not enough” is not configured, the “has refuse” is used.

DialogQuest

DialogQuest class corresponds to the dialog offer in dialog flow. There are two relations depending on the answer of the player. There is also a relation to the quest proposed. A special use of this class is the quest reference is null. In this case, it means that the MEP has to offer a first quest to the player depending on his position.

DialogQuestFinish

This class allows finishing a quest (validating a quest). The relation “has next” is used to get the next dialog. There is also the relation to the quest to know which one to finish.

DialogQuestion

DialogQuestion is not directly related to another dialog because this relation is delegated to the dialog answer classes. The class stores the answers that the player can see.

DialogQuestionPuzzle

To check the answer of a quest puzzle, the DialogQuestionPuzzle keep a relation to the QuestConditionPuzzle corresponding to the quest puzzle. There are also two relations to navigate if the answer is correct or no correct.

DialogText

The DialogText is the simplest class for the dialog model. This is only a textual dialog without interaction. The relation “has next” allows continuing the navigation in the dialog flow.

5.5.4. Model Dialog Answer

The Figure 65 shows the architecture of the dialog answer. These answers correspond to the dialog question. Abstract class contains the reference to the next dialog if the answer is chosen. The abstract method “process” can be used to do something depending on the answer.

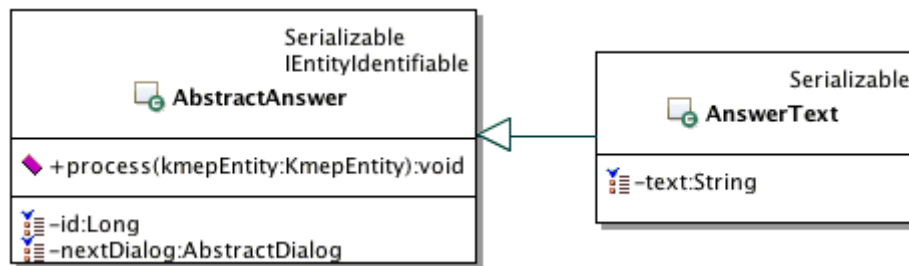


Figure 65 : UML – Model Dialog Answer classes

5.5.5. Model Dialog Condition

The Figure 66 shows the model dialog conditions architecture. These condition are used when it necessary to get the correct dialog between all the dialogs present. The abstract method “isValid()” provides the mechanism to define if the dialog can be read or not by the player based on his current state.

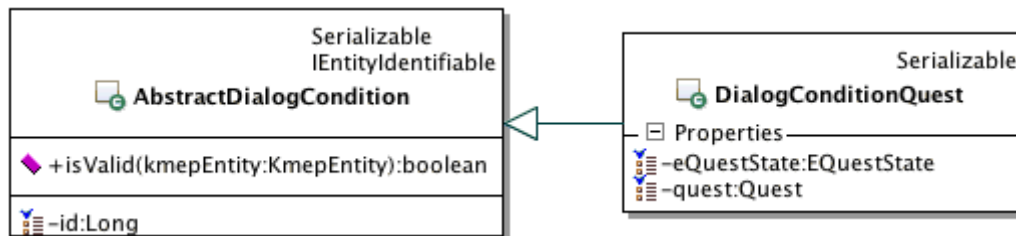


Figure 66 : UML – Mode Dialog Condition classes

5.5.6. Model Item

The item model is decomposed in two categories: ItemType and MultitemType. The Figure 67 shows these two categories and two another utility classes.

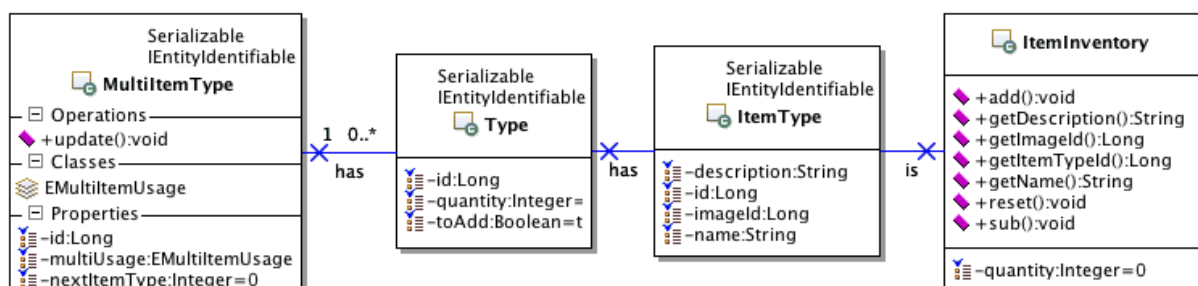


Figure 67 : UML – Model Item classes

ItemType

The item type is designed to represent a family of item. It contains the common data for an item.

ItemInventory

ItemInventory is not directly attached to the model. This is not an entity in the meaning of EJB because there is no annotation @Entity. This class is used to transfer from the inventory

to the player the quantity of a certain item. There are some shortcuts methods to access the data of ItemType object. The methods “add”, “sub” and “reset” allow managing the amount of item inventory.

Type

Type class is not correctly named. It represent the possibility to create an order for some classes like DialogItem or otherwise. It contains a quantity of the ItemType related and the flag to know if the items are to be added or removed.

MultitemType

This class is a complex order to add or remove item during the game execution. It allows configuring some manner to distribute or get items. The intern enumeration defines the rules to give or take items. Actually, there are only two ways. The first allow getting or taking all the items in one time and the second to run a circular distribution (or get). This second one means there is only one item after the other available. It is very useful to allow a uniform distribution of some items. The “update” method can be used to update the state of the item distribution.

5.5.7. Model Message

The message model architecture offers some extended mechanism to run functionalities like exchange item or give quest... The Figure 68 shows the class diagram.

AbstractMessage

The abstract message is the super class for all messages. It contains the flag read to know if the message is read or not.

MessageItemGift

This class allows creating an item gift message. This is used to offer some items to another player. The player can read the offer and answer it with this message.

MessageItemGiftConfirm

This is the confirmation for the message gift item. It is used when the proposition is answered by a player.

MessageItemExchange

MessageItemExchange is used to offer a trade from a player to another one. The second player can answer the exchange by proposing another item to exchange. This is called counter exchange. This message can be used for both, first proposition and counter proposition. The method “isFirstStep” define in which part of the trade the players are.

MessageItemExchangeConfirm

This is the confirmation for a trade. It is similar to the message item exchange for the stepping process. There is the possibility to know if the answer is applicable for the offer or the counter offer.

MessageQuestGift

The principle of this class is the same as MessageItemGift but for a quest.

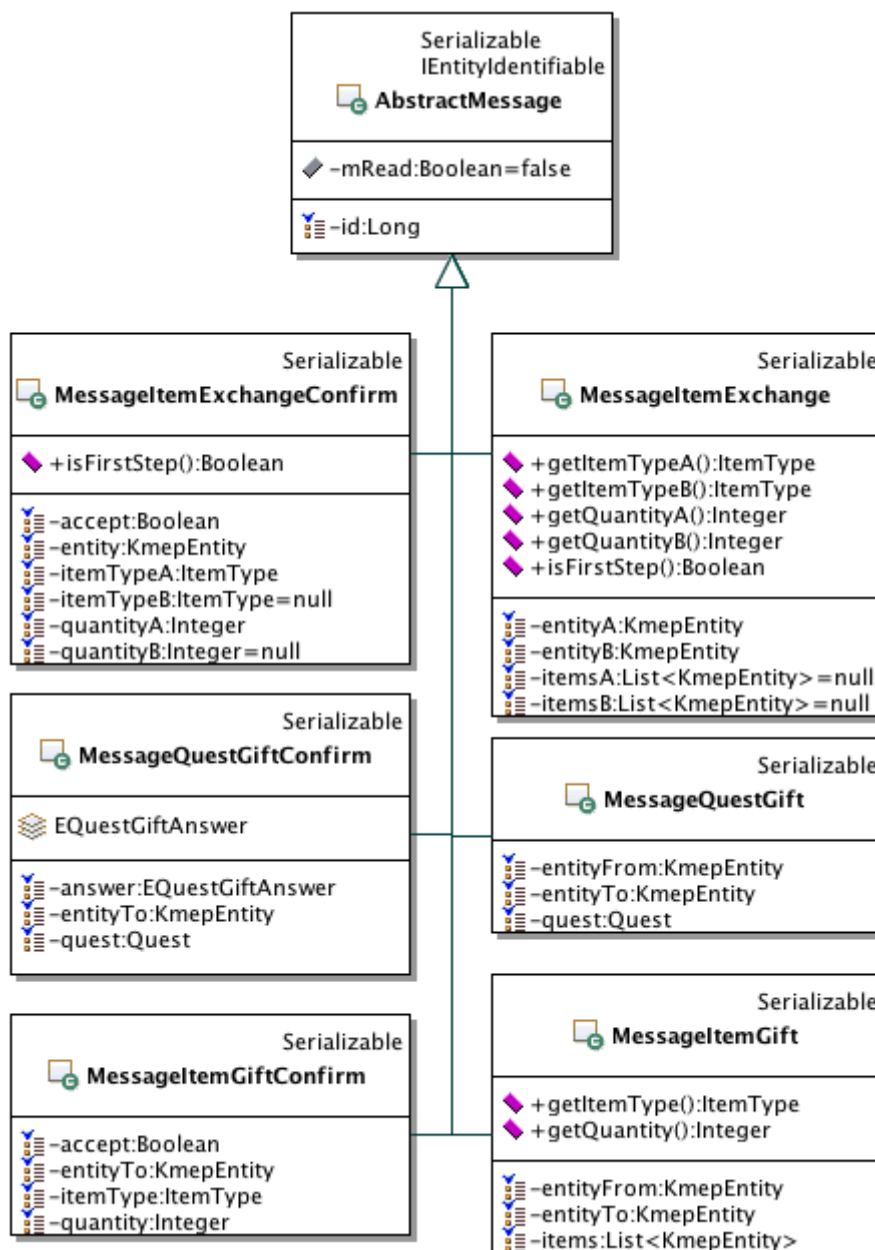


Figure 68 : UML – Model Message classes

MessageQuestGiftConfirm

MessageQuestGiftConfirm is also similar to the MessageItemGiftConfirm. There is a significant difference. The proposed quest can be already in the quest diary of the player that receives the offer. In this case, it is necessary to know if the quest is accepted, refused or already present. This is the reason to have an internal enumeration to manage the answer.

5.5.8. Model Quest

The Figure 69 shows the quest model architecture and the first quest too. The First quest is used to give a quest when the player begins to play. The quest class contains the list of materials, conditions and rewards that define the quest.

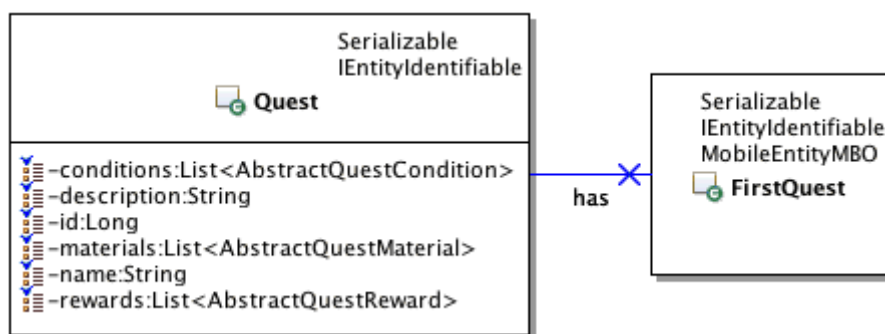


Figure 69 : UML – Model Quest classes

5.5.9. Model Quest Materials

The Figure 70 shows the quest materials design. The method “process()” allows to apply the material to the KmepeEntity concerned. This process has to be done when the quest is added to the player.

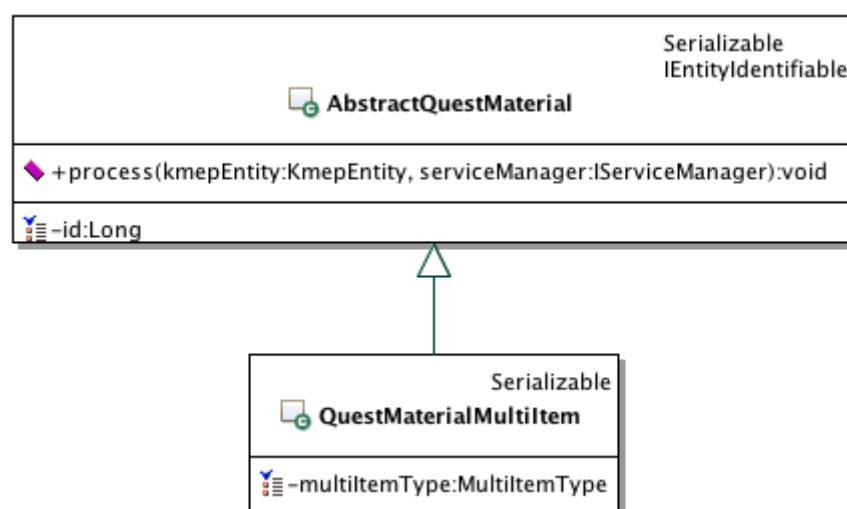


Figure 70 : UML – Model Quest Materials classes

QuestMaterialMultitem

This class allows configuring the material to give to the player in terms of items. The use of MultitemType is necessary for the quest puzzle for example. With the correct configuration, it is possible to give puzzle element with a uniform distribution.

5.5.10. Model Quest Conditions

The Figure 71 shows the model quest conditions architecture. The conditions allow knowing if a quest is finished or not. The “createState()” method permit to create a condition state and the “process()” method can be used to apply the condition when it is reached, generally when the quest is validated.

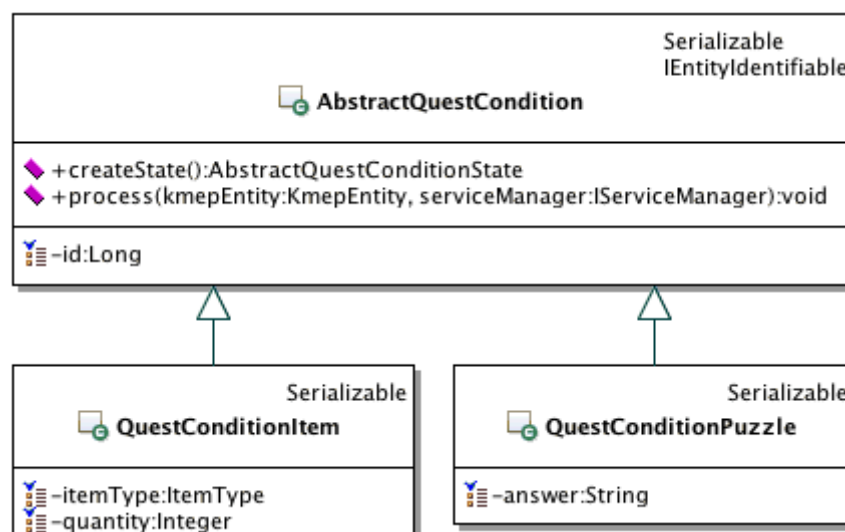


Figure 71 : UML – Model Quest Conditions classes

QuestConditionItem

The quest condition item defines the condition based on the possession of item. It is define by the type and the quantity to own.

QuestConditionPuzzle

The puzzle quest condition defines the text answer that the players have to guess on the base of pictures given.

5.5.11. Model Quest Rewards

The Figure 72 shows the model quest rewards architecture. The rewards are given or applied when the quest is validated. To do that, the “process” method is used.

QuestRewardItem

The quest reward item allows giving item to the player when he successes to do his quest. It is possible to give a certain amount of an item type.

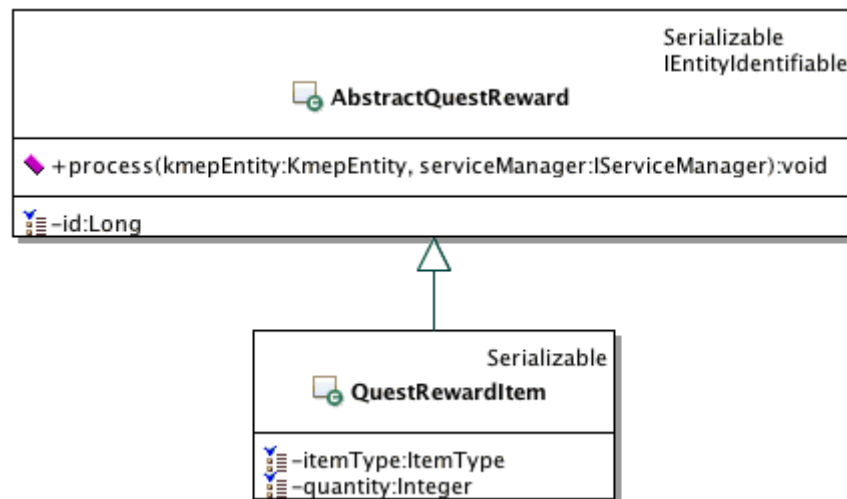


Figure 72 : UML – Model Quest Rewards classes

5.5.12. Model Quest State

To know the state of players' quest, there is a need to build architecture to save the quest state. The Figure 73 shows this architecture.

QuestState

The quest state allows maintaining the global state for a quest. It uses the **EQuestState** enumeration to save the state. The method "evaluate" is used to know the current state of the quest and the "reset" method to clean the state to have a quest state in the same state after its creation. The quest state keeps a link to retrieve the quest concerned by the state.

EQuestState

This enumeration contains the five states for a quest. For more details, see 4.5.4.1 Quest State paragraph.

AbstractQuestConditionState

This abstract class represents the super class for the quest condition states. It contains a reference to the corresponding condition. The two methods "reset" and "evaluate" have the same reason than the methods of quest state. They are abstract and must to be defined in the sub classes. They are used to know the state of condition or reset it.

EQuestConditionState

This enumeration allows knowing if a quest condition state is successes or failed. There is also the pending state.

QuestConditionStateItem

This condition state has no special data because the items can change all the time during the game. A player can give, receive or left items when he wants. To evaluate the condition, this is necessary to count the items in the inventory.

QuestConditionStatePuzzle

The puzzle condition allows remembering the answer given by the player. This is necessary because the state can be evaluated at any time during the game. In this situation, if the answer is not memorized, the condition can be considered as failed rather than successes.

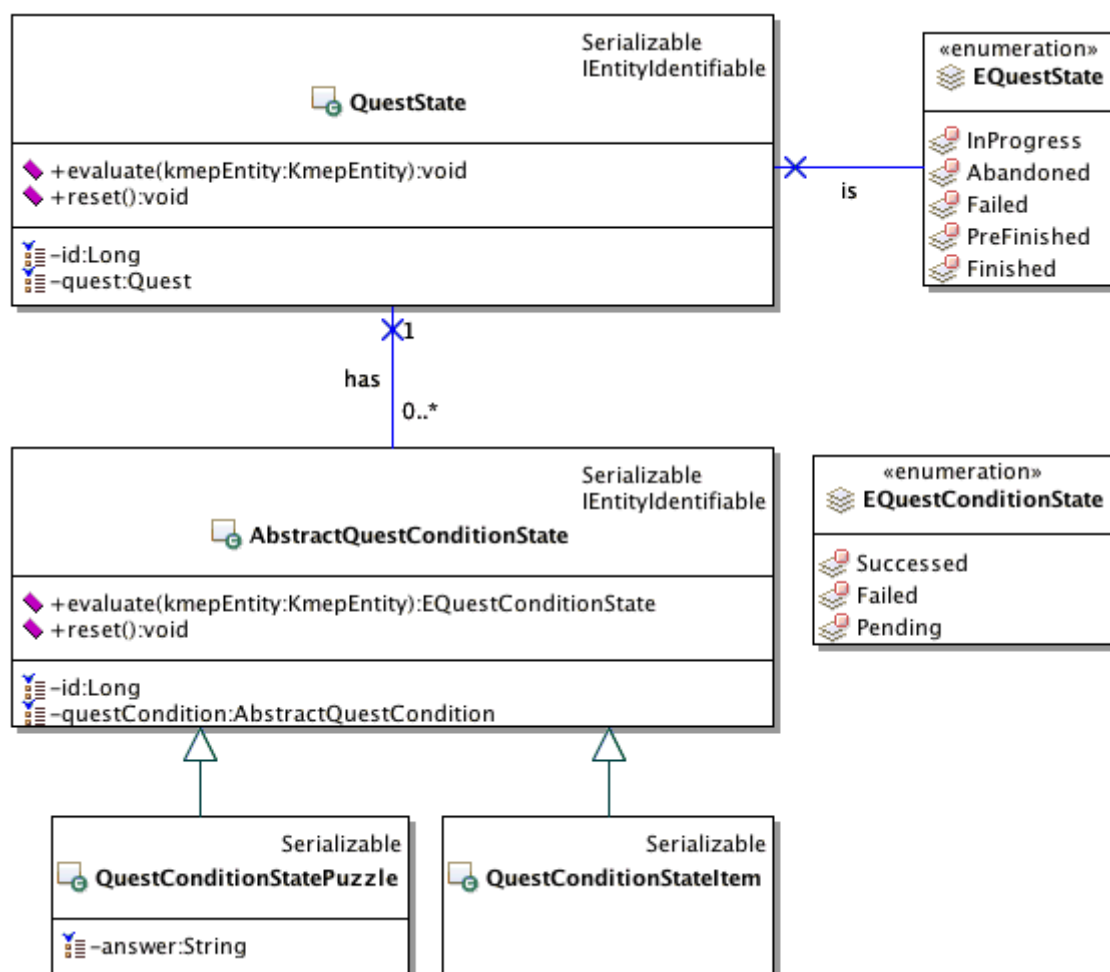


Figure 73 : UML – Model Dialog State classes

5.5.13. Model Statistic

The Figure 74 shows the architecture used to store and manage the statistics used in the game for the players. The statistics can be categorized by the enumeration EStatisticCategory and have a name through the enumeration EStatisticName.

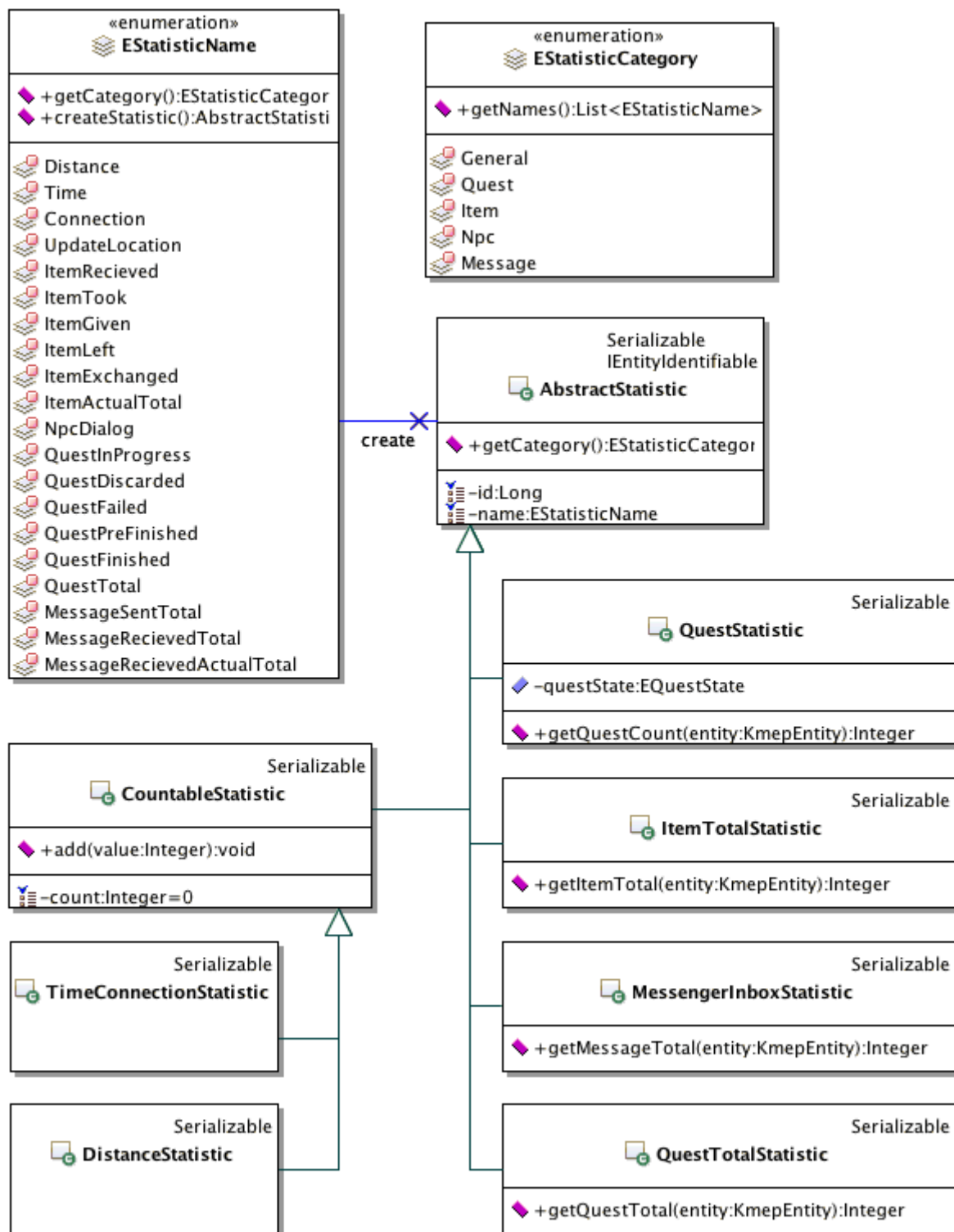


Figure 74 : UML – Model Statistics classes

AbstractStatistic

The **AbstractStatistic** is the super class for all the statistics. It offers a shortcut to get the category of the statistic.

CountableStatistic

CountableStatistic represent the entire statistic that is possible to count and memorize the amount. The method “add()” allows to add a certain amount to the total actually memorized.

EStatisticCategory

The statistics are categorized and this enumeration is here for that. There is the possibility to retrieve all the statistic names from the category with the method “getNames()”.

EStatisticName

This enumeration is used as a factory and a manner to differentiate the statistics. The method “getCategory()” allows retrieving the category of the statistic and “createStatistic()” is used to create the statistic to save in persistence layer.

QuestStatistic

This statistic allows retrieving the number of quest corresponding to the configured quest state. The method “getQuestCount()” retrieve the number of quest at the time of the method call.

ItemTotalStatistic

ItemTotalStatistic class represents the statistic to know how many items are actually owned by the player. The method “getItemTotal()” calculate the item total at each call.

MessengerInboxStatistic

This statistic has the same function as the ItemTotalStatistic. It allows knowing the total amount of inbox messages. The method “getMessageTotal()” calculate the amount at each call.

QuestTotalStatistic

QuestTotalStatistic allows knowing the total of quest in the quest diary without any consideration of their states. The method “getQuestTotal()” evaluate the number of quests at each call.

TimeConnectionStatistic & DistanceStatistic

These two statistics are based on the CountableStatistic but they use a different formatter. The first one uses a formatter to present the data as time notation and the second one as distance notation. For these reasons, they have their own “getFormatter()” method.

5.5.14. Model Utils

For debugging reasons, it is important sometimes to print all the content of an object with its relative object contained in it. The problem is if there is a circular reference, the StackOverflow exception is thrown. To avoid this problem, the class Stringifier shown on the Figure 75 is used.

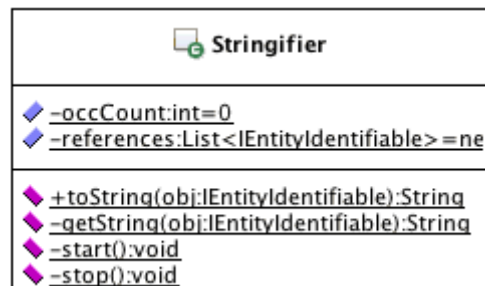


Figure 75 : UML – Model Utils classes

The Stringifier class allows maintaining a directory of already printed objects that implement IEntityIdentifiable. There is only the “toString” method available from this class. This method manage the directory with the “start()”, “stop()” and “getString()” methods. The “occCount” attribute is used to know if the tree is finished or not. The “start()” method increment the counter and “stop()” method decrement it.

Each IEntityIdentifiable has to implement the “getString()” method. The “getString()” from this class call the IEntityIdentifiable “getString()” method. The “toString()” method of these classes call the Stringifier “toString” method.

It is a little bit tricky but it very useful to get a complete trace of the values contains in a connected graph and avoids the StackOverflow exception.

5.6. KMEP-EJB Services

The EJB services part contains the services to manage the game. This application layer is used from external application that can implement the game mechanism in a User Interface for example. There are two categories of services, the “normal” services and the administrative services.

5.6.1. Services

The “normal” services are used to manage the game rules. There are several services for that categorized by type of functionalities. The Figure 76 shows the different services.

IKmepEntityManager

This service allows managing the entity of the MEP. It allows creating item and players during the game. There are also some lookup methods to find the entities or their behaviors.



Figure 76 : UML – Services interfaces

IDialogManager

The dialog manager offers the methods to manage the dialog flow between two entities. The method “getStartDialog()” allows stating a dialog and the other method to continue it. The “find()” method is to find a dialog by its id. The rest of methods are used to do some special actions depending of the dialog types.

IItemManager

The item manager provides all the necessary methods to assume the functionality like give, leave, exchange or take an item. There are also the methods to consult the players’ inventory.

IMessageManager

This service contains the method to manage the inbox of an entity. There are the methods to get the messages, add or remove. There are also the method find specific message or to know if there is unread message in inbox.

IQuestManager

The quest service is used to manage the quests. There are the methods to find some relative quest element like condition... or get some quest by filtering method. The other methods allow managing the quest diary of players.

IPlayerManager

This service provides the methods to manage the players. There are all the method to create, login and logout the players and the management methods for the session management.

IStatisticManager

The statistic manager provides the method to find a statistic and the method to update them. There are some special update methods and standard methods for the countable statistics.

ITrackingManager

This service is designed on the base of the *InTrackManager* from the administrative services. This is a consolidation of the *inTrack* service. The *inTrack* service provides error codes from *inTrack* and the tracking service the KMEP return code. There is the translation between results from *inTrack* and KMEP to do.

Some methods are also consolidated from the methods offers by *inTrack* service. For example the “*rebuildInTrackRelation()*” method is used to redefine the bind and tracking id between KMEP and *inTrack*.

5.6.2. Administrative Services

The administrative services are used to manage the MEP. There are several services for that categorized by type of functionalities. The Figure 77 shows the different services.

IConfigManager

This service allows managing the configuration stored in the database. They are some methods to get simple types values, update the values and check if the value exists. All these methods are based on a key lookup system.

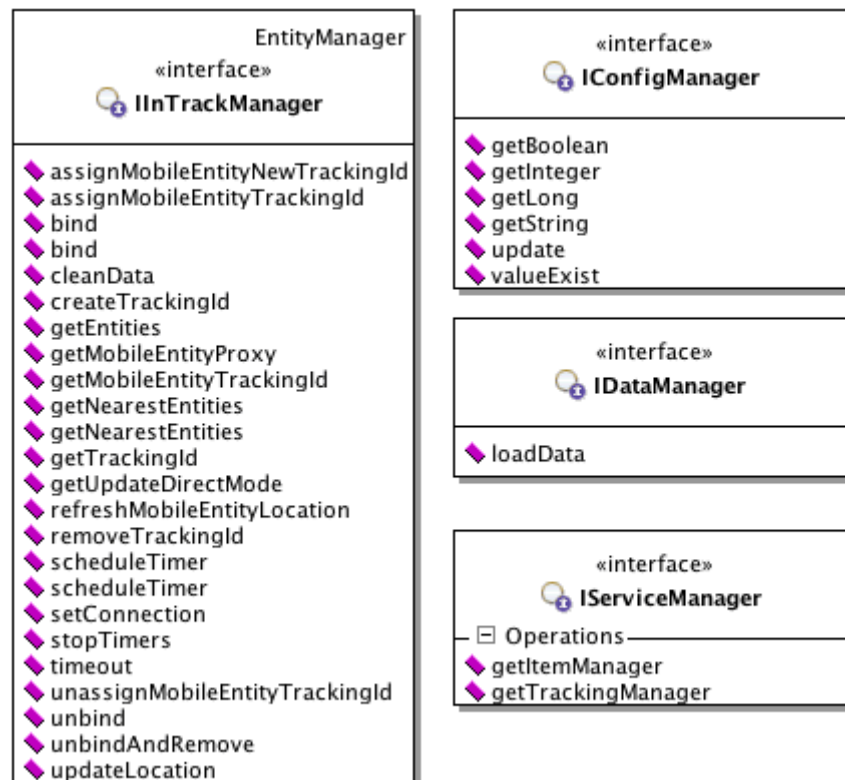


Figure 77 : UML – Administrative Services interfaces

IDataManager

The data service is used to load the data during the application deploy process. The implementation of the interface provide the method “loadData()” code shown on Code 8. The commentaries provided in the code are sufficient to understand the different step executed to load and to persist the XML configuration files.

The first part of the code is the definition to access some other services and configure the persistence access. The methods are used to load and to persist the data.

```

@Stateless
public class DataManager implements IDataManager {
    // The entity manager to access data
    @PersistenceContext(unitName = "kmepPU")
    private EntityManager em;

    // Kmep entity manager
    @EJB private IKmepEntityManager kmepEntityManager;

    // Tracking manager
    @EJB private ITrackingManager trackingManager;

    // Config manager
    @EJB private IConfigManager cfgManager;

    // Service manager
    @EJB private IServiceManager serviceManager;
  
```

```
/**
 * Allow to load the game data and persist them from the base
 * XML files that contains the loading definition
 * @param loader Loader to load base data information
 * @throws KMEPEException See error doc. for more details
 */
public void loadData(String loader) throws KMEPEException {
    // Load the file containing the other data files to load
    LoaderXML loaderXml =
        new LoaderXML(
            loader,
            ServerConfig.getInstance().getString("xsd.loaders"),
            LoaderCollectionXML.class.getName()
        );

    // Load the content
    LoaderCollectionXML lc =
        (LoaderCollectionXML)XMLLoader.load(loaderXml);

    // List of persistables
    List<IPersistable> persistables =
        new ArrayList<IPersistable>();

    // Load the configuration
    ConfigValueCollectionXML config = (ConfigValueCollectionXML)
        XMLElement.load(lc.getConfigLoader());

    // Check if the data must to be loaded
    if (!cfgManager.valueExist("initdata") ||
        cfgManager.getBoolean("initdata")) {

        // Clean MEP Data in inTrack
        trackingManager.cleanProjectData();

        // Create the persister
        Persister persister = new Persister(em, serviceManager);

        // Load the data
        for (LoaderXML l : lc.getLoaders())
            persistables.add((IPersistable)XMLLoader.load(l));

        // Add the config for the persistables
        persistables.add(config);

        // Persist process
        persist(persistables, persister);

        // Update the config state and first dialog npc to use
        cfgManager.update("initdata", false);
        cfgManager.update("npcdialog.firstdialogid",
            persister.find(AbstractDialog.class, EDataTypeXML.Dialog,
                cfgManager.getLong("npcdialog.firstdialogid")).getId());
    }
}
```

```
/**
 * Persists the data
 * @param persistables List of persistables to save
 * @param persister Persister to manage the persistence process
 * @throws KMEPEException See error doc. for more details
 */
private void persist(List<IPersistable> persistables,
    Persister persister) throws KMEPEException {

    // Persist the data
    for (IPersistable p : persistables)
        p.persistData(persister);

    // Flush the entity manager to guarantee all loaded object
    // are saved in the persistence layer before the lookup
    // process.
    em.flush();

    // Persist the references
    for (IPersistable p : persistables)
        p.persistReferences(persister);
}
}
```

Code 8 : Data Manager implementation

IServiceManager

This service can be used inside the beans to lookup some other services. During the project, it was not possible to use directly a lookup from the entities. In this situation, the best solution is to transfer this service to the methods that need a specific service. These methods can retrieve the service from the lookup service.

InTrackManager

This service is provided by the inTrack platform but some modifications can be done to add the missing methods like “updateLocation”. The result of the modification is that there is no more easy maintenance of this service. If there is an update of this service, the update must be done manually and not just with a cut & past of the files.

The methods in this service are not directly used by other services except the ITrackingManager. This is service is the only one to access to the inTrack service method. This implementation is a security for future evolution of the inTrack platform in regard of the MEP.

5.7. KMEP Web Application Global Architecture

The Figure 78 shows the MVC architecture base used for the iMode web application in the KMEP application. The Model View Controller architecture allows a great separation between tasks to accomplish. For remember the Model store and work on the data. The controller organizes the tasks between View and Model and the View is the rendering of the application.

In the KMEP application, the MVC is a little modified for the Controller part. Normally, there is more than one controller. In this application, there is only one controller to organize all the processing between model and tasks. The Model part is called Action in this diagram.

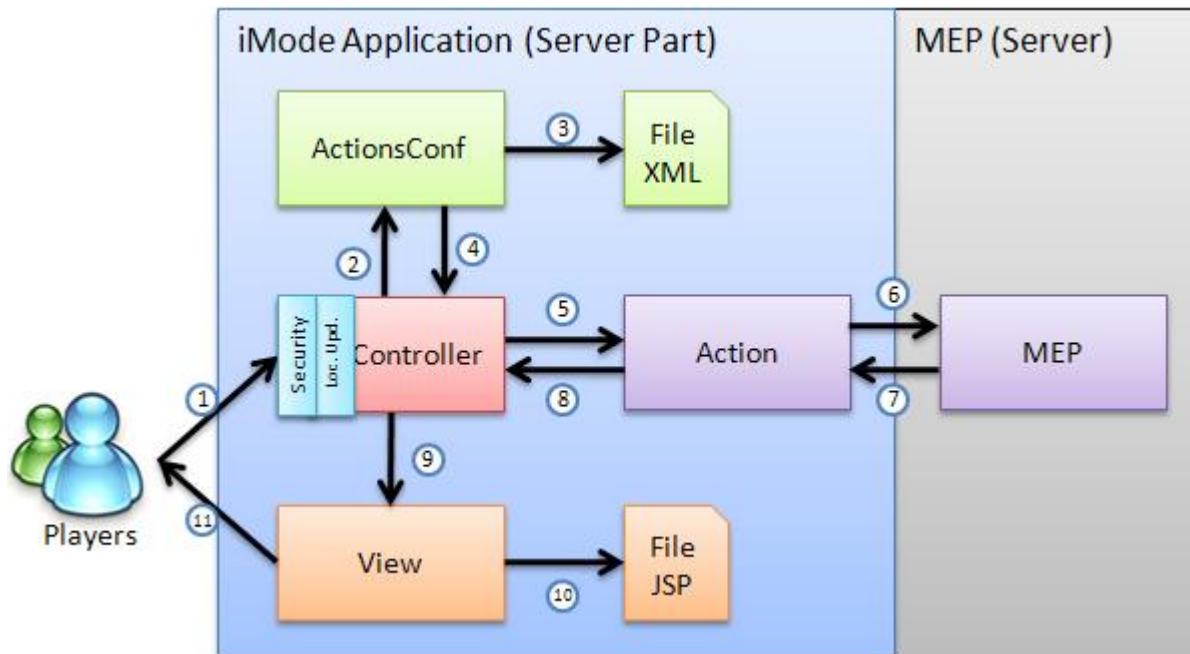


Figure 78 : KMEP Web Application architecture

The Figure 78 also gives the order of operation processed when a player call a page of the application. The next lines describe this process:

1. Player request
2. Security and Update Location processing are done if necessary (they are included in the Controller). The controller does a lookup on the "act" query argument to find the called action.
3. The lookup is done with a specific part of the application that search inside an XML file. If there is no "act" attribute, the default action is chosen.
4. The ActionsConf does the lookup in the XML file and return the result to the controller.
5. The Controller forwards the processing to the called action.
6. The Action does the necessary processing with the calls to the EJB part of the application (Services and Game Engine are called from the actions. This is the communication with MEP).
7. The result of the EJB part is returned to the Action.
8. The Action configures the corresponding view and return an ActionState to the Controller.
9. Finally, the Controller can forward the processing to the correct View configured by the Action
10. The View does the processing of the rendering with the call to JSP. JSP can calls sub views if necessary.
11. The result is finally returned to the player.

5.8. KMEP-WAR Configuration & Infrastructure

This section will present the class diagrams for the common WUI application. There is a lot of code that could be reused for another type of WUI. In this project, the WUI is only for the iMode technology.

Due to the time for this project, the reusable part is with the iMode part that is only usable by phones that support iMode technology. These different points will be discussed later.

5.8.1. Configuration

The Figure 79 shows the different classes used to handle the configuration of the application for the web part.

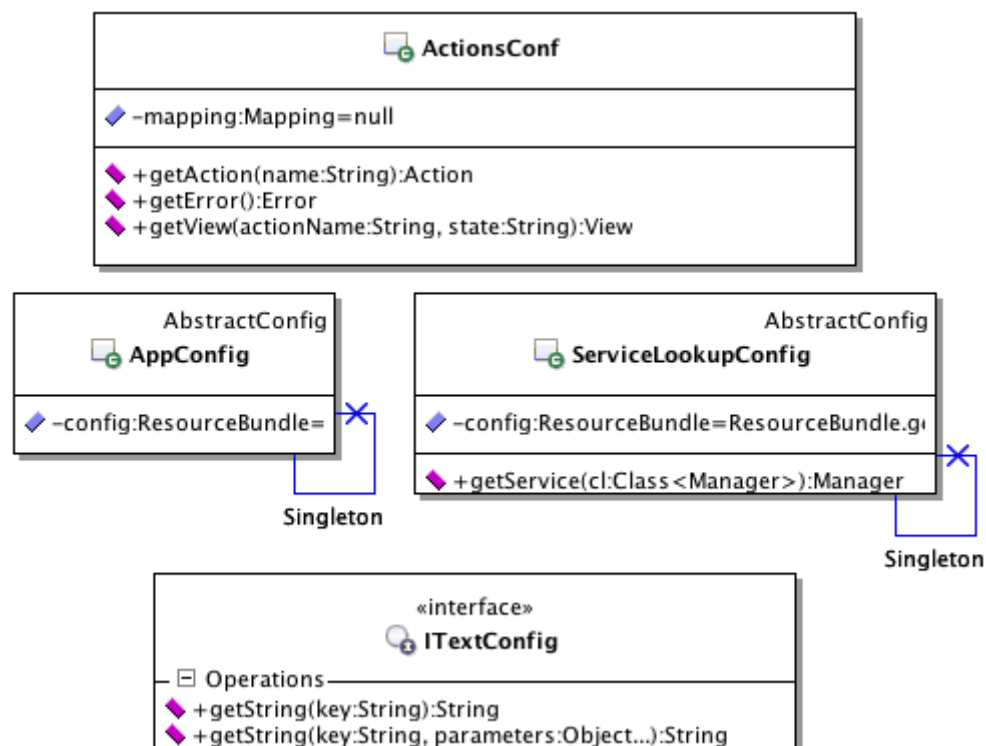


Figure 79 : UML – Configuration classes

ActionsConf

This class is design to be used in the controller to allow retrieving the views and actions corresponding to key words.

AppConfig

The application configuration class contains the mapping between the action names used in the code and the action names used in the XML file. This additional level allows changing a name in the XML configuration file. This change is done without any change in code except the configuration file to update.

ServiceLookupConfig

The service lookup configuration class allows doing lookups for the services that the actions need. The method “getService” with the “genericity” allows retrieving a service through its interface directly cast to the correct type.

ITextConfig

Sometimes, the actions have to prepare specific message to the user that the views cannot know before it occurs. For this reason, the interface ITextConfig provides the methods to implement to get the text message looking for.

5.8.2. Configuration mappings

This is the core of the configuration part of the web application. The Figure 80 shows the relations between classes concerned in the configuration construction. Due to the usage of JAXB, there is some relation only for loading and some classes used as converters.

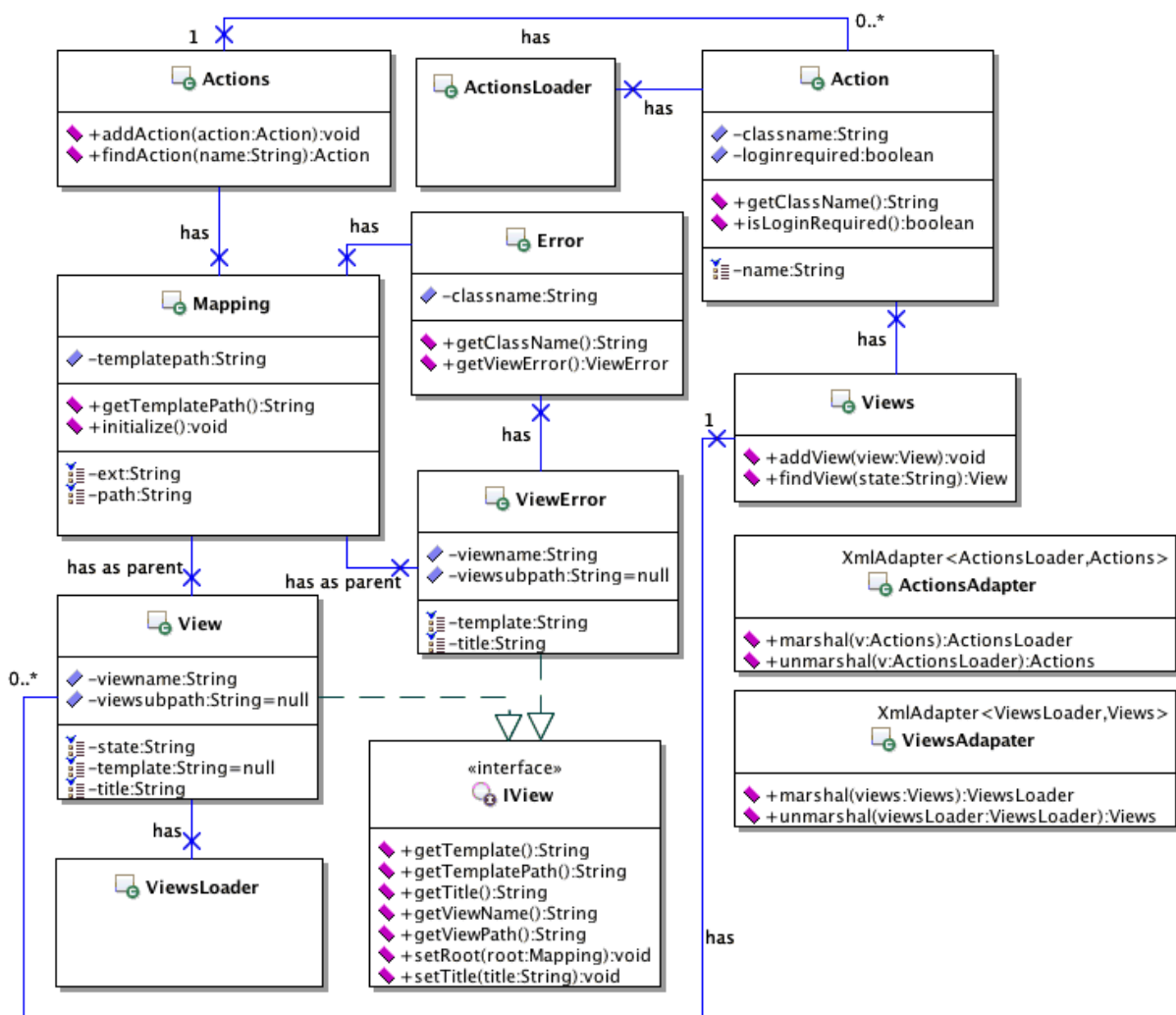


Figure 80 : UML – Configuration Mappings classes

ActionsAdapter & ViewsAdapter

These two classes are especially present to convert an ActionsLoader (ViewsLoader) to Actions (Views) class. Due to the document XML structure, this is not possible to load directly the collection of information. Some processing is necessary after the loading task.

ActionsLoader & ViewsLoader

These two classes are used by the adapters presented just before. Their utility is only to load the data from the XML file, after that, they are not useful.

Mapping

The Mapping class corresponds to the @XmlElement for the mappings between actions and views. This mapping allows retrieving a view from an action with a specific state. The class permits also to retrieve an action.

Actions

This is the collection of the application actions. An action can be found by its name. The name of action must be unique but there is actually no check about this uniqueness. If there is two actions with the same name, the last inserted overwrites the previous one.

Action

The Action class contains the definition to find the action to process (the class name in the complete form) and the collection of views. The Action class includes also a Boolean to determine if the action requires a valid session or not. This is especially used during the check access process.

Views

The collection of views contains the views corresponding to a name state. The name state must be unique but like the Actions, there is no check and the same rule described is applied.

View

The View class contains the data required to find the correct view for the graphical rendering. The View maintains a reference to its parent for an easy navigation.

Error

The Error is a special action that runs when nothing else can be done. This is particularly useful to present the error to the player with a graphical interface like the rest of the game.

ViewError

ViewError class has a similar structure than the View (they have the same interface to implement). It has also the reference to the mappings.

IView

The IView interface allows building some views with the minimum requirements attended. This is necessary when the views are stored in lists...

5.8.3. Factories

The application web uses a class to define the URLs but depending on the web technology used, the parameters can differ a little. For this reason, an interface is used in the actions when this is necessary to create URL directly in them. The Figure 81 shows the interface. The methods to implement allow creating predefined URLs.



Figure 81 : UML – IURLFactory interface

5.8.4. Utilities

To simplify the logging for the request or session parameters, the two classes presented in Figure 82 are offered to retrieve some data in the request.



Figure 82 : UML – Utilities classes

5.8.5. Servlets

Two servlets are especially used during the start of the server. There is the UpdatePeriodically that allows configuring the inTrack synchronization of data with polling method as regularly intervals and the second is designed to load the data in the persistence layer if needed. The Figure 83 shows these two classes.

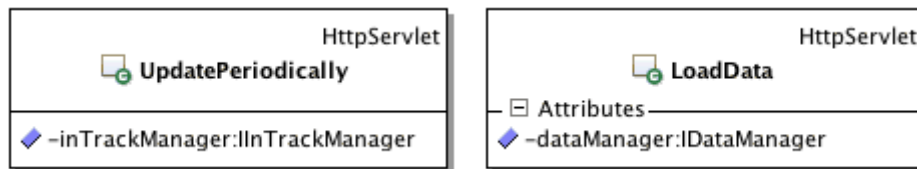


Figure 83 : UML – Servlets classes

5.9. KMEP-WAR Transfer Objects

This part covers the classes that used to transfer information from the actions to the java server pages. The Figure 84 shows the base for all TO.

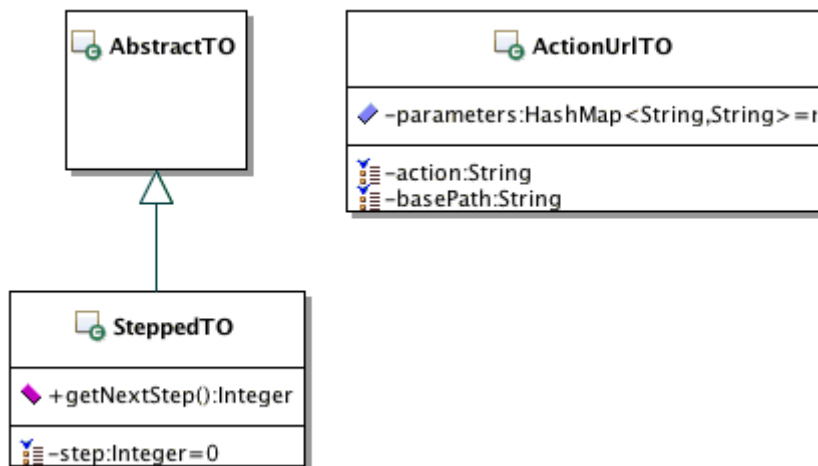


Figure 84 : UML – TO classes

AbstractTO

This is the super class for all the TO.

SteppedTO

In some cases, there is a need to do an action divided in more than one-step. For these actions, the SteppedTO is useful to maintain the actual state of the action.

ActionUrlTO

This TO is particular because it is used to store the actual action and next action to allows coming back from a dialog message in case of error.

5.9.1. TO General

The Figure 85 shows the general TOs used in the application. The MenuTO and PreferencesTO are used for specific views. The DialogTO is more general and is used to present the errors encountered during the application execution.

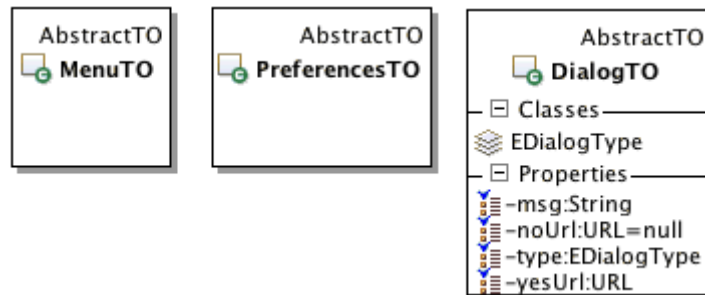


Figure 85 : UML – General TO classes

5.9.2. TO Item

The Figure 86 shows the TOs dedicated to the item actions. Some of them can inherit from the ItemTypeTO super class to have the ItemType. There is no more specificity here.

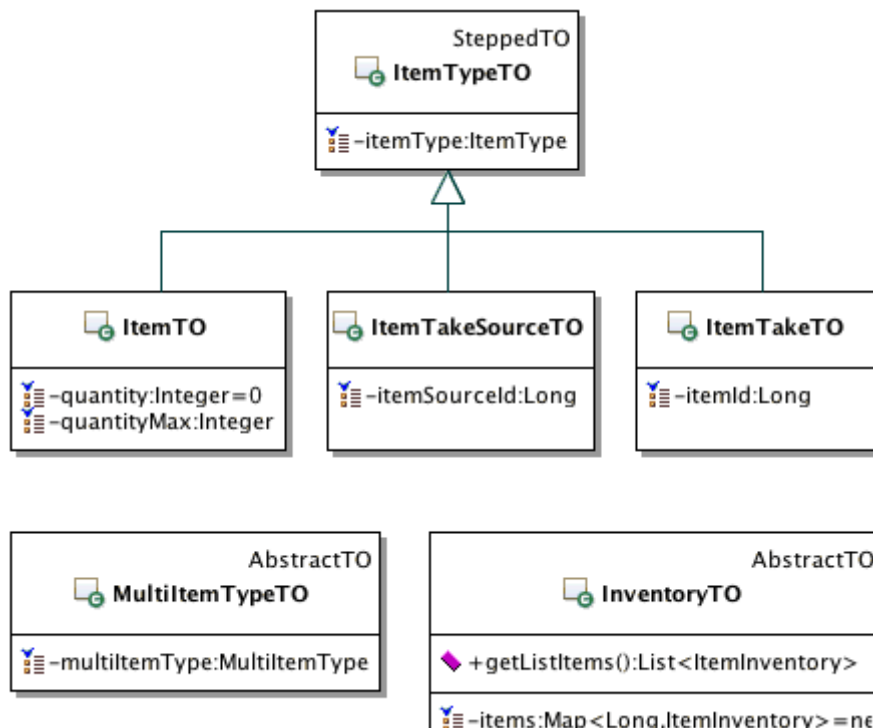


Figure 86 : UML – Item TO classes

5.9.3. TO Map

The Figure 87 shows the MapTO class used to transfer map data.

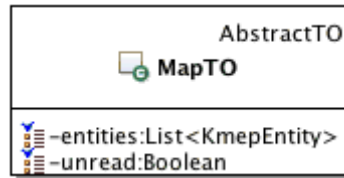


Figure 87 : UML – Map TO class

5.9.4. TO Message

The two classes shown in the Figure 88 are dedicated to transfer the data of message or list of messages.

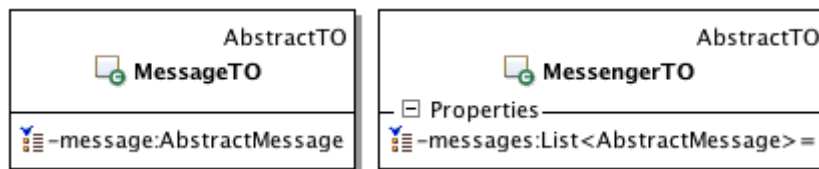


Figure 88 : UML – Message TO classes

5.9.5. TO NPC Dialog

The Figure 89 shows the NPCDialogTO class used to transfer the data during the dialog flow processing.

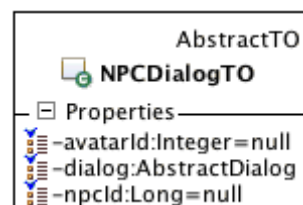


Figure 89 : UML – NPC Dialog TO class

5.9.6. TO Player

The Figure 90 shows the player TO classes used for the different player actions like registering, connecting...

AvatarTO

The AvatarTO allows maintaining the paging during the avatar choice. There is also a distinction between the choice during the registration or during the game.

PlayerChoiceTO

The PlayerChoiceTO is used to manage the player choice during some functionality of the application.

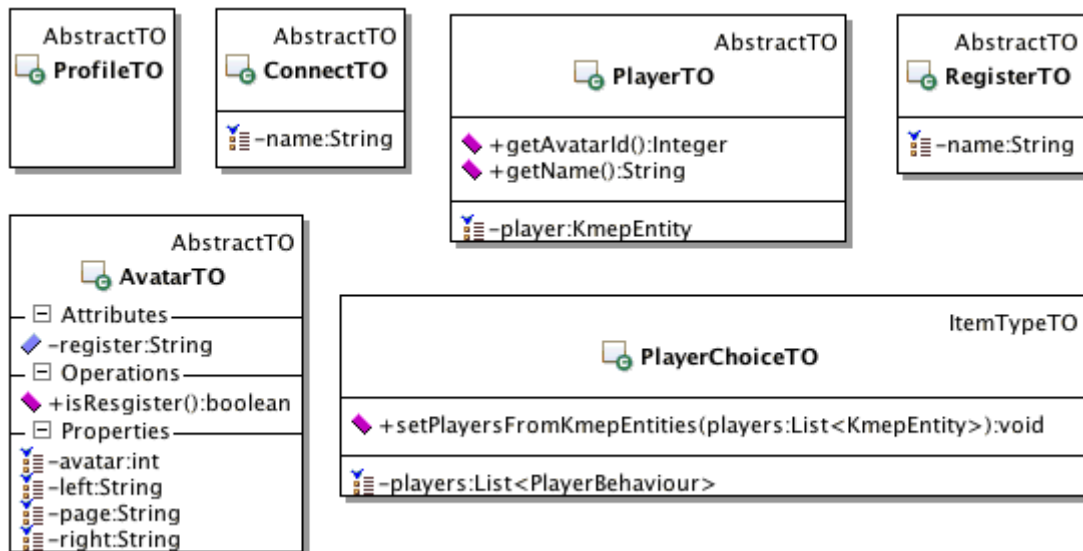


Figure 90 : UML – Player TO classes

5.9.7. TO Quest

The Figure 91 shows the Quest TO classes used to carry the data from the actions to the quest views.

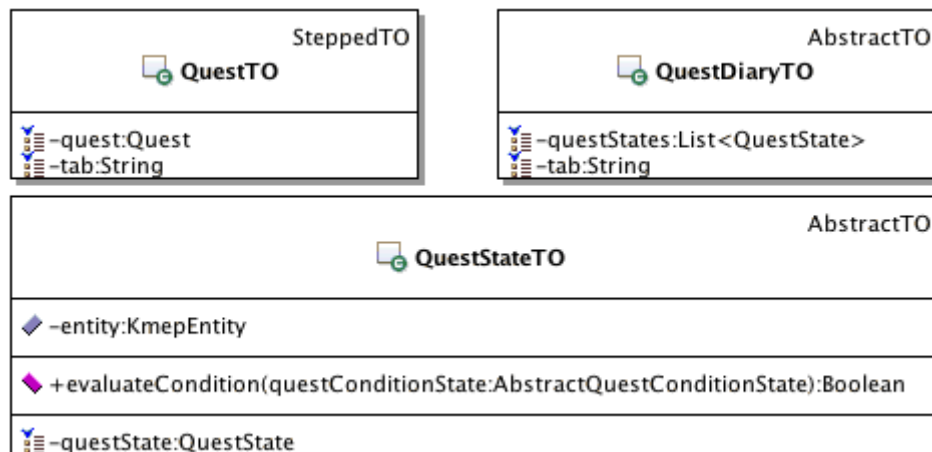


Figure 91 : UML – Quest TO classes

QuestStateTO

The particularity of this TO is that contains a method to evaluate the Quest State to obtain the most recent state of the quest to show.

5.9.8. TO Statistic

The Figure 92 shows the Statistic TO class used to transfer the data for the rendering of statistic. The relative entity is stored in the TO object to be used with some statistic.

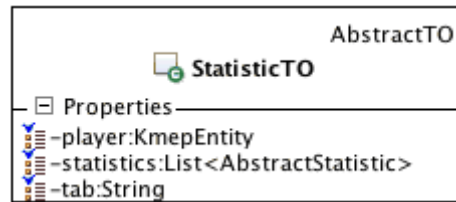


Figure 92 : UML – Statistic TO class

5.10. KMEP-WAR Actions

This part covers the actions used in this part of the project. These actions can be used only in the context of a web application. The Figure 93 shows the base for the Actions architecture construction.

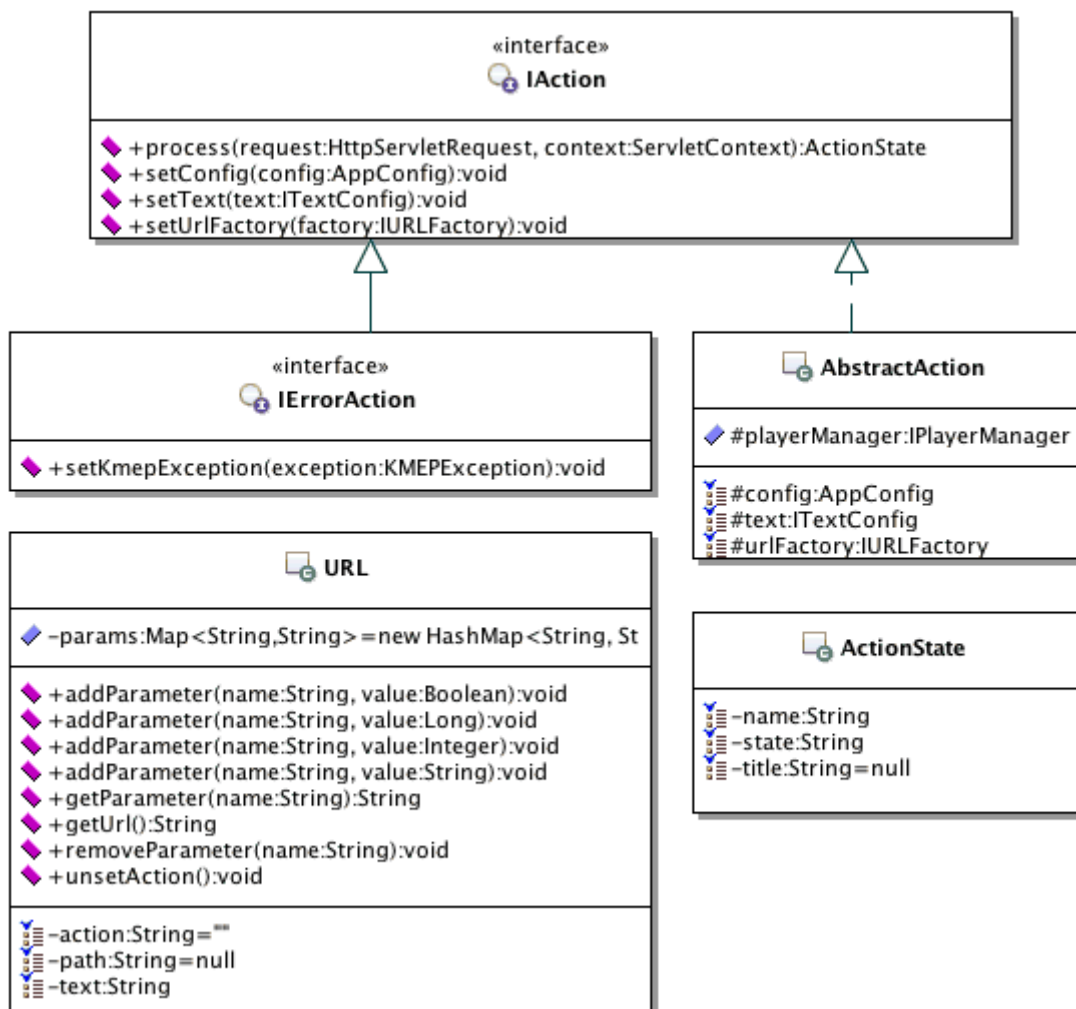


Figure 93 : UML – Actions classes

IAction

This interface allows forcing the classes implement it to have some methods very used to process the actions. The “process()” method is always call by the controller of the application to get the ActionState resulting.

IErrorAction

This interface is a specialization the IAction interface to add the KMEPEXception relative method. This is necessary to use the interface for the error actions.

AbstractAction

The AbstractAction is the super class for the majority of actions used in the application. The interface is implemented here for the basic methods. The “process()” must to be defined in the last stage of inheritance. The Player service is the most used in the application. This is the reason for why there is in this abstract class.

ActionState

ActionState class allows returning the current state for an action after the run of the method “process()” in the actions.

URL

This class allows building general URLs for the application. There are the methods to manage the parameters to add to the query string.

5.10.1. Actions General

The Figure 94 shows the general actions used to process the standard actions in the game. For example, the IndexAction allows processing the “splashscreen” and the ErrorAction processing the errors encountered in the application.



Figure 94 : UML – Actions General classes

5.10.2. Actions Item

The Figure 95 shows the common based item actions. These actions inherit from the `AbstractItemAction` to share the same fields used in the action processed.

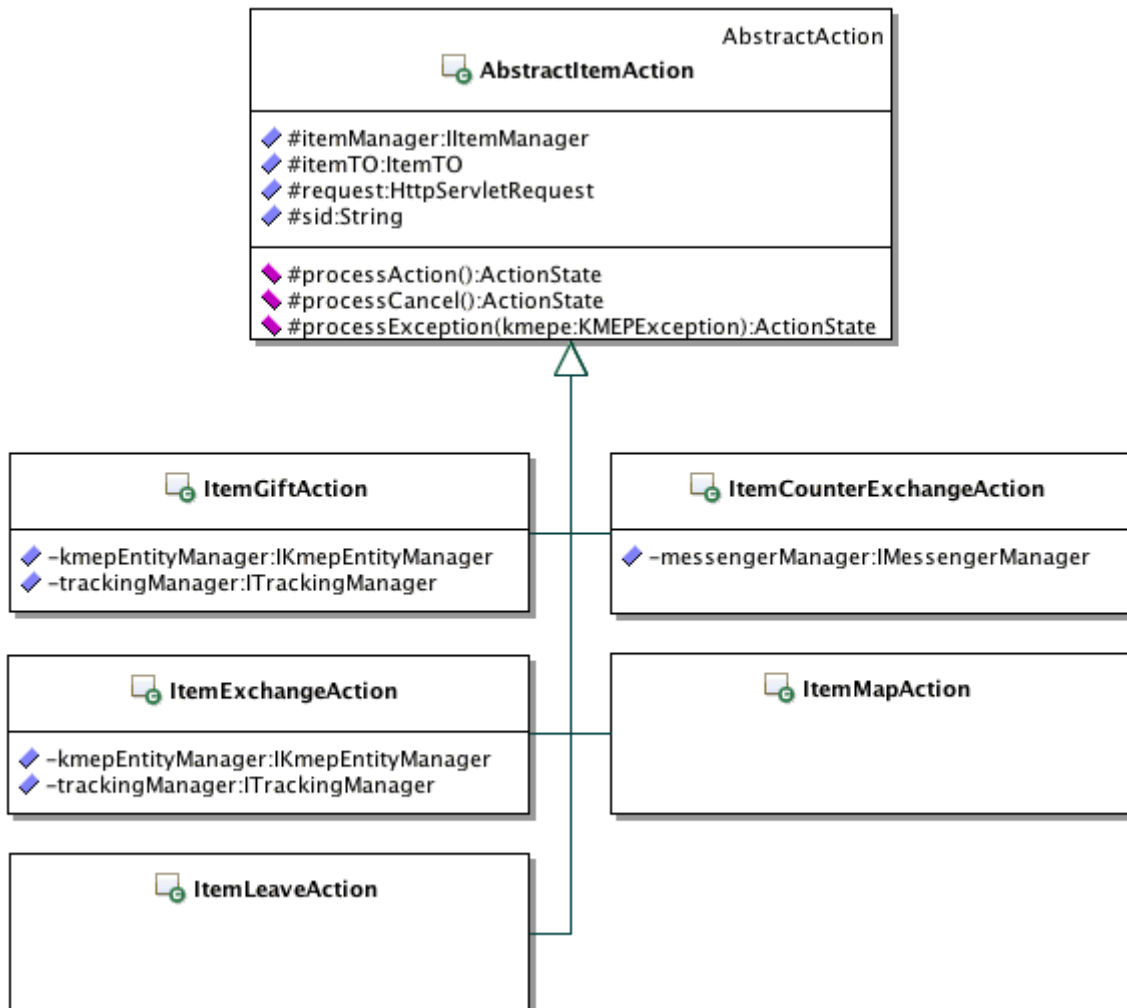


Figure 95 : UML – Actions Item common base classes

AbstractItemAction

This abstract class defines some abstract protected method that the subclasses have to implements. These methods offer the possibility to process the main flow of the action, the exception encountered and the cancel action.

It stores some common useful values to avoid redoing always the same processing. The “process()” method described before is implemented in this class. So all the common processing is done in the mother class. This common processing is shown in Code 9.

```

public ActionState process(HttpServletRequest request,
    ServletContext context) throws KMEPEException {

    // Pre process the request
    preProcess(request);
}
    
```

```

// Process the correct step
try {
    // Check the cancel process
    if (cancel != null && cancel.equalsIgnoreCase("cancel"))
        return processCancel();
    else
        return processAction();
}
catch (KMEPEException kmepe) {
    return processException(kmepe);
}
}
    
```

Code 9 : AbstractItemAction process method

The Figure 96 shows the other item actions that are used to process the different actions relative to the items.

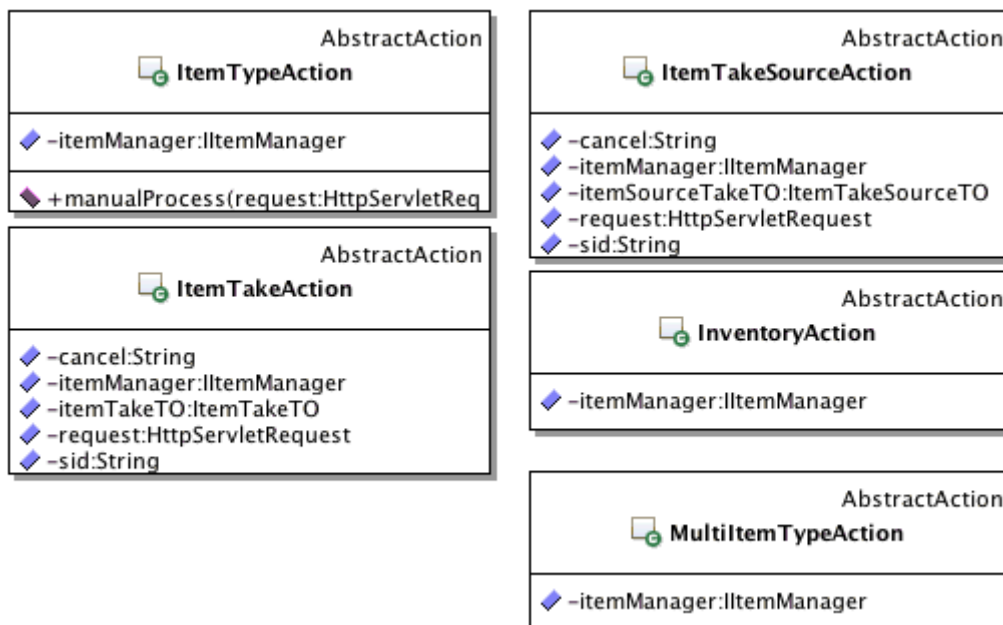


Figure 96 : UML – Other Actions Item classes

5.10.3. Actions Map

The Figure 97 shows the actions relative to the map functionalities. The LocationAction is not a real action. This is more a component used by the controller when it is necessary. The “locate” method can be used to do the localization process of the players.

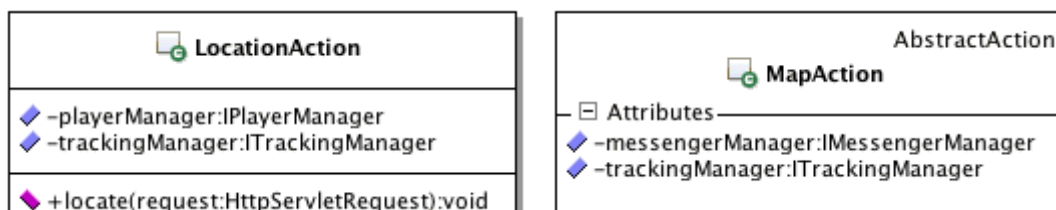


Figure 97 : UML – Actions Map classes

5.10.4. Actions Message

The Figure 98 shows the actions relative to the message functionalities. There are three groups of actions shown on the figure.

The first contains the MessageDetailsAction and MessageDeleteAction. They are used to do some work on the most common messages.

The second one contains the MessageItem*Action. These actions are a little particular in the fact they are used to continue the flow of the ItemActions like giving an item or exchanging an item.

The last one contains only the MessengerAction. This is the action to manage all the messages to show a list of messages for example.



Figure 98 : UML – Actions Message classes

5.10.5. Actions NPC Dialog

The Figure 99 shows the NPC Dialog actions that are used to manage the dialog flow process. All these actions inherit from the AbstractNPCDialogAction.

AbstractNPCDialogAction

This abstract method offers the base for the specific dialog actions. It defines the protected “getDialog()” method to get the correct dialog corresponding to actually dialog flow state.

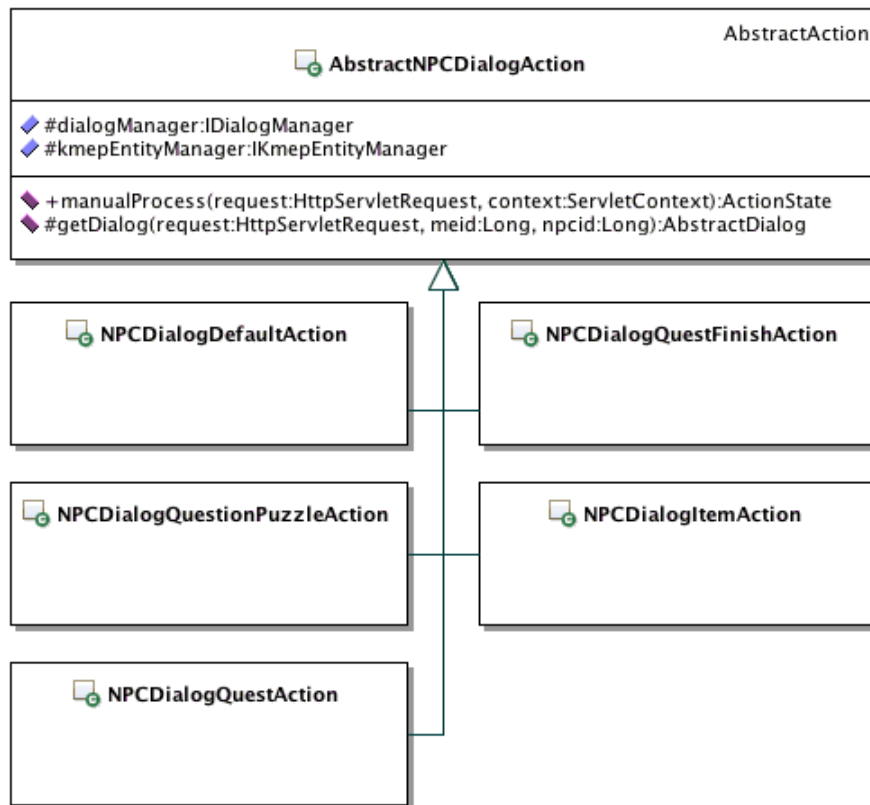


Figure 99 : UML – Actions NPC Dialog classes

5.10.6. Actions Player

The Figure 100 shows the actions classes relative to the player like login/logout, changing the avatar, viewing his profile...

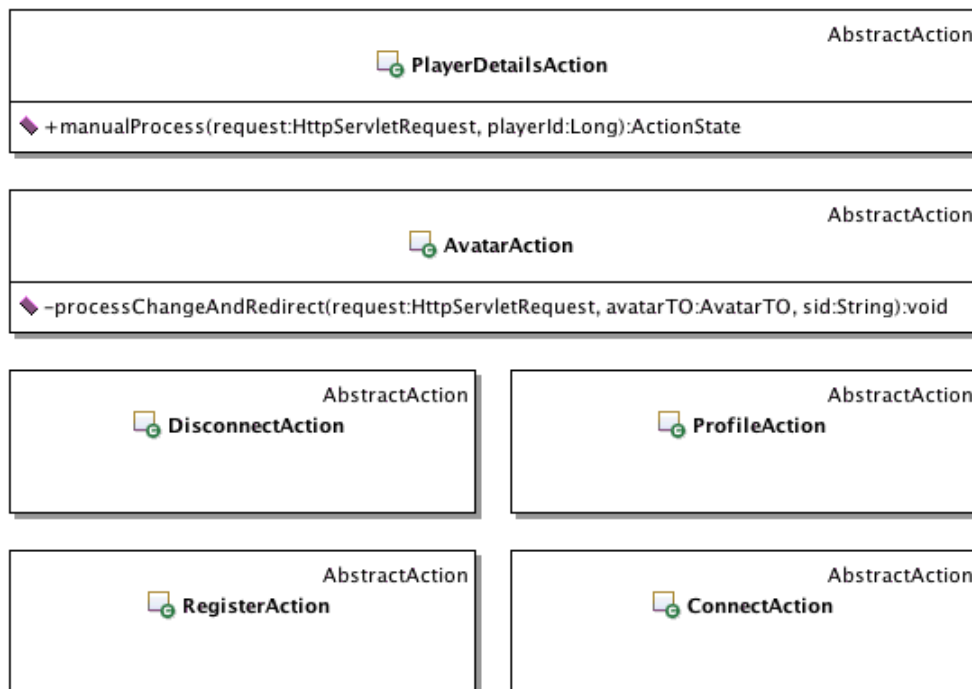


Figure 100 : UML – Actions Player classes

5.10.7. Actions Quest

The Figure 101 shows the common based quest actions. These actions inherit from the AbstractQuestAction to share the same fields used in the action processed.

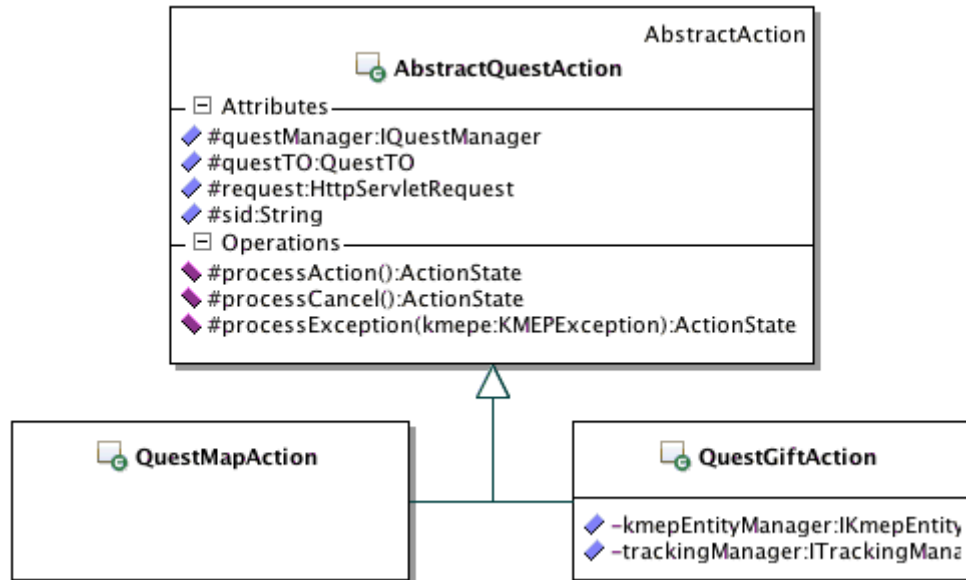


Figure 101 : UML – Actions Quest common base classes

AbstractQuestAction

This abstract class defines some abstract protected method that the subclasses have to implements. Theses methods offer the possibility to process the main flow of the action, the exception encountered and the cancel action.

It stores some common useful values to avoid redoing always the same processing. The “process()” method already described in this part of document, is implemented in this class. So all the common processing is done in the mother class. This common processing is shown in Code 10.

```

public ActionState process(HttpServletRequest request,
    ServletContext context) throws KMEPEException {
    // Pre process the request
    preProcess(request);

    try { // Process the correct step
        // Check the cancel process
        if (cancel != null && cancel.equalsIgnoreCase("cancel"))
            return processCancel();
        else
            return processAction();
    } catch (KMEPEException kmepe) {
        return processException(kmepe);
    }
}

```

Code 10 : AbstractQuestAction process method

This code is the same than the AbstractItemAction. This action comes later in the project and the time is not enough to refactor this part. There are only two classes like but it can be great to create a super class to do the global processing for this kind of actions.

The Figure 102 shows the other quest actions that are used to process the different actions relative to the quests.

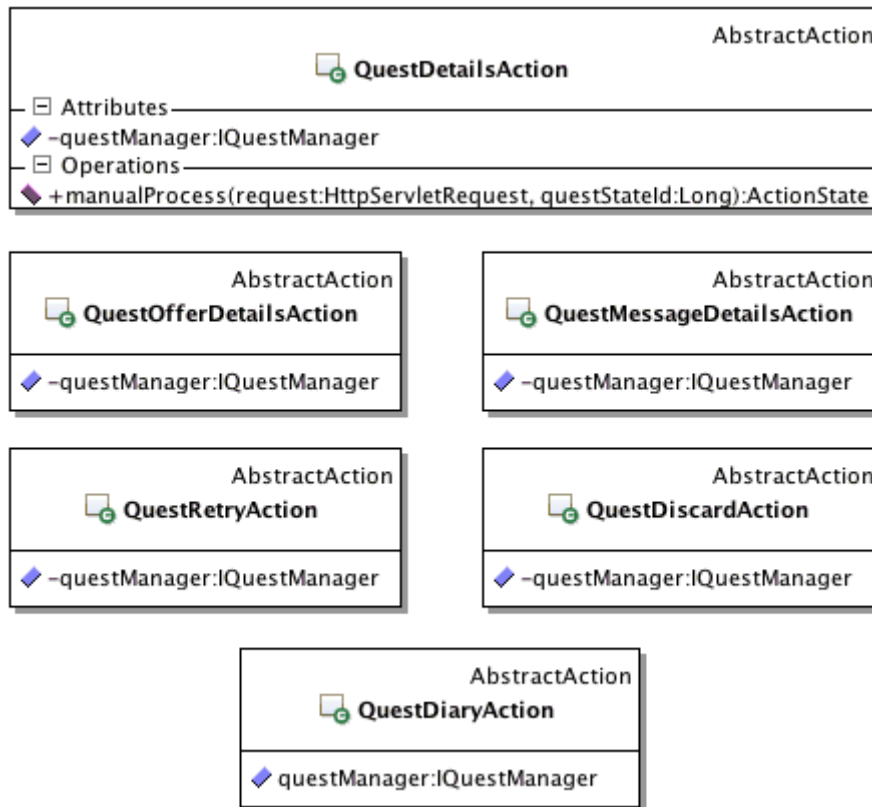


Figure 102 : UML – Other Actions Quest classes

5.10.8. Actions Statistic

The Figure 103 shows the StatisticAction class used to present the statistic to the players.



Figure 103 : UML – Actions Statistic class

5.11. KMEP-WAR iMode application

This part covers the architecture relative to the iMode application. This is the WUI of the application.

5.11.1. iMode Configuration

The Figure 104 shows the implementation for the iMode application of some configuration classes. These classes implement the Singleton design pattern.

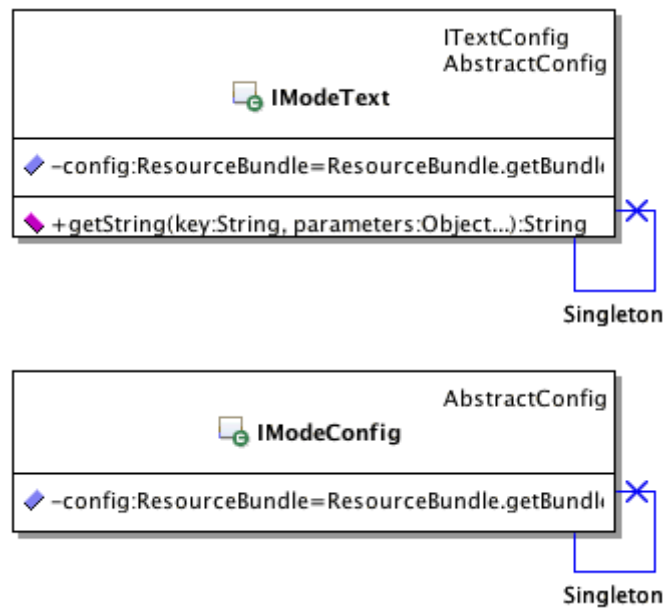


Figure 104 : UML – iMode Configuration classes

5.11.2. iMode Localization

The Figure 105 shows the LocationConverter class used by the iMode application. It implements the ILocationConverter already seen in this document.

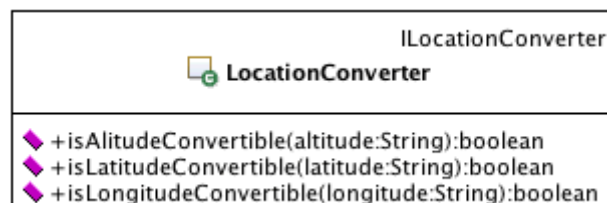


Figure 105 : UML – iMode Location class

5.11.3. iMode Factories

The classes shown in the Figure 106 are the implements of various factories for the iMode application.

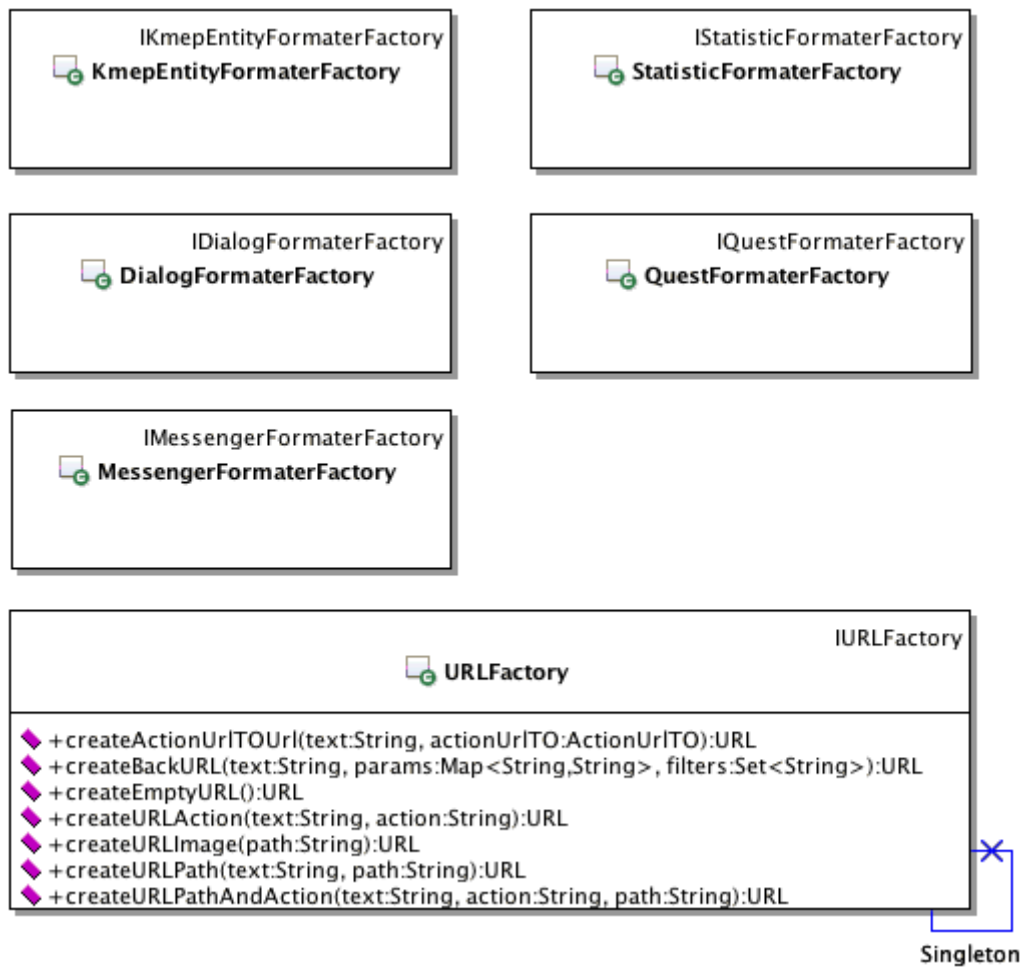


Figure 106 : UML – iMode Factory classes

5.11.4. iMode Servlets

The Figure 107 shows the iMode controller. The controller is used to drive the web application and executing the actions. There are more details in 5.7 KMEP Web Application Global Architecture paragraph.

The Code 11 shows the method “processRequest()” called when the user does a request on the server with the correct URL pattern.

```

...
// Retrieve the action to process
IAction action = getAction(request);

// Check and update location
updateLocation(request);

// Process action and forward to the view for the rendering
forwardToView(request, response,
    action.process(request, getServletContext()));
...

```

Code 11 : processRequest code part

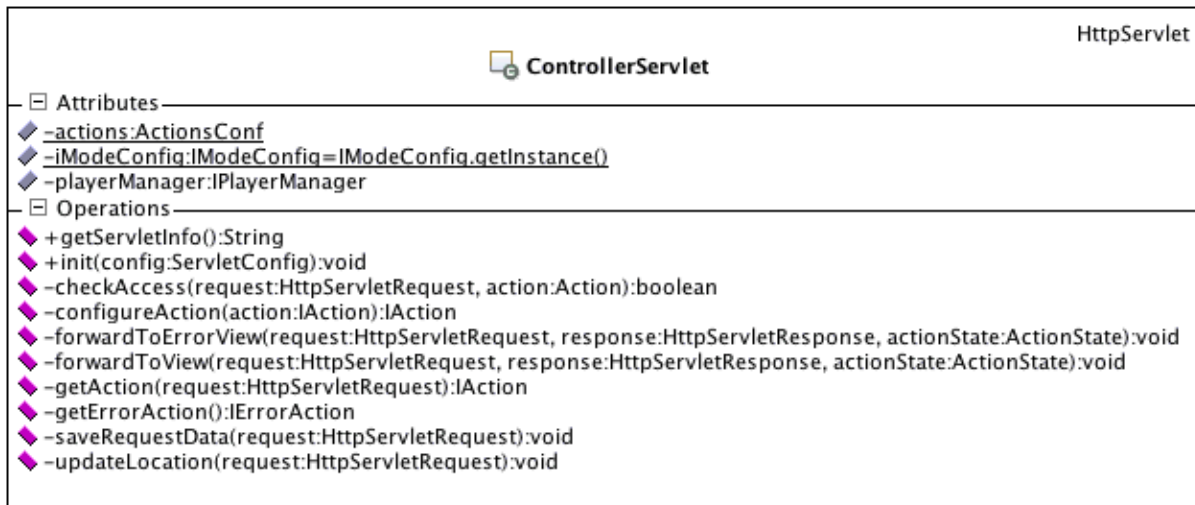


Figure 107 : UML – iMode ControllerServlet class

The Code 12 shows the “getAction()” method. This method allows retrieving the correct action depending on the query string. If there is no action asked, the default one is returned.

```

// Retrieve the class and method to invoke
Action action =
    actions.getAction(RequestParameter.getString(request, "act"));

try {
    Class classRetrieved;

    // Action found
    if (action != null) {
        if (!checkAccess(request, action))
            throw new KMEPEException(...); // Security code exception
        else
            classRetrieved = Class.forName(action.getClassName());
    }

    // Default action
    else
        classRetrieved = IndexAction.class;

    // Invoke the action and configure it
    return configureAction((IAction) classRetrieved.newInstance());
}
    
```

Code 12 : getAction code part

The Code 13 shows the “updateLocation()” method used at each user request. The method check if an update is necessary or not. The method tries to retrieve the session id from the request and the session.

```

/**
 * Update the location
 * @param request Request to get the location data
 * @throws KMEPEException See error doc. for more details
 */
private void updateLocation(HttpServletRequest request)
    throws KMEPEException {
    
```

```
// Check if an update is necessary
if (RequestParameter.getString(request, "geo") != null) {
    // Try to find the sid from the request
    String sid = RequestParameter.getString(request, "sid");

    // If there is no session id in the request,
    // get it from the session
    if (sid == null)
        sid = SessionAttribute.getAttribute(request, "sid");

    // Update the location there is a session
    if (sid != null)
        new LocationAction().locate(request);
}
}
```

Code 13 : updateLocation method code

The Code 14 shows the method “checkAccess”. This method check if the player can access to a certain function or not. The security check is very basic. There is actually only two level of security access. The first not connected and the second connected. The reason is that the check access is more to validate that the player is connected to the game and the player state is coherent than a true security check.

```
/**
 * Check the validity of the session and the access
 * @param request Request to get the Session id
 * @param action The action to verify the rights
 * @return True if the action can be shown, false otherwise
 * @throws KMEPEException See error doc. for more details
 */
private boolean checkAccess(HttpServletRequest request,
    Action action) throws KMEPEException {

    // Check if a connection is required
    if (action.isLoginRequired()) {
        String sid = SessionAttribute.getAttribute(request, "sid");

        // Verifiy if the session is valid and update it
        if (playerManager.isConnected(sid)) {
            playerManager.updateSession(sid);
            return true;
        }

        // Session invalid
        else
            return false;
    }
    else
        return true;
}
```

Code 14 : checkAccess method code

The Code 15 shows the final step of the controller. This is when the action is finished and the result must to be forwarded to the view. This method forwards the result to the correct view.

```
/**
 * Forward to the view servlet the result of the action processed
 * @param request Servlet request
 * @param response Servlet response
 * @param actionState The state of the action
 * @throws ServletException If a servlet-specific error occurs
 * @throws IOException If an I/O error occurs
 */
private void forwardToView(HttpServletRequest request,
    HttpServletResponse response, ActionState actionState)
    throws ServletException, IOException {

    // Retrieve the view
    View view = actions.getView(
        actionState.getName(),
        actionState.getState()
    );

    // Override the title if necessary
    if (actionState.getTitle() != null)
        view.setTitle(actionState.getTitle());

    // Save the view in the session
    SessionAttribute.setAttribute(request, "view", view);

    // Forward to the servlet View
    getServletContext().getRequestDispatcher(
        view.getTemplatePath() == null ?
        view.getViewPath() : view.getTemplatePath()
    ).forward(request, response);
}
```

Code 15 : forwardToView method code

These different code parts bring a good view of the processing did during a user request to the controller.

5.11.5. iMode HTML tags

This part covers the iMode HTML tags classes. These classes allow using “JSP like” tags directly in the JSP pages. There is also a configuration files with the definition of the tags directly stored in the WEB-INF directory.

The Figure 108 shows the base class for the tags. The AbstractBodyTag allows building complex tags with content between opening and closing tag and the AbstractSimpleTag offer the simple version of tag without a closing tag.

The code can be refactored to avoid the repetition of the attributes but with the inheritance of these classes, it becomes not so easy to refactor correctly. The work to do that is not efficient for this project.

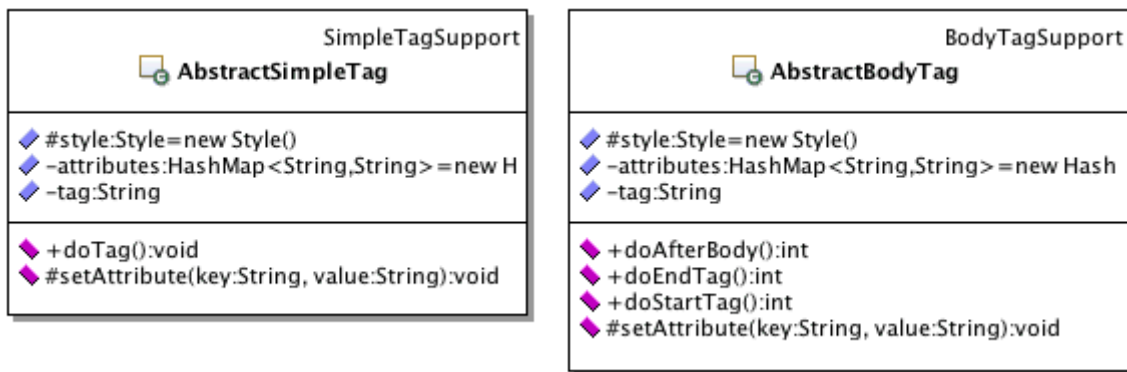


Figure 108 : UML – iMode Tags abstract classes

5.11.5.1. Tags Block

The Figure 109 shows the block type tags classes and the methods to configure the attributes of the tags.

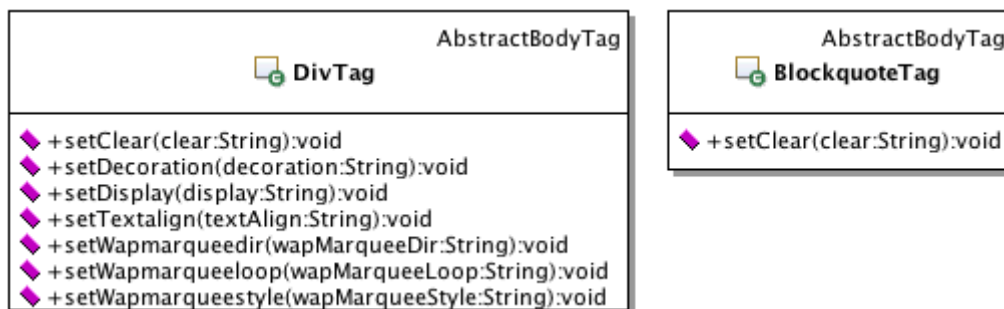


Figure 109 : UML – Tags Block classes

5.11.5.2. Tags Document

The Figure 110 shows the document type tags classes and the methods to configure the attributes of the tags. The document tags are used to create the base of the document.

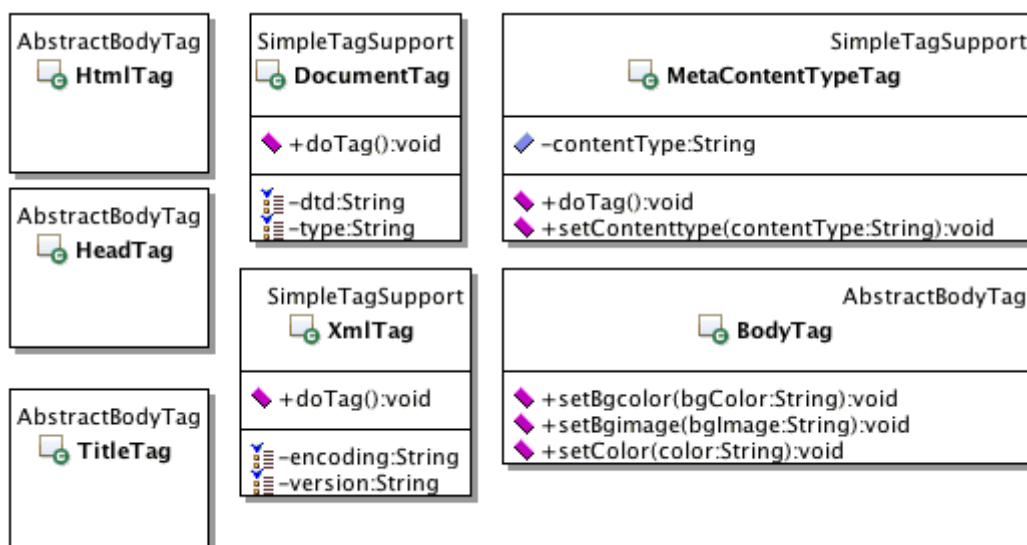


Figure 110 : UML – Tags Document classes

5.11.5.3. Tags Form

The Figure 111 shows the form type tags classes and the methods to configure the attributes of the tags. The form tags allow buildings forms in the web pages. The architecture offers the best way to reuse the maximum of common code. There is a special attribute dedicated to NTT Docomo to add automatically the location data on the client side. This attribute is "lcs". This attribute can be found on the FormTag class.

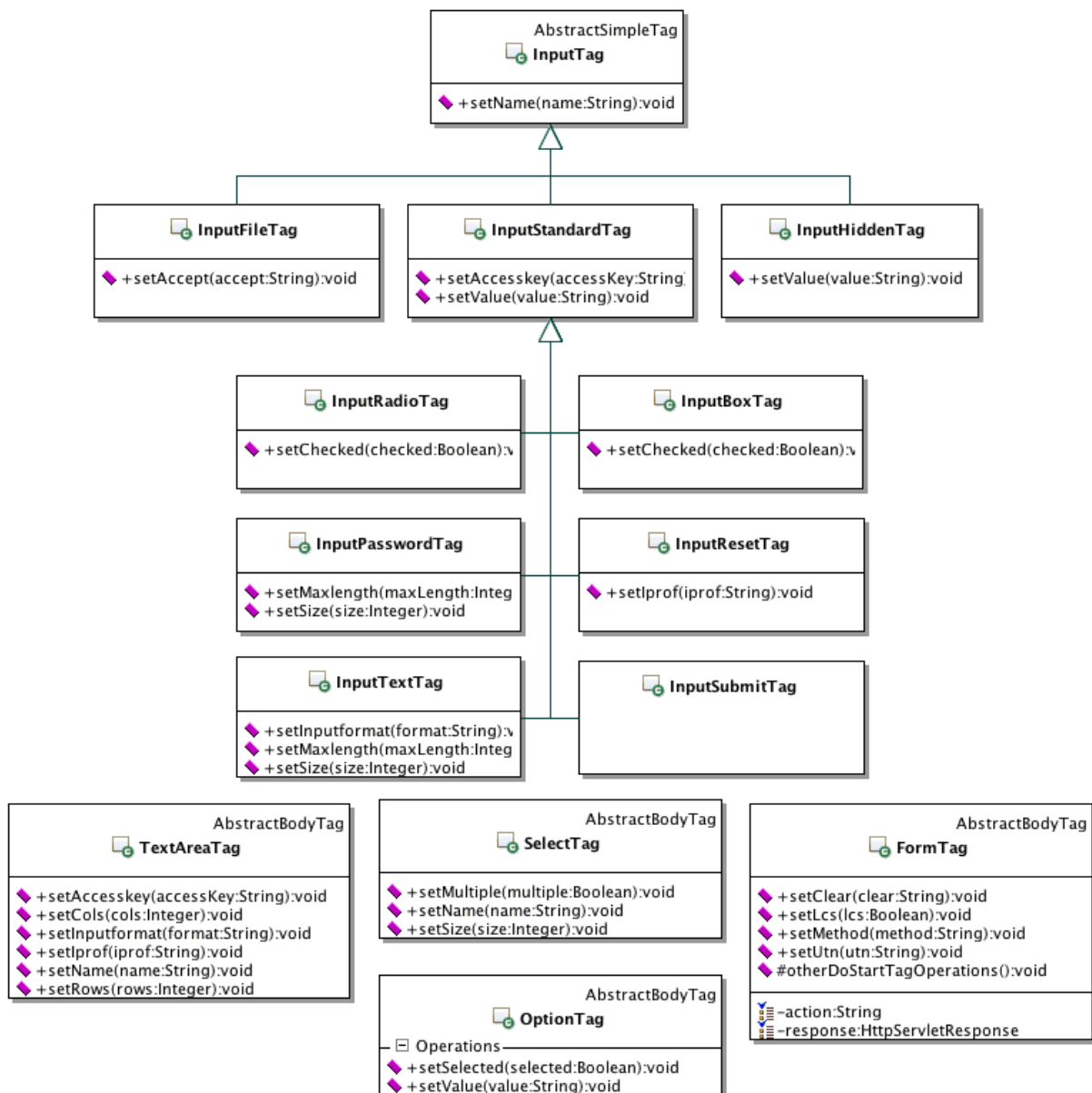


Figure 111 : UML – Tags Form classes

5.11.5.4. Tags Link

The Figure 112 shows the link type tags classes and the methods to configure the attributes of the tags. The link tags are used to create hyperlinks in the output documents. There is the same attribute for the LinkTag class as the form tag for the localization.

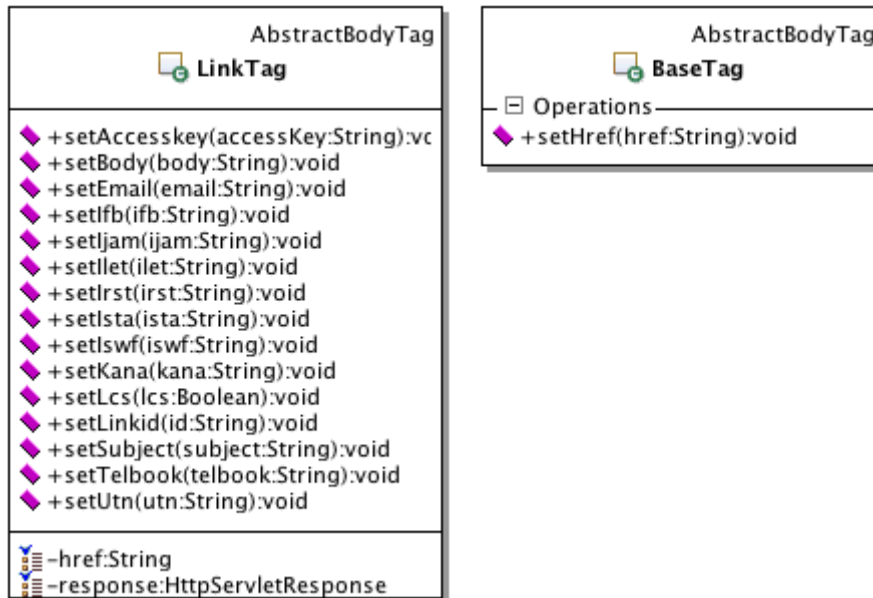


Figure 112 : UML – Tags Link classes

5.11.5.5. Tags List

The Figure 113 shows the list type tags classes and the methods to configure the attributes of the tags. The list tags are used to create bullet list or numbered list...

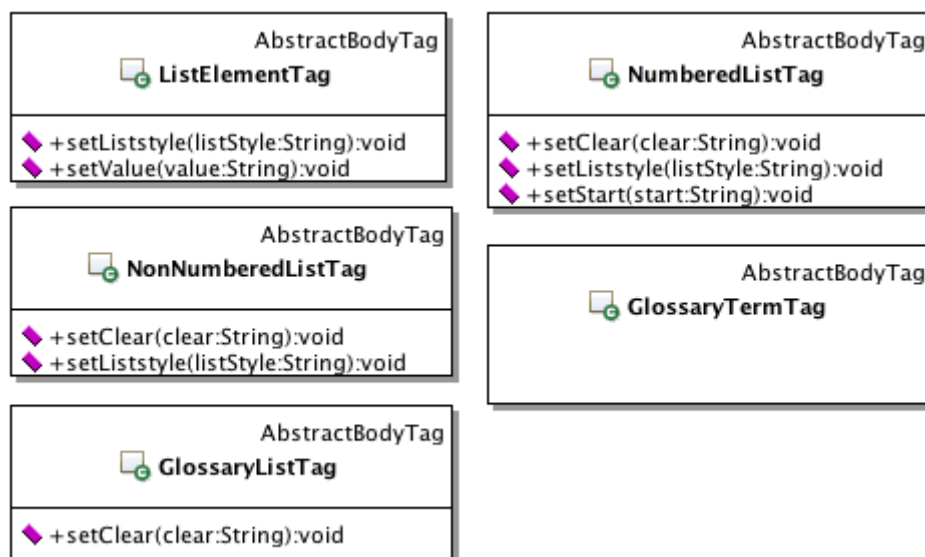


Figure 113 : UML – Tags List classes

5.11.5.6. Tags Media

The Figure 114 shows the media type tags classes and the methods to configure the attributes of the tags. The media tags are used to create pictures and this kind of object. The ParamTag class is used with the ObjectTag to configure it in the JSP code.

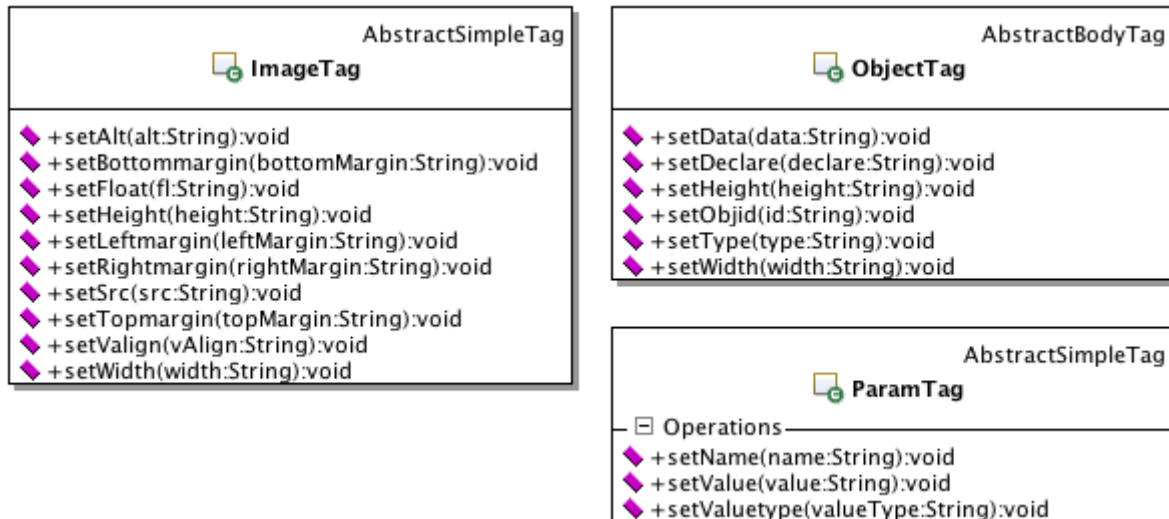


Figure 114 : UML – Tags Media classes

5.11.5.7. Tags Style

The Figure 115 shows the Style class used by the other tags classes to configure the inline style. There is no need to manipulate this class directly because it is used in the mother tags classes.

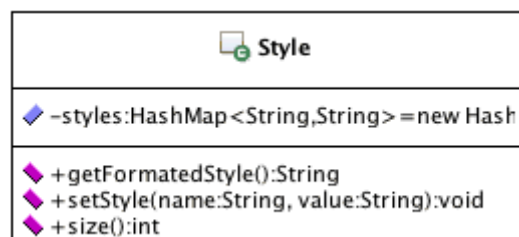


Figure 115 : UML – Tags Style class

5.11.5.8. Tags Table

The Figure 116 shows the table type tags classes and the methods to configure the attributes of the tags. The table tags class and relative classes allows building HTML tables in the JSP pages.

There is no check for the coherence of the usage of the tags. The tags shown here are very simple and quickly developed to help the project development. There is probably a lot of improvement to do for this part.

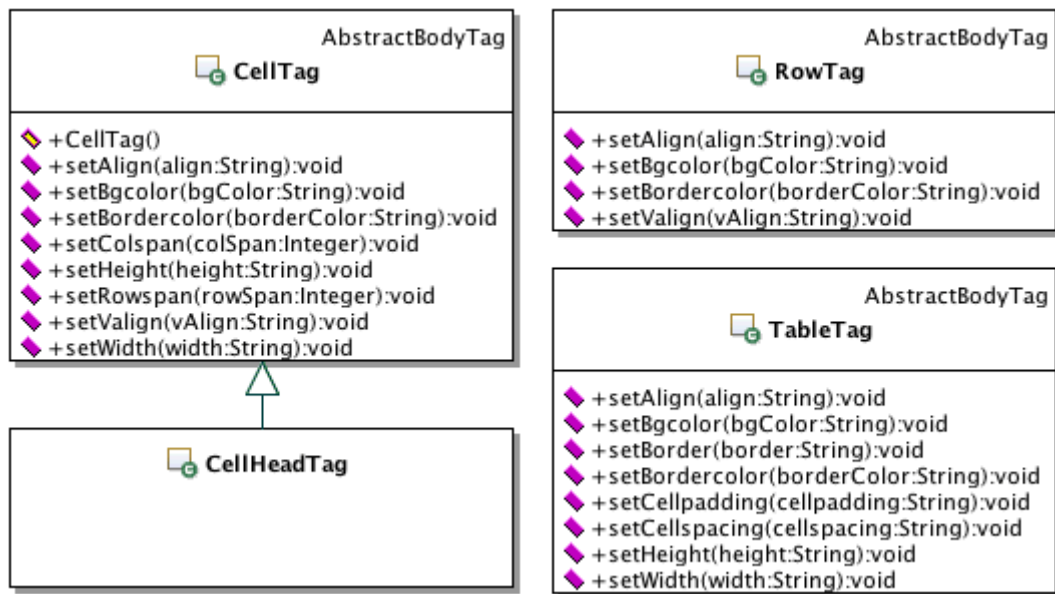


Figure 116 : UML – Tags Table classes

5.11.5.9. Tags Text

The Figure 117 shows the text type tags classes and the methods to configure the attributes of the tags. The text tags allow creating text element in the pages with some presentation possibilities.



Figure 117 : UML – Tags Text classes

5.11.6. iMode Views Formatter

This part focused on the iMode Views Formatter. There is nothing special to discuss because the classes presented are the implements of IFormatter or IMessageFormatter.

The Figure 118 shows the abstract class used by all the formatters and specific to the iMode application. This abstract class provide an abstract method “getViewName()” to retrieve the view to use for the rendering.



Figure 118 : UML – Views Formatter AbstractFormatter class

5.11.6.1. iMode Views Formatter Dialog

The classes presented on Figure 119 are the NPC dialog formatters.

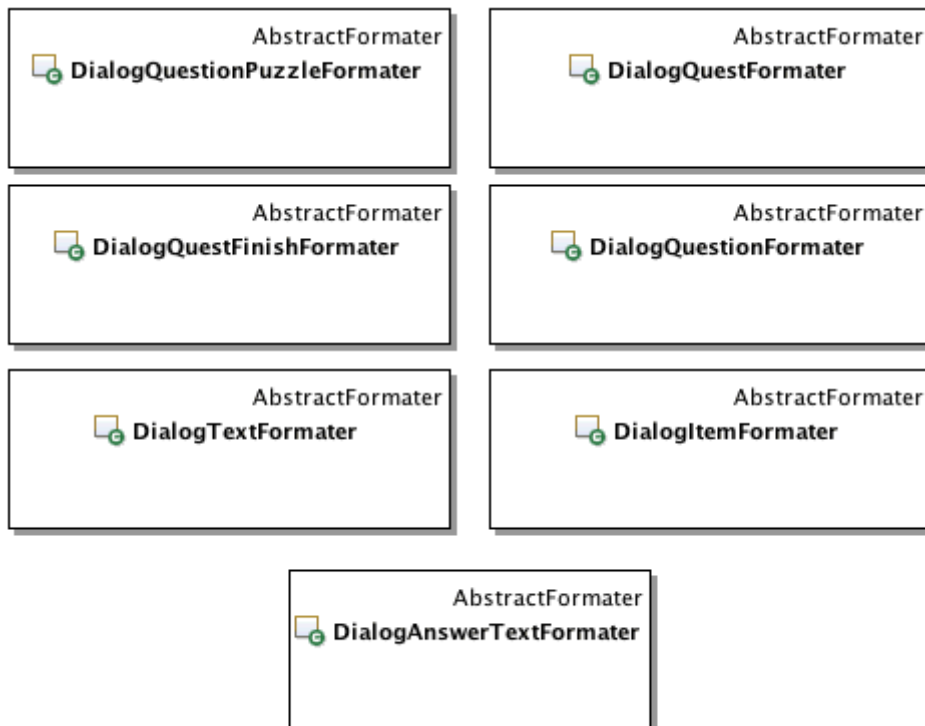


Figure 119 : UML – Views Formatter Dialog classes

5.11.6.2. iMode Views Formatter Entity

The Figure 120 shows the classes for KMEP entities formatters.

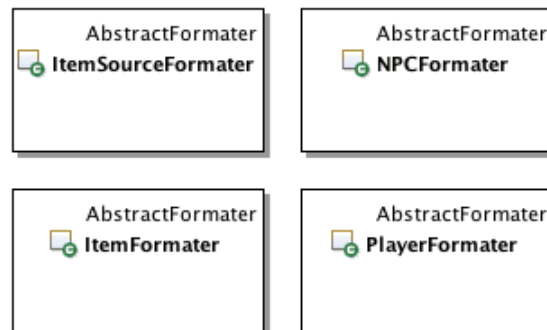


Figure 120 : UML – Views Formatter Item classes

5.11.6.3. iMode Views Formatter Message

The message formatter classes are shown on the Figure 121. They implement the IMessageFormatter and extend the AbstractFormatter.

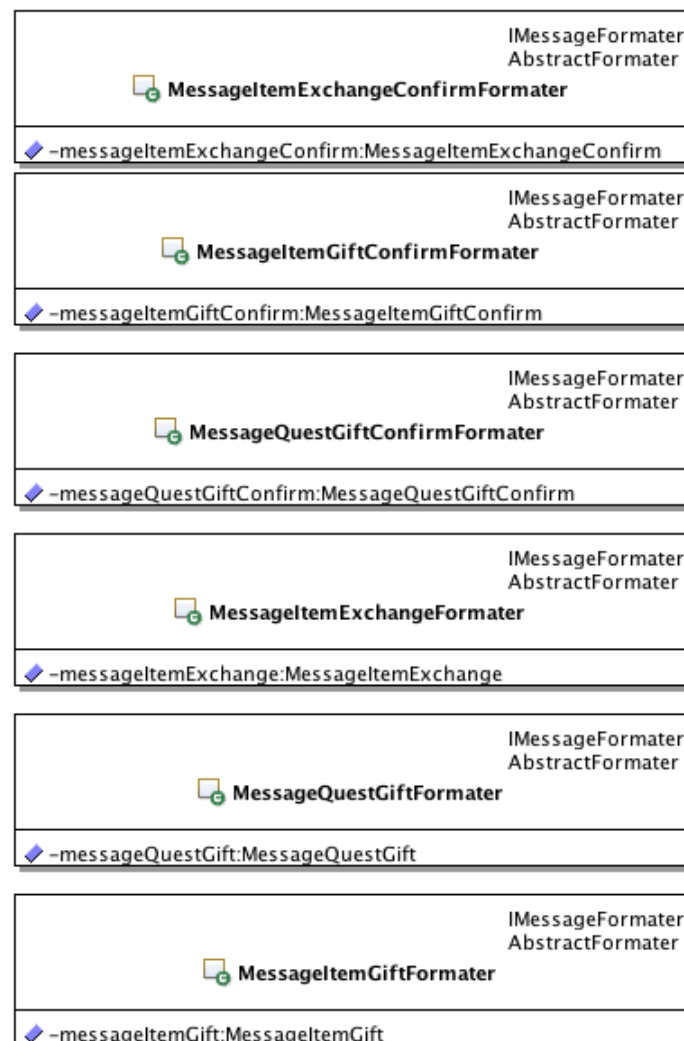


Figure 121 : UML – Views Formatter Message classes

5.11.6.4. iMode Views Formatter Quest

The quest formatter classes are shown on the Figure 122.

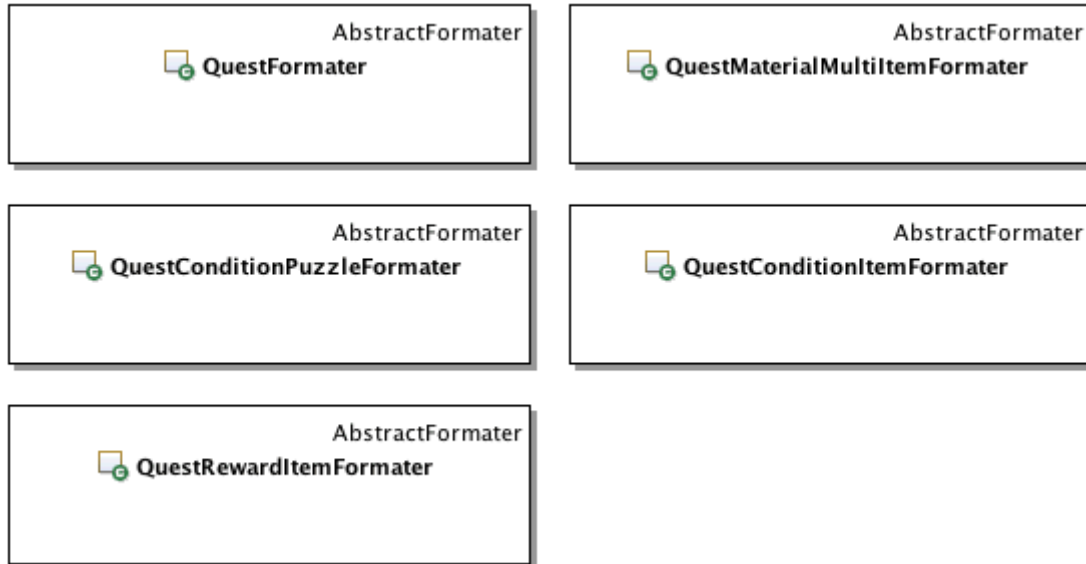


Figure 122 : UML – Views Formatter Quest classes

5.11.6.5. iMode Views Formatter Statistic

The statistic formatter classes are shown on the Figure 123.

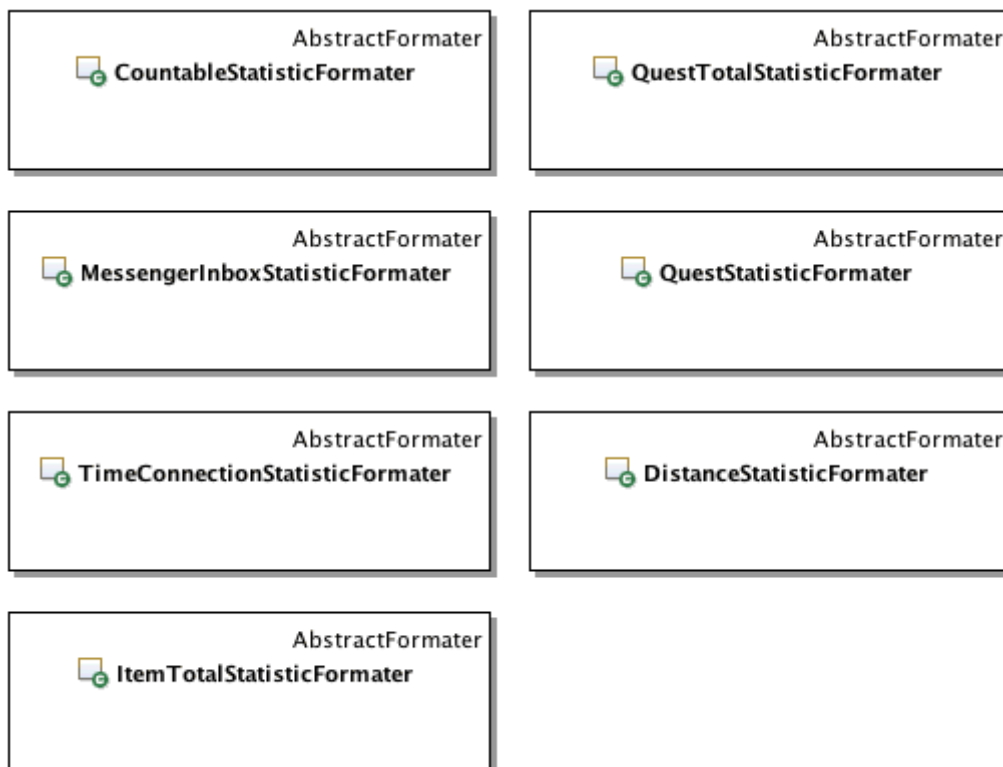


Figure 123 : UML – Views Formatter Statistic classes

5.11.7. iMode Views Helpers

The helper classes shown on the Figure 124 are used directly in the JSP pages to do some useful processing. These classes are rendering utilities to simplify the JSP code.

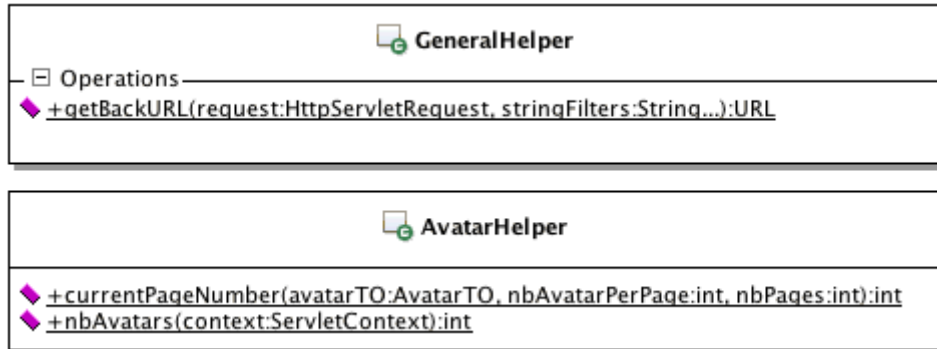


Figure 124 : UML – Views Helpers classes

5.11.8. iMode Views Utils

The Figure 125 presents the utility classes used in the JSP pages or related views processing.

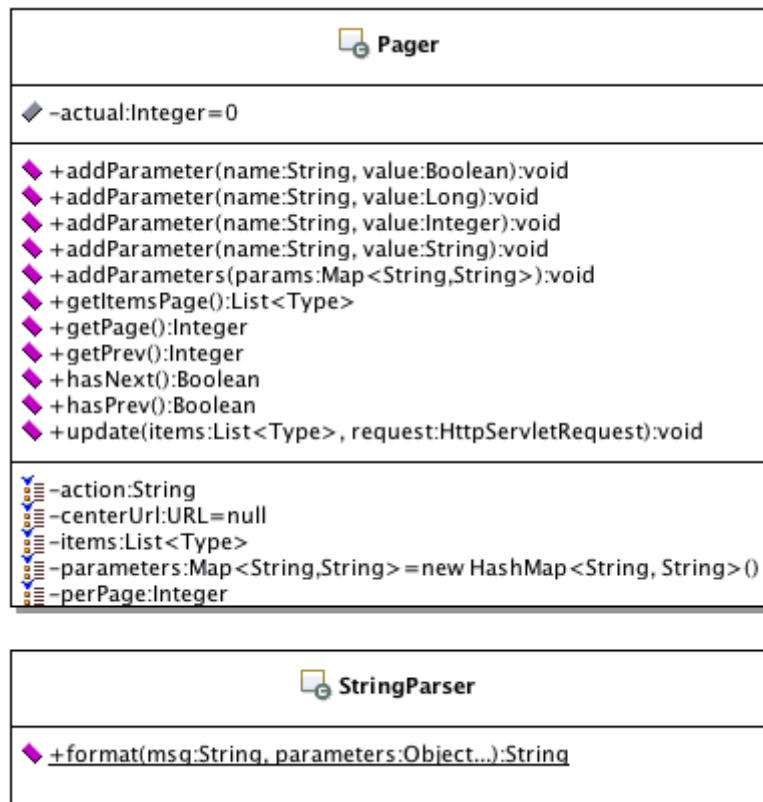


Figure 125 : UML – Views Utils classes

Pager

The Pager class allows managing a list element presentation with more than one page. It allows saving the parameters used in the URLs and save the state of the current page read.

StringParser

The StringParser class allows preparing a string with this kind of format: “This is a %0 with %1 parameters”. After that, the method “format()” is used to replace the “%n” tokens by the Object parameters. The method “toString()” is used to get the textual representation of the objects. The output will be “This is a message with some parameters” if the call is “format(msg, “message”, “some”);” and msg represent the string in previous sample.

5.12. inTrack in use

To use inTrack in this project, it was necessary to do a lot of additional work to prepare the EJB project. Normally, inTrack provides some maven project archetypes that allow directly developing in the chosen archetype. However, when inTrack was added to the project, it was too late to use an archetype directly. It was necessary to add the required element directly in the EJB project to use correctly inTrack in MEP.

5.12.1. Installation

First, it is necessary to create the archetype project to get the necessary data for the real project. To create the client archetype project, the instructions at <http://193.134.218.14/trac/inTrackPartner/wiki/ApplicationClientmodule> must to be followed.

This procedure create a new project that can be opened by NetBeans (navigate the pages on the inTrackPartner documentation to find other information for NetBeans and relative tools installation).

On this base, it is necessary to update the main project with the correct element from the inTrack archetype created project. The first step is to update the pom.xml and nbactions.xml from the parent project and the EJB project.

The dependency shown on Code 16 must be added in the parent pom.xml.

```
<dependency>
  <groupId>ch.intrack</groupId>
  <artifactId>InTrack-ClientRuntime</artifactId>
  <version>0.8</version>
</dependency>
```

Code 16 : inTrack dependency

Check that the nbactions.xml in the main project contains the right actions with the correct values. Maybe you have to add your own specificities and adapt the content shown on Code 17.

```
<action>
  <actionName>build</actionName>
  <packagings>
    <packaging>*</packaging>
  </packagings>
</action>
```

```
</packagings>
<goals>
  <goal>install</goal>
  <goal>-e</goal>
</goals>
<properties>
  <maven.test.skip>>true</maven.test.skip>
</properties>
</action>

<action>
  <actionName>rebuild</actionName>
  <packagings>
    <packaging>*</packaging>
  </packagings>
  <goals>
    <goal>clean</goal>
    <goal>process-resources</goal>
    <goal>install</goal>
    <goal>-e</goal>
  </goals>
  <properties>
    <deploy.server>djohannot</deploy.server>
    <maven.test.skip>>true</maven.test.skip>
    <populate.database>>false</populate.database>
    <wsdl>heig</wsdl>
  </properties>
</action>
```

Code 17 : Main project nbactions.xml

In the pom.xml from the EJB project, check you have the elements shown on Code 18 correctly configured. Maybe you have to configure your own specific values to match your project.

```
<profiles>
  <profile>
    <id>wsdlLocal</id>
    <activation>
      <property>
        <name>wsdl</name>
        <value>local</value>
      </property>
    </activation>
    <properties>
      <wsdl.address>localhost:8080</wsdl.address>
    </properties>
  </profile>

  <profile>
    <id>wsdlHeig</id>
    <activation>
      <property>
        <name>wsdl</name>
        <value>heig</value>
      </property>
    </activation>
    <properties>
```

```
<wsdl.address>193.134.218.14:80</wsdl.address>
</properties>
</profile>
</profiles>

<build>
  <resources>
    <resource>
      <directory>src/main/resources</directory>
      <filtering>true</filtering>
    </resource>
  </resources>

  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-ejb-plugin</artifactId>
      <configuration>
        <archive>
          <manifest>
            <addClasspath>true</addClasspath>
          </manifest>
        </archive>
        <ejbVersion>3.0</ejbVersion>
      </configuration>
    </plugin>

    <plugin>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>2.0.2</version>
      <configuration>
        <encoding>UTF-8</encoding>
      </configuration>
    </plugin>

    <plugin>
      <artifactId>maven-resources-plugin</artifactId>
      <version>2.2</version>
      <configuration>
        <encoding>UTF-8</encoding>
      </configuration>
    </plugin>

    <plugin>
      <groupId>org.codehaus.mojo</groupId>
      <artifactId>jaxws-maven-plugin</artifactId>
      <version>1.9</version>
      <executions>
        <execution>
          <id>entityWSDL</id>
          <goals>
            <goal>wsimport</goal>
          </goals>
          <configuration>
            <wsdlUrls>
              <wsdlUrl>
                (see first link after the code)
              </wsdlUrl>
            </wsdlUrls>
          </configuration>
        </execution>
      </executions>
    </plugin>
  </plugins>
</build>
```



```
<packageName>
  ch.intrack.client.connection.ws
</packageName>
<bindingDirectory>
  src/main/resources/bindingFiles
</bindingDirectory>
<bindingFiles>
  <bindingFile>
    entityManagerBinding.xml
  </bindingFile>
</bindingFiles>
</configuration>
</execution>

<execution>
  <id>trackingWSDL</id>
  <goals>
    <goal>wsimport</goal>
  </goals>
  <configuration>
    <wsdlUrls>
      <wsdlUrl>
        (see second link after the code)
      </wsdlUrl>
    </wsdlUrls>
    <packageName>
      ch.intrack.client.connection.ws
    </packageName>
    <bindingDirectory>
      src/main/resources/bindingFiles
    </bindingDirectory>
    <bindingFiles>
      <bindingFile>
        trackingManagerBinding.xml
      </bindingFile>
    </bindingFiles>
  </configuration>
</execution>

<execution>
  <id>tagWSDL</id>
  <goals>
    <goal>wsimport</goal>
  </goals>
  <configuration>
    <wsdlUrls>
      <wsdlUrl>
        (see third link after the code)
      </wsdlUrl>
    </wsdlUrls>
    <packageName>
      ch.intrack.client.connection.ws
    </packageName>
    <bindingDirectory>
      src/main/resources/bindingFiles
    </bindingDirectory>
    <bindingFiles>
      <bindingFile>
        tagManagerBinding.xml
      </bindingFile>
    </bindingFiles>
  </configuration>
</execution>
```

```
        </bindingFile>
      </bindingFiles>
    </configuration>
  </execution>
</executions>
</plugin>
</plugins>
</build>
```

Code 18 : EJB pom.xml

For presentation reasons, the links above are not directly included in the code shown on Code 18.

```
http://${wsdl.address}/MobileEntityManagerBeanService/MobileEntityManagerBean?wsdl
http://${wsdl.address}/TrackingManagerBeanService/TrackingManagerBean?wsdl
http://${wsdl.address}/TagManagerBeanService/TagManagerBean?wsdl
```

Finally check the nbactions.xml from the EJB project to have something similar to the Code 19.

```
<action>
  <actionName>build</actionName>
  <goals>
    <goal>install</goal>
  </goals>
  <packagings>
    <packaging>*</packaging>
  </packagings>
  <properties>
    <maven.test.skip>>true</maven.test.skip>
  </properties>
</action>

<action>
  <actionName>rebuild</actionName>
  <packagings>
    <packaging>*</packaging>
  </packagings>
  <goals>
    <goal>clean</goal>
    <goal>install</goal>
  </goals>
  <properties>
    <maven.test.skip>>true</maven.test.skip>
  </properties>
</action>
```

Code 19 : EJB nbactions.xml

Now, it is necessary to update the persistence.xml file. Add the code shown on Code 20. This modification allows the EJB mechanism to create the tables for the inTrack local data.

```
<class>ch.intrack.client.model.MobileEntityBO</class>
<class>ch.intrack.model.MobileEntityImpl</class>
<class>ch.intrack.model.TrackingId</class>
<class>ch.intrack.model.Location</class>
<class>ch.intrack.model.Coordinates</class>
<class>ch.intrack.model.CoordinateSystem</class>
<class>ch.intrack.model.CoordinateSystemType</class>
<exclude-unlisted-classes>>false</exclude-unlisted-classes>
```

Code 20 : EJB persistence.xml

With these modifications, the project files are ready to use but there are some other manipulation to do to finish the inTrack installation inside the project. It is necessary to copy some files needed by the inTrack inclusion.

Copy the file “configuration.xml” and the directory “bindingFiles” to the “Other Resources” EJB directory. These files contain the configuration to communicate with inTrack.

Add the “MobileEntityMBO.java” to the correct place in your project. Update the package reference inside the class if necessary. This class is important. It allows the creation of local classes with inTrack facilities for the location system. In simple terms, it allows to keep local data of the remote inTrack data for the location data.

To finish the installation, there are two Java files to add. They are the files that represent the Session Bean to manipulate the remote entities (mobile entities on the inTrack platform). One is the interface and the second is the implementation. Copy the files “EntityBOManagerLocal.java” and “EntityBOManagerBean.java” to the correct place and update the package reference if necessary.

At this point, the integration of inTrack-runtime-client is finished. However, you have some adaptations to do from this installation to match exactly your project. This installation is only in the case you have already your project in a state that does not allow creating and moving your project in the inTrack archetype.

5.12.2. Usage

To use the previous installation, you have to create a class that inherits from the MobileEntityMBO. Pay attention that the EJB annotation must be on the getters/setters and not on the attributes. It is very important.

To manipulate your entities, you have to use the methods offered in the service “EntityBOManagerLocal” to bind/unbind your local entities to the remote entities. They are also the methods to add/remove tracking ids to an entity. Finally, there is the method to update the location.

It is probably necessary to add or update some methods in this service but the best way is to create your own service based on this one. For example, in this project, the service was

updated to match to the exact needs of this project but when there is some update from inTrack platform, the service must to be updated with a lot of attention.

5.12.3. Summary

Normally, the best way to use inTrack is to create the archetype project and after that add your own classes in the projects. However, sometimes it is not possible. In these cases, a manual solution is possible with few works.

5.13. Conclusion

This chapter has shown the details of the MEP and KMEP application. There is also a part especially dedicated to use inTrack in a special way not documented in the inTrack documentation.

The most important things to remember are inTrack can be use in another way that is normally expected and the global architecture is sufficiently flexible to add easily new functionalities.

heig-vd

Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

K-MEP

Conclusions

Laurent Prévost



R RITSUMEIKAN

Table of Content

6.1. INTRODUCTION	171
6.2. CONTRIBUTIONS	171
6.2.1. Game platform	171
6.2.1.1. Quests	171
6.2.1.2. Items	171
6.2.1.3. Dialogs	171
6.2.2. Social aspect	172
6.2.2.1. Culture	172
6.2.3. inTrack evaluation and refactoring	172
6.3. PROJECT REVIEW	172
6.3.1. Mobile phone development	172
6.3.1.1. Solutions	173
6.3.1.2. Remarks	175
6.3.2. Model View Controller	175
6.3.3. inTrack	176
6.3.3.1. inTrack problems	177
6.3.4. Double click	178
6.3.5. Development platform	178
6.3.6. Limits	179
6.3.6.1. Dialog conditions	179
6.3.6.2. Dialog configuration	179
6.3.6.3. Quest puzzles	179
6.3.6.4. Loading	180
6.3.6.5. Player recognition	180
6.3.7. Planning	180
6.4. FUTURE WORK	185
6.4.1. Configuration tool	185
6.4.2. New quest types	185
6.4.3. Items	185
6.4.4. NPC	186
6.4.5. KMEP	186
6.5. CONCLUSION	187

Table of Figures

FIGURE 126 : ORIGINAL PLANNING – PART ONE	181
FIGURE 127 : ORIGINAL PLANNING – PART TWO	182
FIGURE 128 : FINAL PLANNING – PART ONE	183
FIGURE 129 : FINAL PLANNING – PART TWO.....	184

6.1. Introduction

This chapter offers a retrospective of the project and provides some interesting thought for the future.

6.2. Contributions

The different contributions explained in the introduction chapter are covered widely in this project. The different elements are developed and allowed building pervasive games in specific contexts.

6.2.1. Game platform

The MEP offers the platform to build the different games based on the services offered. The different tools developed to help the creation of games are very useful like the configuration mechanism.

The actual architecture is sufficient for a first release but some details have to be refactored to use the different technologies better. The major mechanisms provide the minimal requirements to create great games in a geographical context.

6.2.1.1. Quests

The quests mechanisms are enough to run many different quest types. However, adding new mechanism and improving the actual ones is necessary. Depending on the MEP orientation (in regards of what it was planned originally), the reward part is not necessary and can be removed from the quest part.

6.2.1.2. Items

Items are like the other entities in the game. They can be localized by the platform. The player can carry them and leave them on the map. Actually, they are under used because they are just here to add the support for the realization of some quests.

Improvement of the items in a future release seems to be an important part to do. Actually, there is no other mechanism than the “existent” one. It means that the items have no reasons to exist than the one explained just before.

However, their role in the game is important to realize the “Collect Items” and “Puzzles” quests. They are needed because the quests need them.

6.2.1.3. Dialogs

The dialog tree (graph) mechanism is able to provide rich dialogs from the non-player characters to the players. The different mechanisms to remember the state of the dialogs are very useful too in the perspective to offer a complete dialog system. In this project, the dialog system is very complete and widely enough to create the majority of the dialog.

6.2.2. Social aspect

With the puzzle quest, the interaction between the players is required implicitly. This is a great thing. The players have to communicate in the real world to do some tasks together if they want to success the quests proposed.

6.2.2.1. Culture

In the KMEP realization, the culture is not completely used for the historical places. The time left to the creation of the data was used to create real usable data in easy accessible area for the players.

However, the culture can be approached by the simple fact that the different quests offer to visit some place of the city like the riverside. This a good start in the direction of the cultural aspect of the project. The story construction of the game can bring also cultural facts inside the game.

6.2.3. inTrack evaluation and refactoring

During the development, it was necessary to request some updates to the inTrack team. In a first time, some functionality was missing and in a second time, some functionality was bugged.

The collaboration between the both projects was very interesting and produced a great final product for the both project. The inTrack team was always available to bring some support and help to integrate and use the inTrack platform inside the MEP.

6.3. Project review

6.3.1. Mobile phone development

Due to the mobile phone market in Japan, the original project was not possible to do exactly as it was planned. The principal problem is that each operator has its own technologies with each its own limitation. The data protection is very important and for this reason, it is not so easy to develop an application on a mobile phone.

The choice for NTT Docomo was based on two factors. The first one is that this is possible own a mobile phone with English interface and the second one is that a lot of documentation in English for the use of their platform is available. DoJa is the mobile Java profile for their phones. It is not so far than MIDP but less standard actually. It is a proprietary API. There are some websites with procedure to convert DoJa to MIDP and MIDP to DoJa. Therefore, this is not a real constraint actually.

When the choice was made, the knowledge for the limitations was not enough to know that the use of the functionalities of the mobile phone from the iAppli. The research focuses principally on the deployment aspect. Finally, the NTT Docomo phones allow the best way to

deploy application from a personal platform. It was only when began the GPS development began for the client side that the discovery of the big problem was made.

It was impossible to use the GPS functionality without a real TrustedAPID (this is an id from NTT Docomo to validate the Trusted Content Provider). After checking how to get a real TrustedAPID, it seems there is no relevant information. Apparently, it seems to be only for big companies with NTT Docomo agreement like Google, CNN and so on. After that, the question about the choice of the mobile phone is automatically come back to the mind but the other operators have the same kind of limitation.

6.3.1.1. Solutions

In this situation, the project is in real delicate situation because without localization it was impossible to do correctly the project. After analyzing some other solutions and orientations for the project, some possible ways of solutions were found:

QR Codes

It was possible to read some QR Codes (bar codes in 2D) from an iAppli. It can be possible to create some QR Codes attached to a specific location, put these QR Codes on the right places. The player has to scan a QR Code when he arrives in a certain place. After that, the server does the matching between QR Code and location.

This solution is relatively uneasy to do because of creation of the codes, to check the location where to put them, to go at the right places and put the codes. In addition, it is not possible to put codes everywhere. The player has also some action to do where he scan the QR Code.

With this solution, it is not possible to follow exactly the player due to a non real time localization system and it is not possible to generate events depending on the location. In the first game rules, many elements depend on the location system that is not possible to integrate exactly as it was planned. Therefore, a redefinition of some part of the rules is necessary with this solution.

GPS Tracker

A GPS Tracker is a device that allows doing some localization and sending via a specific network (GPRS...). This device is independent of the mobile phone and cost a relative high amount. In addition, you have to subscribe to a service with another amount per year for example. Depending on the device, you can retrieve the data with some APIs. No more research in details about this possibility due to the cost was done. The budget for this project is limited and is already reached.

With this solution, the player has to wear this device in addition of the mobile phone. The game server has to do the matching between the device and the mobile phone and the server has to get data from the service provider of the GPS Tracker module. This solution, allows following the player in real time and continuing with the original game design. It is

too costly for this project to run a test with more than one person so it is not possible to buy more than one device with the service.

No localization

This is the worst solution at all because with no localization all the game design is to redefine. Imagine a solution to deduce the localization from the quest for example but in this case, it is not very relevant because the players can do their quest at home without going to the place they are supposed to go.

Get a Trusted APID

At all, it can be the best solution but not realistic. To become a Trusted Content Provider from NTT Docomo, there is a long procedure and many criteria to match. Simple student with a six-month project is not able to get this kind of status. Therefore, this solution is not very applicable for this kind of project. It seems that there is not a lot of project with mobile phones and geo-localization in university contexts.

iMode application

NTT Docomo offers a way to get the location by an iMode application. iMode is the technology for the Internet on the NTT Docomo mobile phones. They have two special HTML tags to do that. The first is an <A HREF> modified and the second is a <FORM> modified. You have just to add the "lcs" attribute in the tag like that: "" and after that, when you click on the link, the handsets automatically show you a popup with the choice of the localization source. When you choose the source, another popup is shown to ask you if you want or not to send your location to the content provider.

This solution is not perfect too because the player has to manipulate some menu and take decision but it is better than QR Codes because it is not necessary to go to places to put some tags and take the risk that the tags disappear. This solution implies that this is not possible to do a real time localization of the players.

In addition, there is two ways to explore this solution. The first one is to only do the localization with an iMode webpage and the game continue in an iAppli. The second one is to do all the application in an iMode website. The first solution allows to access of some peripherals but the game play flow will always interrupt by switching between the iAppli and the iMode webpage. The second solution do not allow to access to peripherals but do not interrupt the game play.

iMode Website

With the problems of accessing the GPS location data, there is no choice. It is obligatory to develop an iMode website. The website is in iHTML or iXHTML. iXHTML is chosen for its possibility of design. The possibilities are not so high but it is sufficient for this kind of application. This is not perfect but at least, the location data can be obtained directly in the game flow.

A lot of stuff is not available in iXHTML. For example, the JavaScript is not supported. Therefore, it is not possible to develop a web 2.0 application for the client application. Some other limitations are present. The iXHTML do not allow using external CSS or some style for certain tags. Not all the tags are available too. Finally, the possibilities are very restricted to the basis.

The biggest problem is that there is no JavaScript or something else on the mobile client. For example, this is not possible to draw the map and add some icons on it directly on the client. For this part, the map must be drawn in the server and icons added directly on the server too.

Some limitation with inputs is also present. For example, some inputs accept only 100 – 500 characters. The lists are also limited in size of elements. However, for this application, it is sufficient. It is important to limit the inputs to the minimum and used list, links, buttons or tables for the interaction.

A framework like JSF for iMode application could be very useful but it seems there is no one in English or European language. Maybe there is a framework in Japanese but it is not suitable for this project. In this situation, a homemade framework is required. This is more just a group of components to use rather than a framework. The advantage is that the iXHTML code is only defined at one place. The usage of these components is very easy. All components are IDocument and some of them can store other IDocument for the rendering. For example, a table cell can store other IDocument to render like text, buttons, and so on.

This part is not a very high level of abstraction but it is sufficient for the needs of the application. The iMode application has to be developed because of the problem with the GPS location. If there is no problem with GPS, the client on the mobile phone will be a DoJa application without the need of iMode application.

6.3.1.2. Remarks

During the beginning of the development of the iAppli, a try of the framework J2ME Polish. This is a tool to develop with more efficiency for mobile phones. This suite come from a company based in Germany and focused principally for the European market.

It was impossible to run with success the samples provided in the version 2.0.4 tested in july/august of the project. It was also impossible to focus more on this solution due to the time for the project and after the problem encountered by the use of GPS.

6.3.2. Model View Controller

After some discussion with Professor Liechti, he recommended to implement a simple MVC for this part of the application. A simple framework can be used to develop quickly the web user interface. With the framework developed previously, this goal was reached but for a better architecture and more efficiency productivity, the solution is not adequate.

In this situation, some documentation shows how to set up a simple MVC for the web user interface. To reach this goal, it is necessary to create a library of JSP tags especially for the iMode application and architecture to handle the Model View Controller.

The architecture is relatively simple. A central controller receives the requests and forwards to the correct action the task to do. The choice of the action is based on a request parameter ("act" parameter). After that, the action does its job and returns to the controller an action state with the view name to call. The controller forwards the result to the correct view.

The configuration of the actions and views can be done in an XML file especially created for this purpose. In the XML file configuration, it is possible to prepare the binding between views and actions. One action can have one or more views depending of the result to show. In conclusion, there is a little dependence between actions and views. The actions have to know the view to use but they do not know how the views work. This is the principal aspect to avoid with this architecture.

The controller embedded a basic security check (if the player is connected or not and if the action required that the player has to be connected or not). With this solution, there is only one place to do this check. The developer has to pay attention to the fact that it is possible to call an action from another action. It is possible to break the security check. For example: Action A and B, A do not require a connection from the player but the action B does. If the action A calls the action B, the security check was already done for the action A but for the B. This is a normal behavior and this is the responsibility to the developer to use the actions correctly with this specific point in mind.

6.3.3. inTrack

The KMEP application has to use inTrack to do the geo-localization. This platform developed at HEIG-VD is especially done to track geo-localization data. There are some ways to use it.

Idea is to create a GPS Tracking Module that provides to inTrack the geo-localization data. Originally, the mobile phone has to play this role but with the troubles already explained before in this document, it was not possible to do that.

The second solution was to create a Tracking Module on the server side. The iMode application has to call the Tracking Module to update the geo-localization data. Tests of this solution but were not relevant. The problem is that it is necessary to forward geo-localization via an HTTP Connection from the iMode (WAR Project) application to the Tracking Module (WAR project too). The Tracking Module forwards also the data to inTrack platform. Therefore, the Tracking Module is only an intermediary between the application and inTrack platform. In this situation, it is necessary to merge the two WAR Project to allow the iMode application to become the tracking module.

The merge was possible but the application has also to retrieve inTrack data in EJB Project. For that, MEP has to implement and use a third way of inTrack. The inTrackClient-Archetype. Idea is already the same than the two first explained just before but directly in a way of a rich client. In this third application, the EJB Project becomes the Tracking Module and the

data getter from inTrack. This layer has two roles to play. One is to forward to inTrack the localization updates and the second is to get the relevant data.

6.3.3.1. inTrack problems

During the implementation of the last solution, some problems are encountered a real big problem that takes a lot of time to correct. The project takes some delay (about two weeks) to fix it and a lot of energy. David Johanot from the inTrack team helps to understand the problem without success.

The problem was very special. During application tests without the inheritance all works correctly but when with inheritance for the location purpose, the application changes its behavior.

Normal behavior flow:

- 1) Splash screen
- 2) Register (pseudo, password and password confirmation)
- 3) Choose an avatar
- 4) Register confirmation
- 5) Splash screen
- 6) Connect (pseudo and password)
- 7) **First dialog (first connection)**

Problem behavior flow:

- 1) Splash screen
- 2) Register (pseudo, password and password confirmation)
- 3) Choose an avatar
- 4) Register confirmation
- 5) Splash screen
- 6) Connect (pseudo and password)
- 7) **Error dialog with a message that tells the player is not connected to the game.**

The only change in the code is the inheritance. Without inheritance, all works. With inheritance, connection to the game is impossible.

The problem was discovered in the map that contains the behaviors of a Kmap entity was not the one attended. The class KmapEntity is prepared with a @MapKey annotation to have a map with a specified attribute of the AbstractBehaviour class but it seems that this map is not correct.

Normally, the map waited is composed of <String, AbstractBehaviour>. This is the fact without inheritance. With inheritance, the received map is <Long, AbstractBehaviour>. When the use of map, it is always searched for, a String so with Long values this is not possible to have a correct functionality.

After a lot of try and research, the problem was that the inTrack classes that are inherited from KmapEntity used EJB annotation on the getters/setters method and KmapEntity class uses the EJB annotation on the attributes. The EJB annotation is possible on the both place but only one is considered. In addition, in this situation, it is the first one encountered by the Java Persistence process. This problem was very subtle because there is no error and no exception.

After the move of annotation on the getter/setter method of the map (just created for the occasion), it works. The application was completely tested and there are no more problems for this one. However, there is a trouble for the getter/setter method. Originally, these methods were not planned to be developed because of the specific methods used to access to the map (add and find method with complex functionality).

This is not possible to ask the inTrack team to change their manner to annotate their entities but it is not possible to change all the annotation of this project too. With these getters/setters, the integrity is compromised of the Entities if they are not used properly. The private attribute for the methods can be used. This is not very great to do that but the needs to warranty the integrity of this class is also important and actually, this is the best way to do that.

6.3.4. Double click

In the actual version of KMEP, a problem allows cheating exists. The problem is when a link is activated; it is possible to activate it again directly. For example during a dialog that offers some items to the player, the player can click again on the link and he can receive the items again.

This problem was discovered on the emulator client because it is very easy to do a double click with a mouse. On the real mobile phone, the problem is not easy to reproduce. The speed of communication is enough to avoid the most of this case. However, this is not a final solution. It is just possible to deal with it actually.

This problem is not corrected yet because of the time left for the project. This is not a big problem but is important to notice it and solve it in a future release of the KMEP.

6.3.5. Development platform

During the project, the development platform changed from a Windows XP support to a Mac OS X support. The development has not encountered any inconvenience during the change. The tools under the both platform are based on Java technology and are interoperable.

The only trouble found was about the NTT Docomo iMode (i(X)HTML) emulator used to test the Web User interface. This emulator is only developed for Windows XP and does not run under Mac OS X. The possibility to run a virtual machine under Virtual Box from Sun was sufficient to develop and test the Web User Interface correctly.

The opportunity to change the platform shows the flexibility of the project for the development part. The deployment on a local GlassFish server works with the same manner on the both platform. The configuration to prepare the both platform is nearly the same.

6.3.6. Limits

The MEP is very flexible for the configuration but there are some limitations. These limitations are important to know how to deal with them for the configuration of the application.

6.3.6.1. Dialog conditions

Actually, the conditions are very limited and always evaluated with an “and” conditional clause. It means that all the conditions must to be true to consider the dialog as available to show to the player. In this situation, it is not possible to create a dialog with a quest condition with two different quest states.

To build a dialog with two different quest states, is necessary to build two starting dialog with the two different conditions. The rest of the dialog could be the same. This is not perfect and it must be used with precaution because it can create inconsistent dialogs.

The management of the conditions must to be reviewed and improved by adding some possibilities like group of conditions and operators. This is a major work on the dialog conditions part.

6.3.6.2. Dialog configuration

The dialog configuration is not sufficiently dynamic. The consequence is this is necessary to build many dialogs to do simple things. For example, in the Ritsumeikan data set, a NPC give four or more puzzle quests to the player. It can give the quest only when the player has solved the previous one.

The configuration required to do that is very tricky because for each quest, there is an entire dialog to build with the conditions and state to remember. This is very uncomfortable to configure many dialogs for a same NPC. An idea could be to create a group of quest with an order and the NPC can give the first quest, the second and so on depending on the quest states.

This is probably not the only one problem for the dialog configuration. The mechanism is relatively basic but it allows a good configuration of dialogs for the majority of the cases. This is sufficient actually.

6.3.6.3. Quest puzzles

The quest puzzles give to the player some pictures to help to find the solution required by the quest. These pictures remain after the quest in the player inventory. For the other quest with item conditions, the items are taken back after the quest ending. This is not the case for the quest puzzles.

This is not a true limitation but the consequence is that the player can leave on the map the pictures that are no more useful and allows other players find them. With this behavior, this is possible to avoid the requirement to find other player with the same quests.

Another limitation of this kind of quest is there is no way to know how many pictures are used to compose the puzzle. The only way to know that is to find a sufficient number of players to get all the pictures one time and more. This limitation can be viewed as an opportunity to extend the social interaction but this is not convenient.

A little trouble persist in the presentation of the quest data. The player receive a material (a picture) when he accepts the quest but the material described in the quest details is not the same than he received. This is a little bug to solve in a future version. This bug does not a blocking problem.

6.3.6.4. Loading

This is only possible to load data during the deployment phase. It means when the game is started with real data and real players registered to the game that this is not possible to add new data without cleaning the database. The mechanism to load new data is ready but it is not available actually to add new data.

6.3.6.5. Player recognition

Actually, the game is asynchronous. In this situation, it is not possible to send alerts in real time to the player. For example, a player A is at certain place and leaves them. During maximum ten minutes (session duration or position update) the player remains at his position and could be seen by other players. If the system alerts the other players near the player A, the alert can be obsolete because the player A is no more at the same place.

A good idea to avoid this problem is to invite the player to wear the same distinguished sign to allow the other player to recognize them together. In this case, the players can easily recognize other players and ask them for something related to the game otherwise, the players have to discuss with everybody to know who play to the game.

6.3.7. Planning

The original planning was designed just after the pre-diploma project and was based on the Use Cases written for it. In this situation, it was not possible to avoid the problems encountered during the project itself.

The differences between the both planning are principally due to the major problem encountered in the first month. The discovery of the non-usable GPS embedded functionality inside DoJa program forced to redefine many details of the project.

In this situation, the planning was totally changed to match with the reality. Finally, there is no more iteration in the project. The reason is that the project follows a research model to develop the MEP with the flexibility to change ideas and architecture during the development process. This methodology is more like the Extreme Programming model than the usual one.

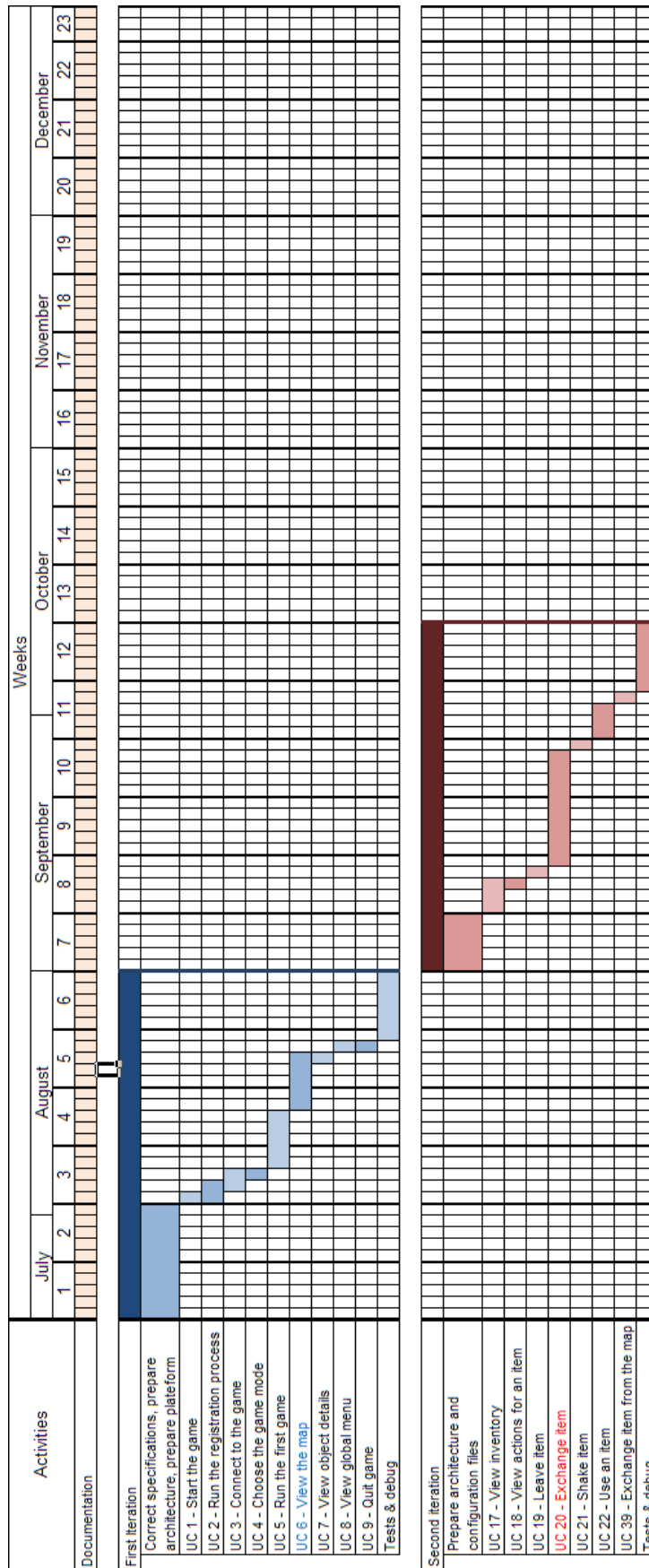


Figure 126 : Original Planning – Part one

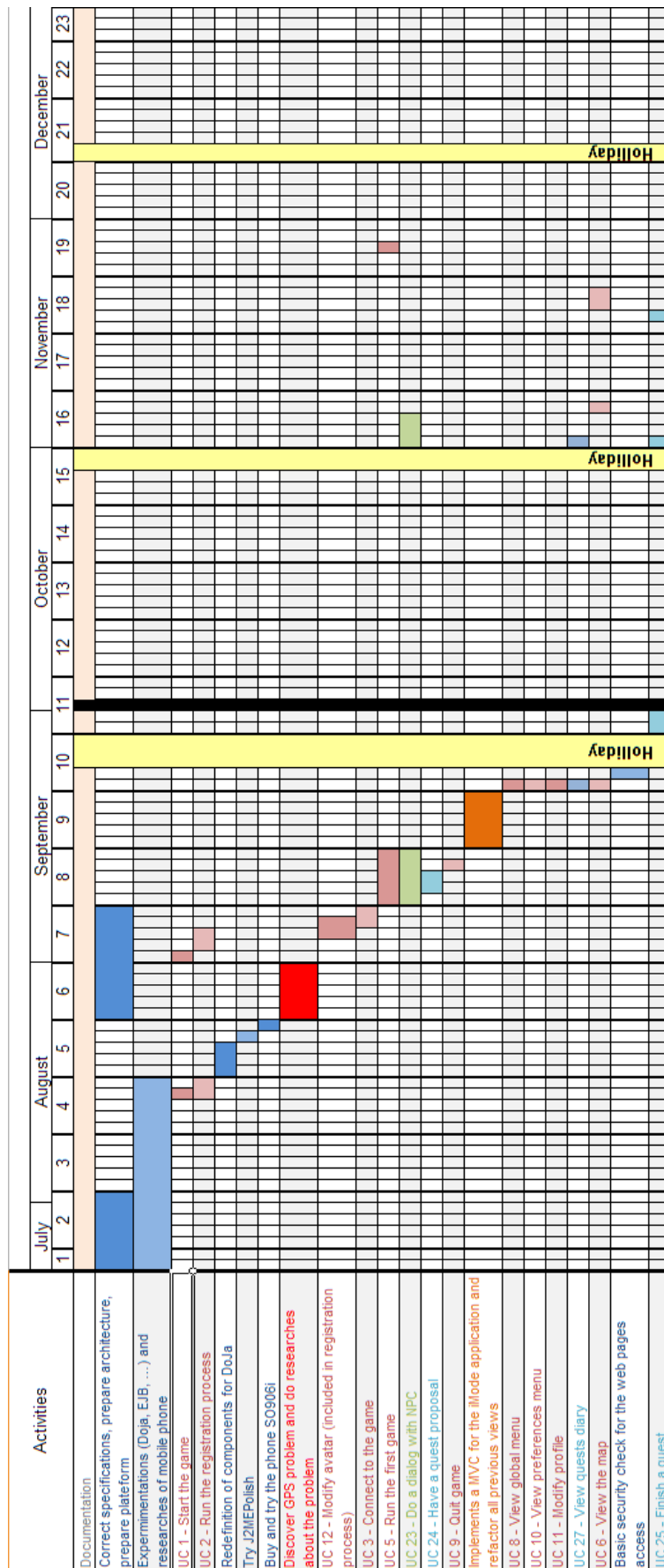


Figure 128 : Final Planning – Part one

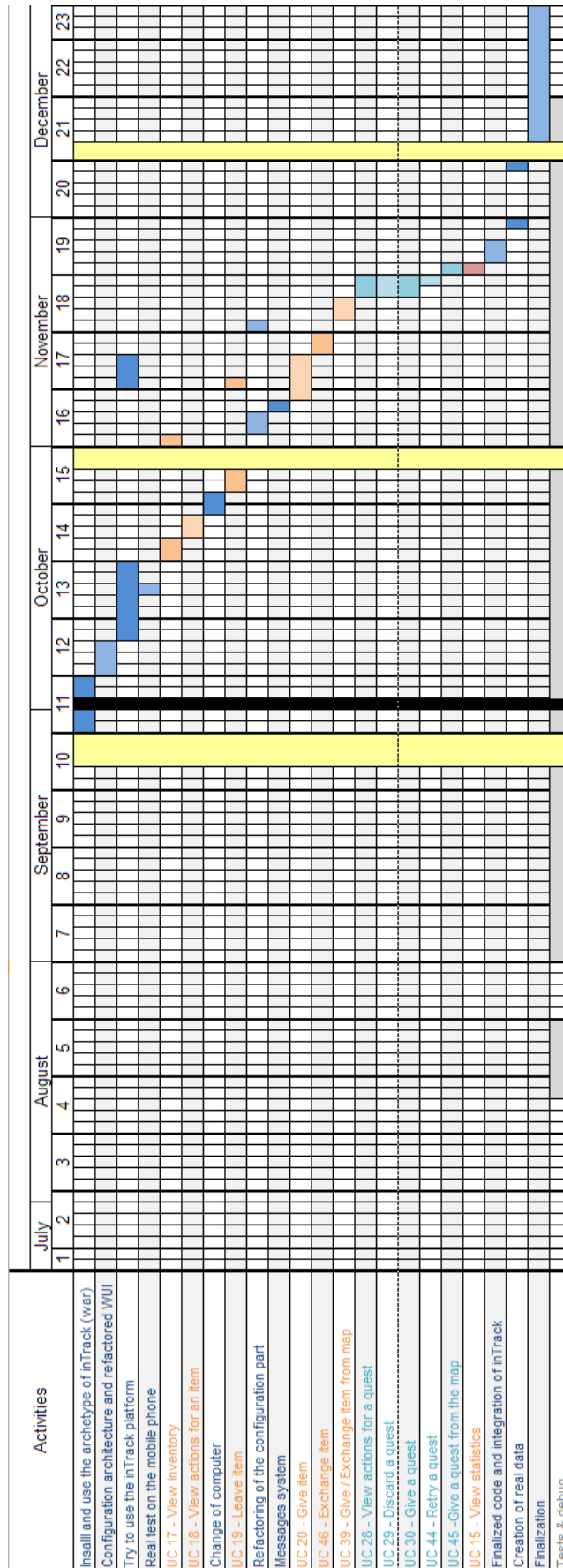


Figure 129 : Final Planning – Part two

It is easy to do this kind of model when the development implies only one person. The constraints for this project were the inTrack Team availability (not easy with the time difference) and the final deadline.

The planning shows equally the cancelled Use Cases. Before the beginning of the project, some of Use Cases were suppressed for realistic reasons. The time for the project does not allow developing all the Use Cases planned. In the final planning, more Use Cases are cancelled. This is due to the discussion with the Professor Liechti and the situation encountered in Japan with the mobile phone technologies.

6.4. Future work

This first version of MEP and KMEP brings much functionality and is sufficient to play. However, there is a lot of work to improve the platform and the game implementation.

6.4.1. Configuration tool

Actually, the most important thing to provide with the MEP is a configuration tool that allows editing the XML files without to write XML code. This kind of tool is very convenient to manage the references between the different files. It allows the data coherence and provides a great view of the data. The tool must to be user-friendly as more as possible.

For example, the dialogs can be shown by a tree (graph) with the relations between each relation. This configuration tool is a very big work to do but it is necessary to offer it for the non-developer people that want to use the MEP.

6.4.2. New quest types

With the actual quest types, the game can be quickly repetitive for players that want to play regularly. It is necessary to imagine and build new quest with new features and not just creating new content (but it is also important).

6.4.3. Items

In this state of the MEP, items are only element used to complete a quest or solve puzzle. In a future, given more importance to the items could be a great idea. For example, some items could be useful to create other ones.

For example, a quest can ask to bring a “soup” but the player must to prepare the “soup”. To do that, he has to collect the different ingredients and to mix them to create the soup. However, he can fail the preparation because his skill in cooking is not enough to prepare the soup. In this situation, maybe the player has to redo his job more than one time.

6.4.4. NPC

The NPCs do not move actually and stay always on the same place and all the time. A great idea is to add time and move mechanisms for the NPCs. With that, the players can only find the NPCs during a certain period of the day or a certain event. Sometimes he has also to follow the NPCs.

The time mechanism could be used for other functionality in the game to add some event mechanisms. It can be interesting to cover special city events or something like that.

6.4.5. KMEP

The KMEP web user interface has to be reviewed to use more efficiently the session mechanism. Actually, the mechanism is not enough used. They are several technical improvements to do for this interface like the security management, the action management and so on.

The actually state of the web user interface is enough to demonstrate the application correctly. The first thing to correct is the double-click problem. For the tests, it is not a problem but for a release, it is very bad.

6.5. Conclusion

At the end of this project, the principal goals have been reached. The platform developed offers the mechanism to create and deploy pervasive games in a geo-localized context. The integration between the game platform and inTrack works perfectly.

The collaboration between the both projects was very interesting. However, it was not easy to collaborate due to the time difference. Time schedules needed to be coordinated to maximize the availability of parties from both sides.

The Japanese context offers a great environment to develop the platform for the cultural aspect but the technological one was very surprising. Normally, the mobile phone technology is in advance in Japan and this is the truth. The actual network generation and services are widely in advance from European countries. However, the providers restrict the usage of the technology for the embedded application. This is the major problem for this project and has required a major change for the game user interface development.

The alternative chosen to develop the game in the context of Kyoto is finally, not bad. This is another approach of the technology offered by the providers. The embedded tags with geo-localization recognition are very helpful for this project. Without this mechanism, it would not have been possible to develop the game user interface.

The different technologies used in the project are not perfectly implemented. However, they are used sufficiently and correctly to offer a stable platform. Not so much work would be required to improve the game platform (without adding new functionalities).

Overall, the project has reached its objectives and offers a useful overview of the possibilities for other, future, applications of the same kind. Hence, the project has served as an excellent development trial prior to full-scale commercial development. With minor refactoring and improvement, the system infrastructure (configuration tool), the platform can be used widely.

heig-vd

Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

K-MEP

Appendix

Laurent Prévost

The logo for RITSUMEIKAN, featuring a large white letter 'R' on the left and the word 'RITSUMEIKAN' in a smaller, white, uppercase sans-serif font to its right, all set against a dark red rectangular background.

R RITSUMEIKAN

Appendix

A.	GAME RULES	193
B.	SPECIFICATIONS REVIEW	223
C.	CONFIGURATION	321
D.	MEP INSTALLATION	379
E.	ERROR CODES	389
F.	MOBILE PHONE RESEARCH	399
G.	DOJA 5.1 QUICK START	407
H.	CODE STANDARDS	433
I.	TOOLS	441

heig-vd

Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

K-MEP

Game Rules

Laurent Prévost

The logo for RITSUMEIKAN, featuring a large white letter 'R' on the left and the word 'RITSUMEIKAN' in a smaller, white, sans-serif font to its right, all set against a dark red rectangular background.

R RITSUMEIKAN

Table of Contents

A.1. INTRODUCTION	199
A.2. DEFINITIONS	199
A.2.1. RPG – Role Playing Game	199
A.2.1.1. Player	199
A.2.1.2. PC – Player Character	199
A.2.1.3. NPC – Non Player Character	199
A.3. SORT OF PLAYERS	199
A.3.1. Short term	200
A.3.2. Medium term	200
A.3.3. Long term	200
A.4. AVATAR	200
A.5. GEO-LOCALIZATION	200
A.5.1. Bicycle	201
A.6. BASE GAME	201
A.6.1. Registration	201
A.6.1.1. Starting Game	201
A.6.2. Games mode	201
A.6.2.1. Simple mode	201
A.6.2.2. Quest mode	203
A.7. GAME MECHANISMS	203
A.7.1. Characteristics	204
A.7.1.1. Characteristics defining	205
A.7.1.2. Precisions	206
A.7.2. Inventory	207
A.7.2.1. Unique items	208
A.7.2.2. Possible actions	208
A.7.2.3. Persistence	209
A.7.3. Monsters	210
A.7.3.1. Remarks	211
A.7.3.2. Combat resolution	211
A.7.4. Combat flow	213
A.7.4.1. Player's death	213
A.7.5. Non Player Characters	214
A.7.6. Non human moves	215
A.8. QUESTS	216
A.8.1. Organization	216
A.8.2. Quest diary	217

A.8.3. Conditions	217
A.8.4. Time	218
A.8.4.1. Relative	218
A.8.4.2. Absolute	218
A.8.5. Giving the quests	218
A.8.6. Grouping	218
A.8.6.1. Group ownership	219
A.8.6.2. Group Member	219
A.9. REWARDS SYSTEM	220
A.10. MAP	220
A.10.1. Fog of war	220
A.10.2. Information	221
A.10.3. Player area	221
A.10.4. Localization	222
A.11. ALERTING	222
A.12. CONCLUSION	222

Table of Figures

FIGURE 130 : SIMPLE GAME MODE MAP SAMPLE	202
FIGURE 131 : QUEST GAME MODE MAP SAMPLE	203
FIGURE 132 : ATTACK FLOW	211
FIGURE 133 : THEFT FLOW	212
FIGURE 134 : ESCAPE FLOW	212
FIGURE 135 : ITEM FLOW	213
FIGURE 136 : COMBAT FLOW	213
FIGURE 137 : FOG OF WAR	220

Table of Tables

TABLE 1 : PLAYER CHARACTERISTICS 204

A.1. Introduction

This appendix will describe the game mechanism as it was planned during the Pre-Diploma Work. The texts with gray backgrounds are some parts that are not available in the current version of the game. The texts with light yellow background are new additions or corrections.

A.2. Definitions

A.2.1. RPG – Role Playing Game

The idea of role-playing game is to bring players in a specific context that they can discover and interact with their environment. Players can talk to non-player characters. They can find items and do some mission usually called “quests”.

During the game, players will improve their characteristics and gain experience points. These points will improve their abilities and permit continuing the progression in the game. Players can discover more and more things in relation with their level of experience.

In KMEP, the idea of RPG is great because this kind of game offers the possibility to add mini games in the principal game. This aspect is very interesting because we could add many different games in the game.

A.2.1.1. Player

In RPG games, we can make the difference between PC and NPC. PC is “player characters” and NPC is “non-player characters”.

A.2.1.2. PC – Player Character

PCs are human players. It means that players can do some action in the game without any sort of scripting actions. Everybody can take this role.

A.2.1.3. NPC – Non Player Character

These players are virtual. Texts they used to speak are written and their actions are “scripted”. They are computer players. They can answer to a player when this player speak with them or do some interaction with it but all the time NPC are only in game and no in real world.

A.3. Sort of players

In the game, we can see several different kinds of players. We can look at players in terms of duration. In this situation, we can considerate: “short”, “medium” and “long” term players.

We must to keep in mind that these terms are informational and not an absolute way to classify players. We must distinguish the playtime against available time. Playtime is the time to spend in the game and available time is the time that players have to play.

A.3.1. Short term

This kind of player is players that came just for a while. Some tourists come to visit Kyoto for a few days. Duration is rarely more than two or three weeks. It is a maximum.

A.3.2. Medium term

These players are coming for a longer period. We can consider that these players do not stay more few months. This kind of player is in the middle. Some of them are considered like short term and others like long term.

A.3.3. Long term

Players who are staying for a (very) long time like six months or more are considered as long term players. Their waiting of game are not the same as the others kinds of players.

A.4. Avatar

Player will be represented by an avatar. This is the representation of the player in the game. Avatar could be chosen between lots of different avatars. This is a simple picture maybe in manga style.

This concept is only for a visual representation in game for players. A representation of players in game can be helpful for interactions between players. When players see others players near them, they can talk with them (just a possibility of interaction).

Avatars are also used for the representation of NPC. It is important for players that they can see PC and NPC in the game.

A.5. Geo-localization

The geo-localization technology offers a great experience to the game. With localization, we are able to start some particular events based on the player position in real life.

For example, a player can take place on a specific temple can see some NPC and/or PC but when he quits this place, he cannot see them.

We can imagine a lot of application to increase the game experience with this “simple” mechanism. We can imagine some scenarios or mini games based on localization. All the game can take place in a geo-localization system.

A.5.1. Bicycle

Bicycle is the preferred method of transportation during the game. One of goals of the game is to use Bicycle for moves in the game but it can be difficult to check if players use a bicycle.

In a first time, we must just to concentrate on the game without too more cheat aspects. We can easily find some hints to force player to do what they have to do but to be sure that they use bicycle will be more difficult. We can just check if players move along a predefined path but players can follow a path on feet or on car.

We have to encourage players to use bicycle to accomplish missions in game. It is one part of exergaming principle used in the game. If players go on feet to play, it is not so bad as well.

A.6. Base game

The game wants to be accessible by all kind of players. In this situation, we must find a simple mechanism that allows all types of play. The game offers two game modes. We will discuss differences in few paragraphs.

A.6.1. Registration

In both modes, a registration is required to play the game. It is necessary for the persistence for player's data. The simple facts that players can be authenticated (in example) provide the ability to create an avatar with specifics abilities and a progression in game.

Avatars of players can be stored in the system. When a player stops the game, he can restore his state because all information is stored. For the player, the story can be continued each time he connects to the game.

A.6.1.1. Starting Game

Both modes start with the same manner. A quest is proposed to the player. The quest consists to reach a specific place where some NPC are placed to help player with other quests.

The first quest is important to guide players who discover the game. It is avatar's player, which guides the player to do his first quest. The player can accept or refuse this first quest.

A.6.2. Games mode

One mode oriented for short and medium term players and one another for medium and long term.

A.6.2.1. Simple mode

In this mode, the player can choose an avatar to represent him in the game. He could do all activities proposed by the game, which offer some cultural interests and guided tour of

Kyoto. The goal of this mode is to propose to players an original way to visit the City of Kyoto and discover some historical facts and stories about the city. The players have to move in the city to reach some historical places organized like a guided tour.

NPC in the game can participate to describe the places with some real interesting details of each place. With this orientation, we can use a narrative form to guide players in the city. It could be more pleasant to players. The goal is to give to player an immersion in the game.

Figure 130 shows an example of a player who run the simple mode. The player talks to NPC 1 and this NPC tells him that he has to go to NPC 2. NPC 2 tells to player some details about place A and B. After his story, NPC 2 tells to player that he has to go to NPC 3. The player continues to go from NPC to NPC.

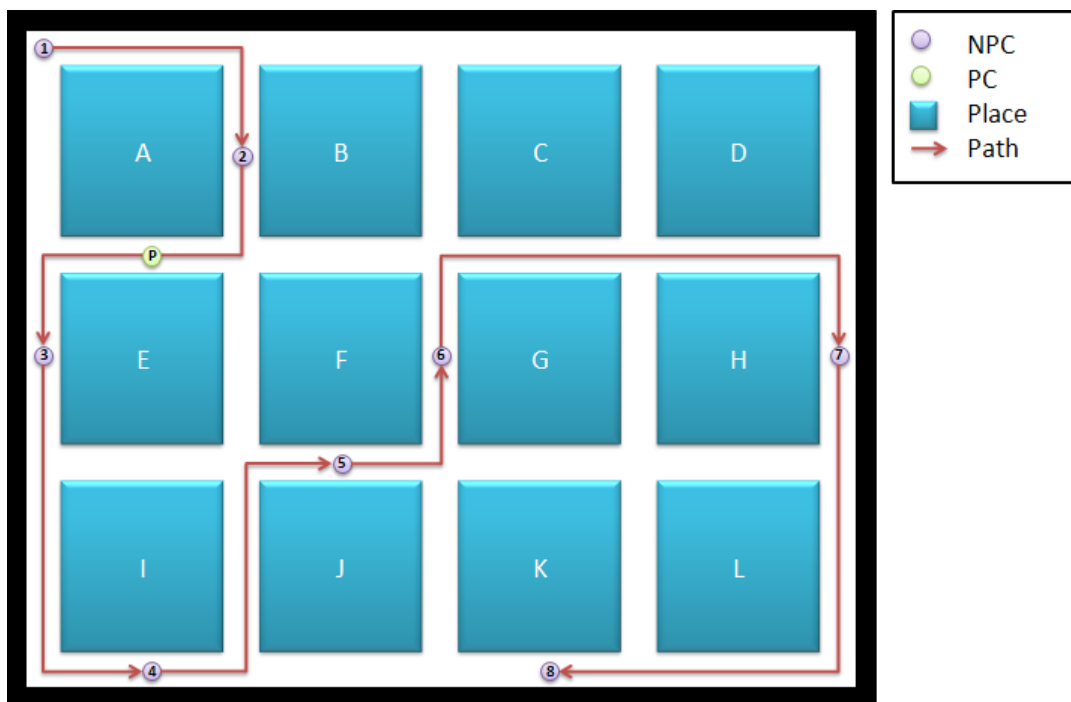


Figure 130 : Simple game mode map sample

In this mode, players do not want to gain experience or other things used in quest mode (the second game mode). Players just want to discover the city. When a player finished a “quest” in simple mode, NPC can propose others quests.

One problem of this mode is to define starting points. These points have to be known by players. Maybe we can introduce specials NPC that re-route players to the starting points.

Transitions between NPC are very dependent of our imagination. We can imagine some little task to give to players like bring a packet to the next NPC or some tasks like this. It is not the central point in this mode but it can add a better immersion to players.

One other thing to consider is the persistence. A player who has already done a quest did not want to redo the quest. In this situation, we must to remember which player did what. It could be preferable this process would be transparent to players. Short players must to be registered like long-term players to remember their actions and game state.

Long term and short-term players do not play for the same motivations. Some mechanisms are hidden for simple mode.

For this reason, we can easily combine the two modes for game mechanisms and just hide “advanced” options to simple players.

A.7.1. Characteristics

When players choose an avatar, some characteristics are automatically configured for their avatar. The Table 1 shows these different characteristics.

Name	Description
(HP) Health	Health points permit to play in the game. When a player lost all his health points, he dies in the game. He must to do something special to recover his health. This aspect will be described later.
(XP) Experience	Experience is the progression of the player in the game. It permits to gain levels. Experience is usually an exponential progression. More the level is high and more the progression is slow.
(LVL) Level	Level represents the degree of progression of players. This characteristic allows restricting some quests or items for specific levels. This is also an indicator for the difficulty when a player meets a monster. This is a good way to know player chance to win against a monster.
(STR) Strength	This characteristic permits to combat monsters in the game. It is useful to bit monsters. It is used in combat resolution.
(STA) Stamina	Like Strength, this characteristic is useful against monster. This is for defense in combats.
(LCK) Luck	Luck can help player to find rare items during his moves. This is also used when a player win a combat and obtain items. Items collected after a combat depends on the chance.
(CHA) Charisma	This characteristic is useful with sellers. This permits to decrease prices of items in shops.
(KM) KMEP	This characteristic is just the money game. It permits players to buy items like potion or other else.

Table 1 : Player Characteristics

These characteristics will be improved when players gain in experience. During the game, players have to accomplished quests that permits experience progression for players.

Players have a level in the game. When a player has sufficient experience points, his level grows up. A consequence is that his characteristics grow up too.

NPC have also these characteristics. In first time, these characteristics do not be useful. At this time, it is only in for a better conceptualization. Not to make the difference is certainly easier to realize than the opposite. Monsters follow the same idea.

A.7.1.1. Characteristics defining

It is important to know how characteristics are defined. This is useful for developers but it must be secret for players. It is a mystery part of a game. Not all is know by the players.

When a player is created, his characteristics will be defined at random in some ranges. After the start, the progression of players depends on what they do. The calculation of progression is dependent of players' actions. NPC and monsters follow these rules. A level is configured for them and characteristics calculation is done.

Determination of different values is based on basics calculation at this time. In later version, we can imagine to modify the calculation to refine balance of values.

Monsters and NPC have the possibility to evaluate with the same rules as players. Now we have to discuss the calculation of different values.

HP

Health points are the calculation of: $\sqrt{STR^2 * STA^2}$.

XP

Experience is not the solution of calculation but of an addition of experience that was given by monsters or quests. After a fight, players obtain some XP points, which are added to his XP points.

LVL

Levels grow up when players gain experience. The progression is organized in stages. The calculation for stages is done like this: $round(lastXPProgression + nextLVL^{coef})$. lastXPProgression is the XP points necessary to reach the current LVL, nextLVL is the LVL to reach and coef is a defined value. Actually, the coef is 2.1.

STR

Strength progress when player hit a monster. A count of his hits is maintained for each players (monster and NPC too). This formula is used to define if player gain a STR point or not: $round(nextSTR^{coef})$. The nextSTR is STR to reach and coef is a defined value. Actually, the coef is 2.1. The counter is compared to the result of formula to know if a point is gain or not.

A condition is used to define if players gain hits points or not. Players must have their $\text{currentSTR} \leq \text{currentLVL} + 5$. When current STR does not grow up, hits points do not count.

STA

STA progresses like STR points. The difference is we count hits received by monsters. Conditions are the same. The formula is: $\text{round}(\text{nextSTA}^{\text{coef}})$. The nextSTA is STA to reach and coef is a defined value. Actually, the coef is 2.1.

LCK

The formula used for the progression of LCK is noticeably the same as STA or STR: $\text{round}(\text{nextLCK}^{\text{coef}})$. The nextLCK is the next step in progression of LCK and coef is a defined value. Actually, the coef is 1.7. The rule that LCK must be $\leq \text{currentLVL} + 5$ is also applicable.

To gain points in LCK, the player must find items on his path. He obtains some points when a combat is run. During a combat, the player will probably dodge some strokes. In this case, the player gain points for his LCK progression.

CHA

When players purchase items to a seller, he gains some points to progress his CHA. We used also the same formula for this progression: $\text{round}(\text{nextCHA}^{\text{coef}})$. The nextCHA is next step of progression and coef is a defined value. Actually, this value is 1.3. The rule that CHA must be $\leq \text{currentLVL} + 5$ is also applicable. Doing quests also permit to grow up the CHA with same rules explained before.

KM

The money can be obtained when a monster is defeated. Players can obtain money with his sales or with luck; he can find some money on his path. Another way to obtain money is when NPC give to players some KMEP.

Distance

This characteristic is just the statistic of the distance that players did in the game. Each meter is counted

A.7.1.2. Precisions

The progression of HP, STR, STA, LCK and CHA do not considerate the equipment (cf. next chapter). It means that the progression is based on original value before applying modifiers. Modifiers are applied after all progression.

A.7.2. Inventory

During a game, players can collect and use items. These items can help the player to progress in the game. A color code is available to differentiate items from each other's. Inventory has no size limit. Players collect add items as much as they want.

Items could be found on monsters after a fight. Some NPC sell items. Some items are specially placed on the map for quests purpose.

They are four item types:

- Normal
- Equipment
- Quest
- Special

Normal items

The items in the game are just an addition and representation of virtual items. This is useful to help the players to accomplish their quests. Items are generally persistent. It means that an item does not disappear since there is no specific action for that (give to a NPC for example).

When an item is fund by a player, this item become persistent. That means item persists in the game since it is used, sold or something else that causes the disappearance of item. Some items are already persistent when players collect them.

An item can be present in more than one category. Categories are principally used for filters.

Normal items (black items)

The normal type is all objects that the player can use all the time (depending on some conditions). These items can help player to give back some health points in example. In short, these items are relatively easily to find.

Equipment items (green items)

These items help players to protect them and improve their characteristics. Applied modifiers are permanent since the player remove or changed his current equipment. These items are more difficult to find than normal items. They are available only in Quest Mode for the purpose of equipment.

Quest items (orange items)

Quest items are especially used for quest. This is just to separate the view of these items to the others items. Sometimes they can be only quests items and they do not be use for other purpose and sometimes they are normal items (or other type) that can be use normally.

Special items (purple items)

These items are particularly interesting because they are persistent. A player can find a special item during his path or by other manners. He decides to sell it to a NPC. The NPC add this special item to his sell list. Another player buys this item and later in the game drops it on the road. Another can find this item and keep it. We can save the history path of the object. The difference with other items is there is no way to destroy item or something like that. Therefore, these items persist always in the game.

An interesting way is to define these special items are “not usable” for any kind of things. This purpose adds mystery to the game. It will create some discussions about these items and add some social interaction.

A great example is .Hack//Roots or .Hack//Sign with the “Twilight Key”. In this manga – represents an MMORPG universe – this item were never seen by any player and nobody knows exactly what sort of item it is. Many players try to learn more about it.

A.7.2.1. Unique items

We can imagine adding unique item in the game. This is a great way to add competition between players. Unique items are a mixed of described items before. They can be classified in one or more category but they have a status that they are present only one time in the game.

A.7.2.2. Possible actions

Players have some possibilities with items. They can buy/sell, collect, drop, use, exchange or shake items.

Buy/Sell

Some NPC hold store, which players can buy or sell some items. This is a possibility to make place in their inventory. Generally, equipment and normal items are separated in two different kinds of shops.

Collect

Players can find some items on their path. They can collect and/or use them.

Leave

Players can take care that their inventory for a better navigation. Players have the possibility to leave on the map one or more items.

Exchange

Players can exchange items with others players. This possibility is interesting to add some interaction between PC. All items can be exchanged.

A great way to do this kind of exchanges is to use peer-to-peer connections between players. For that, players must be near together. We have just to track peer-to-peer connection between players to consider that they can exchange their items or money. It forces players to meet them together for social interaction. This kind of detection can be used for general player's interaction.

Shake

Some actions can be done by shaking an item. Depending on the kind of item, different behavior can be produce.

For example, a simple item can hide a more precious item. Maybe the player will be lucky and when he shakes his collected item, he can find a more powerful or useful item than collected item.

Another example is that shacking item can explode. In this situation, item shacked is lost and players encounters some damage (in Quest Mode only).

A.7.2.3. Persistence

As we discuss above in this document, items are persistent. However, we can now discuss about how to place items in the game. A few ways will be used for that:

- Predefined items
- Item sources
- Left items
- Rewards

Predefined items

These items are configured to be placed at specific places in the game. This sort of placement can be useful for unique items.

Item sources

We can see item sources like a factory of items. That permits to place again some items with an interval of time between two items taken.

Left items

These items are placed when a player left an item from his inventory or after a fight. Other players can take left items. It can be a second solution to exchange items.

Rewards

Items can be produced when a rewards is needed. After quests, fights or during simple talk to NPC.

A.7.3. Monsters

Monsters are interesting addition in this sort of game. They produce to player some actions to do and challenge. For the simple mode, monsters are not visible and they do not interact with players.

They have same characteristics as players. This is normal if we want to compare PC/NPC during combat sequences or other interactions. In this way, they are also persistent in the game. They have a speed characteristic in addition to permit moves.

Monsters are visible on the map like other players or NPC. When player is near the monster, a fight begins. The player can:

- Escape the fight
- Attack the monster
- Try to steal item/money on the monster
- Use an item of his inventory

Combats are solved on turn-by-turn system. With this principle, the player is not obligated to do immediately an action. He can stop his way if he wants or continue since he reaches his next destination. The combat will be solved when the player will do his actions. If the player continues his way, the monster follows him.

If player go away faster than a monster, he has a chance to escape the fight without engage the monster. The monster returns to its area if its moves are constraint.

When a combat is engaged and not solved, other nearest monster can attack the player but these attacks are solved after the first fight. Players can try to avoid meeting monsters. Monsters have not to be in nearest area of player to avoid the fight.

When a monster is engaged in a fight with a player, it cannot see other player in its interaction area. Other player cannot engage a combat with this monster.

Escape the combat

When a player encounters a combat, he can try to escape the combat. It is not possible all the time. Sometimes the player fails to escape a combat and he must to fight the monster.

Attack the monster

Players can attack the monsters. This means player can strike monsters. Monster can also strike the player. Each character (monsters and players) can make a strike when it is their turn to play.

Accelerometers can be used to add some real actions into the combat. In example, player can shake his mobile device to kick his opponent. Depending on how he has shaken his device, the kick can be more powerful or not.

In the same time, the player has the possibility to dodge a kick from his opponent. In this situation, he has to shake his device from left to right and vice versa. With this action, he has luck to dodge the kick.

Steal an item

Instead of attack monsters, player can try to steal an item on monster. The player tries to steal some items on monster. If player succeed in doing his robbery, he gains the stolen item. This is a possibility to obtain some rare items, which are not sold in shops. Money can also be obtained in place of item.

Use an inventory item

Players can use items during a fight instead of attack or stealing an item. The use of an item provides some combat modifier. In example, a player can use a potion to retrieve some health points or he can use a grenade to strike the monster.

A.7.3.1. Remarks

All this actions are considerate as turn action. A player can make only one action per turn. Monsters run the same rules and can use only attack or steal action. The probability of using steal action is less than attack action. This solution is a simplification.

Using items by monsters is difficult because we have to create a very simple “artificial intelligence” that permits to take decisions on which item to use and when use it. This mechanism can be implemented in later version. Stealing items will not be provided in the first version of the game for monsters.

A.7.3.2. Combat resolution

Now we know how combat are occurring but we do not know how they are solved. In this section, we explain the mechanisms of combat resolution.

The fighter who is his turn, has the choice between four actions (attack, steal, flee and use an item) but monster can only use attack.

Attack resolution

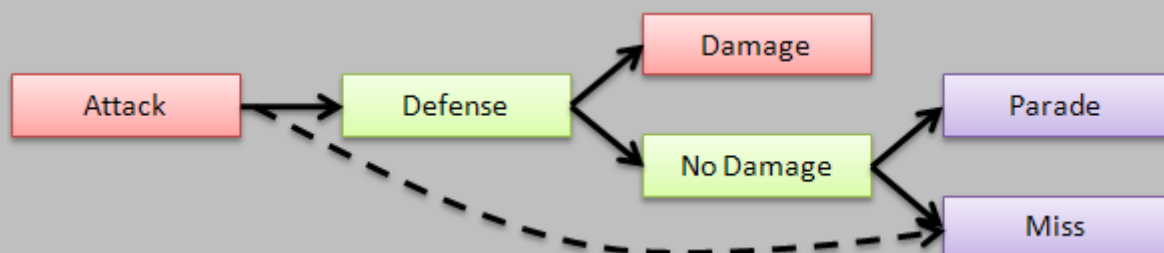


Figure 132 : Attack flow

The Figure 132 does not show theft, escape or item resolution. These resolutions will be approached after Attack resolution.

Attack resolution follows these rules:

1. A roll dice will be done to determine if the attack success or fail. The roll is calculated on the base of this formula: $defenderLCK - attackerLCK$. The roll must be under the result of the previous formula. If the attack fails, it is a miss from attacker. Next turn can begin.
2. Attack points calculation: $attackerSTR * attackerLVL$.
3. Defense points calculation: $defenderSTA * defenderLVL$.
4. Comparison between 2 and 3.
 - a. $2 > 3$ will produce damage points. Damage points will be calculated on this base: $2 - 3$.
 - b. $2 \leq 3$ will produce a parade and no damage.
5. Check if defender has sufficient HP to continue the fight. If he has no more HP, the fight finishes. Otherwise, the fight continues with the next opponent turn.

Theft resolution

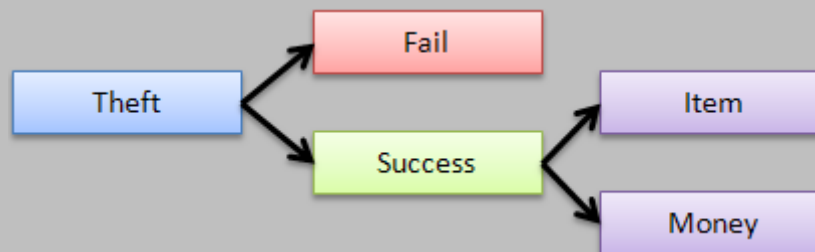


Figure 133 : Theft flow

The Figure 133 shows the flow of a theft resolution. Player tries to steal item on monster and has some luck to success. When he successes, he can have some luck to win money or item.

1. Attacker theft points are calculated: $attackerLCK * attackerLVL$.
2. Defender theft points are calculated: $defenderLCK * defenderLVL$.
3. Comparison between 1 and 2.
 - a. $1 \leq 2$ produces a failure.
 - b. $1 > 2$ produces a success. For the choice between money and item, there is 50% of chance each one.
 - c.
4. After the resolution, the next turn of the opponent begins.

Escape resolution

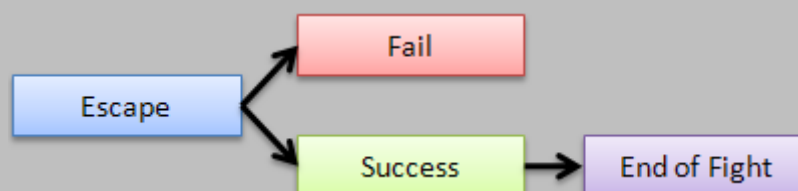


Figure 134 : Escape flow

The Figure 134 shows the flow of an escape resolution. Player tries to escape from the fight. If he succeeds, the fight finishes otherwise, the fight continues with the next turn of the opponent.

1. Player escape points are calculated: $random + (MonsterLVL - PlayerLVL)$
2. Monster escape points are calculated: $MonsterLVL$
3. Comparison between 1 and 2.
 - a. $1 > 2$ produce a failure. The fight continues with the next opponent turn.
 - b. $1 \leq 2$ produce a success. In this case, the fight finishes and player can continue the game.

Item resolution

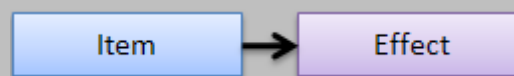


Figure 135 : Item flow

The Figure 135 shows the flow of an item use. Player uses an item on monster or on himself.

1. Item is used.
 - a. Effect of item will be calculated and applied based on target's level and item's level.

A.7.4. Combat flow

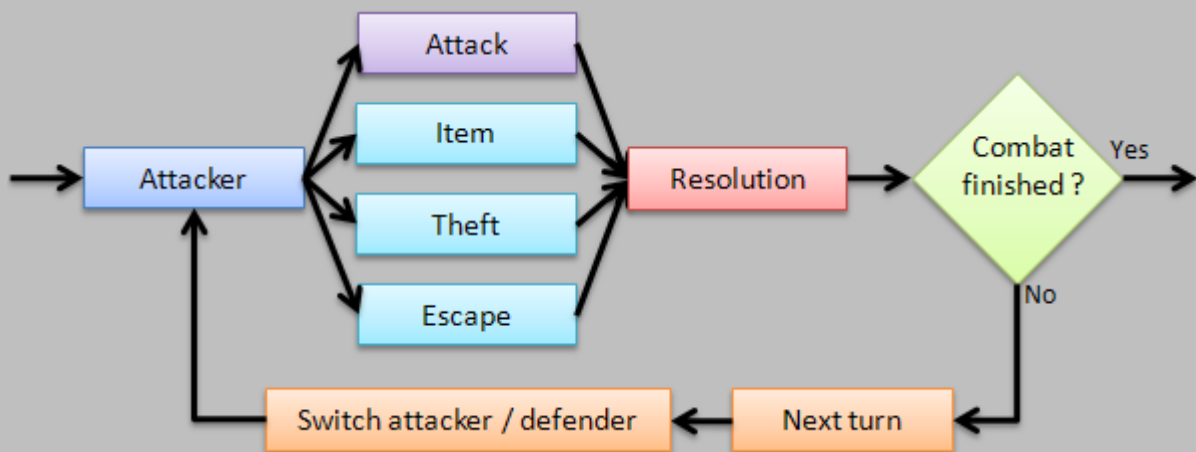


Figure 136 : Combat flow

The Figure 136 shows the combat flow. It indicates the attacker's choices but not all choices are available to monsters. The light blue is only available for players. The combat will finish only when player escape the fight or when player or monster kill his opponent.

A.7.4.1. Player's death

When a player died, two options are available to the player. We will approach these two options:

1. Item to resurrect.

2. Go to a sanctuary to resurrect.

The resurrection of player provides some malus to players. First, HP will be filled to a max of 10% of initial amount of HP. Some others malus will be applied. These modifiers are always applied whatever the chosen method of resurrection.

In the state of death, the player is like a ghost and does not do anything since his resurrection. It means that no fight, no dialog with NPC, no use of item (except resurrect items)...

Item to resurrect

An item is dedicated to facilitate the life of players. This item permits to resurrect without going to a sanctuary.

Sanctuary

Sanctuaries are special places in the game that permits to players to resurrect. These places protect players against monster.

A.7.5. Non Player Characters

As we discuss above in this document, NPC are players controlled by computer. They add possibilities of interaction with players. They can:

- Do a dialog with player
- Offer item
- Offer quest
- Sell/Buy items (Merchants only)

Dialog

NPC can speak with player. Dialog with player can be a mixed of sentences and questions/answers. This is the basic interaction with players.

Item

During a dialog, NPC can offer some items to player. These offers are include in a normal dialog. This is a “special” case of a dialog.

Quest

Like item, NPC can offers quests to the players. This also a “special” case included in normal dialog.

Sell/Buy

Some NPCs can be merchants and can do some trade with players. The beginning of a trade is like normal dialog. That permits a normal dialog introduce the possible actions sell/buy item to the merchant.

A.7.6. Non human moves

NPC and monsters can moves in the game. There are three types of moves for non-controlled entities:

1. Stay
2. Random
 - a. Limited
 - b. Non-limited
3. Predefined way
 - a. Loop
 - b. Simple path

These different moves permit to add more life in the game. Non-human entities gain the possibility to do not stay all the time at the same place.

Stay

This type indicates that entities do not move and stay all the time at the same place.

Random

Random moves allow entities to moves everywhere in the game area. Two different random moves are available.

The limited random move is constraint to a specific area. Entity cannot go away from the configured area.

Non-limited random move is non-constraint. So entities configured with this type of moves can go everywhere in the game area without limits.

Predefined way

This kind of moves allows configuring a specific path that the entity must to follow. Two types of configuration are allowed.

Loop configuration makes that the entity repeats his paths permanently during the game. Begin point and end point of the path are the same.

Simple path configuration makes that the entity follows his path and when it finished, it comes back to its starting points. It follows the reverse path.

A.8. Quests

Quests are the centre of the game. It is the most important and attractive pole of the game. Quests are available in the both type of play but they provide not the same things for the different kinds of play.

Simple mode

In this mode, not all quests are available. Only cultural and useful quests are available. This mode does not offer rewards at the end of quests like other mode.

Quest mode

All the quests are available in this mode with all game mechanisms. No restriction will be introduced to this game mode. Players can do any quests they want.

All the quests are available for the players. There is no more distinction between game modes.

A.8.1. Organization

The specifications of quests are organized in few different sections. Each section has its specific role in the quest.

Description

Description gives to player some general information and story about the quest. This is the general context of a quest.

Parts

A quest can be break up into few parts. Parts are dependent on the part before it. A part must be done before the next part could be active.

A quest could be break into multiple parts but to do that, this is necessary to build more than one quest and link them with a story manner.

Materials

The materials are what the player receives to help him to do his quests. It could be some items for example.

Objectives (Conditions)

This is the summary of what the player must to do to accomplish his quest. Objectives can change during the quest in regards of happening events.

Rewards

It informs the player of what he can win when he finishes his quest. For multi-part quests, rewards can be obtained after each part because of the quest construction.

Places

A lot of quest need that the player reaches some place to discover, find or anything else. In this case, quests have to indicate the next place to reach.

To indicate the places, the description can be use. A textual description of the place to reach can bring the same like a specific part of the quest construction.

A.8.2. Quest diary

The quest diary shows to the player his current quests. The old quest could be viewed with specific filter. Other filters are possible.

Current (In Progress)

Current quests are the quests that players are currently doing. These quests are not finished.

History (Finished)

Quests present in history are quests that players have already finished with success.

Discarded

Discarded quests are quests that players have decided to discard. They are not finished and not active.

Failed

Quests failed are the quests that the players cannot reach objectives fixed by quests. Players can redo quests.

Pre-Finished

Pre-Finished quest state is for the quest that are finished but not already validated by a NPC dialog. This is particularly useful to do the finish process for a quest (give rewards, take items asked ...).

A.8.3. Conditions

To run a quest, some conditions must be satisfied. In example, some quests are restricted to some levels. Others conditions can be imagine. One other condition can be that to do a quest before another quest will be available. This is already the case for multi-part quests.

A.8.4. Time

A great dimension can be the use of time in two directions. One direction is to constraint some quests to certain duration and the second direction is to constraint some quests to a certain date for its availability. It permits to create special events.

A.8.4.1. Relative

In this kind of quests, players have a dead line based on the actual time and date to accomplish his quest. Passed the dead line, quest is failed. Players can redo the quest if they want.

A.8.4.2. Absolute

In this kind of quests, players have a dead line based on the begin time and date of the quest to accomplish his quest. Passed the dead line, quest is failed. Players cannot redo this quest if they fail it because the quest date is passed. The quest is no longer available when the quest date is passed.

A.8.5. Giving the quests

Players have the possibility to give quests to other players. Idea of giving quests is to permit to players to do some quests together without the constraint to find the good NPC to obtain the quest. It is a shortcut to obtain some quests.

A.8.6. Grouping

The grouping system is not technical. The context of some quest give the possibility to the players to work together to solve the quest. This approach is more intuitive that a technical one. Players do not have to do special manipulations to play together.

It can be friendly to do some quests with other players. Idea is to permit players to group together to do a quest. It allows also decreasing the difficulty or the longer of quests.

To group with others players for a quest, some conditions must to be respected. The players must to own the same quest if they do not own the quest, the quest is automatically added to quests diary.

By example, two players grouping in one team to do a quest have to collect different items at different places. Each player can go to different place and collect items. The quest finish when ending conditions are accomplished but the evaluation is based on all collected items by all players in the team.

The rewards of a quest are the same for all players. Each player in a group receives the same rewards provided by the quest.

To group for a quest, players must be near to other players. The same mechanism as exchanging items can be used to detect nearest players.

Different actions can be done on groups depending if the player has the ownership of group or not.

A player can encounter two situations. First case is that the player creates a group and takes the ownership of the group. The second case is when the player is invited to group and he joins the group.

A.8.6.1. Group ownership

Group's owner can manage the group. To manage the group, the players have to :

- Add Member
- Remove Member
- Dissolve Group

Add Member

This action permits to the owner to add a member to the current group. Added players can only be near the group's owner like exchange mechanism.

Remove Member

The owner of a group can remove members from his group. There is no special condition to do this action. The removed player is advertised that he was removed from the group.

Dissolve Group

Dissolving a group permits to the owner to leave the group and liberate all other grouping players. Like Remove Member, there is no special condition to run this action. Members are advertised that the group is dissolved.

A.8.6.2. Group Member

A group member does not do anything on group management. The only things that he can do are to leave the group or join a group when he is invited.

Join

Players can join a group when an owner's group invites him to join his group.

Leave

When a player is in a group for a quest, he can leave the group when he wants. He has just to leave the group. The owner's group is notified that a player has left the group.

A.9. Rewards system

Players can obtain some XP points, Money or HP points when he is walking (or biking). Idea is to add these points during players move. More the players are moving and more they win points and money.

For example, each meter that a player does in the game can be converted to 1KM added to the player's money, or 1HP added...

A.10. Map

An important point in the game is the geo-localization. For that, the player has a view of a map with his avatar representation to localize himself. NPC and PC (only in a defined perimeter) can be appearing on the map. The quests next places to reach are also shown to the player.

A.10.1. Fog of war

An interesting way of research can be the fog of war. In some games, the map is not shown entirely to the player. It is to the player to discover the map during his quests or discussion with NPC.

When a player discovers a part of the global map, this part will be revealed for all the game. This mechanism will only available to the quest mode. The simple mode has the entire map available at any times.

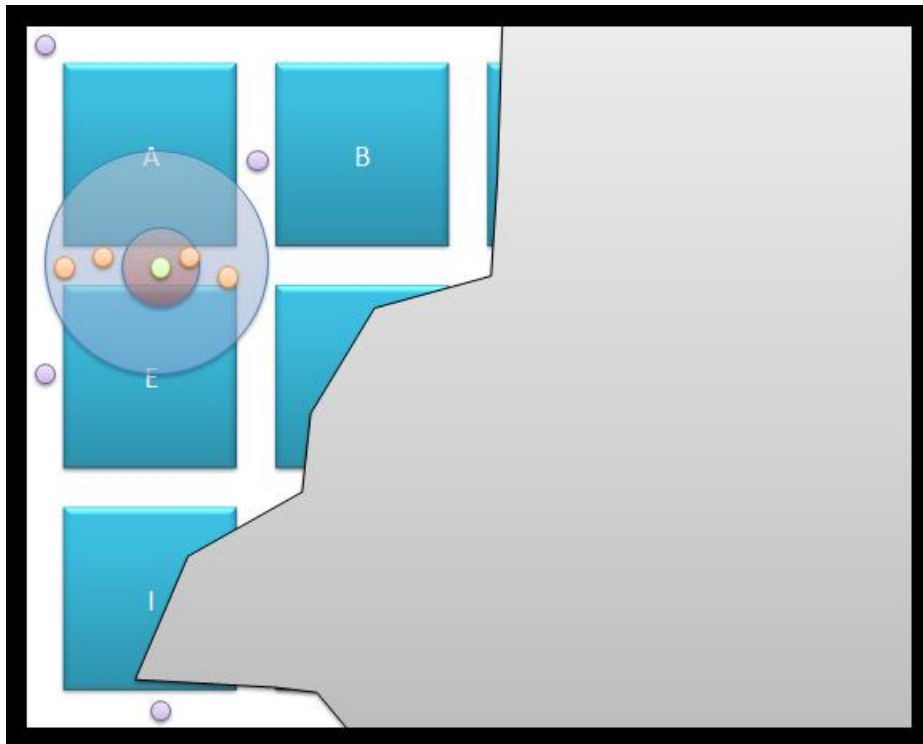


Figure 137 : Fog of war

The Figure 137 shows an example of map that is not entirely discovered by the player. Some places must be discovered by the player. The grey surface shows exactly what player must discover.

Green points represent the player, orange points represent other human players and violet points represent non-human players.

A.10.2. Information

Map contains a lot of useful information for players. Players can see other players with some additional information like quest grouping research, items near the player...

We can imagine also adding some information on the map like items, monsters and so on. The possibilities of information are very large but we cannot forget that the surface to show information is quite small.

Other information can be viewed when the player “click” on icons present on the map. In this case, a dialog shows some information on the selected element.

A.10.3. Player area

During the game, players have a certain field of interaction. They are three types of interaction/visibility:

Peer-to-peer

The peer-to-peer (red zone in Figure 137) is the nearest zone of influence of the player. This area defines how the player can interact with proximity objects. Outside this area, peer-to-peer interactions are not possible.

Radar

This area represents the view of other objects near him (blue zone in Figure 137). He can view human players relatively near him by example. This is a good indicator to locate other players in real/virtual world.

General

The general area is the global map. In this area, we can see some information about quest NPC or Items. Not all information is shown to the player only vital information.

This is also the area of interaction for a player. There is no more difference between Radar and General area.

A.10.4. Localization

There are many ways to do localization. The most known system is GPS. This is the base system that we will use to do the localization. We can imagine the use of other peripherals to do that.

Camera and code bars

This system is based on code bars recognition. Actually, some systems offer to capture a picture of a code bar on a publicity and send it to buy the shown article. In our game, it could be useful for indoor localization. By the way, it is necessary to add some code bars at strategic points. This method requires a few materials to install.

Wifi

If we are capable to know a place of wifi access point and if a mobile phone is connected to this access point, we can make localization of player. This method requires knowing where access points and some other information.

Bluetooth

It is not really a localization system but it can be helpful to localize nearest players. This technology could be used to add local localization. We can considerate this way like radar.

A.11. Alerting

Due to the nature of the game, we must alert players when events encountered. A way to do this is to use the vibration mechanism present on the mobile device. This is a good solution to alert the player without many inconveniences of sounds or something like that.

The alerting system is more like a message notification system. Due to the asynchronous system of the game, a message service is more efficient.

A.12. Conclusion

Finally, this appendix offers a view of game rules applied in very large kinds of situations. Quests are not described because each quest can be different from the others.

Mechanisms of quests can be generic but scenarios can differ for each quest. The resolution of game is a general platform that can offer a base to add some specific little games with quests.

heig-vd

Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

K-MEP

Specifications Review

Laurent Prévost



R RITSUMEIKAN

Table of Contents

B.1. INTRODUCTION	229
B.2. USE CASES	229
B.2.1. Start the game	231
B.2.2. Run the registration process	233
B.2.3. Connect to the game	236
B.2.4. Choose game mode	238
B.2.5. Run the first play	239
B.2.6. View map	242
B.2.7. View object details	246
B.2.8. View global menu	247
B.2.9. Quit game	248
B.2.10. View preferences menu	250
B.2.11. Modify profile	251
B.2.12. Modify avatar	253
B.2.13. View characteristics	255
B.2.14. View and modify equipment	256
B.2.15. View statistics	257
B.2.16. View and modify options	259
B.2.17. View inventory	260
B.2.18. View actions for an item	262
B.2.19. Leave item	264
B.2.20. Give item	266
B.2.21. Shake item	270
B.2.22. Use an item	272
B.2.23. Do a dialog with NPC	273
B.2.24. Have a quest proposal	278
B.2.25. Finish a quest	279
B.2.26. Make a trade with a merchant	281
B.2.27. View quests diary	283
B.2.28. View actions for a quest	285
B.2.29. Discard a quest	287
B.2.30. Give a quest	289
B.2.31. Manage Grouping buttons action	293
B.2.32. Create a group for a quest	294
B.2.33. Manage group	296
B.2.34. Dissolve a group	297
B.2.35. Remove member	298
B.2.36. Leave a group	299
B.2.37. Fight a monster	300
B.2.38. Add annotation	303

B.2.39.	Exchange item from the map	304
B.2.40.	View quests diary from map	306
B.2.41.	View actions for a quest from the map	307
B.2.42.	Equip from inventory	308
B.2.43.	Unequip from inventory	309
B.2.44.	Retry a quest	310
B.2.45.	Give a quest from the map	311
B.2.46.	Exchange item	313
B.3.	CONCLUSION	319

Table of Tables

TABLE 2 : USE CASES STATES.....	230
---------------------------------	-----

B.1. Introduction

This appendix offers a view of the development of the Use Cases. Each Use Case presented here comes with a little report of development and WUI screenshots.

B.2. Use Cases

In this section, all the Use Cases about player point of view are listed. There are all the necessary to develop the client part with interactivity.


The Table 2 shows the actual states of the different Use Cases. The lines in grey indicate Use Case cancelled during the project. The reasons are explained in the Use Case description. The text in red indicates new Use Cases or updates of the Use Case title.

#	Title	State
1	Start the game	100%
2	Run the registration process	100%
3	Connect to the game	100%
4	Choose the game mode	-
5	Run the first play	100%
6	View map	100%
7	View object details	-
8	View global menu	100%
9	Quit game	100%
10	View preferences menu	100%
11	Modify profile	100%
12	Modify avatar	100%
13	View characteristics	-
14	View and modify equipment	-
15	View statistics	100%
16	View and modify options	-
17	View inventory	100%
18	View action for an item	100%
19	Leave item	100%
20	Exchange Give item	100%
21	Shake item	-

#	Title	State
22	Use an item	-
23	Do a dialog with NPC	100%
24	Have a quest proposal	100%
25	Finish a quest	100%
26	Make a trade with a merchant	-
27	View quest diary	100%
28	View action for a quest	100%
29	Discard a quest	100%
30	Give a quest	100%
31	Manage group buttons action	-
32	Create a group for a quest	-
33	Manage group	-
34	Dissolve a group	-
35	Remove member	-
36	Leave a group	-
37	Fight a monster	-
38	Add annotation	-
39	Exchange item from the map	100%
40	View quest diary from the map	-
41	View actions for a quest from the map	-
42	Equip from inventory	-
43	Unequip from inventory	-
44	Retry a quest	100%
45	Give a quest from the map	100%
46	Exchange Item	100%

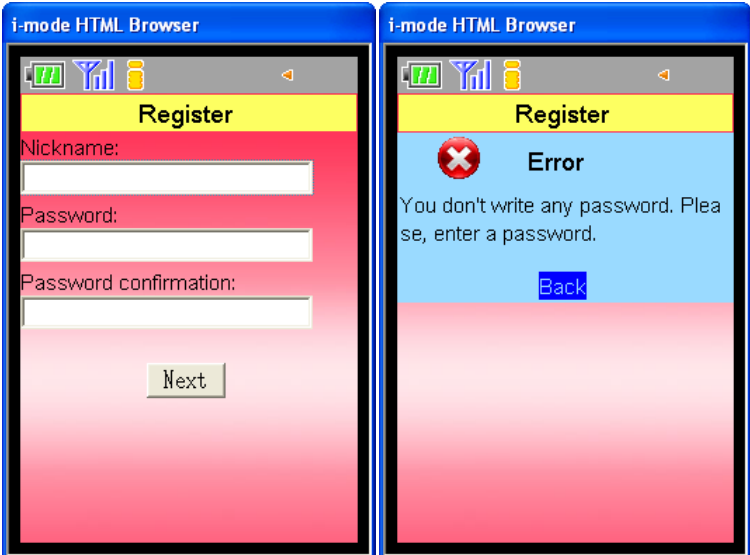
Table 2 : Use Cases States

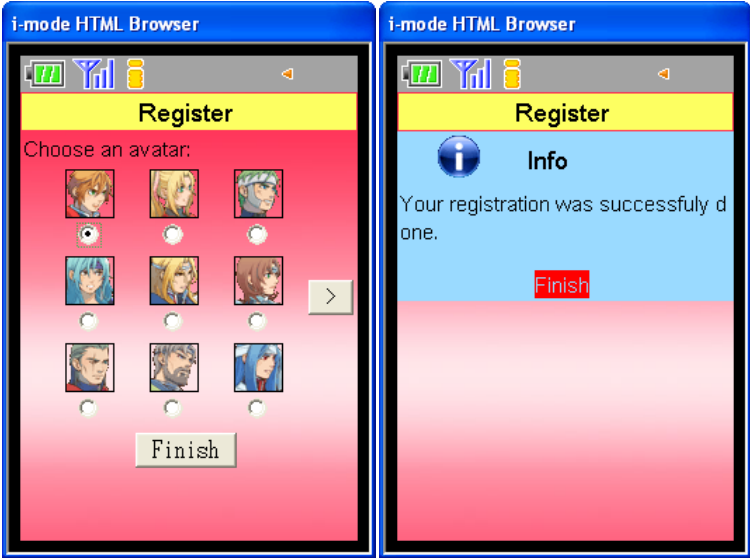
B.2.1. Start the game

UC Number	1
Label	Start the game
Depends	-
Priority	High
Complexity	Low
Description	Players have to run the game to view the first screen of the application and decide what they want to do.
Actors	Player
Running	<ol style="list-style-type: none"> 1) Run the game 2) Splash screen is shown 3) Auto-connect enabled. <ol style="list-style-type: none"> a. True → UC 4, Step 1 b. False → Step 4 4) Players see registration/connection screen <ol style="list-style-type: none"> a. Registration → UC 2, Step 1 b. Connection → UC 3, Step 1
Samples	
Remarks	
State	<p>08.09.2008</p> <p>Finished</p> <p>The splash screen is the first screen that the player sees. Due to the limitation of the iMode application (basic web interface), this is not possible to auto-connect the player. In this situation, point 3 is not implementable.</p> <p>10.09.2008</p> <p>A way to do the auto-connect is to use the cookies. This point must to be analyzed and integrated it if it is possible.</p>

	<p>11.09.2008</p> <p>Now, the cookies are supported. So, if the player chooses to auto-connect when he comes on the iMode site, it will be automatically connected to the game. Actually, this mechanism did not try on emulator. It seems to be not possible to use cookies with emulator. The test will be done directly on the mobile phone. At least, this point is not a problem for the normal process.</p> <p>21.09.2008</p> <p>Due to the implementation of the MVC model, the auto-connect is no more available. Cookies possibilities will be tested directly on real phone and will allow deciding how it is possible to do or not this kind of things. If it is possible to store some cookies, the auto-connect will be redone otherwise this functionality will be abandoned.</p> <p>09.10.2008</p> <p>A simple test on private server web was done. Two php pages were written. The first one starts the session and store a simple string. The second one shows the stored string to the user. With a normal browser like FireFox, it works but with the mobile phone, it does not work. It seems that the emulator has the same behavior. Actually, it not possible to use the session (jsessionid) stored in the cookies with the emulator. To correct the problem, a jsessionid is used directly in the URL with URLRewriting mechanism. The consequence is that it is not possible to store the user/password in a cookie to do the auto-connect. In this situation, this use case is considered as finished.</p>
--	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

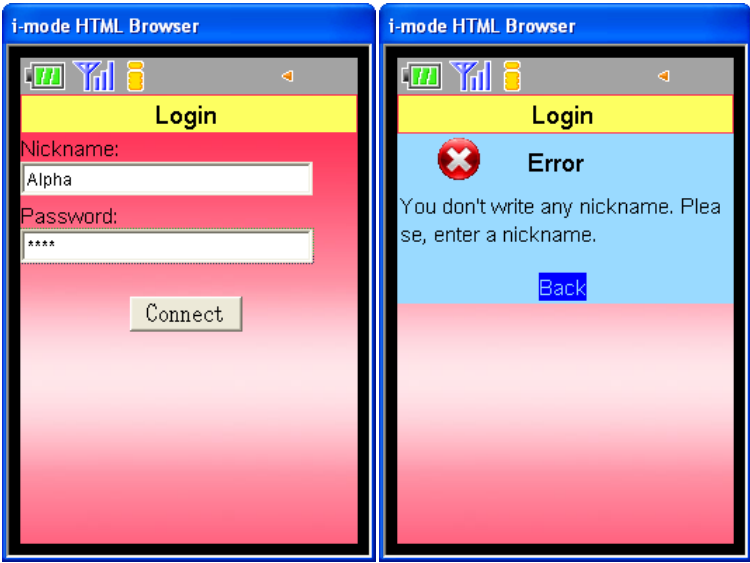
B.2.2. Run the registration process

UC Number	2
Label	Run the registration process
Depends	UC 1
Priority	High
Complexity	Medium
Description	The registration process is here to allow the persistence of players. It's the base of all actions executed in the game. The game mode is not significant at this time.
Actors	Player, Engine
Running	<ol style="list-style-type: none"> 1) Player enters Nickname, Password and Confirm Password and click Next 2) Verification of data validity (unique nickname, password ok) <ol style="list-style-type: none"> a. Ok → Step 2 b. Ko → Step 3 3) Error dialog is shown <ol style="list-style-type: none"> a. Ok → Step 1 4) Player chooses an avatar that represents itself and clicks Finish. 5) Registration process finished. → UC 4, Step 1
Samples	

	
Remarks	All next Use Cases depends on this Use Case. All players need to be registered and connected (see UC 3). Nicknames must be unique. Nicknames are identifiers.
State Finished	<p>08.09.2008 This Use Case is totally implemented. Some little bugs like empty nickname and password are possible actually. It would be corrected soon. There is also a problem when the same avatar is chosen. The Service refuse to change the avatar if this is the same than the previous. By the way, the servlet has to deal with this behavior.</p> <p>09.09.2008 The problem with empty password or/and nickname is solved. The problem with avatar is not corrected actually.</p> <p>10.09.2008 The problem for choosing the same avatar is corrected. Now, when a player chooses the same avatar, there is no change on the server but a message indicates to the player that the change is successfully done. The message shown just after the registration process (before the choice of avatar) is now optional depending on the configuration file.</p> <p>11.09.2008 The register message is added at the end of the process (after the choice of avatar).</p>

	<p>21.09.2008</p> <p>The implementation of the MVC does not affect this use case. Perhaps, some text in the WUI is different now. A little bug is corrected. The possibility to hash the password in the database before the call to the player manger is added but in the player manager, the comparison of password with a blank password was not correct. The blank password has also to be hashed for the comparison with a hashed password.</p>
--	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

B.2.3. Connect to the game

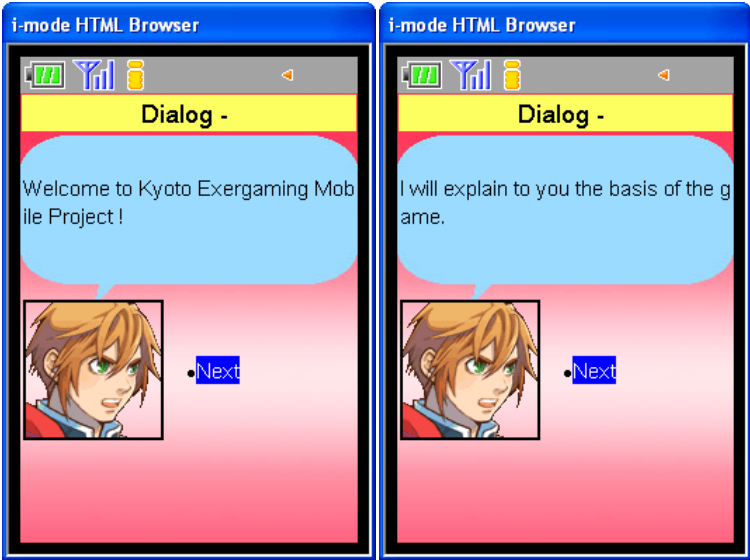
UC Number	3
Label	Connect to the game
Depends	UC 2
Priority	High
Complexity	Medium
Description	To play the game, players need to be connected. For this process, players need to be registered.
Actors	Player, Engine
Running	<ol style="list-style-type: none"> 1) Player gives his nickname and password and press Connect <ol style="list-style-type: none"> a. He can choose to remember his connection information to do not again the connection process. 2) Authentication <ol style="list-style-type: none"> a. Success → UC 4, Step 1, Step 4 b. Fail → Step 3 3) Error dialog is shown <ol style="list-style-type: none"> a. Ok → Step 1 4) First game? <ol style="list-style-type: none"> a. True → UC 5, Step 1 b. False → UC 6, Step 1
Samples	
Remarks	This Use Case depends on the registration Use Case. When player remember his connection information, the process of authentication is processed again but the player do not see it. He does not see the connection screen.

<p>State Finished</p>	<p>08.09.2008 The connection screen is ready but there is no possibility to implement the point 1.a due to the limitation of the iMode application.</p> <p>10.09.2008 Depending on the result of the analysis, the connection could be automated with cookies.</p> <p>11.09.2008 The checkbox is added to remember the player. This checkbox has effect to create cookies with name and password on the mobile phone. This not perfect in terms of security but the risk is limited. The message to confirm the connection is only available in debug mode.</p> <p>21.09.2008 No change due to the implementation of the MVC.</p> <p>23.11.2008 Cookies are not correctly available from the iMode mobile phones. The “remember me” option is not possible to implement easily. The previous implementation was tested only on a normal browser. There is some trouble with the standard used by the application sever and the iMode browser. The both technologies do not use the same standard for the cookies.</p>
----------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

B.2.4. Choose game mode

UC Number	4
Label	Choose game mode
Depends	UC 3
Priority	High
Complexity	Low
Description	When players are registered and connected, they have to choose the Game Mode they want. They can change the Game Mode during a play (See Preferences Use Cases).
Actors	Player, Engine
Running	 5) Game Mode memorized? c. True → Step 5 d. False → Step 2 6) Players see Game Mode choice e. Simple Mode → Step 3 f. Quest Mode → Step 3 7) Memorize CheckBox "remember mode" 8) Memorize Game Mode chosen 9) First game? g. True → UC 5, Step 1 h. False → UC 6, Step 1
Samples	
Remarks	-
State	21.09.2008 Cancelled After a discussion with professor Liechti, we decide to focalize on the simple game mode and just offer quests and items. In this situation, there is no more need to choose the game mode for the player.

B.2.5. Run the first play

UC Number	5
Label	Run the first play
Depends	UC 4, UC 23
Priority	High
Complexity	Medium
Description	The first run is important to explain some mechanisms to new players. For the two modes of the game, the first play is the same.
Actors	Player, NPC
Running	<ol style="list-style-type: none"> 1) Player sees a NPC representing his avatar. 2) NPC explains basic mechanism of quests 3) NPC proposes to the player a little quest to do. The goal of the quest for player is to reach a certain place to discover other quests. <ol style="list-style-type: none"> a. Accept → Step 4 b. Refuse → Step 5 4) Quest is added to Quest Diary (current category) 5) The discussion ended 6) Player sees the map view → UC 6
Samples	

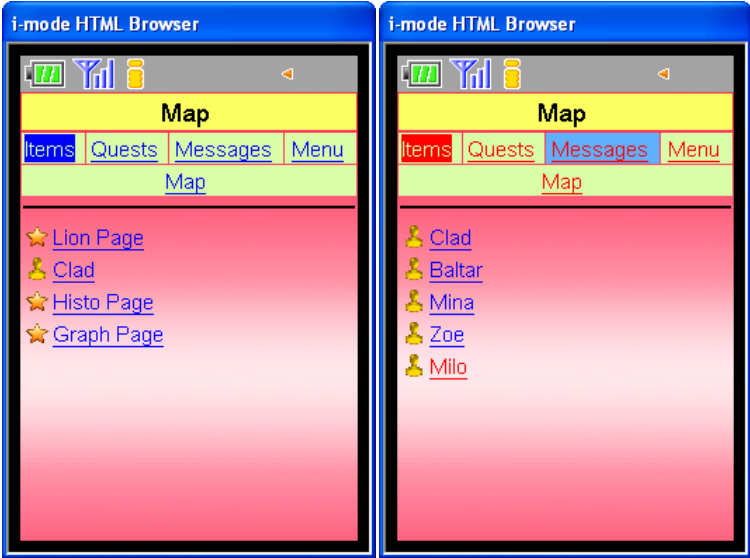
	 <p>The screenshots show a sequence of five dialog screens in an i-mode HTML Browser. Each screen has a blue header with 'i-mode HTML Browser' and a yellow bar with 'Dialog -'. The background is a pink-to-white gradient. A character portrait of a young man with orange hair is on the left of each screen.</p> <ul style="list-style-type: none"> Screen 1: Text: "To understand the quest game mechanism, I offer to you to accomplish a little quest." Buttons: Accept, Refuse. Screen 2: Text: "Are you sure ? You already known t he game ?" Buttons: Yes, No. Screen 3: Text: "You can see the details of your quest in your quest diary present on the map screen after this dialog." Button: Next. Screen 4: Text: "The quest diary offer different views to easily see the states of your quests. Try to find the quest I give you." Button: Next. Screen 5: Text: "Now, let's begin to play. Good luck !" Button: Finish.
<p>Remarks</p>	<p>For the Simple or Quest mode, the first play after registration is the same.</p>

<p>State Finished</p>	<p>08.09.2008 This Use Case is under development. Actually, it is possible to see a first dialog step. The NPC is represented by the player's avatar for the first dialog.</p> <p>10.09.2008 A quest could be proposed and the player can accept or refuse with a correct redirection on the next dialog step. Actually, the first quest must be set manually. The next step is to automate the choice of the nearest quest for the player. When the player accepts the quest, it is added to the quest diary with the current state of the quest.</p> <p>11.09.2008 The point 4 and 5 are implemented. Now the discussion can finish. After the dialog, the player arrives directly on the map.</p> <p>21.09.20098 The implementation of MVC does not cause any modification on this use case. Actually, there are some real pictures for avatars.</p>
----------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

B.2.6. View map

UC Number	6
Label	View map
Depends	UC 4
Priority	High
Complexity	High
Description	The map view permits to players to show a lot of information like their position, position of NPC, position of items, Quests direction . Players can access to the global menu from this screen.
Actors	Player, Engine
Running	<p>Part A:</p> <ol style="list-style-type: none"> 1) Player does an action <ol style="list-style-type: none"> a. Open principal menu → UC 8, Step 1 b. Open Inventory → UC 17, Step 1 c. Open Quest Diary → UC 27, Step 1 d. Move the map → Part B, Step 1 e. Zoom on the map → Part C, Step 1 f. Click on map icon → Part D, Step 1 g. Choose an element in the list → Part D, Step 1 h. Add annotation → UC 38, Step 1 <p>Part B:</p> <ol style="list-style-type: none"> 1) Player chooses a direction where to move the map. <ol style="list-style-type: none"> a. North → Step 2 b. South → Step 2 c. East → Step 2 d. West → Step 2 e. Centre → Step 3 2) Map is moved in the chosen direction → Part A, Step 1 3) The map is centered on the player's position <p>Part C:</p> <ol style="list-style-type: none"> 1) Player can: <ol style="list-style-type: none"> a. Zoom in b. Zoom out c. Restore zoom 2) Apply zoom choice → Part A, Step 1 <p>Part D:</p> <ol style="list-style-type: none"> 1) Map icon is near the player? <ol style="list-style-type: none"> a. True → Step 3

	<p>b. False → UC 7, Step 1</p> <p>2) Player Returns → Part A, Step 1</p> <p>1) Depending on the object type, the player can don an action with (This step replaces the UC 7):</p> <ul style="list-style-type: none">a. NPC → UC 23, Step 1 UC 24, Step 1b. Item → Part E, Step 1c. Player → Part F, Step 1d. Place → Step 2e. Monster → UC 38, Step 1f. Annotation → Part G, Step 1 <p>Part E:</p> <ul style="list-style-type: none">1) Player can choose an action<ul style="list-style-type: none">a. Take → Step 2b. Cancel → Part A, Step 12) Item is added to the player's inventory3) Item disappear from the map <p>Part F:</p> <ul style="list-style-type: none">1) Player can choose an action<ul style="list-style-type: none">a. Item → UC 39, Step 1b. Quest → UC 40, Step 1c. Cancel → Part A, Step 1 <p>Part G:</p> <ul style="list-style-type: none">1) Player sees a dialog with text annotation:<ul style="list-style-type: none">a. Ok → Part A, Step 1b. Suppress → Step 22) Player sees a confirmation dialog (if confirmation is required):<ul style="list-style-type: none">a. Yes → Step 3b. No → Step 13) Annotation is removed from the map → Part A, Step 1
--	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------


Samples	
Remarks	<p>Open the menu can be done by clicking on keyboard of the mobile device. Part B of this Use Case is depending on the mobile device for the possibility to control the moves of the map. If there is a touch screen, we can use it to move more naturally the map.</p> <p>A player is considered near a map icon when his first influence radius area contains the map icon considered.</p> <p>The second influence area is used to detect unknown map icon like other players, NPC, items or monsters.</p> <p>Map icon out of second influence area can be shown under certain conditions. Generally, this information is shown after the player known them.</p> <p>Monsters and annotations icons are not represented on the map dialog above.</p> <p>We must pay attention to the concurrency. This is important when two or more players try to take the same items.</p>
State Finished	<p>10.10.2008 The map is shown without information. The menu elements are usable. The menus Inventory, Quest Diary and Menu are present.</p> <p>10.11.2008 Some menu in the map screen are added. The menu added was Messages and Map. Messages allow showing messages list and Map allows refreshing the map elements. Inventory to Items and Quest Diary to Quests were renamed.</p> <p>14.11.2008 Actually, a list of item is shown with players and items. There is no NPC actually in the game. The next step is to add the interaction with the elements shown in the list.</p>

	<p>17.11.2008</p> <p>This Use Case changed a lot because of the technical problems encountered with the localization and the focusing on the Simple Game mode. There is no picture of the map because no possibility to use JavaScript on the mobile phone. There is a possibility to build the pictures on the server side but the time does not allow doing that.</p> <p>There is no more two area of influence for a player, now there is only one radius that the player can see and interacts with other game elements. This decision is to simplify the game for the simple game mode.</p> <p>The Use Case 7 is directly integrated in this Use Case due to the change of the two areas and other information changed here. The player can see directly details and action when he clicks on a map element.</p> <p>18.11.2008</p> <p>Actually, this possible to see the dialog when a player clicks on a NPC. This is also possible to see the player details when he clicks on a player. The player details show also the actions possible from the map. For example, from the player details, this is permit to do an action for a quest or an item.</p> <p>20.11.2008</p> <p>The last object type could be seen on the map. This item sources allow building multiple items during the game. When a player takes the item from the source, a timer is run to prepare the next drop of the item.</p>
--	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

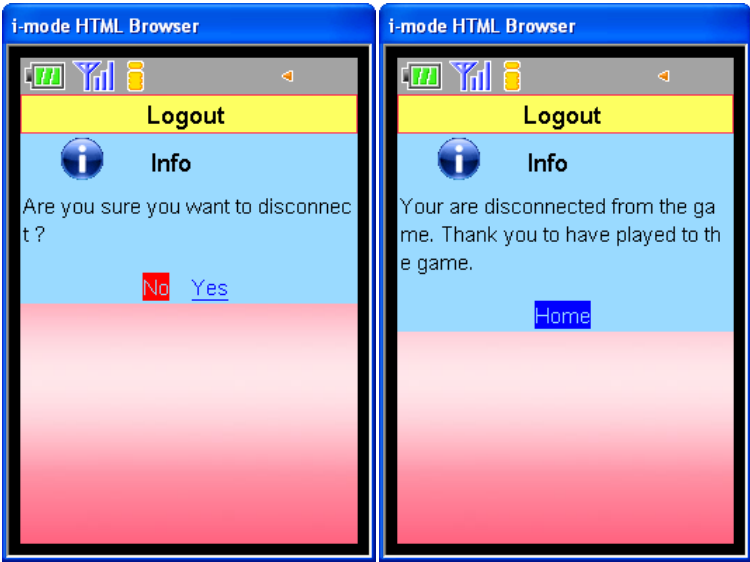
B.2.7. View object details

UC Number	7
Label	View object details
Depends	UC 6
Priority	Medium
Complexity	Low
Description	Players has the possibility to view object details (items, places, players, ...)
Actors	Player
Running	<ol style="list-style-type: none"> 1) Player screen state is memorized 2) Player sees dialog which shows details of item depending on object types: <ol style="list-style-type: none"> a. Item → Dialog 1 b. Place → Dialog 2 c. NPC → Dialog 3 d. Monster → Dialog 4 3) Player clicks on Ok → Step 4 4) He returns to screen memorized
Samples	
Remarks	-
State	17.11.2008
Replaced	Due to the simplification of the game (only Simple Game Mode), this Use Case can be done directly in the Use Case 6 – View map.

B.2.8. View global menu

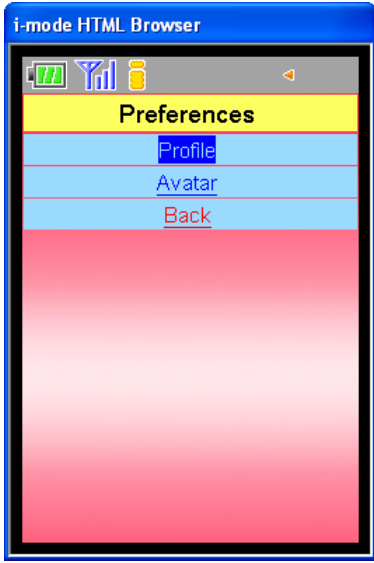
UC Number	8
Label	View global menu
Depends	UC 6
Priority	High
Complexity	Low
Description	To access to some parts of the game, players have to navigate in principal menu. Players can access to Quest Diary, Inventory or Preferences. They can also Quit the application.
Actors	Player
Running	<ol style="list-style-type: none"> 1) Player sees the principal menu 2) Player can access to different parts of the game from the principal menu. <ol style="list-style-type: none"> a. Inventory → UC 17, Step 1 b. Quests Diary → UC 27, Step 1 c. Preferences → UC 10, Step 1 d. Quit → UC 9, Step 1 e. Map → UC 6, Step 1
Samples	
Remarks	-
State Finished	22.09.2008 For a better navigation the Inventory and Quest Diary elements are available directly and only from the map screen. The global menu contains only [Preferences, Quit and Map].

B.2.9. Quit game

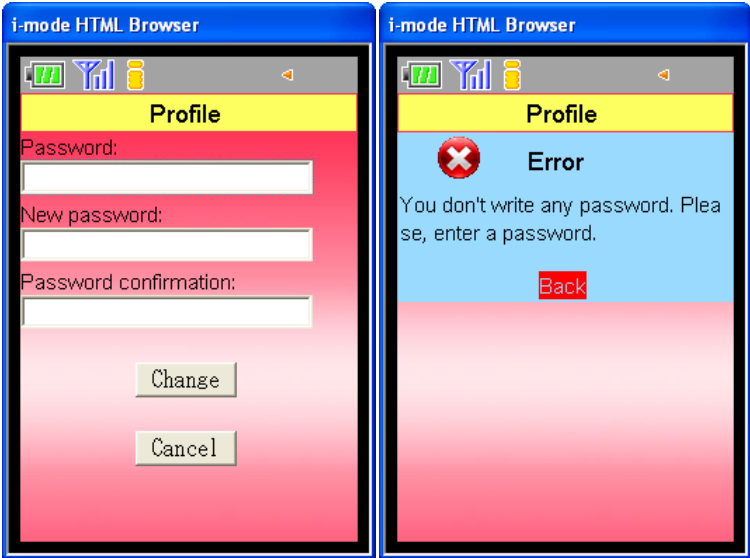
UC Number	9
Label	Quit game
Depends	UC 8
Priority	High
Complexity	Low
Description	Players can normally quit the game by a specific action. The normal way permits to engine to know that the player has left the game and finalize the player state (persistence).
Actors	Player, Engine
Running	<ol style="list-style-type: none"> 1) Showing the confirmation dialog <ol style="list-style-type: none"> a. True → Step 2 b. False → Step 4 2) Players see a confirmation dialog with remember option 3) Memorized remember option 4) Do the action corresponding to the player choice. <ol style="list-style-type: none"> a. Yes → Step 4 b. No → UC 8, Step 1 5) Notify engine player leave the game 6) Quit the application
Samples	
Remarks	-

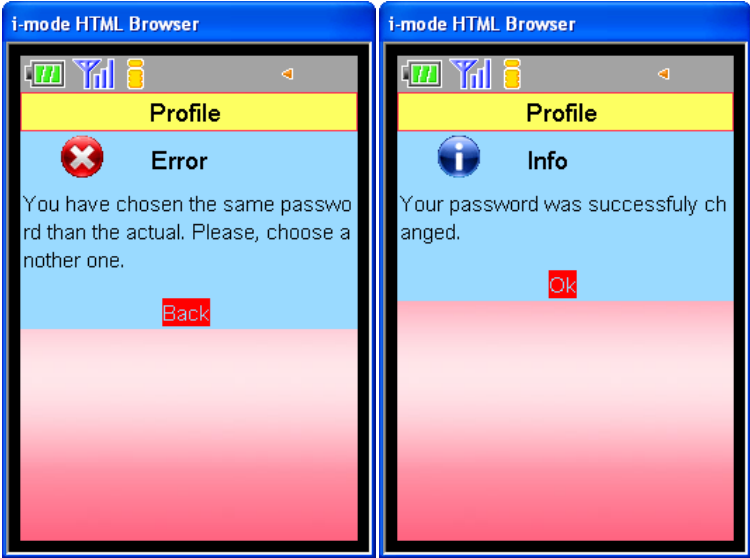
<p>State Finished</p>	<p>11.09.2008 The logout functionality is ready to use. Actually, there is no confirmation dialog. The servlet just do the logout and inform the player if it is ok or not. Next step is to add the confirmation dialog and final message after logout. In addition, to find a way to go back from the previous page in the case of cancel.</p> <p>22.09.2008 The logout process is ready. There is no confirmation actually, when the action is required. The player is directly logged off when he asked for.</p> <p>09.10.2008 Actually, it is not possible to use the cookies to store the Web User Interface options for a user. The WUI options have not to be stored directly in the application model because it means to mix View data with Model data. It is not a good idea. If the time is sufficient, a way to store the WUI options separately will be finding. The confirmation dialog is available and works like it is planned. The no link comes back to the previous page and the yes link disconnects the player and goes to a confirmation dialog.</p>
----------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

B.2.10. View preferences menu

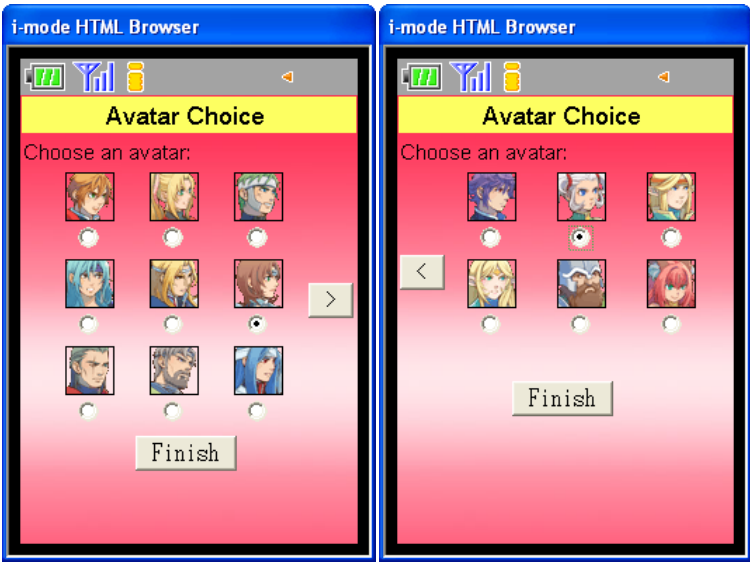
UC Number	10
Label	View preferences menu
Depends	UC 8
Priority	High
Complexity	Low
Description	Players can view their preferences. Actually, they can just change their password and avatar. In Quest Mode, they can view their avatar characteristics.
Actors	Player
Running	<ol style="list-style-type: none"> 1) Player sees a Preferences menu <ol style="list-style-type: none"> a. Profile → UC 11, Step 1 b. Avatar → UC 12, Step 1 c. Characteristics or Statistics → Step 2 UC 15, Step 1 d. Options → UC 16, Step 1 e. Cancel → UC 8, Step 1 2) Depending on the Game Mode <ol style="list-style-type: none"> a. Characteristics → UC 13, Step 1 b. Statistics → UC 15, Step 1
Samples	
Remarks	Characteristics are only available in Quest Mode, if Simple Mode is active; the button is replaced by Statistics button.
State	22.09.2008
Finished	Actually, there is no Options and Statistics entry point in the preferences menu. If the time is sufficient, these functionalities will be available and the preferences menu updated adequately.

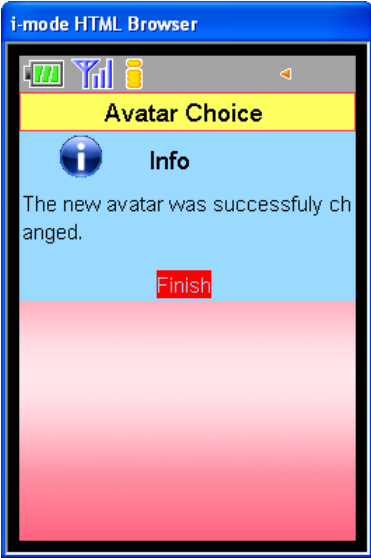
B.2.11. Modify profile

UC Number	11
Label	Modify profile
Depends	UC 10
Priority	Low
Complexity	Medium
Description	Players can change their password with this specific Use Case. A dedicated screen is provided for that.
Actors	Player, Engine
Running	<ol style="list-style-type: none"> 1) Player sees a change password dialog 2) He can write Old Password, New Password and Confirmation Password 3) He clicks <ol style="list-style-type: none"> a. Change → Step 4 b. Cancel → UC 10, Step 1 4) Validation of input is done <ol style="list-style-type: none"> a. Success → Step 6 b. Fail → Step 5 5) Error dialog is shown <ol style="list-style-type: none"> a. Ok → Step 2 6) Confirmation of password modification is done <ol style="list-style-type: none"> a. Ok → UC 10, Step 1
Samples	

	
Remarks	-
State Finished	22.09.2008 This Use Case is totally implemented and ready to use. The necessary checks for the validity of the passwords are done.

B.2.12. Modify avatar

UC Number	12
Label	Modify avatar
Depends	UC 11
Priority	Low
Complexity	Low
Description	Players can change their avatar with this specific Use Case. A dedicated screen is provided for that.
Actors	Player, Engine
Running	<ol style="list-style-type: none"> 1) Player sees a change avatar dialog 2) He can choose an avatar 3) He clicks <ol style="list-style-type: none"> a. Change → Step 4 b. Cancel → UC 10, Step 1 4) Confirmation of avatar modification is done <ol style="list-style-type: none"> a. Ok → UC 10, Step 1
Samples	

	
Remarks	-
State Finished	<p>08.09.2008 This Use Case is partially implemented. The possibility to change the avatar exists because this is the same mechanism than during the registration process. However, there is no way to go to the player profile actually.</p> <p>22.09.2008 There is an entry point to modify the avatar from the preferences menu. This use case is finished.</p>

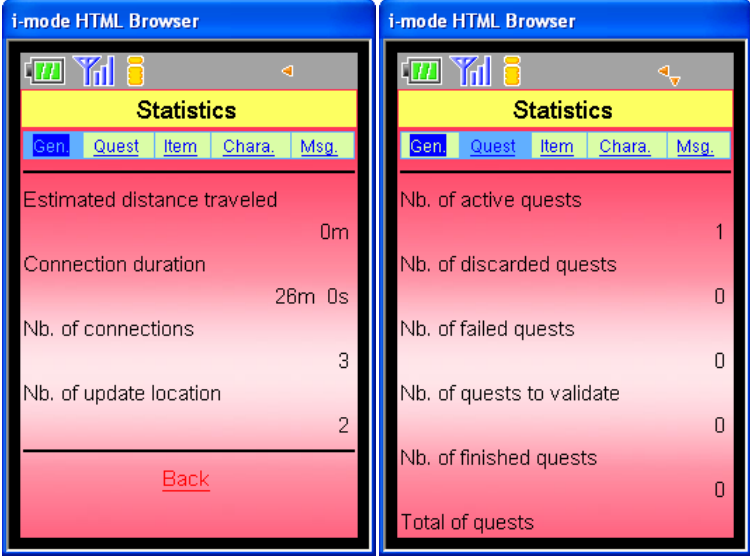
B.2.13. View characteristics

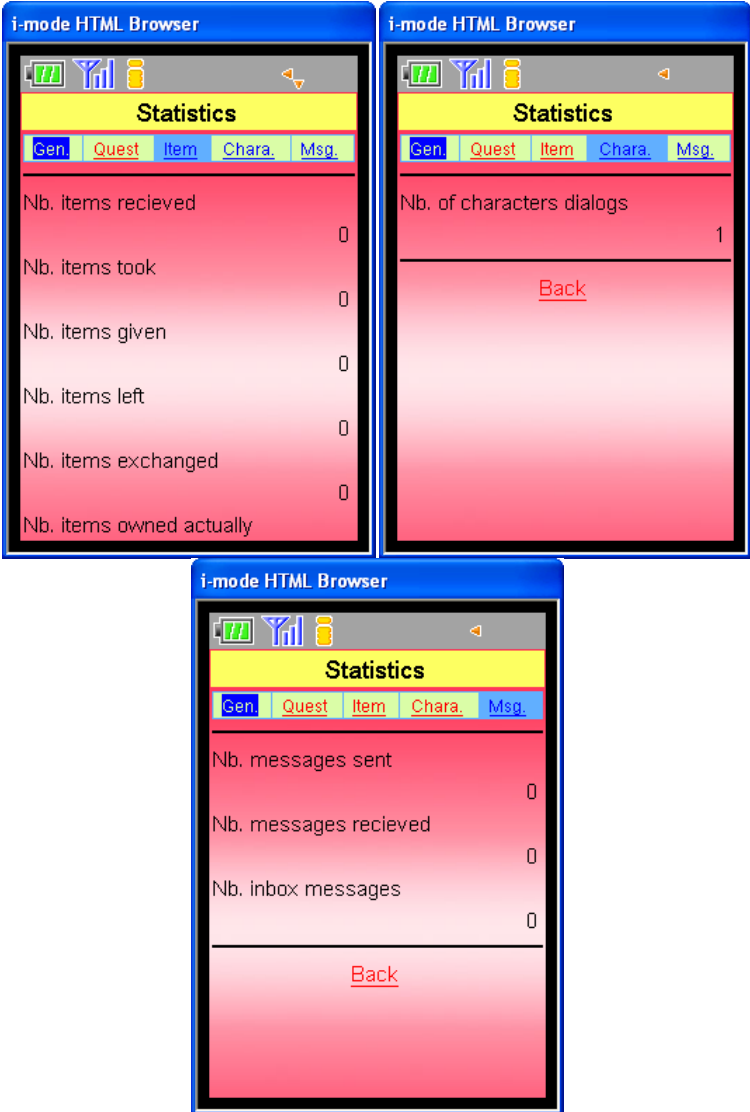
UC Number	13
Label	View characteristics
Depends	UC 11
Priority	Low
Complexity	Low
Description	Players can view their avatar characteristics. By the way, there is a possibility to configure the players' equipment.
Actors	Player, Engine
Running	<ol style="list-style-type: none"> 1) Player sees a dialog with characteristics of his character 2) He can do an action <ol style="list-style-type: none"> a. Back → UC 10, Step 1 b. Equip → UC 14, Step 1 c. Stat → UC 15, Step 1
Samples	
Remarks	Due to the mobile device, the "Equip" action can be activated by a pressure on the avatar picture.
State	<p>21.09.2008</p> <p>Cancelled After a discussion with professor Liechti, we decide to focalize on the simple game mode and just offer quests and items. In this situation, there is no more need to view or manage the player's characteristics (STR, ...).</p>

B.2.14. View and modify equipment

UC Number	14
Label	View and modify equipment
Depends	UC 13
Priority	Low
Complexity	High
Description	Players can view their equipment. By the way, there is a possibility to configure the players' equipment.
Actors	Player, Engine
Running	<ol style="list-style-type: none"> 1) Player sees a dialog with equipment of his character 2) Player chooses the equipment piece to change <ol style="list-style-type: none"> a. Head b. Feet c. Arms d. Legs e. Chest f. Weapon 3) He sees the current equipment with current modifiers. (If no piece of equipment is present, skipped this step → Step 4) <ol style="list-style-type: none"> a. Back → Step 1 b. Unequip → Step 8 c. Change → Step 4 4) He chooses a piece of equipment in the filtered list 5) A dialog with old characteristics values and possible values with equipment is shown. <ol style="list-style-type: none"> a. Cancel → Step 1 b. Equip → Step 6 6) Remove current equipment if necessary 7) Equip the new piece of equipment at the right place → Step 1 8) Remove current equipment → Step 1
Samples	
Remarks	<p>Due to the mobile device, the "Equip" action can be activated by a pressure on the specific place to equip.</p> <p>Current equipment dialog shows the actual piece of equipment modifiers. This view permits to know how the equipment piece modifies player's characteristics.</p> <p>Equip dialog shows actual values and new values. Old values are actual values with current equipment. New values are the values obtain with removing current equipment and adding new equipment.</p>
State	<p>21.09.2008</p> <p>After a discussion with professor Liechti, we decide to focalize on the simple game mode and just offer quests and items. In this case, there is no more need to manage the equipment in relation with characteristics.</p>

B.2.15. View statistics

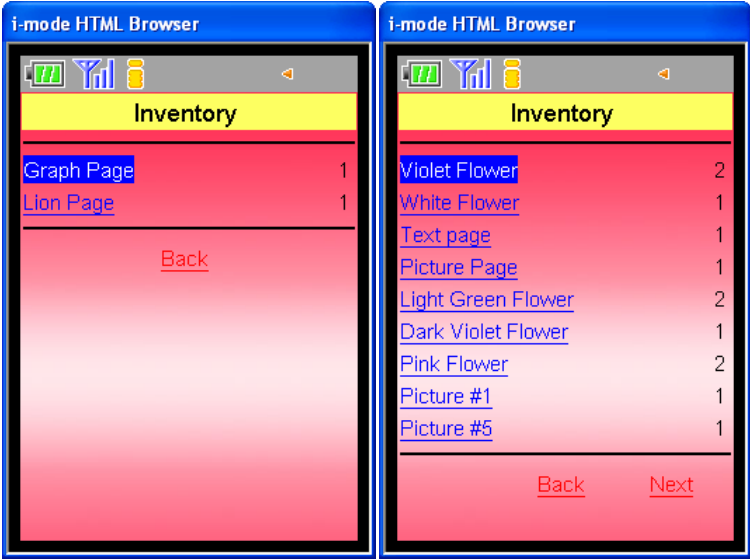
UC Number	15
Label	View statistics
Depends	UC 13
Priority	Low
Complexity	Low
Description	Players can view their statistics. Statistics can be the distance that players have covered during all the game.
Actors	Player, Engine
Running	<ol style="list-style-type: none"> 1) Player sees a dialog with statistics 2) Player can choose: <ol style="list-style-type: none"> a. Tab element → Step 3 b. Back → UC 8, Step 1 3) Data are refreshed with new statistic category → Step 1 4) Players clicks on Back button <ol style="list-style-type: none"> a. Comes from Preferences Menu → UC 10, Step 1 b. Comes from Characteristics screen → UC 13, Step 1
Samples	

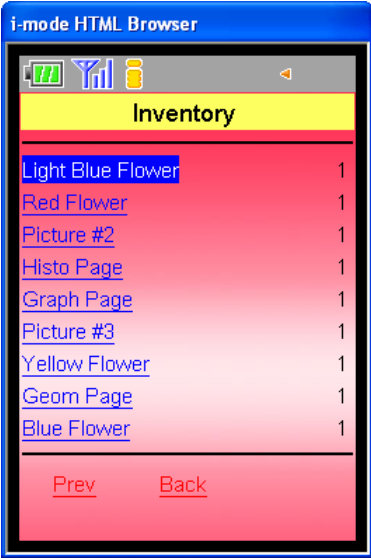
	
Remarks	-
State Finished	23.11.2008 This Use Case is completely finished. There is a screen with tabbed menu that allows changing the category of statistics. If a statistic does not exist, it is created. There is a few statistics like time connection or number of location updates. There are about five categories of statistics. When the player has finished seeing his statistics, he returns to the global menu.

B.2.16. View and modify options

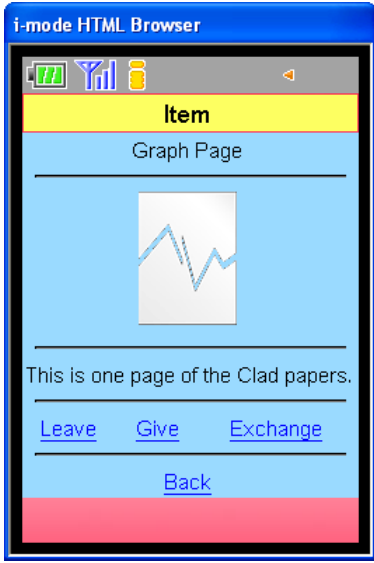
UC Number	16
Label	View and modify options
Depends	UC 10
Priority	Low
Complexity	Low
Description	Players can view and modify game options (client options). The most of options are about the User Interface.
Actors	Player
Running	<ol style="list-style-type: none"> 1) Player sees a dialog with options 2) He can modify options and then choose an action: <ol style="list-style-type: none"> a. Default → Step 3 b. Cancel → Step 5 c. Save → Step 4 3) Restore default program options values → Step 2 4) Save player options values 5) Returns to → UC 10, Step 1
Samples	
Remarks	<p>Options are not clearly know at this time. A possibility can be to configure remembered dialog decision (quit game confirmation, ...):</p> <ul style="list-style-type: none"> • Player connection remember • Game mode remember • Quit game remember • Use item remember <p>This Use Case is subject to changes. Not all options are known at this time.</p>
State	<p>22.11.2008</p> <p>Cancelled Due to the Web User Interface, there is no option to implement for the customization of the interface. This Use Case is no more useful.</p>

B.2.17. View inventory

UC Number	17
Label	View inventory
Depends	UC 8
Priority	Medium
Complexity	Medium
Description	Players have the possibility to view their items in inventory. This is a list of their items in different categories (Normal, Quests, Equipment and Special).
Actors	Player
Running	<ol style="list-style-type: none"> 1) Player sees the list of item in his inventory → Step 4 1) Player chooses a category <ol style="list-style-type: none"> a. Normal, Equipments, Quests or Special → Step 2 b. Cancel → UC 8, Step 2 2) Players see a filtered list of items corresponding to the selected category. <ol style="list-style-type: none"> a. One page → Step 4 b. More than one page → Step 3 3) They can click on Next or Prev buttons to show next or previous page of the current category inventory. They can repeat this action as much as they are available pages. 4) Players can choose Back, Map button or he can choose an item to do an action. <ol style="list-style-type: none"> a. Back → UC 8, Step 2, UC 6, Step 1 b. Map → UC 6, Step 1 c. Item → UC 18, Step 1
Samples	


	
Remarks	<p>Normal and Equipment items are not available in Simple Mode so players cannot see Normal and Equipments button. Items can be presents in more than one category.</p>
State Finished	<p>22.10.2008 Due to the orientation of the project, this Use Case had to be redefined a lot. There is no more item category and in this situation, the possibility to choose a category is no longer necessary. For the same reason, the link Back is now the same link than Map. Therefore, there is no reason too to keep the both links. Only Back link is kept. This Use Case is ready. The item list is available and shows all the items owned by the player.</p> <p>23.10.2008 Now, there is a link for each item to see the details of the item. The details come from the item type.</p> <p>03.11.2008 The navigation contains the ability of paging. Now, if there is more a certain number of items, the navigation shows automatically link to go on the next or previous page.</p>

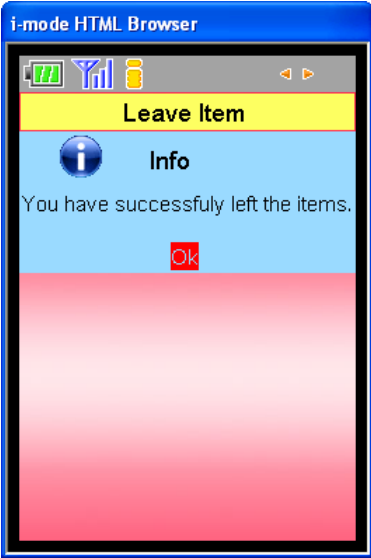
B.2.18. View actions for an item

UC Number	18
Label	View actions for an item
Depends	UC 17
Priority	High
Complexity	Low
Description	Players have the possibility to do some actions on items. They can leave exchange or shake items.
Actors	Player
Running	<p>1) Player chooses an item to do an action. A dialog is shown to the players with details of item and possible actions.</p> <ol style="list-style-type: none"> Leave → UC 19 Exchange Give → UC 20 Exchange → UC 46 Shake → UC 21 Use → UC 22 Equip → UC 42 Unequip → UC 43 Cancel → UC 17, Step 4
Samples	
Remarks	<p>In Simple Mode, only Quests and Special item can be selected.</p> <p>When accelerometers are available, player can shake item without press the shake button.</p> <p>Action Use depends on some conditions like usable from Inventory or not, or Simple Mode usage is allowed...</p> <p>Actions Equip/Unequip depends on inventory category (equipment needed) and if equipment is already equipped or not.</p>

State Finished	23.10.2008 Accelerometers are not available from the Web User Interface and in this condition; there is no easy way to use them to do the shake action. For this reason, we are constraint to cancel this Use Case. Equip/Unequip is no longer necessary due to the Simple Game Mode only. These Use Cases are cancelled too. Due to the asynchronous system with the Web User Interface, there is no way to do a good notification system without break the game flow. Remember that the device cannot use JavaScript. For the rest of this Use Case, the links to Exchange, Give or Leave an item are present on the dialog.
--------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

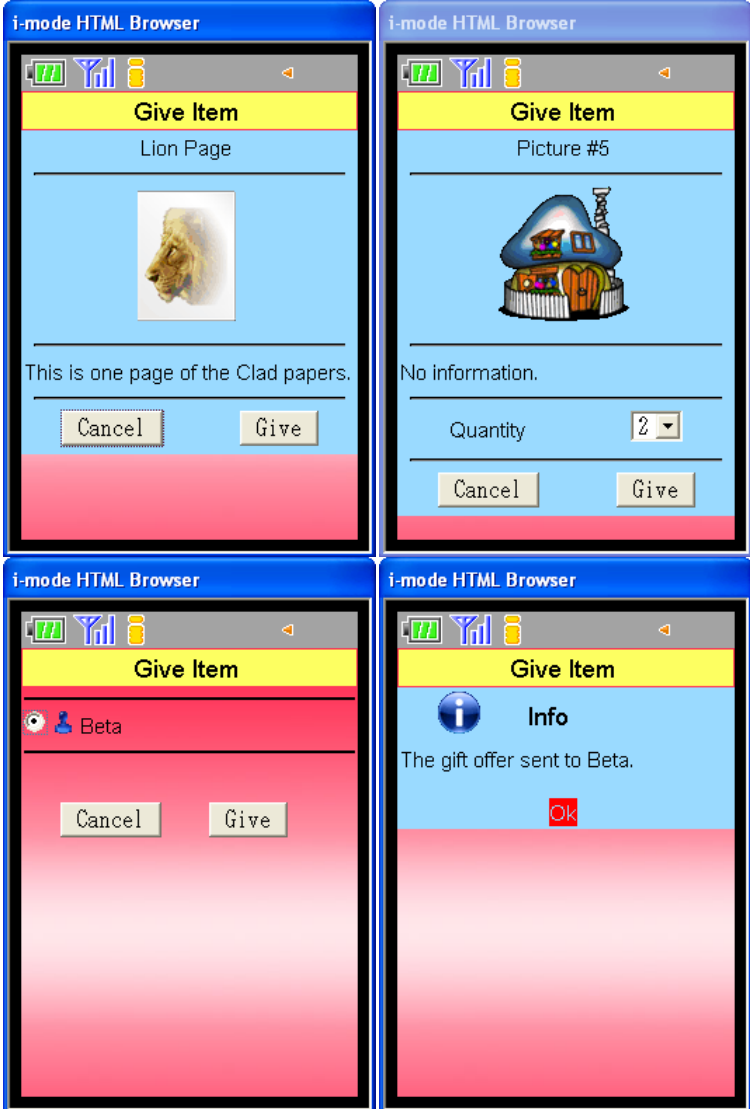
B.2.19. Leave item


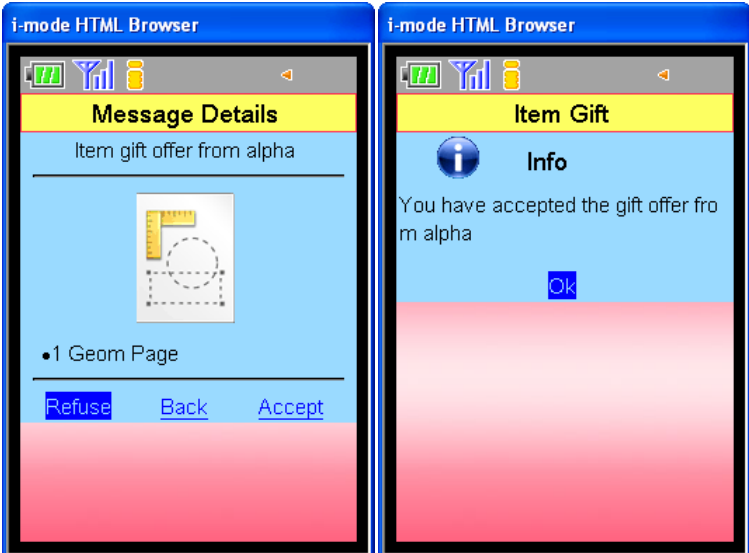
UC Number	19
Label	Leave Item
Depends	UC 18
Priority	Medium
Complexity	Medium
Description	Players can leave items during the game. All items persist in the game when players leave them.
Actors	Player, Engine
Running	<ol style="list-style-type: none"> 1) Player sees a dialog asking him for how many items of the current sort he wants to leave. (See remarks). 2) He clicks on: <ol style="list-style-type: none"> a. Leave → step 3 b. Cancel → UC 18, step 1 3) Item are removed from Inventory <ol style="list-style-type: none"> a. Items are placed in the game (at the current player position) 4) Player returns → UC 17, Step 2 Step 1
Samples	

	
<p>Remarks</p>	<p>If there is only one item of the selected kind of item, the quantity choice can be skipped.</p>
<p>State Finished</p>	<p>28.10.2008 The Use Case is completely implemented without the correct placement of the item(s) on the map. Actually, the localization is not available and this is not possible to use it properly.</p> <p>12.11.2008 The localization is partially available now and with it, this is possible to finish this use case. Now, the left item is placed on the map at the player position.</p>

B.2.20. Give item

UC Number	20
Label	Give Item
Depends	UC 18
Priority	Medium
Complexity	High
Description	<p>Players can give items with other players during the game. To Give items, players must be near other players.</p> <p>There are two starting points to come in this Use Case. Depending on these starting points, this use case can change for the returning points.</p>
Actors	Player, Engine
Running	<p>Player A</p> <p>Part A:</p> <ol style="list-style-type: none"> 1) Player A sees a dialog asking them for how many items of the current sort they want to propose. (See remarks). <ol style="list-style-type: none"> a. Give → Part A, Step 2 b. Cancel → UC 18, Step 1 2) Player A sees a dialog with nearest players that can be able to exchange items with. <ol style="list-style-type: none"> a. Choose a player B by clicking on player nickname → Part A, Step 3 b. Cancel → Part A, Step 1 3) Items offered are removed from the Player A inventory. 4) Player A sees a confirmation dialog that he sent an offer to the player B. <p>Player B</p> <p>Part A:</p> <ol style="list-style-type: none"> 1) Player B goes to the messages screens to see the message of the offer done by the player A. 2) Player B can choose: <ol style="list-style-type: none"> a. Back → Step 1 (Messages screen) b. Accept → Both Players, Part A, Step 1 c. Refuse → Both Players, Part B, Step 1 <p>Both Players:</p> <p>Part A:</p> <ol style="list-style-type: none"> 1) Items offered are added to the Player B inventory. → Part C, Step 1 <p>Part B:</p> <ol style="list-style-type: none"> 1) Items offered are added to the Player A inventory. → Part C, Step 1

	<p>Part C:</p> <ol style="list-style-type: none"> 1) Offer is removed. 2) A confirmation message is sent to the Player A. 3) Player B return → Player B, Part A, Step 1
<p>Samples Player A</p>	

	 <p>The screenshot shows a mobile browser interface with a blue header and a yellow title bar. The title is "Message Details". The main content area is light blue and contains the text "Item gift onfirmation from Beta" and "Beta has accepted your gift offer." Below this is a small image of a gift box. At the bottom, there is a list item "•1 Geom Page" and two buttons: "Back" and "Delete".</p>
Samples Player B	 <p>The left screenshot shows "Message Details" for an "Item gift offer from alpha". It includes a gift box image and a list item "•1 Geom Page". At the bottom, there are three buttons: "Refuse", "Back", and "Accept".</p> <p>The right screenshot shows "Item Gift" confirmation. It features an "Info" icon and the text "You have accepted the gift offer from alpha". An "OK" button is visible at the bottom.</p>

<p>Sequence Diagram</p>	<pre> sequenceDiagram participant A as Player A participant B as Player B A->>A: 1. Choose how many items A->>A: 2. Remove items from inventory A->>A: 3. Choose player A->>A: 4. Create offer A->>B: 1: Offer activate B B->>B: 6. View offer B->>B: 7. Accept offer B->>B: 8. Add items into inventory B->>B: 9. Remove offer B->>B: 10. Create confirmation B->>A: 1.1: Confirmation deactivate B A->>A: 5. Confirmation Dialog </pre>
<p>Remarks</p>	<p>When a player has only one item of an item type, he does not need to select the quantity.</p>
<p>State Finished</p>	<p>11.12.2008 This Use Case is partially done. Actually, this is possible to send an offer to the player B with the quantity of desired item. The Player B can see the offer.</p> <p>13.11.2008 The player B can accept or refuse the offer from the player A. Items are added to the correct inventory depending of the answer. Confirmation is sent to the player A. The player A can see the dialog and delete it or keep it.</p> <p>14.11.2008 The cancel buttons are now operational. When the player comes back from the choice of player, he retrieves the value he selected before to go to the select player page.</p> <p>17.11.2008 The bug for the quantity not memorized when the player come back after a cancel action is corrected. This Use Case is totally finished.</p>

B.2.21. Shake item

UC Number	21
Label	Shake Item
Depends	UC 18
Priority	Medium
Complexity	Low
Description	<p>Some items can be shaken to produce some special behavior on it. To produce special behavior on items, players must shake their mobile device (if accelerometers are available) or push the shake button.</p> <p>If accelerometers are not available, the UC begins at the Part A, Step 2.</p>
Actors	Player, Engine
Running	<p>Part A:</p> <ol style="list-style-type: none"> 1) Player shakes his mobile device as he wants during a certain amount of time. <ol style="list-style-type: none"> a. Nothing happens → Step 2 b. Something happens → Step 3 2) A message asks the player if he wants to retry or not. <ol style="list-style-type: none"> a. Retry → Step 1 or Step 2 (if accelerometers are not available) b. Cancel → UC 18, Step 1 3) Message informs the player on the behavior of item. <ol style="list-style-type: none"> a. Discover new item → Part B, Step 1 b. Item explode → Part C, Step 1 4) Player returns to → UC 17, Step 2 <p>Part B:</p> <ol style="list-style-type: none"> 1) Shacked item is removed from player's inventory 2) Player is invited to decide if he wants to take or leave the new item. <ol style="list-style-type: none"> a. Take → Step 4 b. Leave → Step 3 3) Discovered item is landed on the map. → Step 5 4) Discovered item is added to the player inventory 5) Part A, Step 5 <p>Part C:</p> <ol style="list-style-type: none"> 1) Item exploded (vibration can be used) <ol style="list-style-type: none"> a. Ok → Step 2 2) Some health points are subtracted to player's health points. (Only in Quest Mode) 3) Shacked item is removed from player's inventory 4) Part A, Step 5

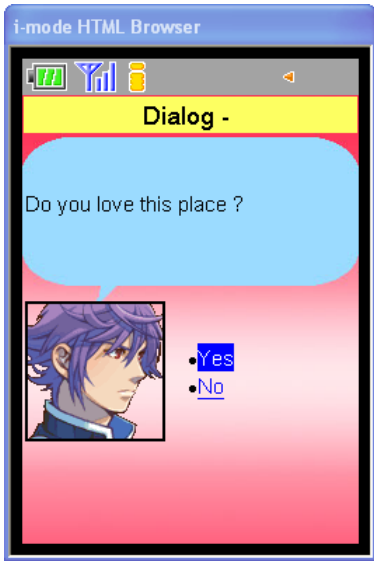
Samples	
Remarks	-
State Cancelled	<p>08.09.2008 Due to the limitation of the iMode application (basic web interface), this is not possible to use the motion sensor for the shake action. The shake action could be done manually by pressing an action button.</p> <p>21.09.2008 After a discussion with professor Liechti, we decide to focalize on the simple game mode and just offer quests and items. Therefore, this functionality is not essential for objectives. In addition to the limitation of the WUI interface, this use case is cancelled.</p>

B.2.22. Use an item

UC Number	22
Label	Use an item
Depends	UC 18
Priority	Low
Complexity	Low
Description	Players can use some items from Inventory. For example, a potion could be used to regenerate the player's health from inventory.
Actors	Player, Engine
Running	<ol style="list-style-type: none"> 1) Showing the confirmation dialog <ol style="list-style-type: none"> a. True → Step 2 b. False → Step 4 2) Players see a confirmation dialog with remember option 3) Memorized remember option 4) Do the action corresponding to the player choice. <ol style="list-style-type: none"> a. Yes → Step 5 b. No → UC 18, Step 1 5) Item effect is applied 6) Item is removed from player's inventory 7) A notification is show to the player to indicate item effect. <ol style="list-style-type: none"> a. Ok → UC 17, Step 2
Samples	
Remarks	-
State	<p>21.09.2008 Cancelled After a discussion with professor Liechti, we decide to focalize on the simple game mode and just offer quests and items. Therefore, this functionality is not essential for objectives. In addition to the limitation of the WUI interface, this use case is cancelled.</p>

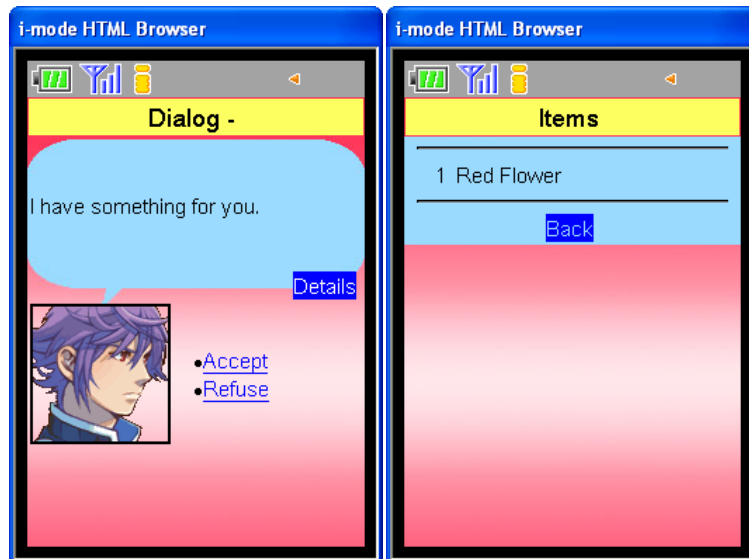
B.2.23. Do a dialog with NPC

UC Number	23
Label	Do a simple dialog with NPC
Depends	UC 6
Priority	High
Complexity	High
Description	Simple dialogs permits for players to obtain some information about the game and about the city or quests. During simple dialogs, some actions are possible depending on the conversation with the NPC. Players can give or receive some Kmeps or items.
Actors	Player, NPC
Running	<p>Part A:</p> <ol style="list-style-type: none"> 1) Player sees an avatar that represents NPC and he can read what NPC says. <ol style="list-style-type: none"> a. Simple dialog → Step 2 b. Quest proposal → UC 24 c. Quest ending → UC 25 d. Merchant → UC 26 2) To read the next part of the discussion, player clicks on the Next button (or Finish button). <ol style="list-style-type: none"> a. Discussion is finished → Step 5 b. Answer of a question is needed → Step 3 c. Players can obtain something → Part B, Step 1 d. Players can give something → Part C, Step 1 e. Next page of the dialog is shown → Step 1 3) Player chooses one answer from those which are proposed. 4) Next dialog step: <ol style="list-style-type: none"> a. Discussion is finished → Step 5 b. Next page of the dialog is shown → Step 2 5) Player returns → UC 6, Step 1 <p>Part B:</p> <ol style="list-style-type: none"> 1) Player receives something: <ol style="list-style-type: none"> a. Kmeps → Step 2 b. Item(s) → Step 4 2) Player sees what how many Kmeps NPC is offering with specific message. He can: <ol style="list-style-type: none"> a. Accept → Step 3 b. Refuse → Part A, Step 4 3) Kmeps are added to the total amount of Kmeps of player. → Part A, Step 4

	<p>4) Player sees which item(s) NPC is(are) offering with specific message. He can:</p> <ul style="list-style-type: none"> a. Accept → Step 5 b. Refuse → Part A, Step 4 <p>5) Item(s) is(are) added to player's inventory. → Part A, Step 4</p> <p>Part C:</p> <p>1) Player is asking to give something:</p> <ul style="list-style-type: none"> a. Kmeps → Step 2 b. Item(s) → Step 4 <p>2) Player sees what how many Kmeps NPC is requesting with specific message. He can:</p> <ul style="list-style-type: none"> a. Accept → Step 3 b. Refuse → Part A, Step 4 <p>3) Kmeps are subtracted from the total amount of Kmeps of player. → Step 6</p> <p>4) Player sees which item(s) NPC is(are) requesting with specific message. He can:</p> <ul style="list-style-type: none"> c. Accept → Step 5 d. Refuse → Part A, Step 4 <p>5) Item(s) is(are) subtracted to player's inventory. → Step 6</p> <p>6) Current quests are updated.</p>
<p>Samples</p>	 <p style="text-align: center;">Question</p>



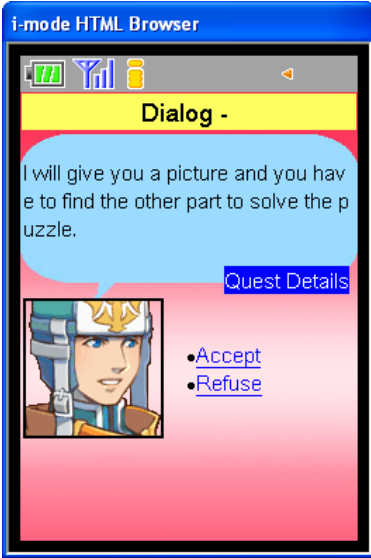
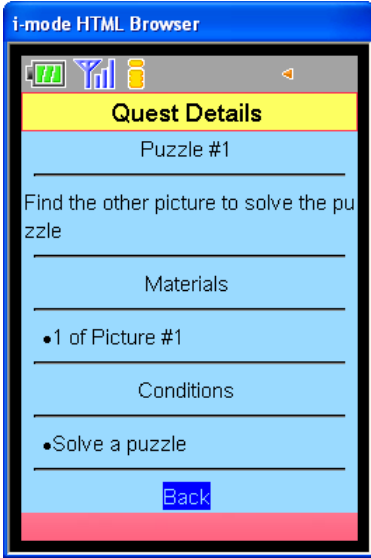
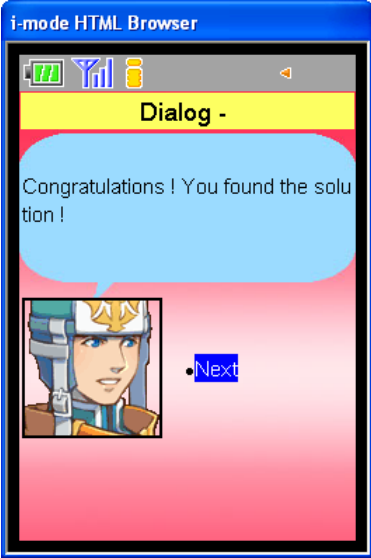
Text



Item offer/request



Puzzle question

	  <p style="text-align: center;">Quest proposition</p> 
Remarks	Each choice can have an influence on the discussion.
State Pending	<p>08.09.2008 This Use Case is currently under development. Actually, the layout is ready to receive different information. There is some server part to do to finish this Use Case. Actually, this is not possible to interact with the dialog and/or retrieve dialogs.</p> <p>09.09.2008 The mechanism to follow the dialog flow is ready but due to a bug in the deployment, it was not tested yet.</p>

	<p>10.09.2008 The mechanism to follow the dialog flow is ready and tested. The bug encountered yesterday was due to a problem with GlassFish/NetBeans. After a reboot, the problem does not appear again. Quest proposal is possible and ready to use (tested too). This is possible to accept or refuse the quest.</p> <p>21.09.2008 The previous state of this use case is established again after the implementation of the MVC.</p> <p>03.11.2008 The dialog to add and/or remove items is now available. There is no problem to offers items or to get items during a NPC dialog. Now, it will be great to add the screen to show which items will be given or removed. The dialog answer will be redefined in the context of Puzzle picture quest. There is also a simplification for the add/remove items dialog. There is no more confirmation dialog.</p> <p>12.11.2008 This Use Case is finished. The Question Puzzle is now implemented. A player can answer a textual quest for a quest puzzle. His answer is remembered and applied for the quest evaluation.</p>
--	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

B.2.24. Have a quest proposal

UC Number	24
Label	Have a quest proposal
Depends	UC 23
Priority	High
Complexity	Low
Description	When player speak with NPC, NPC can propose some quest to the player.
Actors	Player, NPC
Running	<p>1) NPC describes the quest and how to do to finish it. Description is like a simple dialog (UC 23, Step 2 (Letter "a" and "e" only) one or more times). NPC describes the quest and offers the possibility to show the quest details in a specific screen.</p> <p>2) Player has to decide to accept or refuse the quest:</p> <ul style="list-style-type: none"> a. Accept → Step 3 b. Refuse → Step 4 <p>3) Quest is added to the Quest Diary</p> <p>4) Player returns on simple dialog → UC 23, Step 4</p>
Samples	See the samples of the UC 23
Remarks	Some quest can be unavailable to some player depending on their characteristics and game mode.
State Finished	<p>08.09.2008 This Use Case depends on the previous Use Case. The mechanism to show the quest proposal is ready to use. There is just to finalize the server part as the description of the state of the previous Use Case.</p> <p>10.09.2008 The quest details will be appeared in a specific dialog rather than in the dialog flow. The quest details view has to be done.</p> <p>21.09.2008 This use case is already ok after the implementation of the MVC. The player can accept or refuse a quest. All the steps are done (for the first step, the remarks of 10.09.2008 are always true.</p>

B.2.25. Finish a quest

UC Number	25
Label	Finish a quest
Depends	UC 23
Priority	High
Complexity	Low
Description	When player speak with NPC, NPC can propose some quest to the player.
Actors	Player, NPC
Running	<ol style="list-style-type: none"> 1) NPC rewards the player <ol style="list-style-type: none"> a. Item(s) → Step 2 Step 3 b. Kmeps → Step 4 c. Both → Step 6 2) Player can choose if he wants or not offered item(s): <ol style="list-style-type: none"> a. Accept → Step 3 b. Refuse → Step 8 3) Item(s) is(are) added to player's inventory. → Step 8 4) Player can choose if he wants or not offered Kmeps: <ol style="list-style-type: none"> a. Accept → Step 5 b. Refuse → Step 8 5) Kmeps are added to player's Kmeps. → Step 8 6) Player can choose if he wants or not offered Kmeps and item(s): <ol style="list-style-type: none"> a. Accept → Step 7 b. Refuse → Step 8 7) Kmeps are added to player's Kmeps and Item(s) is(are) added to player's inventory. → Step 8 8) Quest is updated <ol style="list-style-type: none"> a. A new step of the quest is available → Step 9 b. Quest is marked as finished → Step 10 9) NPC describes the quest and how to do to finish it. Description is like a simple dialog (UC 23, Step 2 (Letter "a" and "c" only) one or more times). 10) Player returns on simple dialog → UC 23, Step 4
Samples	See the samples of UC 23.
Remarks	Kmeps rewards are only available in Quest Mode.
State Pending	<p>29.09.2008</p> <p>This Use Case is very changed from the original because of simplification and the focalization for the simple game mode. So actually, there is only item as rewards and the player cannot refuse the reward. The player can leave item(s) received from his inventory.</p>

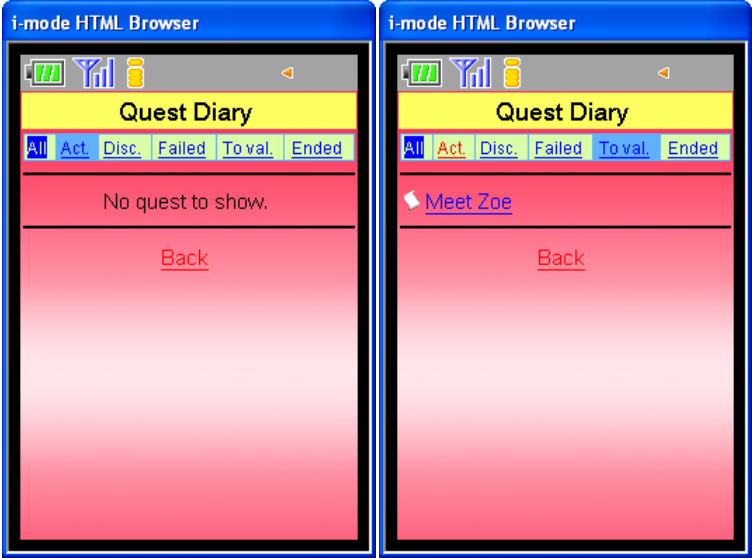
	<p>03.11.2008 Now, the item rewards are added to the player inventory when the quest is finished but the functionality is not actually tested.</p> <p>18.11.2008 This Use Case is totally finished and tested. Now, when a player talk to a NPC that allows finishing a quest and the quest is ready to be validated, the dialog is run and rewards added to the inventory (or other processes). The quest is marked as finished.</p>
--	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

B.2.26. Make a trade with a merchant

UC Number	26
Label	Make a trade with a merchant
Depends	UC 23
Priority	Low
Complexity	Medium – High
Description	In Quest Mode, players can make some trades with NPC merchants. They have the possibility to sell or buy some items.
Actors	Player, NPC
Running	<p>Part A:</p> <ol style="list-style-type: none"> 1) Players see an option dialog: <ol style="list-style-type: none"> a. Buy → Part B, Step 1 b. Sell → Part C, Step 1 c. Cancel → Step 2 2) Returns → UC 6, Step 1 <p>Part B:</p> <ol style="list-style-type: none"> 1) A list of proposed items is shown to the player 2) Player can: <ol style="list-style-type: none"> a. Click on an item → Step 3 b. Click on Cancel → Part A, Step 1 3) Player can choose the quantity that he wants to buy (limited by his Kmeeps in his possession): <ol style="list-style-type: none"> a. Buy → Step 4 b. Cancel → Step 2 4) Items are added to player's inventory 5) Kmeeps are subtracted from player's total kmeeps. 6) Player returns to option dialog → Part A, Step 1 <p>Part C:</p> <ol style="list-style-type: none"> 1) Inventory is opened 2) Player choose item's category 3) Player can: <ol style="list-style-type: none"> a. Click on an item → Step 4 b. Click on Cancel → Part A, Step 1 4) Player can choose the quantity that he wants to sell (limited by his quantity in his possession): <ol style="list-style-type: none"> c. Sell → Step 5 d. Cancel → Step 3 5) Items are removed from player's inventory 6) Kmeeps are added to player's total kmeeps.

	7) Player returns to option dialog → Part A, Step 1
Samples	
Remarks	This Use Case is valid only for Quest Mode. Items prices depend on NPC Merchant.
State Cancelled	21.09.2008 After a discussion with professor Liechti, we decide to focalize on the simple game mode and just offer quests and items. In this situation, there is no more need to have merchant because players have no kmep money (no characteristics).

B.2.27. View quests diary

UC Number	27
Label	View quest diary
Depends	UC 8
Priority	Medium
Complexity	Medium
Description	Quest Diary permits to players to view their different quests classed by categories. They can do some action on it.
Actors	Player
Running	<ol style="list-style-type: none"> 1) Active quests are evaluate to determine if they are finished. 2) Player can choose the category directly on the Quest Diary <ol style="list-style-type: none"> a. Category changed → Step 3 b. Back (back to the map) → UC 6, Step 1 2) Player chooses a category <ol style="list-style-type: none"> c. Category Quest → Step 3 d. Cancel → UC 8, Step 1 3) Player sees a list of quest corresponding to the selected category. <ol style="list-style-type: none"> a. One page → Step 4 b. More than one page → Step 3 4) He can click on Next or Prev buttons to show next or previous page of the current quests diary category. He can repeat this action as much as they are available pages. 5) Player can choose Back or Map button <ol style="list-style-type: none"> a. Back → Step 1 UC 6, Step 1 b. Map → UC 6, Step 1 c. Quest → UC 28
Samples	

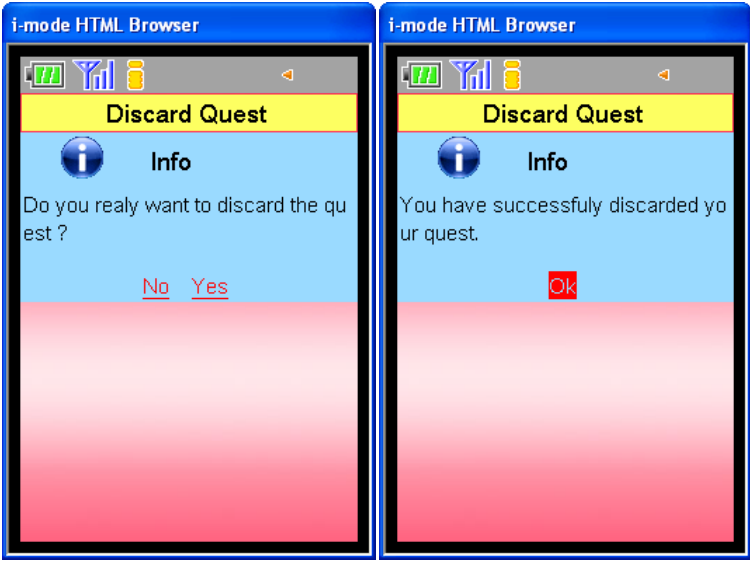
Remarks	When a quest has more than one part, each part is visible as a sub quest of the principal quest.
State Finished	<p>22.09.2008 This Use Case is near the end. There is only the paging to add. Actually, this Use Case is considered as finished because the paging is not a primordial need at this time of the project.</p> <p>03.11.2008 The navigation contains the ability of paging. Now, if there is more a certain number of quests, the navigation show automatically the links to go on the next or previous page.</p>

B.2.28. View actions for a quest

UC Number	28
Label	View actions for a quest
Depends	UC 27
Priority	Low
Complexity	Low
Description	<p>Players can do some action on quests depending on current quest state. Current quests can be discarded or given. Discarded quests have no action available. History quests can be given. Failed quests can be redone. Depending on the quest state, the player can do some actions like Give a quest, Discard it or Try again.</p>
Actors	Player
Running	<p>Part A:</p> <ol style="list-style-type: none"> 1) Player views a dialog with quest details. 2) He can: <ol style="list-style-type: none"> a. Back → UC 27, Step 2 1 b. Action → Step 3 c. Prev → Step 4 d. Next → Step 5 3) Depending on Quest Category <ol style="list-style-type: none"> a. Discarded → No action available b. Failed → Part D, Step 1 a. InProgress → Part B, Step 1 b. Abandoned → Part D, Step 1 c. Failed → Part D, Step 1 d. PreFinished → Part B, Step 1 e. Finished → Part C, Step 1 4) Players view previous information of the quest if available → Step 2 5) Players view next information of the quest if available → Step 2 <p>Part B:</p> <ol style="list-style-type: none"> 1) Chose an action to do <ol style="list-style-type: none"> a. Discard → UC 29 b. Give → UC 30 c. Grouping Buttons depending on UC 31 d. Cancel → Part A, Step 32 <p>Part C:</p> <ol style="list-style-type: none"> 1) Chose an action to do <ol style="list-style-type: none"> a. Give → UC 30 b. Cancel → Part A, Step 32

	<p>Part D:</p> <ol style="list-style-type: none"> 1) Chose an action to do <ol style="list-style-type: none"> a. Redo Retry → UC 32 UC 44 b. Give → UC 30 c. Cancel → Part A, Step 32
<p>Samples</p>	
<p>Remarks</p>	<p>Group mechanisms (and buttons) depending of the UC 31. The UC 31 describes the conditions of Group Buttons apparition.</p>
<p>State Pending</p>	<p>22.11.2008</p> <p>The entry points are ready to be used and tested relative to the quest states. XML configuration file is used to create dummy states to check if the actions links are the correct ones depending on the quest state. All the actions are implemented too.</p>

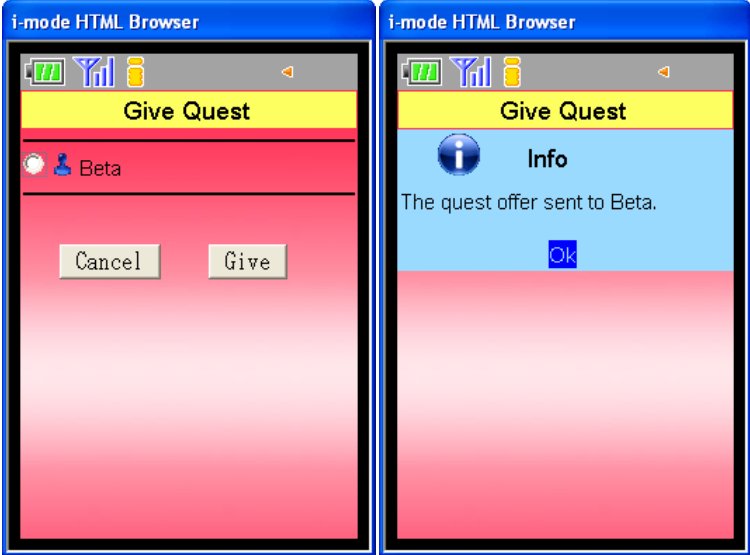
B.2.29. Discard a quest

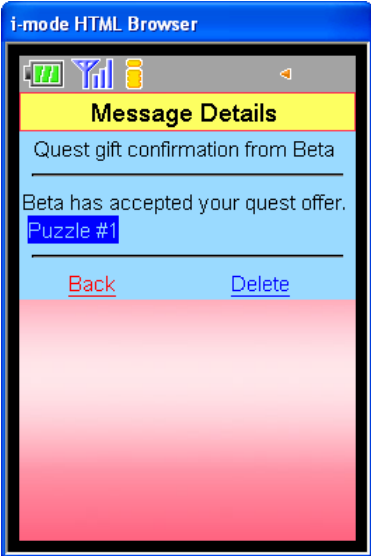
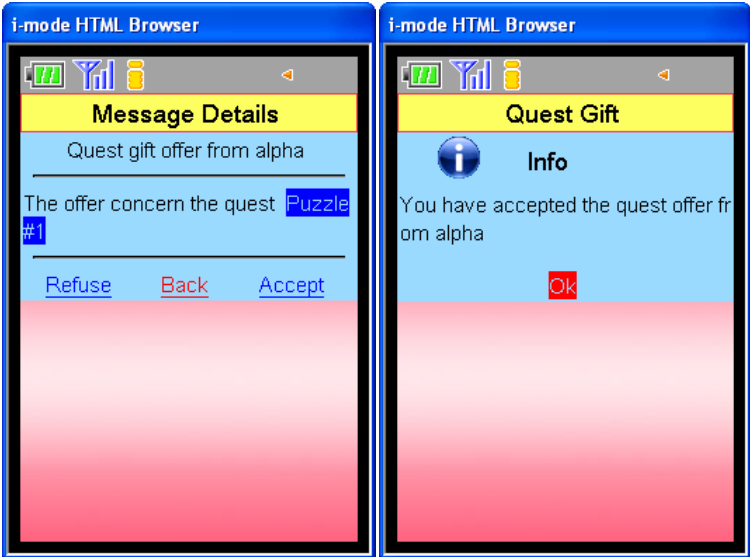
UC Number	29
Label	Discard a quest
Depends	UC 28
Priority	Low
Complexity	Low
Description	<p>Players can discard a quest in InProgress or PreFinished state current state. When a quest is discarded, one way to do the quest again is to go to discuss with the NPC which has proposed the quest. The second way is that another player gives this quest (UC 30). To do the quest again, the player has to activate the Retry action on the quest.</p>
Actors	Player, Engine
Running	<ol style="list-style-type: none"> 1) Players see a dialog asking if they are sure to discard the chosen quest. 2) They click on: <ol style="list-style-type: none"> a. Yes → step 3 b. No → UC 28, Part A, Step 21 3) The state of the quest is changed to Abandoned state. 4) Player sees a dialog with confirmation: <ol style="list-style-type: none"> a. Ok → UC 27, Step 1 5) Quest is removed from current quests state diary. 6) Quest is added to discarded quests state diary. 7) Players returns → UC 27, Step 2
Samples	
Remarks	-

State Finished	22.11.2008 Players can discard a quest in the allowed state. The quest state is updated. The dialogs are correctly shown to the player.
--------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------

B.2.30. Give a quest

UC Number	30
Label	Give a quest
Depends	UC 28
Priority	Medium
Complexity	Medium
Description	<p>Players can give to other players their quests. This is useful to escape the discussion with NPC and doing a quest together. Without too more constraints (via grouping by example). Players can come from different starting points to do this Use Case. In this situation, returning points can depends on starting points.</p>
Actors	Player, Engine
Running	<p>Part A:</p> <ol style="list-style-type: none"> 1) Player A sees a dialog with nearest players that can be able to give receive a quest to them. <ol style="list-style-type: none"> a. Choose a player B by clicking on player nickname → Step 2 b. Cancel → UC 28, Part A, Step 2 2) A notification is sent to the player B <ol style="list-style-type: none"> a. Player A return to UC 28, Part A, Step 1 3) Player A sees a dialog asking if he wants to give the selected quest. <ol style="list-style-type: none"> a. Give → Step 3 b. Cancel → Step 1 or UC 40, Step 2 4) Player A sees a waiting dialog. <ol style="list-style-type: none"> a. Cancel → Part C, Step 1 5) Player B current screen state is memorized 6) Player B is notified for the given quest. <ol style="list-style-type: none"> a. Accept → Step 6 b. Refuse → Part B, Step 1 7) Quest is added to the player's B current quests state. 8) Player A sees player's B answer. 9) Player B returns to memorized screen. 10) Player A returns → Step 1 or UC 40, Step 2 <p>Part B:</p> <ol style="list-style-type: none"> 1) Player B sees the message from player A <ol style="list-style-type: none"> a. Accept → Step 2 b. Refuse → Part C, Step 1 2) Check if the player B doesn't own the quest <ol style="list-style-type: none"> a. True → Step 3 b. False → Part D, Step 1 3) New quest state is added to the player B quest diary

	<p>4) Accept notification is sent to the player A → UC 28, Part A, Step 1</p> <p>Part C:</p> <p>1) Refuse Notification is sent to player A → UC 28, Part A, Step 1</p> <p>Part D:</p> <p>1) Quest already owned by player B is sent to the player A → UC 28, Part A, Step 1</p> <p>Part B:</p> <p>1) Player A is notified that player B refuses his given quest.</p> <p>2) Player A return to player choice after he clicks on Ok. (or UC 40, Step 2)</p> <p>3) Player B returns to memorized screen.</p> <p>Part C:</p> <p>1) Player B is notified that the Player A canceled his proposition.</p> <p>2) Player B returns to memorized screen after he clicks on Ok.</p>
<p>Samples Player A</p>	

	 <p>The screenshot shows a mobile browser interface with a blue header 'i-mode HTML Browser'. Below the header is a yellow bar with the title 'Message Details'. The main content area is light blue and contains the text: 'Quest gift confirmation from Beta', a horizontal separator, 'Beta has accepted your quest offer.', and 'Puzzle #1'. At the bottom of the content area are two links: 'Back' and 'Delete'. The background of the browser is a red-to-white gradient.</p>
<p>Samples Player B</p>	 <p>The row contains two side-by-side screenshots of the mobile browser. The left screenshot shows a yellow bar with the title 'Message Details'. The text below reads: 'Quest gift offer from alpha', a horizontal separator, 'The offer concern the quest Puzzle #1', and three links: 'Refuse', 'Back', and 'Accept'. The right screenshot shows a yellow bar with the title 'Quest Gift'. Below it is an 'Info' icon and the text: 'You have accepted the quest offer from alpha', followed by a red 'OK' button. Both screenshots have the same blue header and red-to-white gradient background as the first screenshot.</p>

<p>Sequence Diagram</p>	<pre> sequenceDiagram participant A as Player A participant B as Player B A->>A: 1. Choose a player A->>A: 2. Ask for confirmation A->>A: 3. Send a notification A->>B: 1: Offer activate B B->>B: 4. Show the message B->>B: 5. Accept the proposition B->>B: 6. Check the quest already owned B->>B: 7. Create new quest state B->>B: 8. Send a notification B->>A: 1.1: Confirmation deactivate B </pre>
<p>Remarks</p>	<p>Give a quest is dependant of the same condition than to accept a quest proposed by a NPC. It is not possible to propose a quest that a player has not pre-requisites. Multi part quest can be given. The gift is possible even if some parts are already done.</p> <p>Player list contains only player that can receive the quest from the player A. do the quest depending on remarks just before.</p>
<p>State Finished</p>	<p>22.11.2008</p> <p>This Use Case is finished. Players can give a quest to another player with a similar process than give item. One message is sent to the player to give the quest. This player has to accept or refuse. Depending on the answer, a notification is sent to the player who offers the quest. There is also a check for to validate if the player who receives the quest has or not the corresponding quest.</p>

B.2.31. Manage Grouping buttons action

UC Number	31
Label	Manage Grouping buttons action
Depends	UC 28
Priority	Medium
Complexity	Medium
Description	This Use Case is particular because it describes the management of User Interface for the grouping mechanisms. Grouping mechanism depends on some conditions that interfere with screen views.
Actors	"GUI Client"
Running	<p>Part A:</p> <ol style="list-style-type: none"> 1) Evaluation of current grouping state. <ol style="list-style-type: none"> a. Group exists and player is owner → Part B, Step 1 b. Group doesn't exist and player is not in a group → Part C, Step 1 c. Group doesn't exist and player is in a group → Part D, Step 1 <p>Part B:</p> <ol style="list-style-type: none"> 1) Manage button is shown. → UC 33 <p>Part C:</p> <ol style="list-style-type: none"> 1) Create button is shown. → UC 32 <p>Part D:</p> <ol style="list-style-type: none"> 1) Leave button is shown. → UC 34
Samples	
Remarks	-
State	<p>04.11.2008</p> <p>Cancelled</p> <p>After the meeting with professor Liechti, we conclude that the time does not allow implementing the group functionalities. With the quest of puzzle pictures, the players have to group implicitly with other players. This grouping is a way to bypass the technical way of grouping with a natural way. There is another reason that the grouping functionalities are not possible. This is because that the asynchronous mode disallows retrieving in real time the location and uses it correctly in the game.</p>

B.2.32. Create a group for a quest

UC Number	32
Label	Create a group for a quest
Depends	UC 28
Priority	Medium
Complexity	Medium
Description	<p>Players can create a group to do quests. This possibility is a plus to accomplish tasks with other players and offer some richer game experience.</p> <p>Players can arrive in this UC from the map for another way of navigation.</p>
Actors	Player, Engine
Running	<p>Part A:</p> <ol style="list-style-type: none"> 1) Group is created 2) Player A sees a dialog with nearest players that can be able to group with him. (If player B is already known, this step is skipped → Step 3) <ol style="list-style-type: none"> a. Choose a player by clicking on player nickname → Step 3 b. Cancel → UC 28, Part A, Step 2 (or UC 33, Step 1 if player A comes from Manage group action). 3) Players see a dialog asking them if they want to invite the player B to join the group. <ol style="list-style-type: none"> a. Invite → Step 4 b. Cancel → Step 2 or UC 40, Step 2 4) Player A sees a waiting dialog. <ol style="list-style-type: none"> a. Cancel → Part C, Step 1 5) Player B current state screen is memorized. 6) Player B is notified to join player A group. <ol style="list-style-type: none"> a. Accept → Step 7 b. Refuse → Part B, Step 1 7) Player B is added to the group. <ol style="list-style-type: none"> a. Player B owns the quest → Step 9 b. Player B doesn't own the quest → Step 8 8) Quest is added to Player B quests diary. 9) Player A sees chosen player B answer and click Ok. 10) Player B returns to memorized screen. 11) Player A returns to → Step 2 or UC 40, Step 2 <p>Part B:</p> <ol style="list-style-type: none"> 1) Player A is notified that player B refuses his grouping invitation. 2) Player A return to the player choice after he clicks Ok. (or UC 40, Step 2)

	<p>3) Player B returns to memorized screen.</p> <p>Part C:</p> <p>1) Player B is notified that the player A cancelled his proposition.</p> <p>2) Player B returns to the memorized screen after he clicks on Ok.</p>
Samples Player A	
Samples Player B	
Remarks	<p>Group a quest is dependant of the same condition than to accept a quest proposed by a NPC. It is not possible to propose a quest that a player has not pre-requisites.</p> <p>Additional pre-requisites are necessary. To join a group, a player must not be in another group.</p> <p>Nearest players shown in player choice dialog are only players that they fit all conditions describes above.</p> <p>If player does not own the proposed quest, the quest is automatically added to his quests diary when he joins the group.</p>
State Cancelled	<p>04.11.2008</p> <p>After the meeting with professor Liechti, we conclude that the time does not allow implementing the group functionalities. With the quest of puzzle pictures, the players have to group implicitly with other players. This grouping is a way to bypass the technical way of grouping with a natural way. There is another reason that the grouping functionalities are not possible. This is because that the asynchronous mode disallows retrieving in real time the location and uses it correctly in the game.</p>

B.2.33. Manage group

UC Number	33
Label	Manage group
Depends	UC 28
Priority	Medium
Complexity	Medium
Description	When a player has created a group, he has the possibility to manage it. The management tasks available are Add Member, Remove Member and Dissolve the Group. Only group's owner can manage their group.
Actors	Player, Engine
Running	<ol style="list-style-type: none"> 1) Players see management actions available. <ol style="list-style-type: none"> a. Dissolve → UC 34 b. Add Member → UC 32, Step 2 c. Remove Member → UC 35 d. Cancel → UC 28, Part A, Step 2
Samples	
Remarks	-
State	<p>04.11.2008</p> <p>Cancelled After the meeting with professor Liechti, we conclude that the time does not allow implementing the group functionalities. With the quest of puzzle pictures, the players have to group implicitly with other players. This grouping is a way to bypass the technical way of grouping with a natural way. There is another reason that the grouping functionalities are not possible. This is because that the asynchronous mode disallows retrieving in real time the location and uses it correctly in the game.</p>

B.2.34. Dissolve a group

UC Number	34
Label	Dissolve a group
Depends	UC 33
Priority	Low
Complexity	Medium
Description	A group's owner can dissolve his group for any kind of reason. The procedure is very simple. The player must to click on dissolve and confirm the dissolution. After that, all grouping players are notified of the dissolution.
Actors	Player, Engine
Running	<ol style="list-style-type: none"> 1) Player A has to confirm their decision. <ol style="list-style-type: none"> a. Yes → Step 2 b. No → UC 33, Step 1 2) Group is dissolved 3) Notification is sent to each member of the group <ol style="list-style-type: none"> a. Ok → Returns to current memorized screen. 4) Player A sees a confirmation dialog and has to click Ok to returns to → UC 28, Part A, Step 2
Samples Player A	
Samples Members	
Remarks	-
State Cancelled	04.11.2008 After the meeting with professor Liehti, we conclude that the time does not allow implementing the group functionalities. With the quest of puzzle pictures, the players have to group implicitly with other players. This grouping is a way to bypass the technical way of grouping with a natural way. There is another reason that the grouping functionalities are not possible. This is because that the asynchronous mode disallows retrieving in real time the location and uses it correctly in the game.

B.2.35. Remove member

UC Number	35
Label	Remove member
Depends	UC 33
Priority	Low
Complexity	Medium
Description	<p>Group's owner has the possibility to remove member from their group. This possibility is useful to remove a member that is no more connected to the game or for any kind of reasons.</p> <p>Players can arrive from the map for another way of navigation.</p>
Actors	Player, Engine
Running	<ol style="list-style-type: none"> 1) Player A sees a dialog with members of his group <ol style="list-style-type: none"> a. Choose a player by clicking on player nickname → Step 2 b. Cancel → UC 33, Step 1 2) Player A sees a confirmation dialog. <ol style="list-style-type: none"> a. Yes → Step 3 b. No → Step 1 or UC 40, Step 2 3) Player B is removed from the group 4) Player B received a notification that he has been removed from the group. 5) Player A sees a confirmation dialog and has to click Ok → Step 1 or UC 40, Step 2 6) Player B returns to memorized screen state.
Samples Player A	
Samples Player B	
Remarks	-
State Cancelled	<p>04.11.2008</p> <p>After the meeting with professor Liechti, we conclude that the time does not allow implementing the group functionalities. With the quest of puzzle pictures, the players have to group implicitly with other players. This grouping is a way to bypass the technical way of grouping with a natural way. There is another reason that the grouping functionalities are not possible. This is because that the asynchronous mode disallows retrieving in real time the location and uses it correctly in the game.</p>

B.2.36. Leave a group

UC Number	36
Label	Leave a group
Depends	UC 28
Priority	Low
Complexity	Medium
Description	A player has the possibility to leave a group for his own reasons (by example to group with other players). For this, an action is available to permit leaving a group.
Actors	Player, Engine
Running	<ol style="list-style-type: none"> 1) Confirmation dialog is shown to the player A <ol style="list-style-type: none"> a. Yes → Step 3 b. No → UC 28, Part A, Step 2 2) Player A is removed from his current group 3) Player B (owner) is notified that the player A has left the group. <ol style="list-style-type: none"> a. Ok → returns to memorized screen state. 4) Player A sees a confirmation dialog and has to click Ok to return to → UC 28, Part A, Step 2
Samples Player A	
Samples Player B	
Remarks	-
State Cancelled	04.11.2008 After the meeting with professor Liechti, we conclude that the time does not allow implementing the group functionalities. With the quest of puzzle pictures, the players have to group implicitly with other players. This grouping is a way to bypass the technical way of grouping with a natural way. There is another reason that the grouping functionalities are not possible. This is because that the asynchronous mode disallows retrieving in real time the location and uses it correctly in the game.

B.2.37. Fight a monster

UC Number	37
Label	Fight a monster
Depends	UC 6
Priority	Low
Complexity	Medium
Description	This Use Case concern only Quest Mode. It is not available in Simple Mode. During the game, players can meet monsters and fight them. Monsters are visible on the map and when players are too near them, a fight is engaged.
Actors	Player, Engine
Running	<p>Part A:</p> <ol style="list-style-type: none"> 1) Player sees a dialog with a monster and player's life bar 2) Player is set as Attacker and monster as Defender 3) Attacker can choose (monster can only attack): <ol style="list-style-type: none"> a. Attack → Part B, Step 1 b. Steal → Part C, Step 1 c. Use an item → Part D, Step 1 d. Try to escape → Part E, Step 1 4) Evaluate fight ending: <ol style="list-style-type: none"> a. True → Part F, Step 1 b. False → Step 5 5) Switch Attacker/Defender → Step 3 <p>Part B:</p> <ol style="list-style-type: none"> 1) Determine if the attack successes <ol style="list-style-type: none"> a. True → Step 2 b. False → Step 6 2) Calculate Attacker points 3) Calculate Defender points 4) Compare Attacker and Defender points: <ol style="list-style-type: none"> a. Attacker "win" → Step 5 b. Defender "win" → Step 7 5) Calculate damage and show them → Part A, Step 4 6) Show "miss" message → Part A, Step 4 7) Show "parade" message → Part A, Step 4 <p>Part C:</p> <ol style="list-style-type: none"> 1) Calculate Attacker theft points 2) Calculate Defender theft points 3) Compare Attacker and Defender points: <ol style="list-style-type: none"> a. Theft fails → Step 11

	<p>b. Theft successes → Step 4</p> <p>4) Determine money or item to be stolen:</p> <p>a. Money → Step 5</p> <p>b. Item → Step 8</p> <p>5) Add money to the Attacker</p> <p>6) Remove money to the Defender</p> <p>7) Show result → Part A, Step 4</p> <p>8) Choose randomly an item from Defender inventory and remove it.</p> <p>9) Add chosen item to Attacker inventory</p> <p>10) Show result → Part A, Step 4</p> <p>11) Show “miss” message → Part A, Step 4</p> <p>Part D:</p> <p>1) Dialog is shown to choose an item (see remarks):</p> <p>a. Cancel → Part A, Step 3</p> <p>b. Chosen item → Step 2</p> <p>2) Attacker choose a target (by default, it is the opponent of Attacker):</p> <p>a. Ok → Step 3</p> <p>b. Cancel → Step 1</p> <p>3) Apply item effect on chosen target (see remarks)</p> <p>4) Remove item from Attacker inventory</p> <p>5) Show result (near correct target) → Part A, Step 4</p> <p>Part E:</p> <p>1) Calculate Attacker escape points</p> <p>2) Calculate Defender escape points</p> <p>3) Compare Attacker and Defender points</p> <p>a. Attacker successes → UC 6, Part A, Step 1</p> <p>b. Attacker fails → Part A, Step 4</p> <p>Part F:</p> <p>1) Determine if player has won or lose:</p> <p>a. Win → Step 4</p> <p>b. Lose → Step 2</p> <p>2) Change player state to “ghost” state</p> <p>3) Show the dead dialog</p> <p>a. Ok → UC 6, Part A, Step 1</p> <p>4) Calculate money gain</p> <p>5) Choose item gain</p> <p>6) Show combat rewards (see remarks):</p> <p>a. Ok → UC 6, Part A, Step 1</p>
Samples Attacker	
Samples Defender	

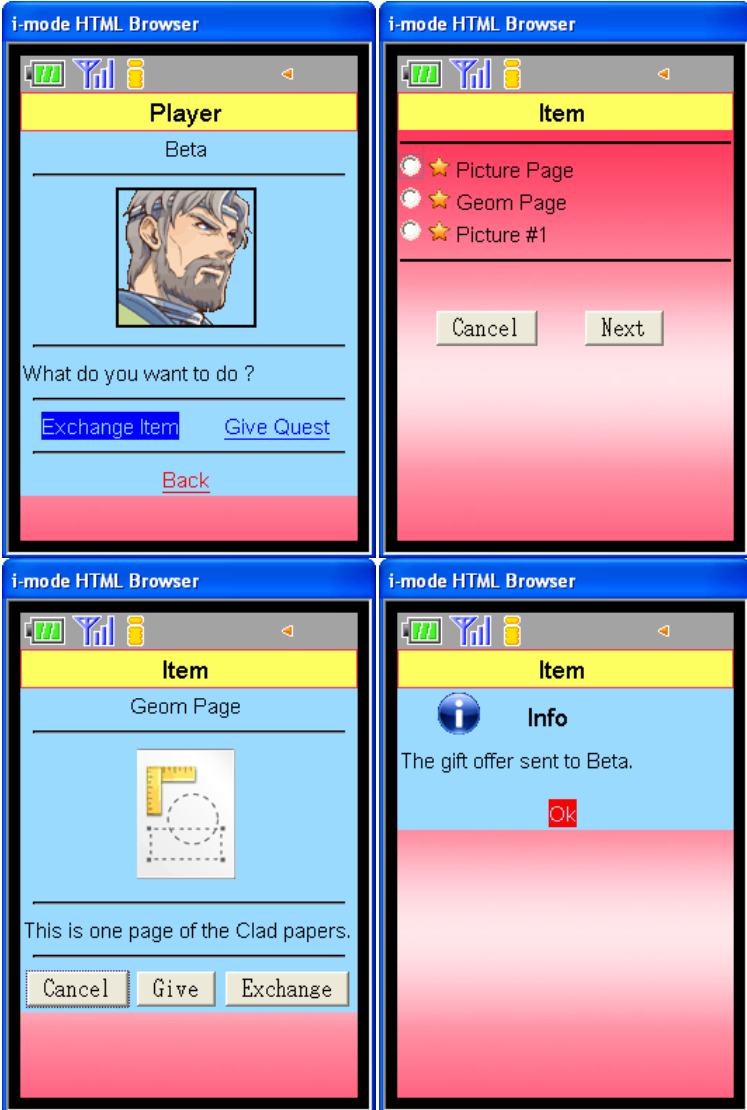
Remarks	<p>The dialog to choose an item show red lines. These red lines indicate items are not available to be used in combat or for other reasons.</p> <p>Items effects are not known in details at this time. We can see items as Attacker to apply their effects and use usual rules (attack solving...)</p> <p>The reward dialog can show only money or item indication if there is no item or money to gain.</p>
State Cancelled	<p>08.09.2008</p> <p>Due to the time for the project, a decision is necessary to drop some Use Cases. This Use Case could not be implemented during the project.</p>

B.2.38. Add annotation

UC Number	38
Label	Add annotation
Depends	UC 6
Priority	Low
Complexity	Low
Description	Players have a basic mechanism to add some text annotation on the map.
Actors	Player
Running	<ol style="list-style-type: none"> 1) Player sees a dialog to add text and has to write title and text annotation: <ol style="list-style-type: none"> a. Save → b. Cancel → UC 6, Part A, Step 1 2) Validation of non-empty fields are done: <ol style="list-style-type: none"> a. Correct → Step 4 b. Incorrect → Step 3 3) Error dialog is shown <ol style="list-style-type: none"> a. Ok → Step 1 4) Save text annotation on local device at the current player's position → UC 6, Part A, Step 1
Samples	
Remarks	The mechanism is deliberately basic with no interaction. This is for simplification reasons. Next version of the game could integrate a better framework for sharing text annotations.
State	This Use Case is abandoned at this time. It has to offer more functionality than actually proposed.

B.2.39. Exchange item from the map

UC Number	39
Label	Exchange item from the map
Depends	UC 6
Priority	Medium
Complexity	Medium
Description	Players can exchange items from the map. This Use Case describes how to choose an item before exchanging it. The player to exchange item is already chosen.
Actors	Player
Running	<p>Part A</p> <ol style="list-style-type: none"> 1) Player sees inventory categories and choose <ol style="list-style-type: none"> a. Normal, Equipments, Quests, Special → Step 2 b. Cancel → UC 6, Part F, Step 1 2) Player do an action: <ol style="list-style-type: none"> a. Next or Prev → Step 3 b. Choose an item → UC 20, Part A, Step 2 Step 4 c. Cancel → Step 1 UC 6, Part F, Step 1 3) Page of inventory change if it is possible → Step 2 4) Player sees item details and choose the quantity: <ol style="list-style-type: none"> a. Give → Part B, Step 1 b. Exchange → Part C, Step 1 c. Cancel → Step 2 5) Player returns to the map → UC 6, Step 1 <p>Part B</p> <ol style="list-style-type: none"> 1) Remove item(s) from player's inventory 2) Send notification of gift to the other player → Part A, Step 5 <p>Part C</p> <ol style="list-style-type: none"> 1) Remove item(s) from player's inventory 2) Send notification of exchange to the other player → Part A, Step 5

<p>Samples</p>	 <p>The figure displays four screenshots of the i-mode HTML Browser interface, arranged in a 2x2 grid. Each screenshot shows a different screen from the game's inventory or notification system.</p> <ul style="list-style-type: none"> Top-left: A screen titled "Player" for a character named "Beta". It features a character portrait and a question "What do you want to do?". Below the question are two buttons: "Exchange Item" and "Give Quest". A "Back" link is at the bottom. Top-right: A screen titled "Item" showing a list of items: "Picture Page", "Geom Page", and "Picture #1". Each item has a star icon. At the bottom are "Cancel" and "Next" buttons. Bottom-left: A screen titled "Item" for "Geom Page". It shows a small image of a document and the text "This is one page of the Clad papers.". At the bottom are "Cancel", "Give", and "Exchange" buttons. Bottom-right: A screen titled "Item" with an "Info" icon and the text "The gift offer sent to Beta.". An "OK" button is centered below the text.
<p>Remarks</p>	<p>-</p>
<p>State Finished</p>	<p>19.11.2008 The players can access to the inventory to choose the item they want use for exchange or gift. They can choose the quantity and click on Exchange or Give buttons to run the action relative to their choice. The notifications are sent to the other player. After all, the player returns to the map.</p>

B.2.40. View quests diary from map

UC Number	40
Label	View quest diary from map
Depends	UC 6
Priority	Medium
Complexity	Medium
Description	Players can do some actions on quests from the map when they chosen a player to interact with him. This is the starting point of these interactions.
Actors	Player
Running	<ol style="list-style-type: none"> 1) Player chooses a category <ol style="list-style-type: none"> a. Category Quest → Step 3 b. Cancel → UC 6, Part F, Step 1 2) Player sees a list of quest corresponding to the selected category. <ol style="list-style-type: none"> a. One page → Step 4 b. More than one page → Step 3 3) He can click on Next or Prev buttons to show next or previous page of the current quests diary category. He can repeat this action as much as they are available pages. 4) Player can choose Back or Map button <ol style="list-style-type: none"> a. Back → Step 1 b. Map → UC 6, Step 1 c. Quest → UC 41, Step 1
Samples	
Remarks	When a quest has more than one part, each part is visible as a sub quest of the principal quest.
State	22.11.2008
Cancelled	This use case is no more possible because of the change in the Use Cases concerning the quests. There are no more grouping actions. There is only the Give quest action available from the map described in the UC 45.

B.2.41. View actions for a quest from the map

UC Number	41
Label	View actions for a quest from the map
Depends	UC 40
Priority	Medium
Complexity	Medium
Description	Players can interact with other player to share or give a quest. This Use Cases is the starting points for these interactions from the map view. It allows navigating by another way than the menu and quest diary.
Actors	Player
Running	<p>Part A:</p> <ol style="list-style-type: none"> 1) Player views a dialog with quest details. 2) He can: <ol style="list-style-type: none"> a. Back → UC 40, Step 2 b. Action → Step 3 c. Prev → Step 4 d. Next → Step 5 3) Depending on Quest Category <ol style="list-style-type: none"> a. Current → Part B, Step 1 b. History → Part C, Step 1 4) Players view previous information of the quest if available → Step 2 5) Players view next information of the quest if available → Step 2 <p>Part B:</p> <ol style="list-style-type: none"> 1) Choose an action to do <ol style="list-style-type: none"> a. Give → UC 30, Part A, Step 2 b. Create → UC 32, Step 1 c. Invite → UC 32, Step 3 d. Remove → UC 35, Step 2 e. Cancel → Part A, Step 3 <p>Part C:</p> <ol style="list-style-type: none"> 1) Choose an action to do <ol style="list-style-type: none"> a. Give → UC 30, Part A, Step 2 b. Cancel → Part A, Step 3
Remarks	-
State	22.11.2008
Cancelled	This use case is no more possible because of the change in the Use Cases concerning the quests. There are no more grouping actions. There is only the Give quest action available from the map described in the UC 45.

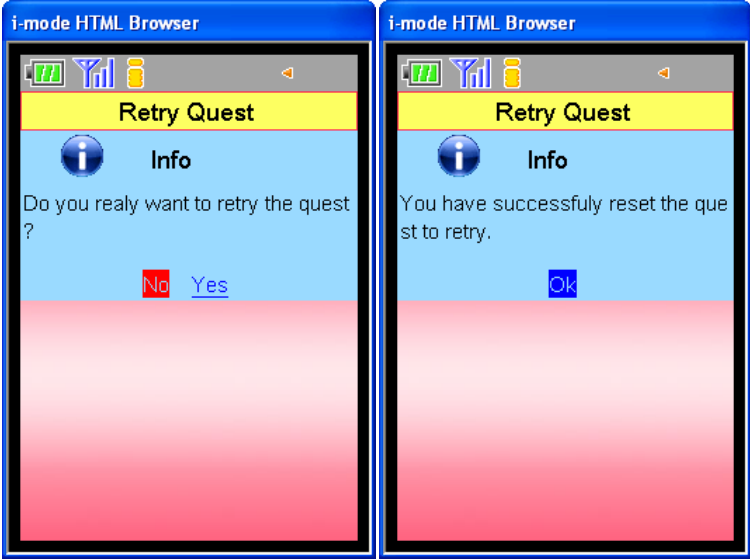
B.2.42. Equip from inventory

UC Number	42
Label	Equip from inventory
Priority	Low
Complexity	Low
Description	The player can equip his avatar from inventory. This is for a different navigation than the menu navigation.
Actors	Player, Engine
Running	<ol style="list-style-type: none"> 1) A dialog with actual characteristics values and possible values with equipment is shown <ol style="list-style-type: none"> a. Cancel → UC 18, Step 1 b. Equip → Step 2 2) Remove current equipment if necessary 3) Equip the new piece of equipment at the right place → UC 18, Step 1
Samples	
Remarks	-
State	08.09.2008 Cancelled Due to the time for the project, a decision is necessary to drop some Use Cases. This Use Case could not be implemented during the project.

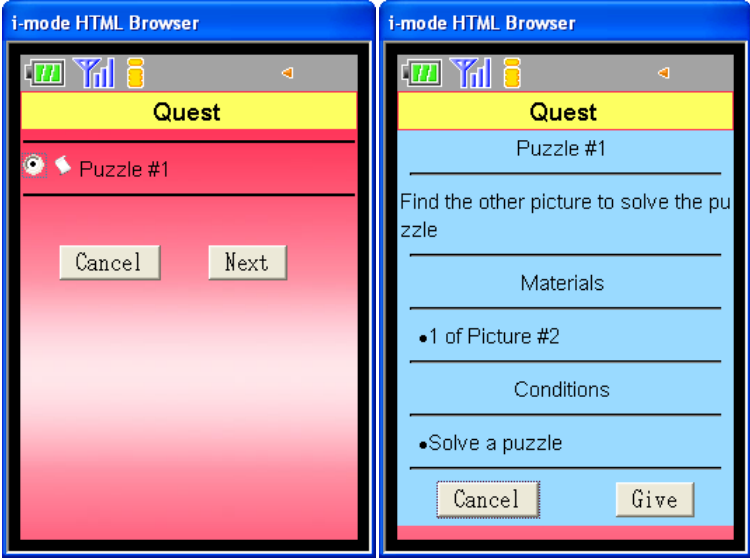
B.2.43. Unequip from inventory

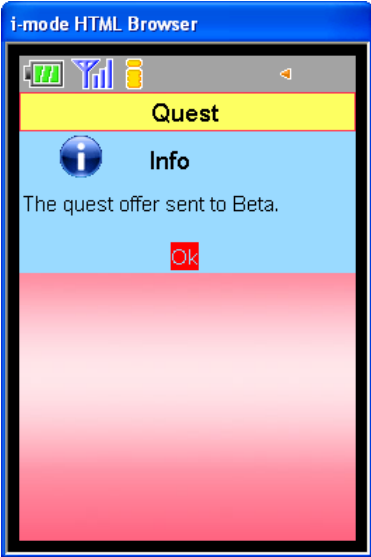
UC Number	43
Label	Unequip from inventory
Priority	Low
Complexity	Low
Description	The player can unequip his avatar from inventory. This is for a different navigation than the menu navigation.
Actors	Player, Engine
Running	<ol style="list-style-type: none"> 1) A dialog with actual characteristics values and possible values with no equipment is shown <ol style="list-style-type: none"> a. Cancel → UC 18, Step 1 b. Unequip → Step 2 2) Remove current equipment → UC 18, Step 1
Samples	
Remarks	-
State	08.09.2008 Cancelled Due to the time for the project, a decision is necessary to drop some Use Cases. This Use Case could not be implemented during the project.

B.2.44. Retry a quest

UC Number	44
Label	Retry a quest
Priority	Low
Complexity	Low
Description	The player can retry a quest in the state of Abandoned or Failed. The quest will reset and the player can redo it.
Actors	Player, Engine
Running	<ol style="list-style-type: none"> 1) A confirmation dialog is shown to the player <ol style="list-style-type: none"> a. Yes → Step 2 b. No → UC 28, Part A, Step 1 2) The quest is reset 3) A confirmation dialog is shown to confirm the reset. <ol style="list-style-type: none"> a. Ok → UC 27, Part A, Step 1
Samples	
Remarks	-
State	22.11.2008
Finished	Players can discard a quest in the allowed state. The quest state is updated. The dialogs are correctly shown to the player.

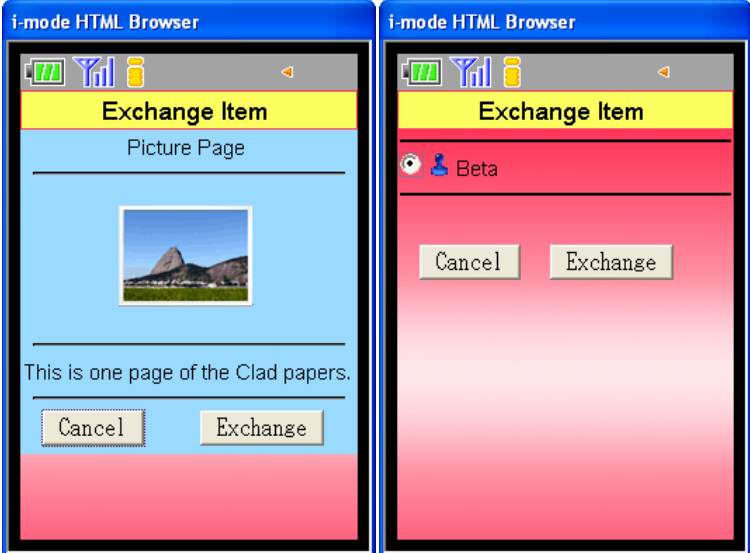
B.2.45. Give a quest from the map

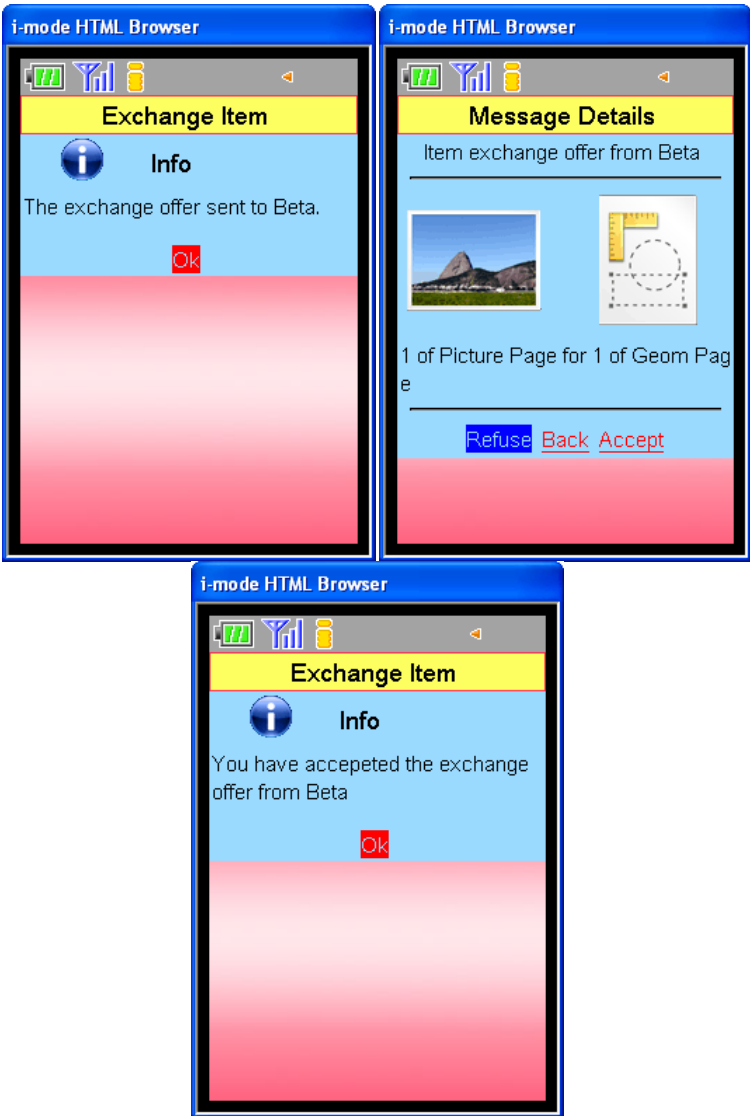
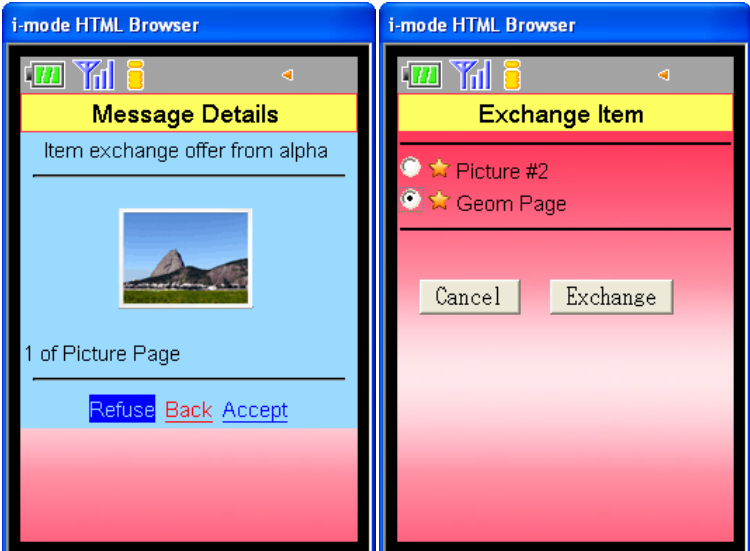
UC Number	45
Label	Give a quest from the map
Priority	Low
Complexity	Low
Description	Directly from the map, a player can choose another player and give a quest. This is particularly useful to avoid the navigation by the quest diary.
Actors	Player, Engine
Running	<ol style="list-style-type: none"> 1) Player A clicks on the action Give Quest from a player's details from the map. 2) He sees a list of quest available to give to the player B 3) He can: <ol style="list-style-type: none"> a. Choose a quest and click on Give → Step 4 b. Cancel → Step 1 4) He sees the quest details <ol style="list-style-type: none"> a. Give → Step 5 b. Cancel → Step 3 5) A notification is sent to the player B for the quest gift offer 6) The player A returns → UC 6, Step 1
Samples	

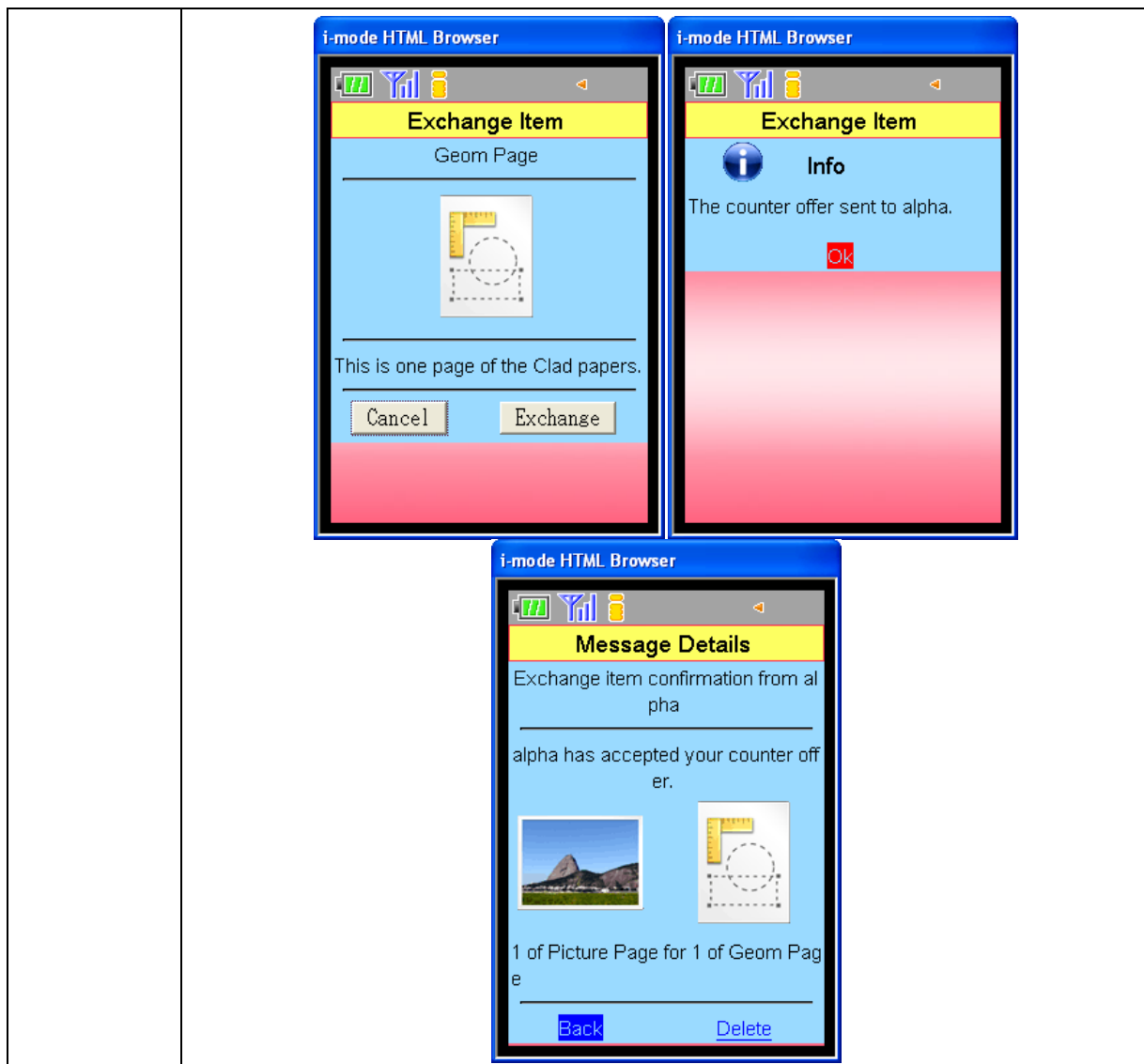
	
Remarks	-
State Finished	23.11.2008 This Use Case is completely implemented. Now, players can give a quest from the map screen through the player's details. The process also checks if the player chosen does not already have the quest in his diary. The end of the give quest process is the same than for give a quest from the quest diary.

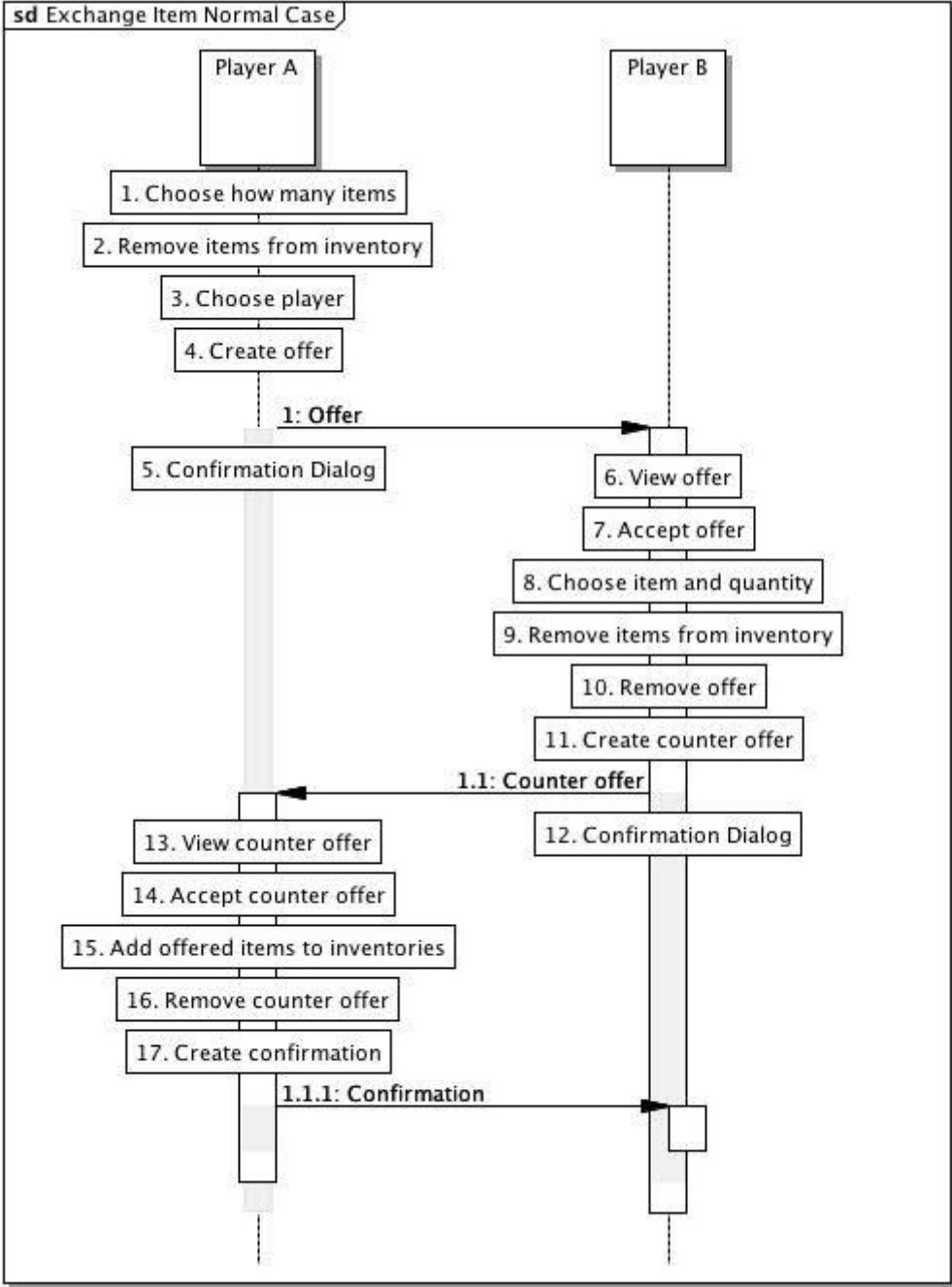
B.2.46. Exchange item

UC Number	46
Label	Exchange Item
Depends	UC 18
Priority	Medium
Complexity	High
Description	<p>Players can exchange items with other players during the game. To exchange items, players must to be near other players.</p> <p>There are two starting points to come in this Use Case. Depending on these starting points, this use case can change for the returning points.</p>
Actors	Player, Engine
Running	<p>Player A</p> <p>Part A:</p> <ol style="list-style-type: none"> 1) Player A sees a dialog asking them for how many items of the current sort they want to propose. (See remarks). <ol style="list-style-type: none"> a. Exchange → Part A, Step 2 b. Cancel → UC 18, Step 1 2) Player A sees a dialog with nearest players that can be able to exchange items with. <ol style="list-style-type: none"> a. Choose a player B by clicking on player nickname → Part A, Step 3 b. Cancel → Part A, Step 1 3) Items offered are removed from the Player A inventory. 4) Player A sees a confirmation dialog that he sent an offer to the player B. <p>Part B:</p> <ol style="list-style-type: none"> 1) Player A wait the counter offer of the player B (messages screen) 2) Player A can: <ol style="list-style-type: none"> a. Accept → Both Players, Part A, Step 1 b. Refuse → Both Players, Part B, Step 1 <p>Player B</p> <p>Part A:</p> <ol style="list-style-type: none"> 1) Player B goes to the messages screens to see the message of the offer done by the player A. 2) Player B can choose: <ol style="list-style-type: none"> a. Back → Step 1 (Messages screen) b. Accept → Part B, Step 1 c. Refuse → Part C, Step 1

	<p>Part B:</p> <ol style="list-style-type: none"> 1) Player B sees the list of item to choose for the counter offer. <ol style="list-style-type: none"> a. Cancel → Part A, Step 2 b. Exchange → Step 3 2) Items of the counter offer are removed from the Player B inventory. 3) Player B sees a dialog confirmation indicates that the counter offer is sent to Player A 4) The offer is removed from the messages. → Part A, Step 1 <p>Part C:</p> <ol style="list-style-type: none"> 1) Items are placed again in the inventory of Player A 2) A message is sent to the Player A 3) The offer is removed from the messages. → Part A, Step 1 <p>Both Players:</p> <p>Part A:</p> <ol style="list-style-type: none"> 1) Items offered by Player A are added to the Player B inventory. 2) Items offered by Player B are added to the Player A inventory → Part C, Step 1 <p>Part B:</p> <ol style="list-style-type: none"> 1) Items offered by player B are added to the Player B inventory. 2) Items offered by player A are added to the player A inventory → Part C, Step 1 <p>Part C:</p> <ol style="list-style-type: none"> 1) Counter offer is removed (from Player A). 2) A confirmation message is sent to the Player B. 3) Player A return → Player A, Part B, Step 1
<p>Samples Player A</p>	

	
<p>Samples Player B</p>	



Sequence Diagram	 <pre> sequenceDiagram participant A as Player A participant B as Player B A->>A: 1. Choose how many items A->>A: 2. Remove items from inventory A->>A: 3. Choose player A->>A: 4. Create offer A->>B: 1: Offer B->>B: 6. View offer B->>B: 7. Accept offer B->>B: 8. Choose item and quantity B->>B: 9. Remove items from inventory B->>B: 10. Remove offer B->>B: 11. Create counter offer B->>A: 1.1: Counter offer A->>A: 13. View counter offer A->>A: 14. Accept counter offer A->>A: 15. Add offered items to inventories A->>A: 16. Remove counter offer A->>A: 17. Create confirmation A->>B: 1.1.1: Confirmation B->>B: 12. Confirmation Dialog A->>A: 5. Confirmation Dialog </pre>
Remarks	When a player has only one item of an item type, he does not need to select the quantity.
State Finished	14.11.2008 All this Use Case is finished. The players can exchange items like it is described in this Use Case. The player A chooses an item, quantity and a player. He sends the offer. The player B can accept or refuse. The player A can accept or refuse the offered item in exchange. The refuse options are available and cancel too. Notifications (messages) are sent to the players corresponding to the state of the process.

	<p>17.11.2008 The bug for the quantity not memorized when the player come back after a cancel action is corrected. This Use Case is totally finished.</p>
--	---------------------------------------------------------------------------------------------------------------------------------------------------------------

B.3. Conclusion

This appendix covers the state of development for the players' functionalities. There are some reports for each Use Case to follow the difficulties and the word progression during the project. This is a good base to continue the development of the KMEP game.

heig-vd

Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

K-MEP

Configuration

Laurent Prévost

The logo for RITSUMEIKAN, featuring a large white letter 'R' on the left and the word 'RITSUMEIKAN' in a smaller, white, sans-serif font to its right, all set against a dark red rectangular background.

R RITSUMEIKAN

Table of Contents

C.1. INTRODUCTION	327
C.1.1. Distribution	327
C.1.1.1. Common project	327
C.1.1.2. EJB project	327
C.1.1.3. WAR Project	327
C.2. PROPERTIES FILES	328
C.2.1. Legends	328
C.2.2. technicalMessages	328
C.2.3. userMessages	328
C.2.4. inTrackConfig	328
C.2.5. serverConfig	329
C.2.6. application	330
C.2.7. iModeConfig	331
C.2.8. iModeText	335
C.2.9. serviceLookup	336
C.3. XML CONFIGURATION FILES	337
C.3.1. Legends	337
C.3.2. KMEP-EJB – Game Data	337
C.3.3. Game data XML files	346
C.3.3.1. loaders	346
C.3.3.2. dialogs	347
C.3.3.3. dialogstates	353
C.3.3.4. firstquests	354
C.3.3.5. items	355
C.3.3.6. itemsources	355
C.3.3.7. itemtypes	356
C.3.3.8. messages	357
C.3.3.9. multiitemtypes	363
C.3.3.10. npcs	364
C.3.3.11. players	365
C.3.3.12. quests	366
C.3.3.13. questmaterials	367
C.3.3.14. questconditions	368
C.3.3.15. questrewards	370
C.3.3.16. queststates	371
C.3.3.17. statistics	373
C.3.4. KMEP-WAR – Model View Controller	375
C.4. CONCLUSION	377

Table of Figures

FIGURE 138 : ABSTRACTBEHAVIOUR XML SCHEMA	338
FIGURE 139 : DIALOGBEHAVIOUR XML SCHEMA.....	338
FIGURE 140 : DIALOGSTATEBEHAVIOUR XML SCHEMA	339
FIGURE 141 : ITEMBEHAVIOUR XML SCHEMA	339
FIGURE 142 : ITEMINVENTORYBEHAVIOUR XML SCHEMA.....	339
FIGURE 143 : ITEMSOURCEBEHAVIOUR XML SCHEMA	340
FIGURE 144 : MESSENGERBEHAVIOUR XML SCHEMA	340
FIGURE 145 : NPCBEHAVIOUR XML SCHEMA	340
FIGURE 146 : PLAYERBEHAVIOUR XML SCHEMA	341
FIGURE 147 : QUESTDIARYBEHAVIOUR XML SCHEMA	341
FIGURE 148 : STATISTICBEHAVIOUR XML SCHEMA.....	341
FIGURE 149 : KMPEENTITY XML SCHEMA.....	344
FIGURE 150 : LOADERS XML SCHEMA	346
FIGURE 151 : ABSTRACTDIALOG IN DIALOGS	347
FIGURE 152 : ABSTRACTCONDITION IN DIALOGS	347
FIGURE 153 : DIALOGTEXT XML SCHEMA.....	348
FIGURE 154 : DIALOGQUEST XML SCHEMA	348
FIGURE 155 : DIALOGQUESTFINISH XML SCHEMA	348
FIGURE 156 : DIALOGQUESTION XML SCHEMA	349
FIGURE 157 : DIALOGQUESTIONPUZZLE XML SCHEMA	349
FIGURE 158 : DIALOGITEM XML SCHEMA.....	349
FIGURE 159 : DIALOGS XML SCHEMA.....	350
FIGURE 160 : DIALOGSTATES XML SCHEMA.....	353
FIGURE 161 : FIRSTQUESTS XML SCHEMA.....	354
FIGURE 162 : ITEMS XML SCHEMA.....	355
FIGURE 163 : ITEMSOURCES XML SCHEMA	355
FIGURE 164 : ITEMTYPES XML SCHEMA.....	356
FIGURE 165 : ABSTRACTMESSAGE XML SCHEMA	357
FIGURE 166 : ITEMIDS XML SCHEMA	357
FIGURE 167 : MESSAGEQUESTGIFT XML SCHEMA.....	357
FIGURE 168 : MESSAGEITEMEXCHANGE XML SCHEMA.....	358
FIGURE 169 : MESSAGEITEMEXCHANGECONFIRM XML SCHEMA	358
FIGURE 170 : MESSAGEITEMGIFT XML SCHEMA	359
FIGURE 171 : MESSAGEITEMGIFTCONFIRM XML SCHEMA	359

FIGURE 172 : MESSAGEQUESTGIFTCONFIRM XML SCHEMA	360
FIGURE 173 : MESSAGES XML SCHEMA	360
FIGURE 174 : MULTIITEMTYPES XML SCHEMA	363
FIGURE 175 : NPCS XML SCHEMA	364
FIGURE 176 : PLAYERS XML SCHEMA	365
FIGURE 177 : QUESTS XML SCHEMA.....	366
FIGURE 178 : ABSTRACTQUESTMATERIAL XML SCHEMA.....	367
FIGURE 179 : MATERIALS XML SCHEMA.....	368
FIGURE 180 : ABSTRACTQUESTCONDITION XML SCHEMA.....	368
FIGURE 181 : CONDITIONS XML SCHEMA	369
FIGURE 182 : ABSTRACTQUESTREWARD XML SCHEMA	370
FIGURE 183 : REWARDS XML SCHEMA.....	370
FIGURE 184 : ABSTRACTQUESTCONDITIONSTATE XML SCHEMA	371
FIGURE 185 : QUESTCONDITIONSTATES XML SCHEMA.....	371
FIGURE 186 : QUESTSTATES XML SCHEMA	371
FIGURE 187 : ABSTRACTSTATISTIC XML SCHEMA	373
FIGURE 188 : COUNTABLE XML SCHEMA	373
FIGURE 189 : STATISTICS XML SCHEMA	373
FIGURE 190 : MAPPINGS ROOT XML SCHEMA.....	375
FIGURE 191 : MAPPINGS ACTION XML SCHEMA	375

Table of Tables

TABLE 3 : INTRACKCONFIG PROPERTIES	329
TABLE 4 : SERVERCONFIG PROPERTIES	330
TABLE 5 : APPLICATIONS PROPERTIES.....	331
TABLE 6 : IMODECONFIG PROPERTIES	334
TABLE 7 : IMODETEXT PROPERTIES.....	335
TABLE 8 : SERVICELOOKUP PROPERTIES.....	336
TABLE 9 : BEHAVIOURS TAGS	344
TABLE 10 : KMEPENTITY TAGS.....	345
TABLE 11 : LOADERS TAGS	347
TABLE 12 : DIALOGS TAGS.....	353
TABLE 13 : DIALOGSTATES TAGS.....	354
TABLE 14 : FIRSTQUESTS TAGS	355
TABLE 15 : ITEMS TAGS	355
TABLE 16 : ITEMSOURCES TAGS.....	356
TABLE 17 : ITEMTYPES TAGS.....	357
TABLE 18 : MESSAGES TAGS	362
TABLE 19 : MULTIITEMTYPES TAGS	364
TABLE 20 : NPCS TAGS	364
TABLE 21 : PLAYERS TAGS	366
TABLE 22 : QUESTS TAGS	367
TABLE 23 : QUESTMATERIALS TAGS.....	368
TABLE 24 : QUESTCONDITIONS TAGS.....	369
TABLE 25 : QUESTREWARDS TAGS	370
TABLE 26 : QUESTSTATES TAGS	372
TABLE 27 : STATISTICS TAGS	374
TABLE 28 : MAPPINGS TAGS	377

C.1. Introduction

This appendix offers a view of the game configuration. There is a lot of configuration file for each application part. Each configuration file used in KMEP will be explained.

There are two types of configuration file. The first one is the simplest one with the system of properties files offered by Java itself and allow the language management. The second type is the usage of XML files with XML Schema. The second one is more complicated but allows a better control of the configuration data.

C.1.1. Distribution

The configuration is distributed in the different part of the project. Actually, there are two principal projects and a reinforcement project (the classes that are common between projects). Each one has its own configuration files.

C.1.1.1. Common project

The common project uses only properties files to handle the configuration. In this project, there are only two files to represent the technical and user messages corresponding to the errors encountered during the program execution.

C.1.1.2. EJB project

This project uses the both mechanism of configuration. There is some configuration to handle some simple data configuration and the complex data for the game data are composed by XML files.

C.1.1.3. WAR Project

For this project, there are many different configuration files in the two ways of configuration. There is also a TLD file to define the iMode tags. The properties files handle simple configuration data and externalize the strings used in the web user interface. The XML and XML Schema files allow configuring the part of MVC for the iMode application that requires a better granularity of configuration.

C.2. Properties files

This section describes each properties file and its usage in the correct project.

C.2.1. Legends

The next tables needs some code of colors and presentation to focus on some relevant data. This paragraph explain the different codes used in the next tables.

Light blue:

Define the different sections in a property file.

C.2.2. technicalMessages

The technicalMessages properties file offers the possibility to manage the technical sentences in case of problems during the execution. Each message in this file corresponds to an error code describes in the “Error Codes” documentation. The other fact to signal is the class named “ReasonProvider” that contains all the error codes constants. Each constant must to have a corresponding message in the technical message file.

The format in the file is very simple. There is a key and an associate value. The key follow this rule: msg.<abs(error_code)>=<value>. For example: msg.1=Unknown Exception.

The messages in this file are for the developer and are not shown to the user in general. For the user, another file follows the same rules. This file takes place in the “KMEP-Common” in the “Resources Files”.

C.2.3. userMessages

The userMessages properties file offers the possibility to manage the user sentences in case of problems during the execution. This file has exactly the same construction than the technicalMessages. All that is true for the technicalMessages are true for this file too.

C.2.4. inTrackConfig

The inTrackConfig properties file is used to configure the inTrack usage in the project. With the configuration values, this is possible to switch from inTrackPartner prod environment to inTrackPartner dev environment. The Table 3 shows the different values.

Key	Value	Description
InTrack Configuration		
intrack.usage	true	Defines if the inTrack platform is used or not. This value is especially used for development purpose.

Key	Value	Description
intrack.dev	false	Defines which environment is used for the inTrackPartner platform.
Dev Platform Configuration		
Intrack.dev.user	kmep_dev	The user used to clean the data in inTrackPartner.
Intrack.dev.pwd	n874Fdb8qSo904W	The password used to clean the data in inTrackPartner.
Intrack.dev.moduleid	<moduleid>	Define the tracking module id in inTrack for the KMEP project. Actually, the value is: INTRACK_ASSIGNED_ID_KMEP_DEV
Intrack.dev.trackingidprefix	INTRACK_KMEP_DEV_	Define the prefix used to define all the tracking ids.
Prod Platform Configuration		
Intrack.prod.user	kmep_prod	The user used to clean the data in inTrackPartner.
Intrack.prod.pwd	fy5jxKHLGYyiqT0	The password used to clean the data in inTrackPartner.
Intrack.prod.moduleid	<moduleid>	Define the tracking module id in inTrack for the KMEP project. Actually, the value is: INTRACK_ASSIGNED_ID_KMEP_PROD
Intrack.prod.trackingidprefix	INTRACK_KMEP_PROD_	Define the prefix used to define all the tracking ids.

Table 3 : inTrackConfig properties

This document of properties takes place in the “KMEP-EJB” and especially in the “Resources” files.

C.2.5. serverConfig

The serverConfig property file contains all the configuration data to set up the server. The Table 4 shows all the configuration value with the default (original) value and the description of the usage of the value.

Key	Value	Description
Path Configuration		
path.xsd	ch/kmep/xsd/	The path to access to the XSD files.

Key	Value	Description
path.xml	ch/kmep/xml/	The path to access to the XML files.
Extensions Configuration		
ext.xsd	.xsd	Extension for the XSD files.
ext.xml	.xml	Extension for the XML files.
XSD files Configuration		
xsd.loaders	loaders	This is the XML Schema for the loaders configuration.
Session Configuration		
session.duration	600'000	This is the duration time of a player session on the game. After this delay, the session will be deleted. The value is in milliseconds and represents 10 minutes.
session.hash	2	This is the hash mechanism used to hash the players' password. The numeric value corresponds to the ordinal value of the enumeration "EHashType" presents in the "KMEP-Common".
Entity Configuration		
entity.radius	0.050	Define the default radius to use when a new entity is created. The radius is used to retrieve the nearest entities for the map for instance.

Table 4 : serverConfig properties

This document of properties takes place in the "KMEP-EJB" and especially in the "Resources" files.

C.2.6. application

This file contains the application WAR properties. In this file, there are all the actions available in the WAR project and an option to enable or disable some additional dialog boxes for the player. The Table 5 shows these properties.

Key	Value	Description
Debug Configuration		
debug.dialog	false	Indicates if the additional dialog boxes have to be shown or not to the user.
Data Load Configuration		
data.init	rits_loaders	Define the base file to load the game data into the persistence.

Key	Value	Description
Action Configuration		
act.<name>	<avalue>	The <name> is replaced by a name of the action and the value corresponds to the action configured in the "iModeActions.xml".

Table 5 : applications properties

This file takes place in the "Resources" folder of the "KMEP-WAR".

C.2.7. iModeConfig

This properties file configuration contains the entire basic configuration for the iMode Web User Interface application. The Table 6 shows the properties.

Key	Value	Description
XML Files		
xml.mapping	/ch/kmep/xml/iModeActions.xml	Define the path and filename to the mapping configuration for the iMode actions.
XSD Files		
xsd.mapping	/ch/kmep/xsd/iModeActions.xsd	Define the path and filename to the XML Schema for the validation of the mapping XML files.
iMode Application Configuration		
imode.basepath	imode	Define the base path for the construction of URL.
imode.doctype	text/xhtml+xml	Define the content type header sent when a response is sent to the player. It seems to have some troubles to include a resource string into the JSP tag "Page ContentType".
Background Pictures Configuration		
bg.img	img/screen/bg.jpg	The background pictures of the application.
bg.text	img/screen/bg_txt.gif	The splash screen text.
Screen Configuration		
size.maxwidth	240px	Define the maximum screen size for the width.

Key	Value	Description
size.maxheight	320px	Define the maximum screen size for the height.
iMode Sub Views Configuration		
Views + .<cat>.<subcat>.<name>	<path>/<filename>	Define the sub views specific to the iMode application.
Dialog Boxes Configuration		
dialog.status.ext	.gif	Define the extension file for the icons to show in the dialog boxes.
dialog.status.path	img/dialog	Define the path to access to the dialog boxes pictures.
dialog.status.width	32px	Define the width of the dialog boxes icons.
dialog.status.height	32px	Define the height of the dialog boxes icons.
NPC Dialogs Configuration		
npcdialog.bg	img/dialog/bg_dialog.jpg	The background image for the NPC dialog screens.
Character Configuration		
characters.smalldir	img/characters/small	Define the path to the directory that contains the small pictures of avatars (for players and/or NPC).
characters.bigdir	img/characters/big	Define the path to the directory that contains the big pictures of avatars (for players and/or NPC).
characters.ext	.gif	Define the file type for characters pictures.
Item Configuration		
item.smalldir	img/item/small	Define the path to the directory that contains the small pictures of items.
item.bigdir	img/item/big	Define the path to the directory that contains the big pictures of item.
item.ext	.gif	Define the file type for items pictures.

Key	Value	Description
Map Configuration		
map.size	208px	Define the width and height of the map. [Not used actually]
map.icons.up	img/map/up.gif	Define the icon to the up direction. [Not used actually]
map.icons.down	img/map/down.gif	Define the icon to the down direction. [Not used actually]
map.icons.left	img/map/left.gif	Define the icon to the left direction. [Not used actually]
map.icons.right	img/map/right.gif	Define the icon to the right direction. [Not used actually]
map.icons.zoomin	img/map/zoomin.gif	Define the icon to the zoom in. [Not used actually]
map.icons.zoomout	img/map/zoomout.gif	Define the icon to the zoom out [Not used actually]
map.icons.player	img/map/player.gif	Define the icon to represent a player.
map.icons.item	img/map/item.gif	Define the icon to represent an item.
map.icons.itemsource	img/map/item.gif	Define the icon to represent an item source.
map.icons.npc	img/map/npc.gif	Define the icon to represent a NPC.
map.icons.quest	img/map/quest.gif	Define the icon to represent a quest.
map.icons.message	img/map/message.gif	Define the icon to represent a message.
map.icons.size	16px	Define the icon size for the map icons. [Not used actually]
Colors Configuration		
color.bgdialog	#9adaff	Define the color for the background of the dialog boxes.
color.tab.inactive	#dafda7	Define the color for an inactive tab in tabmenu.
color.tab.active	#61afff	Define the color for an active tab in tabmenu.

Key	Value	Description
color.tab.bg	#61afff	Define the color for the tabmenu background.
color.menu	#61ffa0	Define the color for a menu element.
color.title	#fdff61	Define the color of page titles.
color.quest.condition.complete		Define the color used to present a quest condition reached.
color.quest.condition.notcomplete		Define the color used to present a quest condition not reached.
color.quest.condition.no color	#000000	Define the color used when there is no other possibility for a quest condition.
Pagers Configuration		
pager.inventory	9	Define the number of elements shown on one page inventory.
pager.questdiary	9	Define the number of elements shown on one page quest diary.
pager.messenger	9	Define the number of elements shown on one page message inbox.
pager.playerchoice	9	Define the number of elements shown on one page player choice.
pager.itemchoice	9	Define the number of elements shown on one page item choice.
pager.questchoice	9	Define the number of elements shown on one page quest choice.

Table 6 : iModeConfig properties

This file takes place into the “Resources” folder in the “KMEP-WAR”.

C.2.8. iModeText

This properties file configuration contains all the visible strings that the player can view during the play. These strings are specific to the iMode application. Strings are grouped by categories. The

Key	Value	Description
Labels Text		
lb.<name>	<value>	These strings represent, in general, the labels associate with a form field. The <name> and <value> depends on the context.
Titles Text		
title.<name>	<value>	Define the strings that represent the page title. The <name> and <value> depends on the context.
Message Title Text		
msg.title. <name>	<value>	Define the strings that represent the messages title. The <name> and <value> depends on the message.
Button Text		
bt. <name>	<value>	Define the string for the forms' button. The <name> and <value> depends on the form.
Links Text		
Ink.<name>	<value>	Defines the strings used to show a link in general. The <name> and <value> depends on the context.
Ink.tab.<category>.<name>	<value>	Define the strings that represent the tab text link in tabbed menu. The <category> represents the name of the tabbed menu. The <name> and <value> depends on the context.
Messages Text		
msg. <prefix>.<name>	<value>	Define short messages to show to the player. The <name> and <value> depends on the context. The <prefix> allows adding hierarchical logic into the naming of messages.

Table 7 : iModeText properties

This file takes place in the "Resources" folder of the "KMEP-WAR".

C.2.9. serviceLookup

This file contains the bindings between names of service and JNDI addresses for the lookup of service. This configuration file helps to avoid changing names of the services. The Table 8 shows the properties.

Key	Value	Description
Service Configuration		
service.<name>	<address>	Define the address to reach a service of the application. The service configured here are the services from KMEP-EJB.

Table 8 : serviceLookup properties

This file takes place in the “Resources” folder of the “KMEP-WAR”.

C.3. XML Configuration files

This part presents all the XML files by their XML Schema. Each Schema is described with their tags.

There are two big categories of XML configuration files:

- KMEP-EJB – Game data
- KMEP-WAR – iMode Model View Controller

C.3.1. Legends

The next tables needs some code of colors and presentation to represent some relevant data. This paragraph explain the different codes used in the next tables.

Light blue:	Definition of a complex type
Light orange:	Allows the choice between one or more consecutive tags. At least, one of them must be chosen.
Light red:	Define a root tag of document or principal element.
Green annotation:	Define a tag that inherits from a parent tag element.

C.3.2. KMEP-EJB – Game Data

This part of XML configuration allows loading the game data to the game server at the installation. The loading is automated with a servlet located in KMEP-WAR. There is a XML file to prepare a bunk data loading. All the XML files in this part use some common XML Schema files. These files are not directly usable by themselves:

- base.xsd
- behaviour.xsd
- kmepentity.xsd

All the XML Schema for this part takes place into the “ch/kmep/xsd” resources directory into the “KMEP-EJB”. The XML files take place into the “ch/kmep/xml” resources directory in the same project.

There is an important point to notice. In all the XML Schema documents, you can see some id fields. These ids are used only for the loading phase. They are replaced by other generated id by the application. They are called XML ids and they must be unique between all the files for a same category of data.

base.xsd

This XML Schema contains the commons elements usable in other files like identifiers or these kinds of definitions.

behaviours.xsd

This XML Schema contains the definitions for the different behaviors used in the game. The behaviors are used by different mobile entities. The Figure 138 shows the base for all the next behaviors.

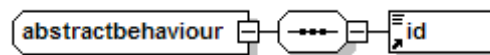


Figure 138 : AbstractBehaviour XML Schema

There is the list of behaviors:

- DialogBehaviour
- DialogStateBehaviour
- ItemBehaviour
- ItemInventoryBehaviour
- ItemSourceBehaviour
- NPCBehaviour
- PlayerBehaviour
- QuestDiaryBehaviour
- StatisticBehaviour

DialogBehaviour

This behavior represents the ability for a NPC to discuss with a player. It stores the different dialogs according to the player situation. There is always a default dialog and sometimes, there are additional dialogs that could be shown depending on some condition and the state of the player. The Figure 139 shows the XML Schema.

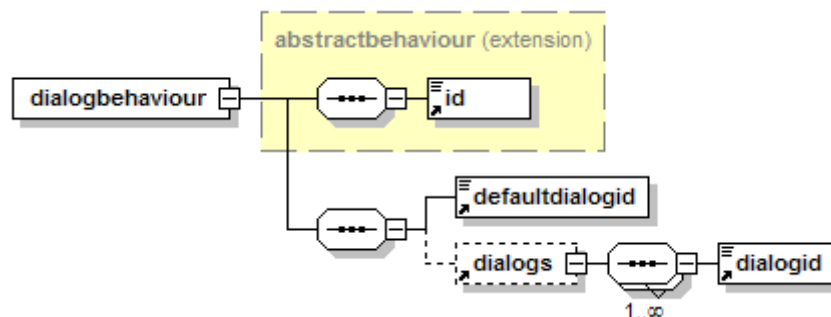


Figure 139 : DialogBehaviour XML Schema

DialogStateBehaviour

The dialog state behavior allows maintaining the state between the player and certain dialogs. In definitive, this mechanism allows avoiding the appearance of already read dialogs. For example: dialogs that reward the player with items do not be replayed after the first time they appeared. The Figure 140 shows the XML Schema

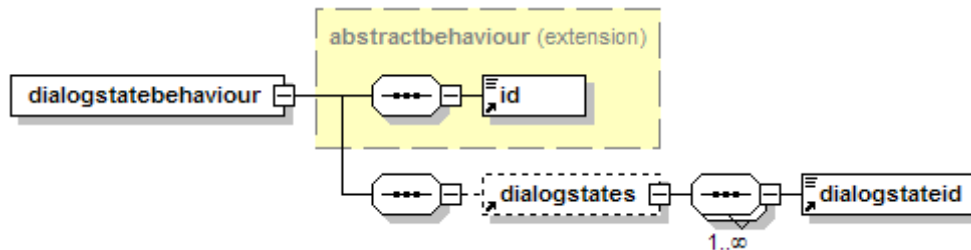


Figure 140 : DialogStateBehaviour XML Schema

ItemBehaviour

This behavior represents concrete items usable in the game. The Figure 141 shows the XML Schema.

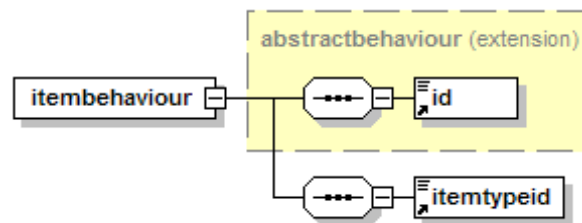


Figure 141 : ItemBehaviour XML Schema

ItemInventoryBehaviour

The ItemInventoryBehaviour allows storing the items by players or other KmepeEntities. The ItemInventoryBehaviour contains items usable by the player or other KmepeEntities. The Figure 142 shows the XML Schema.

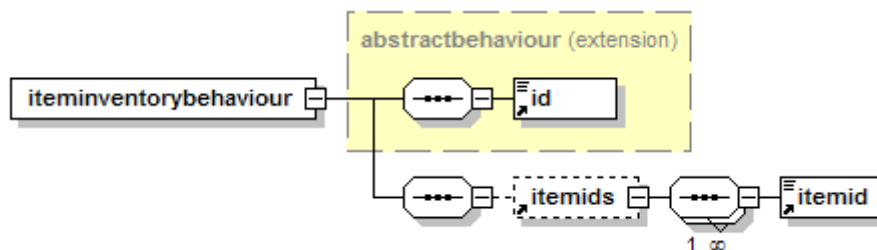


Figure 142 : ItemInventoryBehaviour XML Schema

ItemSourceBehaviour

The ItemSourceBehaviour allows placing sources of items on the map with a delay between two apparitions of item. The Figure 143 shows XML Schema.

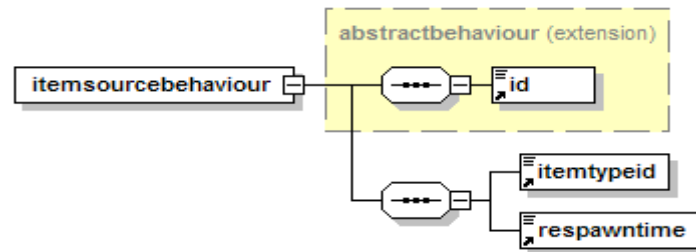


Figure 143 : ItemSourceBehaviour XML Schema

MessengerBehaviour

The messenger behavior allows an entity receiving some alerts depending on some actions done by other entities. The Figure 144 shows the XML Schema.

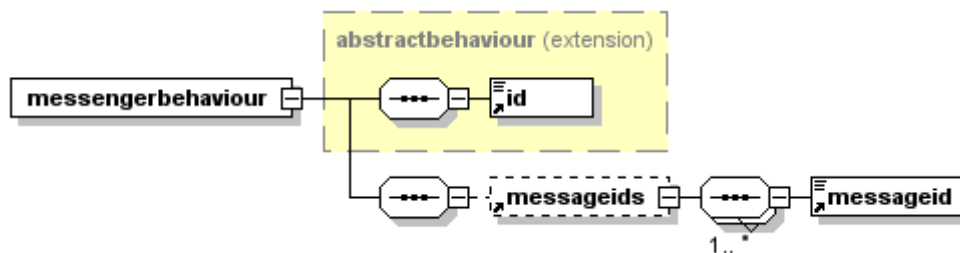


Figure 144 : MessengerBehaviour XML Schema

NPCBehaviour

The NPCBehaviour allows creating NPC. The Figure 145 shows the XML Schema.

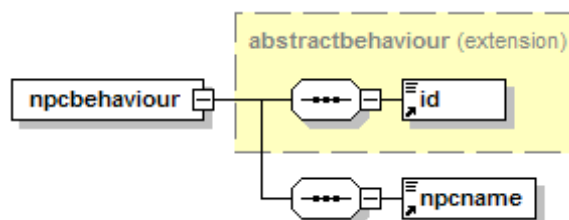


Figure 145 : NPCBehaviour XML Schema

PlayerBehaviour

The PlayerBehaviour represents the player with its connection information. The Figure 146 shows the XML Schema.

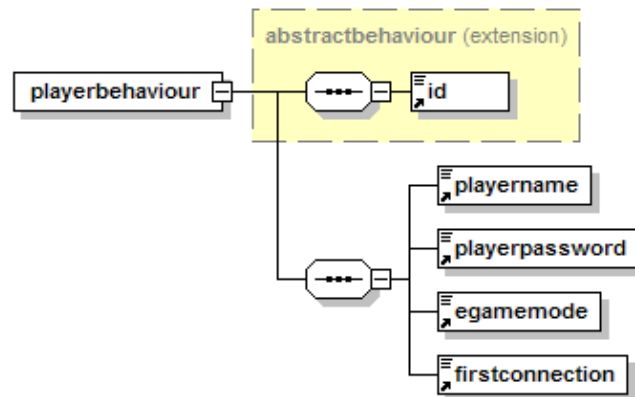


Figure 146 : PlayerBehaviour XML Schema

QuestDiaryBehaviour

The QuestDiaryBehaviour allows storing the state of quest for a KmpEntity. The Figure 147 shows the XML Schema.

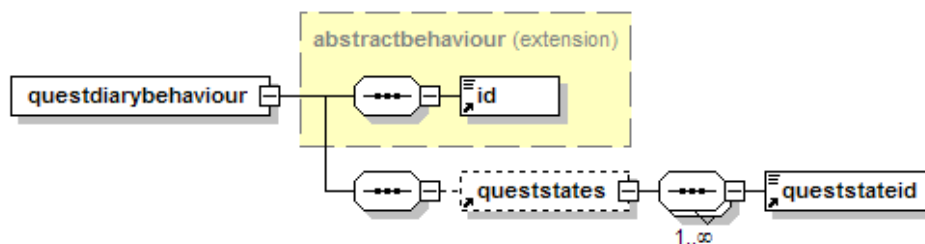


Figure 147 : QuestDiaryBehaviour XML Schema

StatisticBehaviour

The StatisticBehaviour allows storing the statistic for a KmpEntity. The Figure 148 shows the XML Schema.

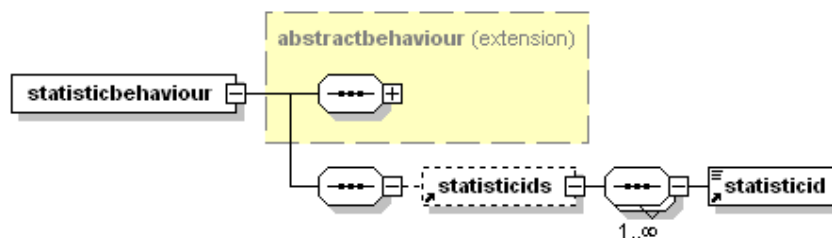


Figure 148 : StatisticBehaviour XML Schema

The Table 9 shows all the tags with their cardinality and meaning.

Tag	Cardinality	Type	Description
Abstractbehaviour Tag			
<id>	1	Positive	Correspond to the identifier of behaviors.
Dialogbehaviour Tag [extends abstractbehaviour tag]			
<defaultdialogid>	1	Positive	Define the default dialog id to show to the player when there is no other dialog to show (in regards of the conditions and player situation).
<dialogs>	0-1	ComplexType	Define the addition dialog collection that is available to show to the player depending on the player situation.
Dialogs Tag			
<dialogid>	1-N	Positive	Define the dialog id to reference the dialog in the collection.
Dialogstatebehaviour Tag [extends abstractbehaviour tag]			
<dialogstates>	0-1	ComplexType	Define the collection of the dialog state. The dialog states allow checking the visibility of some dialogs.
DialogStates Tag			
<dialogstateid>	1-N	Positive	Define the dialog state id. The dialog states are stored in another XML file.
Itembehaviour Tag [extends abstractbehaviour tag]			
<itemtypeid>	1	Positive	Define the item type that the item is from. The item types are stored in another XML file.
Iteminventorybehaviour Tag [extends abstractbehaviour tag]			
<itemids>	0-1	ComplexType	Define the list of id representing the items that the player owns.
Itemids Tag			
<itemid>	1-N	Positive	Define the id of the item owned by the player. The items are stored in another XML file.
Itemsourcebehaviour Tag [extends abstractbehaviour tag]			
<itemtypeid>	1	Positive	Define the item type used in the source. The item types are stored in another XML file.

Tag	Cardinality	Type	Description
<respawntime>	1	Positive	Define the time between two apparitions of an item from the source. The value is in seconds.
MessengerBehaviour Tag [extends abstracbehaviour tag]			
<messageids>	0-1	ComplexType	This is a collection of identifiers of messages.
Messageids Tag			
<messageid>	1-N	Positive	This is the identifier of messages.
Npcbehaviour [extends abstracbehaviour tag]			
<npcname>	1	String	Define the name of the NPC. The length of the name cannot be longer than 25 characters.
Playerbehaviour Tag [extends abstracbehaviour tag]			
<playername>	1	String	Define the name of the player. The length of the player name cannot be longer than 25 characters.
<playerpassword>	1	String	Define the password of the player. The length of the player password cannot be longer than 256 characters. In addition, the password is stored hashed.
<egamemode>	1	Enumeration	Defines the game mode in which the player is. To know the values of the enumeration, refer directly to the "EGameMode" enumeration.
<firstconnection>	1	Boolean	Defines if the player has already played or not. This is a flag for the first connection.
Questdiarybehaviour Tag [extends abstracbehaviour tag]			
<queststates>	0-1	ComplexType	Define the collection of the quest states of the player.
Queststates Tag			
<queststateid>	1-N	Positive	Define the quest state. The quest states are stored in another XML file.
Statisticbehaviour Tag [extends abstracbehaviour tag]			
<statisticids>	0-1	ComplexType	Define the collection of the statistic id corresponding to the statistics.

Tag	Cardinality	Type	Description
Statisticids Tag			
<statisticid>	1-N	Positive	Define the statistic id. The statistics are stored in another XML file.

Table 9 : behaviours tags

kmepentity.xsd

This XML Schema contains the definitions for a KmepEntity. The KmepEntity can be used to define the Players, NPC, and Items... The Figure 149 shows the XML Schema.

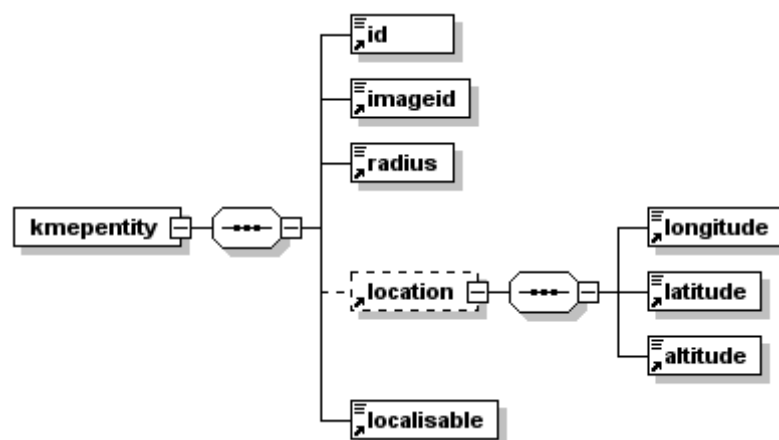


Figure 149 : KmepEntity XML Schema

The Table 10 shows all the tags with their cardinality and meaning.

Tag	Cardinality	Type	Description
Root Tag			
<kmepentity>	1	ComplexType	Root tag that contains other tags.
Kmepentity Tag			
<id>	1	Positive	This is the identifier of the KmepEntity.
<imageid>	1	Positive	The image id represents the graphical representation of the KmepEntity.
<radius>	1	Double	Define the radius of influence/visibility of the KmepEntity. The value must be included in this range: 0.0 to 1'000.0.
<location>	0-1	ComplexType	Define the location of the entity.
<localisable>	1	Boolean	Defines if the location of an entity can be retrieved or not.

Tag	Cardinality	Type	Description
Location Tag			
<longitude>	1	Double	Define the longitude of the location.
<latitude>	1	Double	Define the latitude of the location.
<altitude>	1	Double	Define the altitude of the location.

Table 10 : kmepentity tags

C.3.3. Game data XML files

C.3.3.1. loaders

This configuration files contains the definition to load others configuration files for the data game loading. It allows creating a bunk data loading. The Figure 150 shows the XML Schema used to define this kind of XML file.

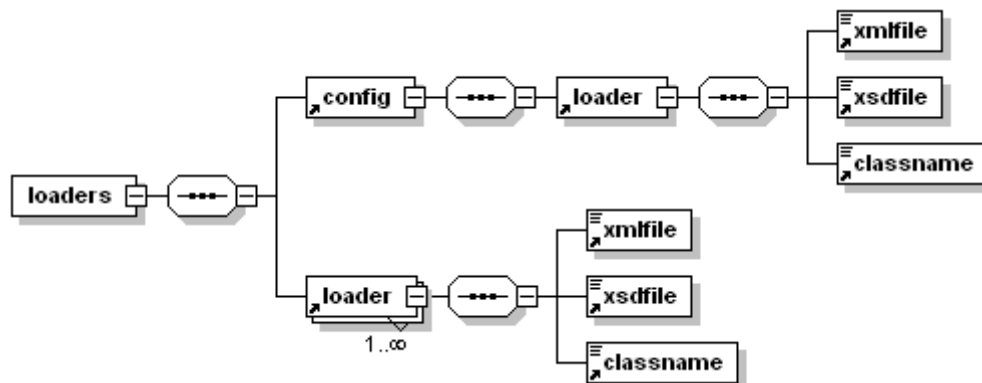


Figure 150 : loaders XML Schema

The Table 11 shows all the tags with their cardinality and meaning.

Tag	Cardinality	Type	Description
Root Tag			
<loaders>	1	ComplexType	Root tag that contains other tags.
Loaders Tag			
<config>	1	ComplexType	Define a special XML file to load for that contains the configuration values to store in the persistence layer.
<loader>	1-N	ComplexType	This tag defines the configuration to load.
Config Tag			
<loader>	1	ComplexType	Define the configuration to load.
Loader Tag			
<xmlfile>	1	String	Represent the name of the XML file without extension and path. The path is known to be in the "ch/kmep/xml" in the "EJB Project Resources". The extension is XML.
<xsdfile>	1	String	Represent the name of the XSD file without extension and path. The path is known to be in the "ch/kmep/xsd" in the "EJB Project Resources". The extension is XSD.

Tag	Cardinality	Type	Description
<classname>	1	String	Define the class that can receive the data stored in the XML file to load. The class name must be written in its complete form. For example: "ch.kmep.server.config.item.ItemTypeCollectionXML"

Table 11 : loaders tags

C.3.3.2. dialogs

This configuration files allow loading the dialogs data for the NPC. The figures above shows all the elements used for the dialogs configuration.

AbstractDialog

The Figure 151 shows the abstract dialog represents the base for dialog.

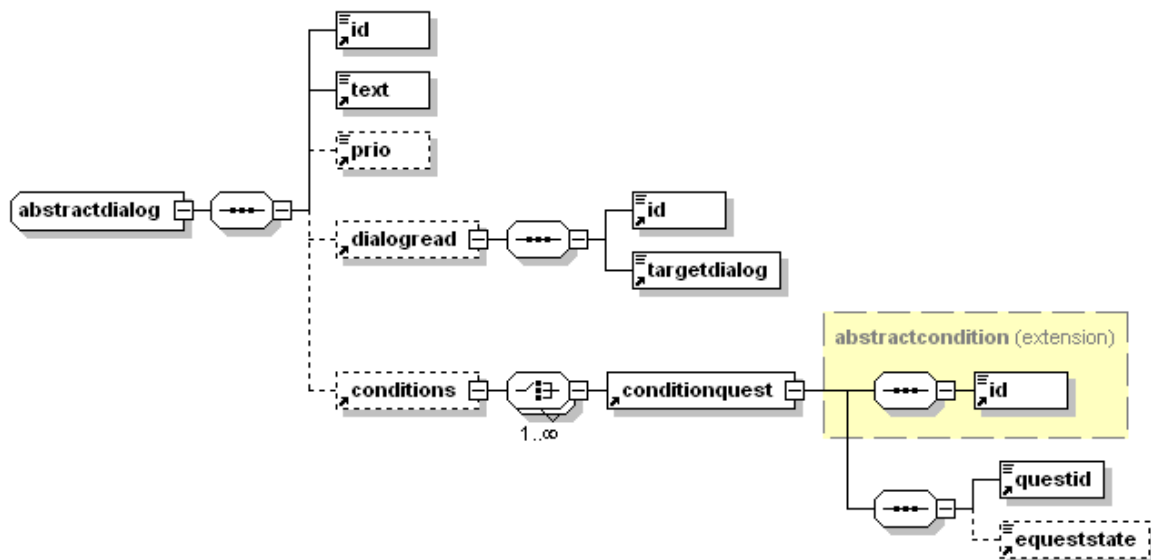


Figure 151 : AbstractDialog in dialogs

AbstractCondition

The abstract condition represents the base for all the dialogs conditions. The Figure 152 shows the XML Schema.

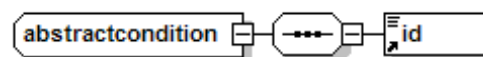


Figure 152 : AbstractCondition in dialogs

DialogText

The dialog text is a simple dialog with a piece of text. The Figure 153 shows the XML Schema.

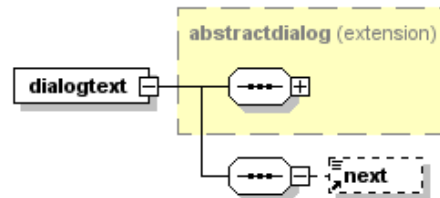


Figure 153 : DialogText XML Schema

DialogQuest

The dialog quest represents a quest proposal for the player. The Figure 154 shows the XML Schema.

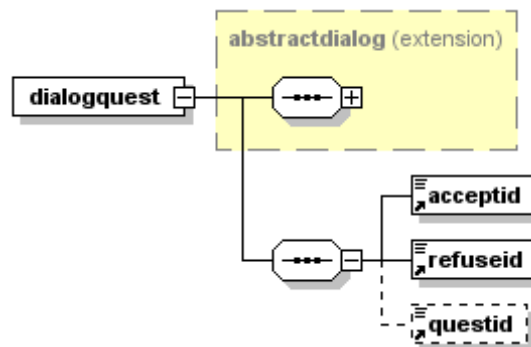


Figure 154 : DialogQuest XML Schema

DialogQuestFinish

The dialog quest finish allows finishing a quest by a player. The Figure 155 shows the XML Schema.

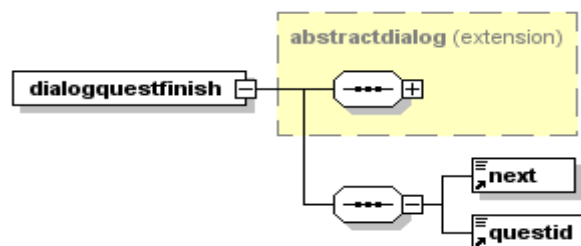


Figure 155 : DialogQuestFinish XML Schema

DialogQuestion

The dialog question allows asking something to the player like a choice in the discussion. The Figure 156 shows the XML Schema.

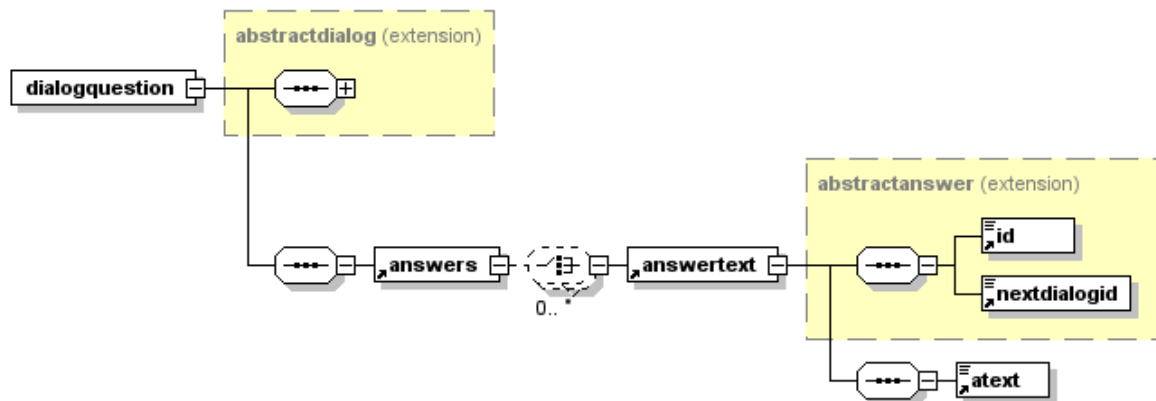


Figure 156 : DialogQuestion XML Schema

DialogQuestionPuzzle

The dialog question puzzle allows asking the answer of a picture puzzle corresponding to a certain quest. The Figure 157 shows the XML Schema.

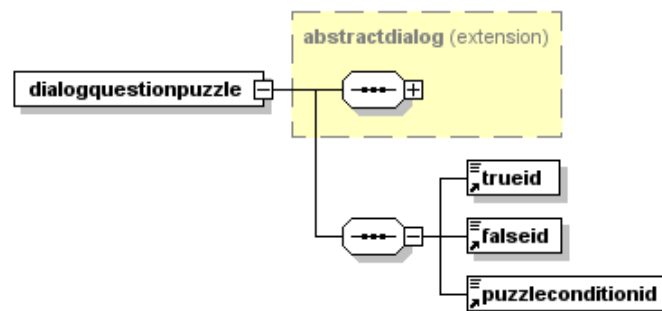


Figure 157 : DialogQuestionPuzzle XML Schema

DialogItem

The dialog item allows giving or asking some items. The players can receive or give items with this kind of dialogs. The Figure 158 shows the XML Schema.

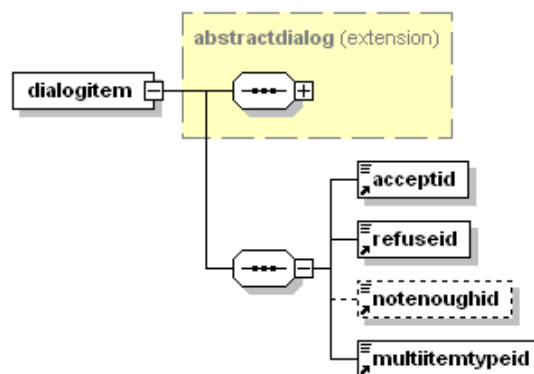


Figure 158 : DialogItem XML Schema

Dialogs

The Figure 159 shows the dialog collection to store the dialogs.

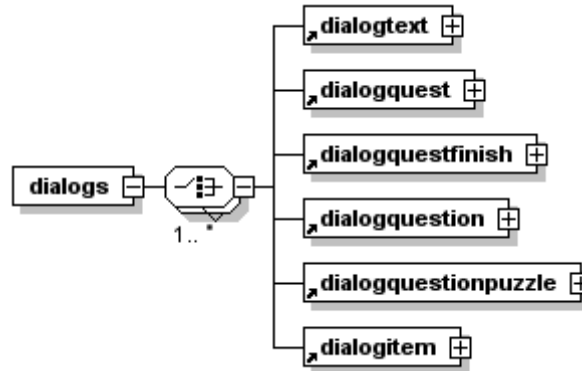


Figure 159 : dialogs XML Schema

The Table 12 shows all the tags with their cardinality and meaning.

Tag	Cardinality	Type	Description
AbstractAnswer			
<id>	1	Positive	Define the identifier for an abstract answer. An abstract answer determines the answer to a question asked by a NPC.
<nextdialogid>	1	Positive	Define the next dialog to show after activating the answer.
AbstractCondition			
<id>	1	Positive	Define the identifier for an abstract condition. An abstract condition determines if a dialog could be read or not according to the condition state.
AbstractDialog			
<id>	1	Positive	Define the identifier for an abstract dialog. An abstract dialog determines what can say a NPC.
<text>	1	String	The text that NPCs can use. This text cannot be longer than 160 characters.
<prio>	0-1	Positive	Define the priority of the dialog when there are two dialogs possible in the same time. The highest priority is "0". If no value is provided, the lowest priority will be used.

Tag	Cardinality	Type	Description
<dialogread>	0-1	ComplexType	Define a flag to create a dialog state when the flag is encountered during the dialog process.
<conditions>	0-1	ComplexType	Define a collection of conditions.
Dialogread Tag			
<id>	1	Positive	Determines the identifier of the dialog read flag.
<targetdialog>	1	Positive	Determine the identifier of the dialog to create the dialog state during the dialog process.
Conditions Tag			
<conditionquest>	0-N	ComplexType	Define a quest condition to show a dialog. A quest condition determines a quest and a quest state to have to see a specific dialog.
Conditionquest Tag [extends abstractcondition tag]			
<questid>	1	Positive	Define the identifier of the quest concerned by the quest condition.
<equeststate>	0-1	Enumeration	Define the state that the quest has to be for the player to see the dialog. For the quest state enumeration, refer directly to the "EQuestState" enumeration in the "EJB Project". No quest state means that the quest must not be present in the quest diary.
Root Tag			
<dialogs>	1	ComplexType	It defines the root tag of the document.
Dialogs Tag			
<dialogtext>	0-N	ComplexType	This is the simplest dialog type. This is a simple text to show to the player.
<dialogquest>	0-N	ComplexType	A dialog quest represents a dialog with a quest proposal.
<dialogquestfinish>	0-N	ComplexType	A dialog quest finish represents the end of a quest with the validation (apply the rewards and so on).
<dialogquestion>	0-N	ComplexType	A dialog question represents a question asked by a NPC to the player with some proposed answers.

Tag	Cardinality	Type	Description
<dialogquestionpuzzlw>	0-N	ComplexType	A dialog question puzzle represents the possibility to the player to solve a picture puzzle giving the textual answer to the puzzle.
<dialogitem>	0-N	ComplexType	A dialog item represent a dialog with a gift of item from the NPC to the player. This is particularly useful to give item after a quest proposal acceptance.
Dialogtext Tag [extends abstractdialog tag]			
<next>	0-1	Positive	Represent the next dialog id. No next dialog id indicates the end of a dialog.
Dialogquest Tag [extends abstractdialog tag]			
<acceptid>	1	Positive	Represent the next dialog id when the user accepts the quest.
<refuseid>	1	Positive	Represent the next dialog id when the user refuses the quest.
<questid>	0-1	Positive	Represent the quest id that the player can accept or refuse. No id means that the quest will be chosen in the FirstQuest.
Dialogquestfinish Tag [extends abstractdialog tag]			
<next>	1	Positive	Represent the next dialog id to show the dialog to the player.
<questid>	1	Positive	Defines the quest id concerned by the dialog.
Dialogquestion Tag [extends abstractdialog tag]			
<answers>	1	ComplexType	Define a collection of answers that the player can choose to answer the question.
Answers Tag			
<answertext>	0-N	ComplexType	Answer text represents a simple answer with only text to show and process.
Dialogquestionpuzzle Tag [extends abstractdialog tag]			
<>trueid>	1	Positive	Represents the next dialog id when the user solve the puzzle
<>falseid>	1	Positive	Represent the next dialog id when the user does not solve the puzzle.

Tag	Cardinality	Type	Description
<puzzleconditionid>	1	Positive	Represent the identifier of the quest condition puzzle concerned by the dialog.
Answertext Tag [extends abstractanswer tag]			
<text>	1	String	Define the text to represent the answer.
Dialogitem Tag [extends abstractdialog tag]			
<acceptid>	1	Positive	Represent the dialog when the player accepts the offer.
<refuseid>	1	Positive	Represent the dialog when the player refuses the offer.
<notenoughid>	0-1	Positive	It defines the id of the dialog to go when there are not enough items to get from the player. This only applicable to a dialog that take item from the player. In the other case, this is not useful. When this dialog id is not filled, the refuse dialog is used.
<multiitemtypeid>	1	Positive	Define the identifier of multi item type. This multi item type allow the usage of more than one type of item in one time

Table 12 : dialogs tags

C.3.3.3. dialogstates

This configuration files allow loading the dialog states data for the players. The Figure 160 is the XML Schema for a dialog states XML file.

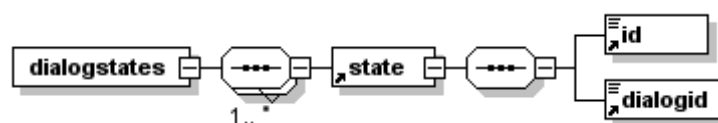


Figure 160 : dialogstates XML Schema

The Table 13 shows all the tags with their cardinality and meaning.

Tag	Cardinality	Type	Description
Root Tag			
<dialogstates>	1	ComplexType	Root tag that contains other tags.
Dialogstates Tag			
<state>	1-N	ComplexType	Define the state for a dialog that the player read.

Tag	Cardinality	Type	Description
State Tag			
<id>	1	Positive	Define the identifier for the dialog state.
<dialogid>	1	Positive	Define the identifier of the dialog concerned by the dialog state.

Table 13 : dialogstates tags

C.3.3.4. firstquests

This configuration files allow loading the first quests data to allow showing a first quest to the player for its first connection to the game. The Figure 161 is the XML Schema for a first quests XML file.

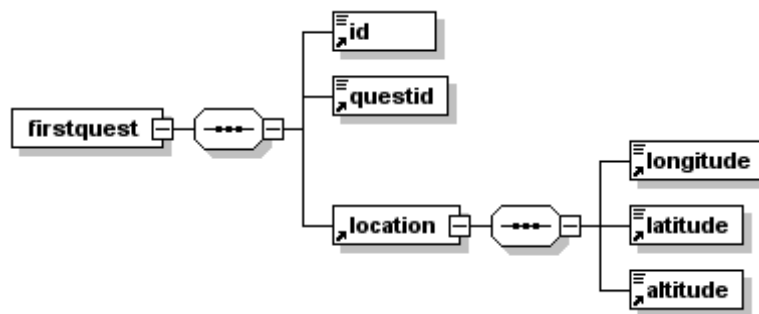


Figure 161 : firstquests XML Schema

The Table 14 shows all the tags with their cardinality and meaning.

Tag	Cardinality	Type	Description
Root Tag			
<firstquests>	1	ComplexType	Root tag that contains other tags.
Firstquests Tag			
<firstquest>	1-N	ComplexType	Define the first quest that a player can see at the beginning of the game (during its first connection to the game).
Firstquest Tag			
<id>	1	Positive	Define the identifier for the first quest.
<questid>	1	Positive	Define the identifier of the quest to propose.
<location>	1	ComplexType	Define the location of the first quest. This location is here to find the nearest first quest to the location of the player.

Tag	Cardinality	Type	Description
Location Tag			
<longitude>	1	Double	Define the longitude of the location.
<latitude>	1	Double	Define the latitude of the location.
<altitude>	1	Double	Define the altitude of the location.

Table 14 : firstquests tags

C.3.3.5. items

This configuration file allows loading the items present in the game. The Figure 162 shows XML Schema.

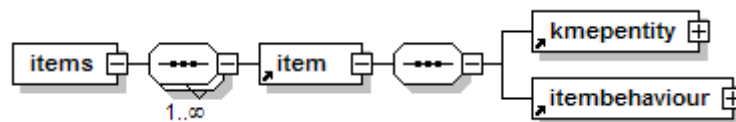


Figure 162 : items XML Schema

The Table 15 shows all the tags with their cardinality and meaning.

Tag	Cardinality	Type	Description
Root Tag			
<items>	1	ComplexType	Root tag that contains other tags.
Items Tag			
<item>	1-N	ComplexType	Define items of the game.
Item Tag			
<kepentity>	1	ComplexType	Define the KmepEntity representing the item.
<itembehaviour>	1	ComplexType	Define the item behavior of the item.

Table 15 : items tags

C.3.3.6. itemsources

This configuration file allows loading the item sources present on the map in the game. The Figure 163 shows XML Schema.

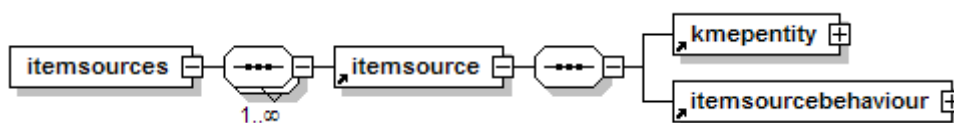


Figure 163 : itemsources XML Schema

The Table 16 shows all the tags with their cardinality and meaning.

Tag	Cardinality	Type	Description
Root Tag			
<itemsources>	1	ComplexType	Root tag that contains other tags.
Itemsources Tag			
<itemsource>	1-N	ComplexType	Define the item sources on the map.
Itemsource Tag			
<kepentity>	1	ComplexType	Define the KmpEntity representing the item.
<itemsourcebehaviour>	1	ComplexType	Define the item source behavior of the item source.

Table 16 : itemsources tags

C.3.3.7. itemtypes

This configuration file allows loading the item type. These data represents the types for items. They allow generalizing some data. The Figure 164 shows XML Schema.

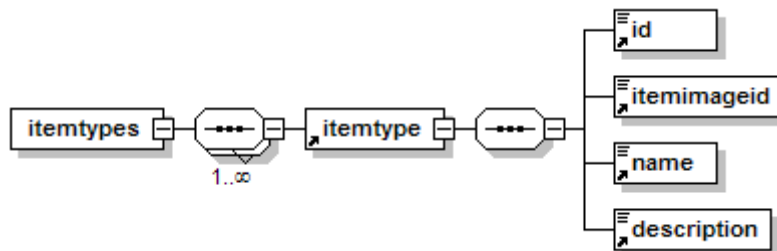


Figure 164 : itemtypes XML Schema

The Table 17 shows all the tags with their cardinality and meaning.

Tag	Cardinality	Type	Description
Root Tag			
<itemtypes>	1	ComplexType	Root tag that contains other tags.
Itemtypes Tag			
<itemtype>	1-N	ComplexType	Define the item types.
Itemsource Tag			
<id>	1	Positive	Define the identifier of the item type.

Tag	Cardinality	Type	Description
<itemimageid>	1	Positive	Define the graphical representation for the item of this type.
<name>	1	String	Define the name of the item type. The maximum length is of 25 characters.
<description>	1	String	Define the description of the item type. The maximum length is of 255 characters.

Table 17 : itemtypes tags

C.3.3.8. messages

Messages represent the ability to notify the players of events occurred during the game and concerning the player. For example, when a player wants to give an item to another one, messages are used to alert players of the gift process.

AbstractMessage

The abstract message is the base to build other messages. The Figure 165 shows the XML Schema.

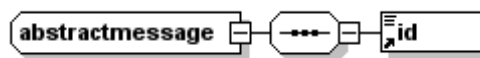


Figure 165 : abstractmessage XML Schema

ItemIds

Item ids are the collection of identifier representing items. This a type used more than one time in the next XML Schema. The Figure 166 shows the XML Schema.

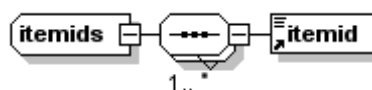


Figure 166 : itemids XML Schema

MessageQuestGift

Message Quest Gift represents the message when a player wants to give a quest to another one. The Figure 173Figure 167 shows the XML Schema.

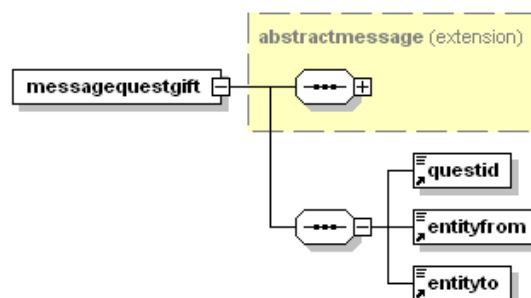


Figure 167 : messagequestgift XML Schema

MessageItemExchange

Message Item Exchange is the category of message used during an exchange of items between two players. The Figure 168 shows the XML Schema.

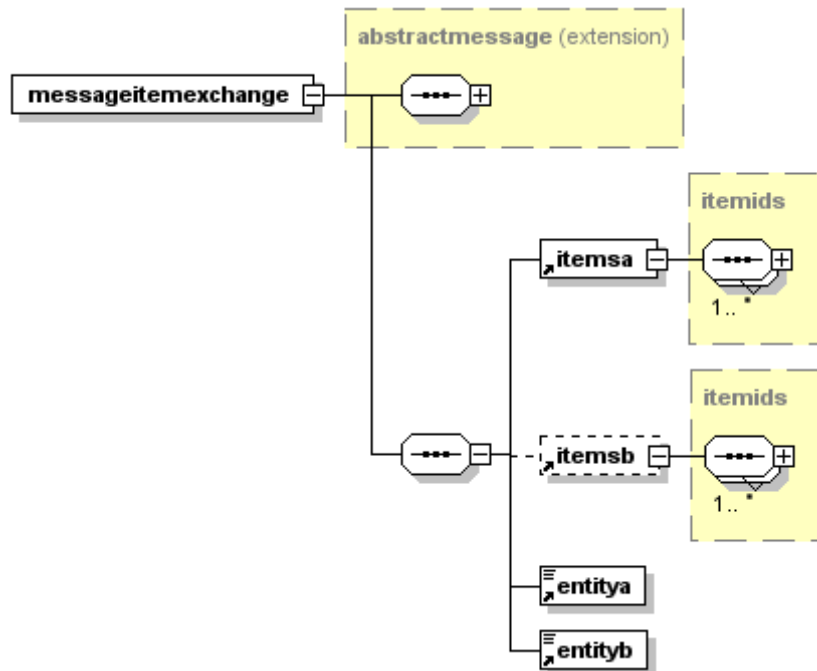


Figure 168 : messageitemexchange XML Schema

MessageItemExchangeConfirm

Message Item Exchange Confirmation represents a confirmation message during the exchange of items between two players. The Figure 169 shows the XML Schema.

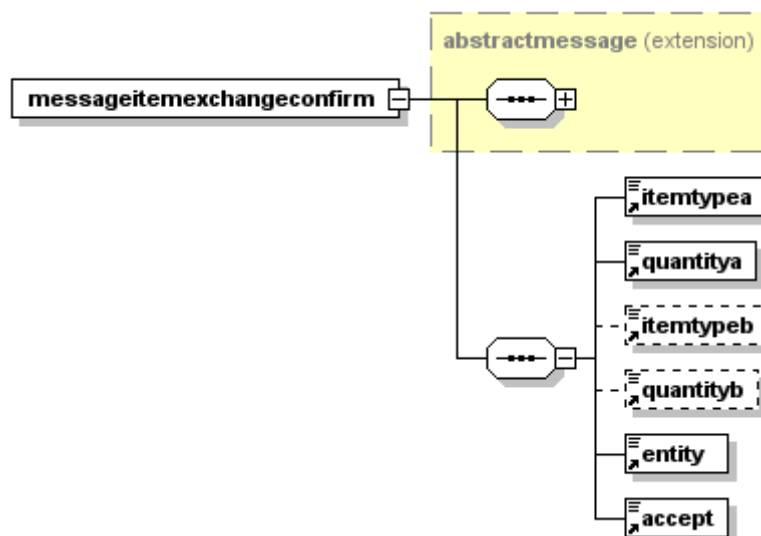


Figure 169 : messageitemexchangeconfirm XML Schema

MessageItemGift

Message Item Gift represents the message when a player wants to give an item to another one. The Figure 170 shows the XML Schema.

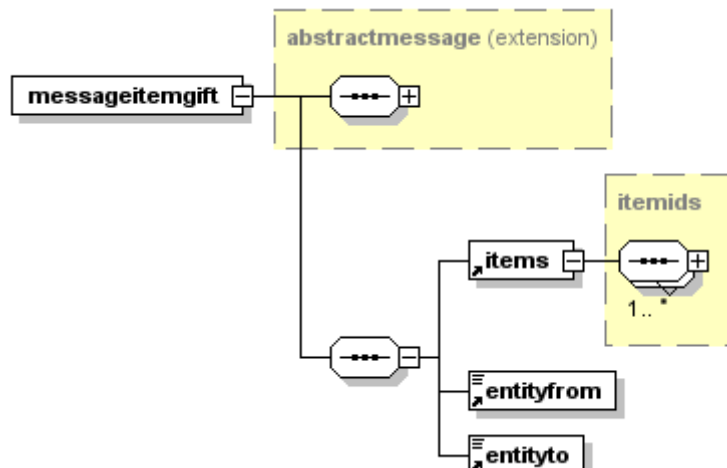


Figure 170 : messageitemgift XML Schema

MessageItemGiftConfirm

Message Item Gift Confirmation represents the confirmation message after a player has accepted or refused a gift offer from another player. The Figure 171 shows the XML Schema.

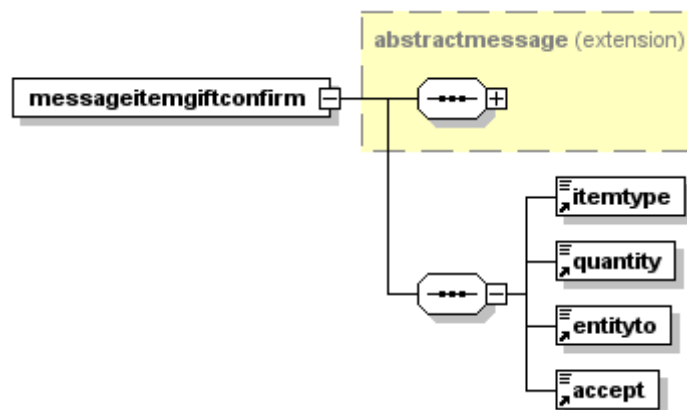


Figure 171 : messageitemgiftconfirm XML Schema

MessageQuestGiftConfirmation

Message Quest Gift Confirmation represents the confirmation message after a player has accepted or refused a gift offer from another player. This is also used when the receiver player has already the quest in its quest diary. The Figure 171 shows the XML Schema.

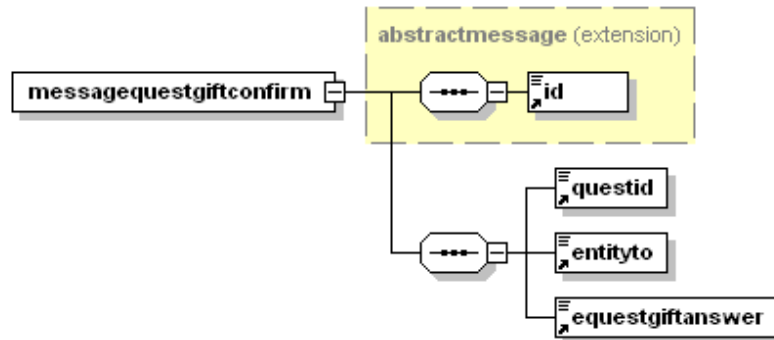


Figure 172 : messagequestgiftconfirm XML Schema

Messages

Messages are the collection of the messages described just before. The Figure 173 shows the XML Schema.

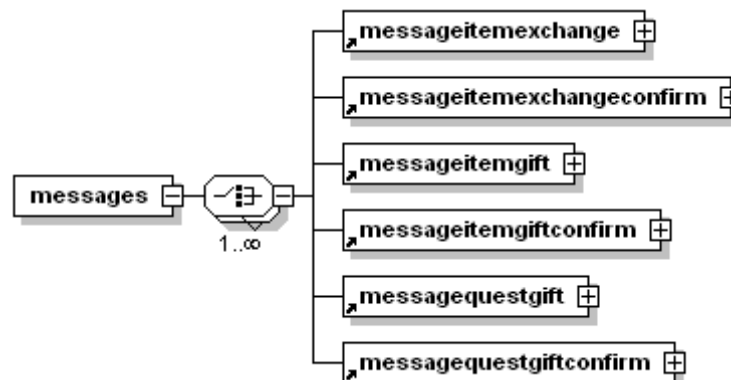


Figure 173 : messages XML Schema

The Table 18 shows all the tags with their cardinality and meaning. For presentation reasons, the tags for Root Tag are composed only by the first letter of each word that composed the tag name.

Tag	Cardinality	Type	Description
Abstractmessage Tag			
<id>	1	Positive	Define the identifier of the message.
Itemids Type			
<itemid>	1-N	Positive	Define the identifier of items.

Tag	Cardinality	Type	Description
Root Tag			
<mie>	0-N	ComplexType	Message Item Exchange (see the remarks before the diagram).
<miec>	0-N	ComplexType	Message Item Exchange Confirmation (see the remarks before the diagram).
<mig>	0-N	ComplexType	Message Item Gift (see the remarks before the diagram).
<migc>	0-N	ComplexType	Message Item Gift Confirmation (see the remarks before the diagram).
<mqq>	0-N	ComplexType	Message Quest Gift (see the remarks before the diagram)
<mqqc>	0-N	ComplexType	Message Quest Gift Confirmation (see the remarks before the diagram).
Messageitemexchange Tag [extends abstractmessage tag]			
<itemsa>	1	Itemids	Defines the items offered by the entity that initiate the exchange.
<itemsb>	0-1	Itemids	Defines the items offered by the entity to answer the proposition.
<entitya>	1	Positive	Define the entity that initiates the exchange.
<entityb>	1	Positive	Define the entity that answers the exchange.
Messageitemexchangeconfirm Tag [extends abstractmessage tag]			
<itemypea>	1	Positive	Define the item type from the player that initiates the exchange.
<quantitya>	1	Positive	It defines the quantity of the item type an exchanged.
<itemypeb>	0-1	Positive	Define the item type from the player that answers the exchange.
<quantityb>	0-1	Positive	It defines the quantity of the item type b exchanged.
<entity>	1	Positive	Define the entity that answers the exchange.
<accept>	1	Boolean	Define the answer of the entity answer the exchange.

Tag	Cardinality	Type	Description
Messageitemgift Tag [extends abstractmessage tag]			
<items>	1	Itemids	Defines the items offered by a player to another player.
<entityfrom>	1	Positive	Define the entity that offers the items.
<entityto>	1	Positive	Define the entity that receives the items.
Messageitemgiftconfirm Tag [extends abstractmessage tag]			
<itemtype>	1	Positive	Define the item type offered.
<quantity>	1	Positive	Define the quantity of items offered.
<entityto>	1	Positive	Define the entity that receives the items.
<accept>	1	Boolean	Define the answer of the entity that receives the items.
Messagequestgift Tag [extends abstractmessage tag]			
<questid>	1	Positive	Define the quest offered by a player to another player.
<entityfrom>	1	Positive	Define the entity that offers the quest.
<entityto>	1	Positive	Define the entity that receives the quest.
Messagequestgiftconfirm Tag [extends abstractmessage tag]			
<questid>	1	Positive	Define the quest offered.
<entityto>	1	Positive	Define the entity that answers the proposition.
<questgiftanswer>	1	Enumeration	Answer of the entity that receives the offer. The enumeration can be view directly in XSD definition file.

Table 18 : messages tags

C.3.3.9. multiitemtypes

The multi item types configuration allows preparing some bunk of items for reward, material or other purpose. This is particularly useful to configure some kind of gift or something like that. For example, it is possible to give only one type of item inside the list of items available. The Figure 174 shows the XML Schema.

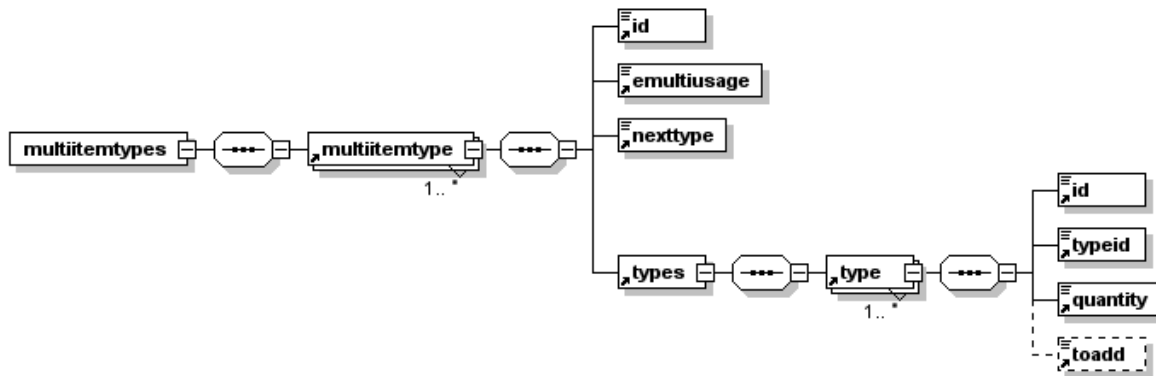


Figure 174 : multiitemtypes XML Schema

The Table 19 shows all the tags with their cardinality and meaning.

Tag	Cardinality	Type	Description
Root Tag			
<multiitemtypes>	1	ComplexType	Root tag that contains other tags.
Multiitemtypes Tag			
<multiitemtype>	1-N	ComplexType	Define the multi item type.
Multiitemtype Tag			
<id>	1	Positive	This is the identifier of the multi item type.
<emultiusage>	1	Enumeration	Define the enumeration of different usages possible for the multi item type. See directly the class EMultiitemUsage for more details.
<nexttype>	1	NonNegative	Define the next item type to use for the Next EMultiitemUsage enumeration value.
<types>	1	ComplexType	This is the collection of types for the multi item type.
Types Tag			
<Type>	1-N	ComplexType	Define the type containing the data to know how to proceed with the item type concerned.

Tag	Cardinality	Type	Description
Type tag			
<id>	1	Positive	Define the identifier of the Type.
<typeid>	1	Positive	Define the identifier of the item type concerned by the type.
<quantity>	1	Positive	This is the quantity to add or remove from the item type id.
<toadd>	0-1	Boolean	Defines if the item type must to be removed or added to a player inventory. By default, item type is added.

Table 19 : multiitemtypes tags

C.3.3.10. npcs

This configuration file allows loading the NPCs present in the game. The Figure 175 shows the XML Schema.

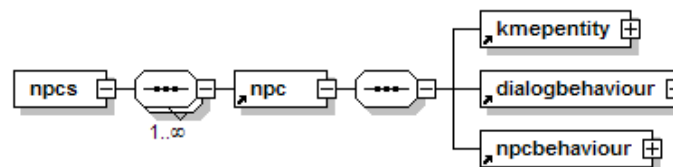


Figure 175 : npcs XML Schema

The Table 20 shows all the tags with their cardinality and meaning.

Tag	Cardinality	Type	Description
Root Tag			
<npcs>	1	ComplexType	Root tag that contains other tags.
Npcs Tag			
<npc>	1-N	ComplexType	Define the NPC entity for the game.
Npc Tag			
<kmepentity>	1	ComplexType	Define the KmepEntity representing the NPC.
<dialogbehaviour>	1	ComplexType	Define the dialog behavior that allows showing the dialogs to the player.
<npcbehaviour>	1	ComplexType	Defines the NPC behavior to store the specific characteristics of a NPC

Table 20 : npcs tags

C.3.3.11. players

This configuration file allows loading the players present in the game. The Figure 176 shows the XML Schema.

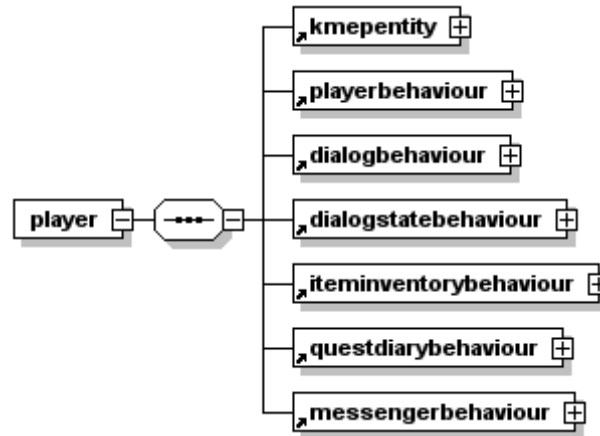


Figure 176 : players XML Schema

The Table 21 shows all the tags with their cardinality and meaning.

Tag	Cardinality	Type	Description
Root Tag			
<players>	1	ComplexType	Root tag that contains other tags.
Players Tag			
<player>	1-N	ComplexType	Define the player entity.
Player Tag			
<kmepentity>	1	ComplexType	Define the KmepEntity representing the Player.
<playerbehaviour>	1	ComplexType	Define the player behavior that contains the connection data of the player.
<dialogbehaviour>	1	ComplexType	Define the dialog behavior that allows showing the dialogs to the player especially in the case of the first connection.
<dialogstatebehaviour>	1	ComplexType	Define the dialog state behavior that allows storing the states of the dialogs read by the player.
<iteminventorybehaviour>	1	ComplexType	Define the item inventory behavior that allows storing the items owned by the player.

Tag	Cardinality	Type	Description
<questdiarybehaviour>	1	ComplexType	Define the quest diary behavior that allows storing the states of the quests of the player.
<messengerbehaviour>	1	ComplexType	Define the messenger behavior that allows storing messages for the player.

Table 21 : players tags

C.3.3.12. quests

This configuration file allows loading the quests that the players can do during the game when they speak with NPC.

Quests

The quests represent the collection of quests available for the players in the game. The Figure 177 shows the XML Schema.

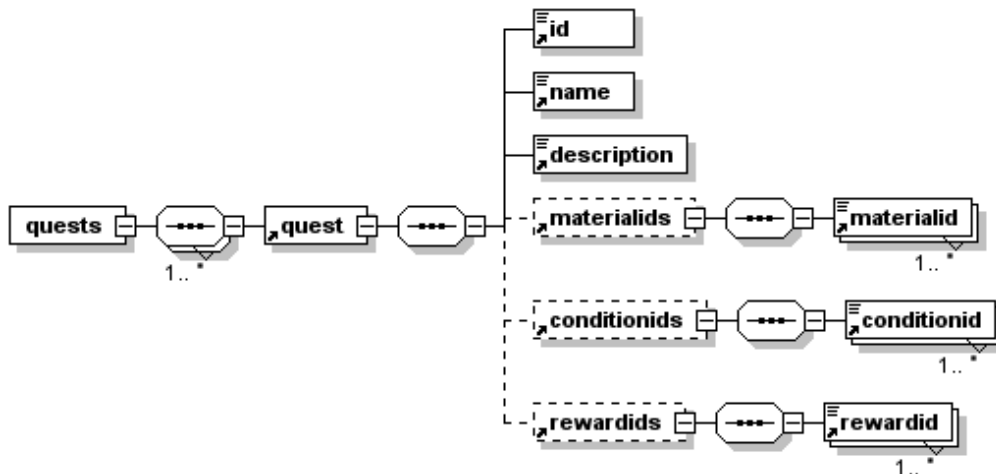


Figure 177 : quests XML Schema

The Table 22 shows all the tags with their cardinality and meaning.

Tag	Cardinality	Type	Description
Abstractquestmaterial Tag			
<id>	1	Positive	This is the identifier of the quest material.
Abstractquestcondition Tag			
<id>	1	Positive	This is the identifier of the quest condition.
Abstractquestreward Tag			
<id>	1	Positive	This is the identifier of the quest reward.

Tag	Cardinality	Type	Description
Root Tag			
<quests>	1	ComplexType	Root tag that contains other tags.
Quests Tag			
<quest>	1-N	ComplexType	Define the quests available for the palyers.
Quest Tag			
<id>	1	Positive	Define the identifier of the quest.
<name>	1	String	Define the name of the quest. The length cannot be longer than 25 characters.
<description>	1	String	Define the description of the quest. The length cannot be longer than 255 characters.
<materialids>	0-1	ComplexType	Define the material that the player receives when he accepts to do a quest.
<conditionids>	0-1	ComplexType	Define the condition that the player has to reach to finish with success a quest.
<rewardids>	0-1	ComplexType	Define the rewards that player won when he successfully finish a quest.
Materialids Tag			
<materialid>	1-N	Positive	Define the identifiers of materials.
Conditionids Tag			
<conditionid>	1-N	Positive	Define the identifiers of conditions.
Rewardids Tag			
<rewardid>	1-N	Positive	Define the identifiers of rewards.

Table 22 : quests tags

C.3.3.13. questmaterials

AbstractQuestMaterial

The material defines the elements necessary to give to the KmapEntity to run the quest correctly. For example, an item can be given to help the player to do some task in his quest. AbstractQuestMaterial is the base for all the materials. The Figure 178 shows the XML Schema.

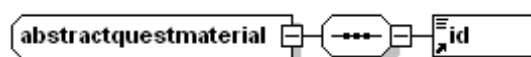


Figure 178 : abstractquestmaterial XML Schema

Materials

The materials are the collection of materials for quest. The Figure 179 shows the XML Schema.

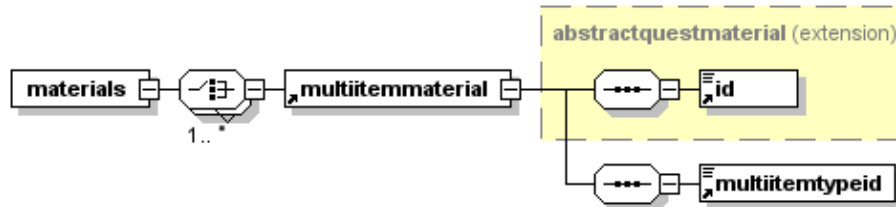


Figure 179 : Materials XML Schema

The Table 23 shows all the tags with their cardinality and meaning.

Tag	Cardinality	Type	Description
Abstractquestmaterial Tag			
<id>	1	Positive	This is the identifier of the quest material.
Root Tag			
<materials>	1	ComplexType	Define the material that the player receives when he accepts to do a quest.
Materials Tag			
<multiitemmaterial>	0-N	ComplexType	Define a multi item material to give to the player when he accepts the quest.
Multitemmaterial Tag [extends abstractquestmaterial tag]			
<multiitemtypeid>	1	Positive	Defines the identifier of the multi item type

Table 23 : questmaterials tags

C.3.3.14. questconditions

AbstractQuestCondition

The abstract quest condition defines the condition to reach to consider the quest is successfully done by the player. The Figure 180 shows the XML Schema.

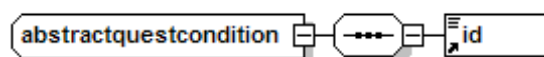


Figure 180 : abstractquestcondition XML Schema

Conditions

The conditions are the collection of conditions for quest. The Figure 181 shows the XML Schema.

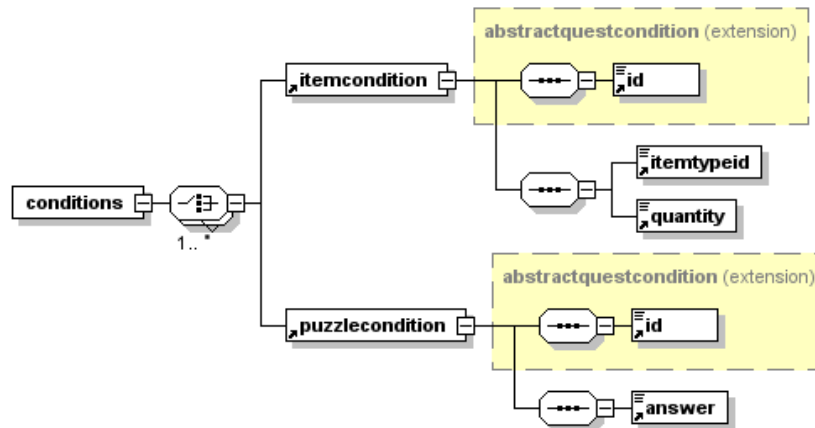


Figure 181 : Conditions XML Schema

The Table 24 shows all the tags with their cardinality and meaning.

Tag	Cardinality	Type	Description
Abstractquestmaterial Tag			
<id>	1	Positive	This is the identifier of the quest material.
Root Tag			
<conditions>	1	ComplexType	It defines the condition that the player has to reach to finish with success a quest.
Conditions Tag			
<itemcondition>	0-N	ComplexType	Define a condition to own a number of a certain item type.
<puzzlecondition>	0-N	ComplexType	Define a condition to answer a question regarding of a picture puzzle. The answer represents the puzzle picture.
Itemcondition Tag [extends abstractquestcondition tag]			
<itemtypeid>	1	Positive	Define the identifier of the item type concerned by the condition.
<quantity>	1	Positive	Define the quantity to have to reach the condition.
Puzzlecondition Tag [extends abstractquestcondition tag]			
<answer>	1	String	This is the textual answer for the puzzle condition.

Table 24 : questconditions tags

C.3.3.15. questrewards

AbstractQuestReward

The abstract quest reward defines the reward that the player won when he successfully done a quest. The Figure 182 shows the XML Schema.



Figure 182 : abstractquestreward XML Schema

Rewards

The rewards are the collection of rewards for quest. The Figure 183 shows the XML Schema.

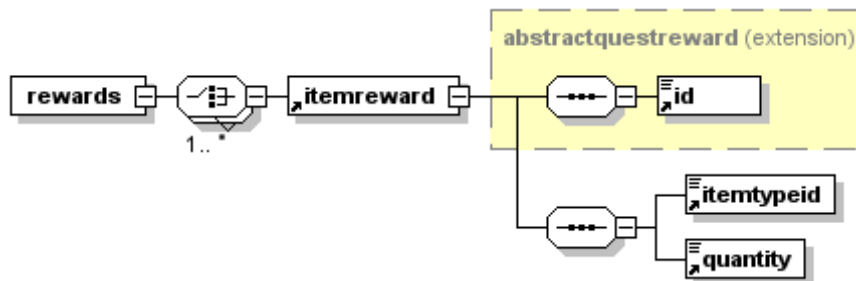


Figure 183 : Rewards XML Schema

The Table 25 shows all the tags with their cardinality and meaning.

Tag	Cardinality	Type	Description
Abstractquestreward Tag			
<id>	1	Positive	This is the identifier of the quest reward.
Root Tag			
<rewards>	0-1	ComplexType	Define the rewards that player won when he successfully finish a quest.
Rewards Tag			
<itemreward>	0-N	ComplexType	Define a reward to win a number of a certain item type.
Itemreward Tag [extends abstractquestreward tag]			
<itemtypeid>	1	Positive	Define the identifier of the item type concerned by the reward.
<quantity>	1	Positive	Define the quantity won when the reward is applied.

Table 25 : questrewards tags

C.3.3.16. queststates

This configuration file allows loading the quest states for the players

AbstractQuestConditionState

The abstract quest condition state is the base for all quest condition states used in the game. The Figure 184 shows the XML Schema.

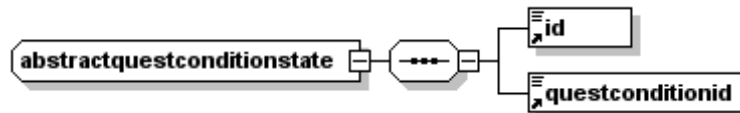


Figure 184 : abstractquestconditionstate XML Schema

QuestConditionStates

The quest condition states is the collection of quest condition states that allows storing the condition states for a quest. The Figure 185 shows the XML Schema.

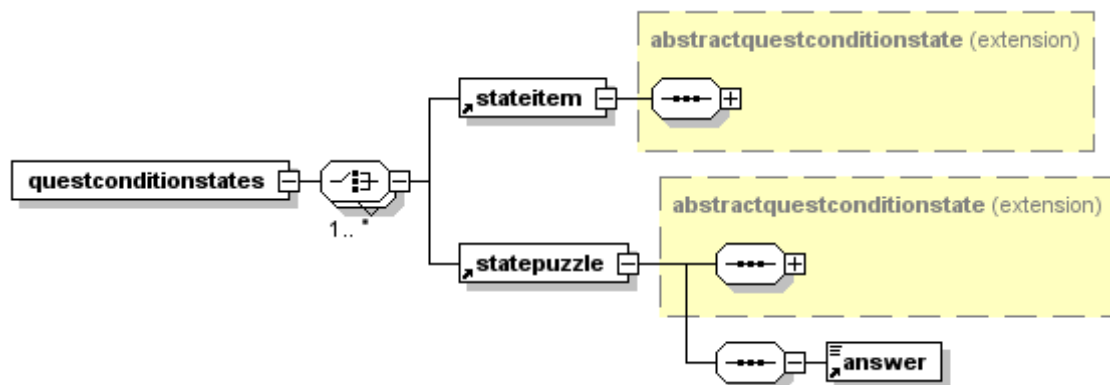


Figure 185 : questconditionstates XML Schema

QuestStates

Quest states represent the states for quests. The Figure 186 shows the XML Schema.

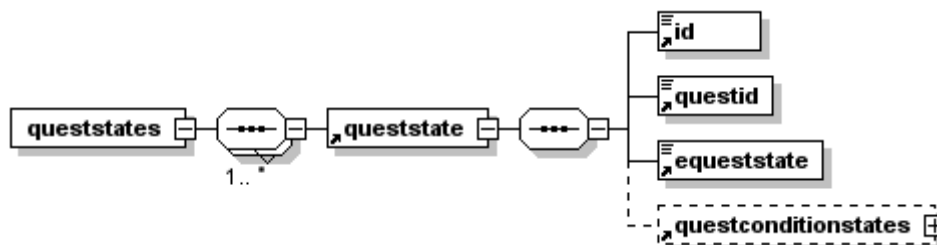


Figure 186 : queststates XML Schema

The Table 26 shows all the tags with their cardinality and meaning.

Tag	Cardinality	Type	Description
Abstractquestconditionstate Tag			
<id>	1	Positive	This is the identifier of the abstract quest condition state.
<questconditionid>	1	Positive	This is the identifier of the quest condition concerned by the state.
Root Tag			
<queststates>	1	ComplexType	Root tag that contains other tags.
Queststates Tag			
<queststate>	1-N	ComplexType	Define the quest states.
Queststate Tag			
<id>	1	Positive	Define the identifier of the quest state.
<questid>	1	Positive	Defines the quest id concerned by the quest state.
<equeststate>	1	Enumeration	Define the current state for the quest. The enumeration values are available directly for "EQuestState" enumeration.
<questconditionstates>	0-1	ComplexType	This is the collection of quest condition states.
Questconditionstates Tag			
<stateitem>	0-N	ComplexType	A state item defines the state for a condition item.
<statepuzzle>	0-N	ComplexType	A state puzzle defines the state for a condition puzzle.
Stateitem Tag [extends abstractquestconditionstate tag]			
No fields.			
StatePuzzle Tag [extends abstractquestconditionstate tag]			
<answer>	1	String	The answer proposed by the player to a quest puzzle question.

Table 26 : queststates tags

C.3.3.17. statistics

This configuration file allows loading the statistics for the players

AbstractStatistic

The abstract statistic is the base for all statistics used in the game. The Figure 187 shows the XML Schema.

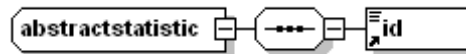


Figure 187 : abstractstatistic XML Schema

Countable

The countable statistic defines statistics with a value that represents the count of something. For example: the total of item given in the game... The Figure 188 shows the XML Schema.

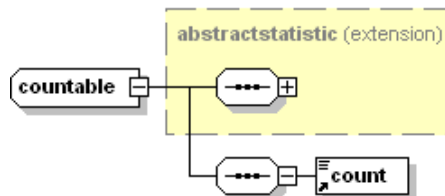


Figure 188 : countable XML Schema

Statistics

Statistics is the main tag that contains all the statistics for the players. This is a collection of different statistic. The Figure 189 shows the XML Schema.

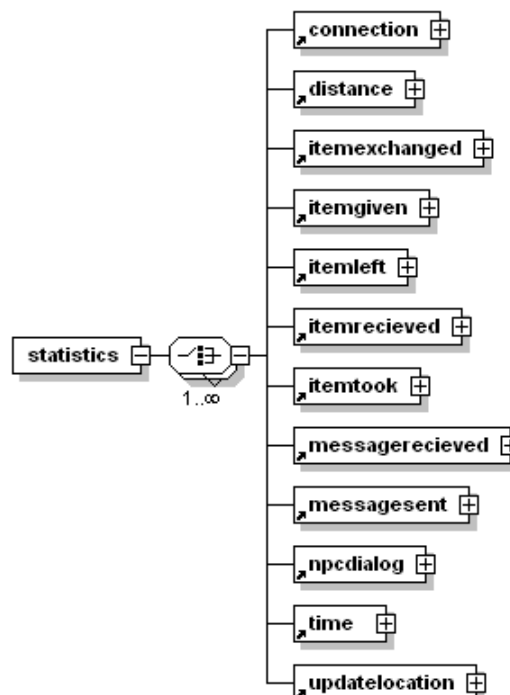


Figure 189 : statistics XML Schema

The Table 27 shows all the tags with their cardinality and meaning.

Tag	Cardinality	Type	Description
Abstractstatistic Tag			
<id>	1	Positive	This is the identifier of the abstract statistic.
Countable Tag [extends abstractstatistic tag]			
<count>	1	Positive	This is the amount value for the statistic.
Root Tag			
<statistics>	1	ComplexType	Root tag that contains other tags.
Statistics Tag (all sub tag [extends countable tag])			
<connection>	0-N	ComplexType	The number of player's connection.
<distance>	0-N	ComplexType	This is the estimated distance did by the player.
<itemexchanged>	0-N	ComplexType	This is the number of item that the player has exchanged.
<itemgiven>	0-N	ComplexType	This is the number of item that the player has given.
<itemleft>	0-N	ComplexType	This is the number of item that the player has left.
<itemrecieved>	0-N	ComplexType	This is the number of item that the player has received.
<itemtook>	0-N	ComplexType	This is the number of item that the player has taken.
<messagerecieved>	0-N	ComplexType	This is the number of message that the player has received.
<messagesent>	0-N	ComplexType	This is the number of message that the player has sent.
<npcdialog>	0-N	ComplexType	This is the number of dialog that the player did.
<time>	0-N	ComplexType	This is the connection time done by the player.
<updatelocation>	0-N	ComplexType	This is the number of update location that the player did.

Table 27 : statistics tags

C.3.4. KMEP-WAR – Model View Controller

This part of XML configuration is dedicated to the configuration of the Model View Controller for the Web User Interface. There is only one XML file with its XML Schema file. The Figure 190 shows the root XML Schema of this part of configuration and the Figure 191 shows the action part of the XML Schema.

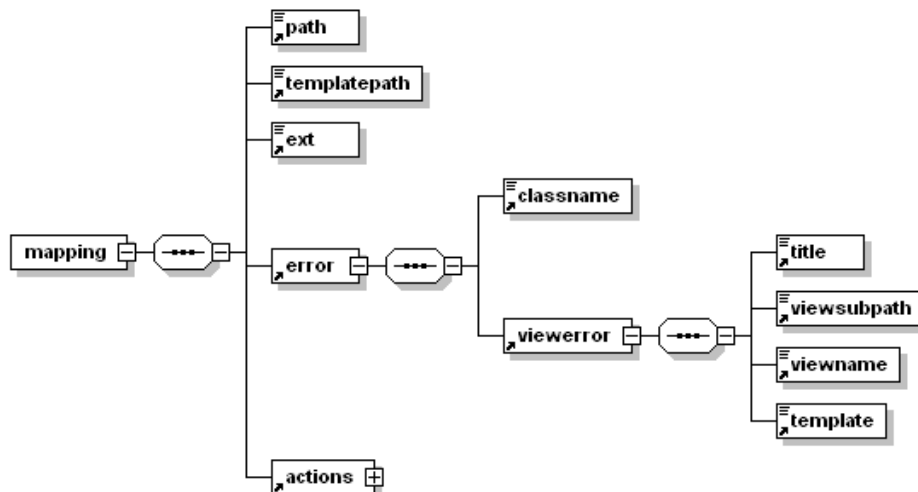


Figure 190 : mappings root XML Schema

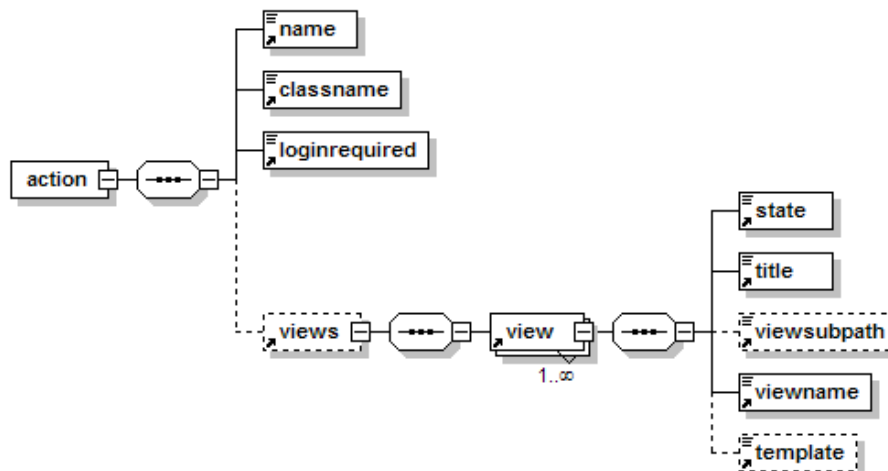


Figure 191 : mappings action XML Schema

The Table 28 shows all the tags with their cardinality and meaning.

Tag	Cardinality	Type	Description
Root Tag			
<mapping>	1	ComplexType	Root tag that contains other tags.
Mapping Tag			
<path>	1	String	This is the path to retrieve the views.

Tag	Cardinality	Type	Description
<templatepath>	1	String	The path to access to the template directory
<ext>	1	String	This is the file extension for the views.
<error>	1	ComplexType	Special view dedicated to the errors encountered in the application.
<actions>	1	ComplexType	This is the collection of actions that represent the mapping action – view.
Error Tag			
<classname>	1	String	Represent the name of the Action classname to handle the errors encountered in the application.
<viewerror>	1	ComplexType	Define the view to show the errors.
Viewerror Tag			
<title>	1	String	Define the title of the view to show to the player.
<viewname>	1	String	Define the name of the view to retrieve the correct view. The name is the part of the file name without extension and path.
<viewsubpath>	0-1	String	Define the path to the category of views. The path must not finish by “/”. The “/” is automatically added to the complete path.
<template>	1	String	The template used to process the rendering with the correct view corresponding to the name and according to the path and extension.
Actions Tag			
<action>	1-N	ComplexType	The action represents the mapping between the action and multiple views depending on the states of the action.
Action Tag			
<name>	1	String	Define the name of the action.
<classname>	1	String	Define the name of the class that handles the action before the rendering process.
<loginrequired>	1	Boolean	Defines if the action needs a connection from the player or not.
<views>	0-1	ComplexType	Define the collection of views that the action can produce as result of action processing.

Tag	Cardinality	Type	Description
Views Tag			
<view>	1-N	ComplexType	The view defines the necessary data to process the rendering.
View Tag			
<state>	1	String	The state for the view corresponding of what an action can return as result after the processing.
<title>	1	String	Define the title to show to the player.
<viewname>	1	String	Define the name of the view to retrieve the correct view. The name is the part of the file name without extension and path.
<template>	0-1	String	The template used to process the rendering with the correct view corresponding to the name and according to the path and extension. If the template is not set, the view name is directly used. The template has to call the view name.

Table 28 : mappings tags

This configuration file is relatively dependant of the implementation of the actions. The actions define the state and names attended to process the views but the actions do not know how the views process the rendering. In this present solution, it is possible to manage different type of Web User Interface based on the same model. For another Web User Interface, it is necessary to create a new controller and new views. The model part and actions can be the same. This is only true for Web User Interface.

C.4. Conclusion

This appendix shown in details all the important points to configure properly the KMEP application for a correct use.

It is important to have a very precise overview of how the configuration has to do. In such kind of application, the complexity of the configuration part is important and has to be clearly defined to help the administrators. In this, the developer and administrator is the same person but this is rarely the truth in big projects. This kind of documentation is very precious.

heig-vd

Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

K-MEP

MEP Installation

Laurent Prévost

R RITSUMEIKAN

Table of Content

D.1. INTRODUCTION	385
D.2. PREREQUISITES	385
D.3. DATABASE CONFIGURATION	385
D.4. PROJECT CONFIGURATION	386
D.5. MEP CONFIGURATION	388
D.6. CONCLUSION	388

Table of Figures

FIGURE 192 : CONNECTION POOL CREATION	386
FIGURE 193 : CONNECTION POOL ADVANCE PROPERTIES	386
FIGURE 194 : JDBC RESOURCES CONFIGURATION	386
FIGURE 195 : PERSISTENCE CONFIGURATION	388

Table of Codes

CODE 21 : PROFILE IN EAR PROJECT.....	387
CODE 22 : DEPLOY ACTION	387
CODE 23 : INTRACK CONFIGURATION.....	388

D.1. Introduction

This appendix covers the installation of the platform. With the installation, the MEP is not only the platform. This is also the implementation of a game in a specific context. For this manual, the explanations are based on some prerequisites explained too. The installation focused on a manual installation. The installation takes place on a localhost.

D.2. Prerequisites

For a correct installation, some elements are needed. The list above shows the different tools supposed installed on your computer. Use the same version as used here or you encounter some troubles are not supported here.

- NetBeans 6.5
- Maven 2.0.9
- MavenIDE (NetBeans plugin) 4.0.5
- MySQL 5.0.67
- MySQL Connector 5.1.5
- Glassfish 2 ur 2
- Java SDK 1.4+
- SVN 1.4.4+

All these tools are necessary to prepare and install the MEP. In the rest of this appendix, they are considered as installed and configured correctly.

The other consideration in this appendix is to have an account on the inTrack platform. For more details about that, contact directly the inTrack team.

D.3. Database configuration

The application need a database correctly configured with the properly connection in glassfish. For that, you have to create a new database and set a user with a password.

In this project, the database configuration is:

Database name:	KMEP
User:	kmep
Password:	kmepPW

When your database is ready, you have to configure the connection inside Glassfish. Go to the web admin console and choose “Resources > JDBC > Connection pools” in the tree menu. After that, click on the “New...” button. Configure the fields with the appropriate data like shown on Figure 192.

Nouveau pool de connexions JDBC (étape 1 sur 2)

Identifiez les paramètres généraux du pool de connexion.

Paramètres généraux

Nom : *	<input type="text" value="kmep"/>
Type de ressource :	<input type="text" value="javax.sql.XADataSource"/>
<small>Doit être spécifié si la classe de source de données implémente plusieurs interfaces.</small>	
Fournisseur de base de données :	<input type="text" value="MySQL"/>

Figure 192 : Connection pool creation

Click on the “Next...” button and go to the list of advance properties. Add the properties shown on Figure 193 with the correct values. Click on “Finish” button to finalize the creation of the connection pool.

<input checked="" type="checkbox"/>	Password	<input type="text" value="kmepPW"/>
<input checked="" type="checkbox"/>	URL	<input type="text" value="jdbc:mysql://localhost:3306/kmep"/>
<input checked="" type="checkbox"/>	User	<input type="text" value="kmep"/>

Figure 193 : Connection pool advance properties

Now, go to the “Resources > JDBC > JDBC Resources” and click on the “New...” button. Configure the fields with the correct data as shown on Figure 194. Finally, click on the “OK” button. The JDBC resource is created.

Nouvelle ressource JDBC

Spécifiez un nom JNDI unique identifiant la ressource JDBC que vous souhaitez créer. Le nom ne doit

Nom JNDI : *	<input type="text" value="jdbc/kmep"/>
Nom du pool : *	<input type="text" value="kmep"/>
<small>Utilisez la page Pools de connexions JDBC pour créer des pools</small>	
Description :	<input type="text"/>
Statut :	<input checked="" type="checkbox"/> Activé

Figure 194 : JDBC Resources configuration

Now, the configuration for the database is done and ready to be used inside the project.

D.4. Project configuration

It is necessary to configure some project parts to run properly the installation of the project. The first thing to do is to create a file “.asadminpassword” in the “/config” of the Glassfish installed folder. The file must to contain this: “AS_ADMIN_PASSWORD=adminadmin” (change with the appropriate password if necessary).

Now, create a profile inside the “pom.xml” in the EAR project. Use the profile shown on Code 21 and update the red values with your own values. Maybe other changes are needed due to your specificities.

```
<profile>
  <id>lprevostMBPProfil</id>
  <activation>
    <property>
      <name>deploy.server</name>
      <value>lprevostMBP</value>
    </property>
  </activation>
  <properties>
    <appserver.home>
      /Applications/NetBeans/glassfish-v2ur2
    </appserver.home>
    <appserver.user>admin</appserver.user>
    <appserver.passwordfile>
      ${appserver.home}/config/.asadminpassword
    </appserver.passwordfile>
    <appserver.domain>domain1</appserver.domain>
    <appserver.host>localhost</appserver.host>
    <appserver.port>4848</appserver.port>
  </properties>
</profile>
```

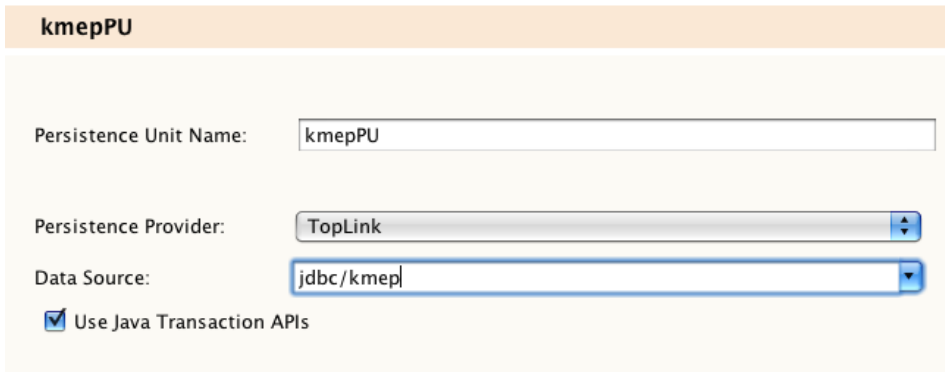
Code 21 : Profile in EAR project

After that, create the deployment action to allow deploying the application from NetBeans. Go to the “nbactions.xml” in the EAR project and create your action as shown on Code 22. Change the red values with the correct values that suit to your project.

```
<action>
  <actionName>CUSTOM-deploy locally (lprevostMBP)</actionName>
  <displayName>deploy locally (lprevostMBP)</displayName>
  <goals>
    <goal>install</goal>
    <goal>exec:exec</goal>
    <goal>-e</goal>
  </goals>
  <properties>
    <deploy.server>lprevostMBP</deploy.server>
  </properties>
</action>
```

Code 22 : Deploy action

Finally, to finish the project configuration, it is necessary to update the “persistence.xml” inside the EJB project. The medication is to use the correct JDBC resource configured before. Change the Data Source value to the correct one as shown on Figure 195.



The image shows a web-based configuration form for a persistence unit named 'kmepPU'. It includes three input fields: 'Persistence Unit Name' with the value 'kmepPU', 'Persistence Provider' with a dropdown menu showing 'TopLink', and 'Data Source' with a dropdown menu showing 'jdbc/kmep'. There is also a checked checkbox labeled 'Use Java Transaction APIs'.

Figure 195 : Persistence configuration

The project is correctly configured to use the database and to be deployed on the local Glassfish server.

D.5. MEP configuration

Another important part to configure is the data for the MEP. This part does not go in details for the game data configuration but provides the necessary to the deployment. To configure the game data, refer you to the Configuration appendix. This part focused on the inTrack configuration.

To configure the inTrack communication between the both platforms, go to the "inTrackConfig.properties" and edit the values to your needs like shown on Code 23. The data in orange are to be changed by your data.

```
# inTrack configuration
intrack.usage=true
intrack.dev=false

# inTrackPartner dev environment
intrack.dev.user=kmep_dev
intrack.dev.pwd=*****
intrack.dev.moduleid=INTRACK_ASSIGNED_ID_KMEP_DEV
intrack.dev.trackingidprefix=INTRACK_KMEP_DEV_

# inTrackPartner prod environment
intrack.prod.user=kmep_prod
intrack.prod.pwd=*****
intrack.prod.moduleid=INTRACK_ASSIGNED_ID_KMEP_PROD
intrack.prod.trackingidprefix=INTRACK_KMEP_PROD_
```

Code 23 : inTrack configuration

The MEP allows to inTrack environment. Ask the inTrack team to get the correct data to place in the configuration.

D.6. Conclusion

With the previous configuration, it is possible to deploy the application on a computer and communicate with the inTrack platform provided by the inTrack team. For the game data configuration, the appendix on the configuration is more efficient.

heig-vd

Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

K-MEP

Error Codes

Laurent Prévost

The logo for RITSUMEIKAN, featuring a large white letter 'R' on the left and the word 'RITSUMEIKAN' in white uppercase letters to its right, all set against a dark red rectangular background.

R RITSUMEIKAN

Table of Content

E.1. INTRODUCTION	393
E.2. PRINCIPLES	393
E.3. CATEGORIES	393
E.3.1. General errors	394
E.3.2. Player errors	395
E.3.3. Kmeq Entity errors	396
E.3.4. Quest Errors	396
E.3.5. Dialog errors	396
E.3.6. Tracking errors	397
E.3.7. Item errors	397
E.3.8. Message errors	398
E.3.9. Config errors	398
E.3.10. Statistic errors	398
E.4. CONCLUSION	398

Table of Tables

TABLE 29 : ERROR CATEGORIES.....	394
----------------------------------	-----

E.1. Introduction

This appendix offers description of the different Error Codes that the application can encounter. The idea is to have an Exception Class to handle all errors in the application with the support of a ReasonProvider Class. This second class brings the error messages (technical or user) to show.

E.2. Principles

The goal to this working manner is to provide a high degree of error management. Everywhere in the application, the need to manage the errors is present. For example, when a player wants to register and he chooses a same nickname than another player. In this situation, we have to advertise the player of the problem.

The KMEPEXception class was developed for that purpose. This exception can take a Return Code in the constructor. In association with the ReasonProvider Class, it is possible to retrieve an error message (technical message or user message). The user's messages can directly be shown to the player without any other treatment. This is very useful for an easy management of errors.

In the code, the others exceptions are handled and transformed into the KMEPEXception with the appropriate Return Code. The two classes are provided into the common part of the application and can be available for the client part than the server part. In this project, it means for the EJB part and WAR part (model and web application).

Two property files offer the possibility to manipulate and translate the technical and user messages. Each file is dedicated to one of message type. The first one "technicalMessages.properties" is dedicated to technical purpose and the second "useMessages.properties" is dedicated to the user purpose. All these files are located in the resources part of the common project.

E.3. Categories

Each error code is placed into a category. Some categories allow differentiating the source of the error. For example, the error code for the use of an existent nickname is the Player category. An error occurred in the treatment of an HTTP parameter is considered like a General error.

The Table 29 shows the different categories and adds a short description of the utility of it.

Category	Description
General	General errors can encounter everywhere in the application. This is all errors that cannot be placed in other specific category.

Category	Description
Player	Player errors are specific to the PlayerManager. These errors are dedicated to handle the problems in the player management.
Kmep Entity	Kmep Entity errors are dedicated to handle problems in kmep entity management. All errors in this category are referred to the Kmep entity manager.
Quest	Quest errors are specific to the QuestManager. In this category, there are only the errors produced by the quest management.
Dialog	Dialog errors are dedicated to manage errors from the dialog management. These errors are encountered from the dialog manager.
Tracking	Tracking errors is used to manage the errors that come from the tracking management and inTrack usage.
Item	Item errors are especially used in the Item Manager. They are used to manage the errors relative to the management of items.
Message	This category allows managing errors relative to the message management especially done in the MessageManager.
Config	The configuration errors are used anywhere of configuration management and especially during the data loading when the game is deployed.
Statistic	These errors are used for the statistic management. The StatisticManager generate them in general.

Table 29 : Error categories

Errors in each category were numbered in the code apparition order. There is no specific logic for the numbers. This is the same for categories. The first category (General) has the numbers -1 to -99 (not the zero), the second category has the numbers -100 to -199 and so on. The zero number is reserved for the normal case (no problem encountered). The positives numbers are not used actually.

E.3.1. General errors

Code	Reason	Constant
-1	Unexpected exception	ERR_GENERAL_EXCEPTION
-2	Invalid parameter type	ERR_GENERAL_TYPE
-3	Connection required for a function	ERR_GENERAL_FUNC_DENY

Code	Reason	Constant
-4	No configuration value found	ERR_GENERAL_NO_VALUE
-5	Wrong type value from configuration	ERR_GENERAL_INVALID_VALUE
-6	Unsupported hash type	ERR_GENERAL_INVALID_HASH
-7	Invalid longitude	ERR_GENERAL_INVALID_LON
-8	Invalid latitude	ERR_GENERAL_INVALID_LAT
-9	Invalid altitude	ERR_GENERAL_INVALID_ALT
-10	Requested parameter not found	ERR_GENERAL_PARAM_NOT_FOUND
-11	Invalid Transfer Object	ERR_GENERAL_INVALID_TO
-12	Security access violation	ERR_GENERAL_SECURITY_VIOLATION
-13	XML Loading error	ERR_GENERAL_XML_LOADING_ERROR
-14	Naming error (service retrieving)	ERR_GENERAL_SERVICE_NAME_ERROR

E.3.2. Player errors

Code	Reason	Constant
-100	Player already exists	ERR_PLAYER_EXISTS
-101	Password and Password confirmation don't match	ERR_PLAYER_PWD_PWDCONF
-102	Unknown player	ERR_PLAYER_UNKNOWN
-103	Player already connected	ERR_PLAYER_CONNECTED
-104	Wrong password	ERR_PLAYER_PWD
-105	Player not connected	ERR_PLAYER_DISCONNECTED
-106	Same password than the previous	ERR_PLAYER_SAME_PWD
-107	Same avatar than the previous	ERR_PLAYER_SAME_AVATAR
-108	Chosen game mode is not available	ERR_PLAYER_GM_INVALID
-109	Same game mode than the previous	ERR_PLAYER_SAME_GM
-110	Avatar unavailable	ERR_PLAYER_AVATAR_INVALID
-111	The nickname is null	ERR_PLAYER_NULL_NICK
-112	The password is null	ERR_PLAYER_NULL_PWD
-113	The confirmation password is null	ERR_PLAYER_NULL_CONF_PWD
-114	The new password is null	ERR_PLAYER_NULL_NEW_PWD

E.3.3. Kmep Entity errors

Code	Reason	Constant
-200	Behavior already added	ERR_KE_BEHAVIOUR_EXISTS
-201	Mobile entity doesn't exist	ERR_KE_NOT_EXIST
-202	Same picture tat the previous	ERR_KE_SAME_IMAGE
-203	Behavior not present	ERR_KE_NO_BEHAVIOUR

E.3.4. Quest Errors

Code	Reason	Constant
-300	No first quest available	ERR_REQUEST_NO_FIRST
-301	No quest diary for the player	ERR_REQUEST_NO_DIARY
-302	No quest found	ERR_REQUEST_NOT_FOUND
-303	No quest state corresponding to the quest	ERR_REQUEST_NO_STATE
-304	Quest condition is not from the correct type	ERR_REQUEST_BAD_CONDITION
-305	Quest is not "discardable"	ERR_REQUEST_NOT_DISCARDABLE
-306	Quest is not "retryable"	ERR_REQUEST_NOT_RETRYABLE

E.3.5. Dialog errors

Code	Reason	Constant
-400	Dialog doesn't exist	ERR_DIALOG_UNKNOWN
-401	Dialog is not "statable"	ERR_DIALOG_NOT_STABLE
-402	Dialog is not quest proposal	ERR_DIALOG_NOT_REQUEST
-403	Dialog quest proposal has no quest	ERR_DIALOG_NO_REQUEST
-404	No refuse dialog available	ERR_DIALOG_NO_REFUSE_DIALOG
-405	No accept dialog available	ERR_DIALOG_NO_ACCEPT_DIALOG
-406	Dialog is not quest finish	ERR_DIALOG_NOT_REQUEST_END
-407	Dialog is not item dialog	ERR_DIALOG_NOT_ITEM
-408	Class cast dialog error.	ERR_DIALOG_CAST_ERROR

E.3.6. Tracking errors

Code	Reason	Constant
-500	An exception occurred in inTrack	ERR_TCK_EXCEPTION
-501	Parameter is null	ERR_TCK_PARAM_NULL
-502	MD5 Error occurred	ERR_TCK_MD5_ERROR
-503	No connection with inTrack	ERR_TCK_NOT_CONNECT
-504	Mobile entity not found	ERR_TCK_ME_NOT_FOUND
-505	Tracking id not found	ERR_TCK_TID_NOT_FOUND
-506	Tracking id already owned by a mobile entity	ERR_TCK_TID_OWNED
-507	Tracking id is not owned by any mobile entity	ERR_TCK_TID_NO_OWNER
-508	Tracking id already exists	ERR_TCK_TID_EXISTS
-509	Tracking id name already exists	ERR_TCK_TID_NAME_EXISTS
-510	User not found	ERR_TCK_USER_NOT_FOUND
-511	Wrong password	ERR_TCK_WRONG_PASSWORD
-512	Parameter not found	ERR_TCK_PARAM_NOT_FOUND
-513	Parameter name already exists	ERR_TCK_PARAM_NAME_EXISTS
-514	Tracking module not found	ERR_TCK_MODULE_NOT_FOUND
-515	Tracking module exists	ERR_TCK_MODULE_EXISTS
-516	Local mobile entity not found	ERR_TCK_LOCAL_ME_NOT_FOUND
-595	Entity is not from the correct type	ERR_TCK_NOT_CORRECT_TYPE
-596	Null coordinate	ERR_TCK_COORD_NULL
-597	Binding error between local and remote entity	ERR_TCK_BINDING_ERROR
-598	Unknown error	ERR_TCK_UNKNOWN_ERROR
-599	An exception occurred in the inTrackManager (local)	ERR_TCK_EX_MANAGER

E.3.7. Item errors

Code	Reason	Constant
-600	Added item is not an item	ERR_ITEM_NOT_ITEM
-601	Item type does not exist	ERR_ITEM_NO_ITEM_TYPE

Code	Reason	Constant
-602	Multi Item Type does not exist	ERR_ITEM_NO_MULTI_ITEM_TYPE
-603	Item was already took	ERR_ITEM_ALREADY_TOOK
-604	Item is not available from the item source	ERR_ITEM_NOT_AVAILABLE

E.3.8. Message errors

Code	Reason	Constant
-700	Message not found	ERR_MSG_NOT_FOUND
-701	Message is from the wrong type	ERR_MSG_WRONG_TYPE

E.3.9. Config errors

Code	Reason	Constant
-800	There is no value for the key	ERR_CFG_NO_VALUE
-801	The value is not from integer	ERR_CFG_NOT_INTEGER
-802	The value is not from Boolean	ERR_CFG_NOT_BOOLEAN
-803	The value is not from long	ERR_CFG_NOT_LONG
-804	The XML id was already loaded	ERR_CFG_XML_ID_EXISTS
-805	The XML id doesn't exist	ERR_CFG_XML_ID_NOT_EXIST

E.3.10. Statistic errors

Code	Reason	Constant
-900	Statistic already exists	ERR_STAT_EXISTS
-901	Statistic is not from the correct type	ERR_STAT_WRONG_TYPE
-902	Statistic is not from the correct category	ERR_STAT_WRONG_CATEGORY

E.4. Conclusion

There is a possibility to handle easily the errors everywhere in the application and this is possible to retrieve specific message with a return code depending on user or technical view.

The ReasonProvider offers all the constants for the error code numbers and allow modifying all codes easily in the case of a global change of codes.

heig-vd

Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

K-MEP

Mobile Phone Research

Laurent Prévost



R RITSUMEIKAN

Table of Contents

F.1. INTRODUCTION	403
F.2. CRITERIA	403
F.2.1. GPS	403
F.2.2. Bluetooth	403
F.2.3. Touch-Screen	403
F.2.4. Bilingual	403
F.2.5. Java	403
F.2.6. Public target	403
F.3. MOBILE PHONE ENVIRONMENT	404
F.4. OPERATORS	404
F.4.1. NTT Docomo	404
F.4.2. Softbank	405
F.4.3. AU by KDDI	405
F.5. FINAL CHOICE	405

F.1. Introduction

This appendix will explain the choices made for the mobile phone used in KMEP project. Criteria of the choice are exposed.

F.2. Criteria

In a first research, a mobile phone with many embedded functionalities was researched.

F.2.1. GPS

This functionality is the most important for the project because this is the way to get location data. Localization is the major functionality of KMEP. It is more convenient to have an embedded GPS than a separate peripheral.

F.2.2. Bluetooth

Near interactions, require a local communication between players. Bluetooth is a good approach for this kind of communication.

F.2.3. Touch-Screen

This is not the most important needs for the project but it can be a very interesting approach for the mobile phone user interface. It can be more users friendly to have a touch screen to use the game.

F.2.4. Bilingual

The development of the project is done only in English. A mobile phone with English menus is very important to avoid lost of time during the tests and configuration of the device.

F.2.5. Java

The programming language is Java for convenient reason. This is the best-known language for this project. The planning for the project is very short and saving some learning time is very useful.

F.2.6. Public target

Another important thing is if the game could be tested by many people, they need to have the right technology to run the game. This fact forces to choose a mobile phone with same functionalities than the most of other mobile phones already present on the market.

F.3. Mobile phone environment

Actually, in Japan, Bluetooth is not very used. The population of mobile phone does not have the Bluetooth.

For the GPS, this is not the same. The luck to find a phone with GPS services is more important than Bluetooth.

Touch-Screen is like Bluetooth. There is not a lot of mobile phone with this capability. iPhone has this functionality but Java is not supported yet.

F.4. Operators

In Japan, there three majors mobile phone operators. They are others operators but often they are virtual providers. It means they use the infrastructure of one of the major operators to offer their services. The best is to concentrate on the most used operators.

F.4.1. NTT Docomo

This provider offers a platform to develop Java application based on its own proprietary Java API. This API is called DoJa (Docomo Java). This API is very similar to MIDP1.0. The constraint is the application developed for this provider cannot be used on other provider phones. There is also a CLDC1.0/1.1

Applications on this provider are called “iAppli” in reference of “iMode”. “iMode” is the service to offer Internet connection to the NTT Docomo customers.

The deployment of the applications can be done by a website. In this case, the application can only communicate with this website.

Applications are limited in the size. Depending of the DoJa version, application can vary between 30KB and 1MB (DoJa 5.0), and there is another space to store data during the life of the application. This space is called Scratchpad and his size varies between 100KB to 1MB (DoJa 5.0). In DoJa 5.0, the application space and Scratchpad are a share space of 1MB.

Their price plans are upper than AU by KDDI for about 30'000 Yen (about 300 CHF). For 4 – 5 months, the estimated total cost is about 110'000 to 125'000 Yen (1'100 to 1'250 CHF).

Pay attention to the fact that you cannot do everything you want with an iAppli. The technical document is not so clear after a first read. To use some functionalities of the mobile phone with an iAppli, you have to validate your application. For that, you must to be a Trusted Content Provider agreed by NTT Docomo. If you do not have this state, you cannot use the GPS directly from an iAppli for example. The restrictions concerned some functions from the API like location. In this case, the iAppli have to be an iAppli DX. It means that the provider of the content must to be trusted by NTT Docomo.

F.4.2. Softbank

This provider uses a MIDP1.0/2.0 base with an additional API for accessing their phones hardware. There is also CLDC1.0/1.1. The difficulty is to deploy the application on the mobile phone. This is necessary to use the Softbank service to offer the application. Actually, it seems that there is not many information about deploying the application by another way. Applications developed for Softbank are called S! Appli.

Softbank offers the iPhone 3G that can be a great platform for this kind of project but the SDK is only for MAC OS. Actually, the development is done under Windows XP and another problem is the language for development. The language is Objective C never used by the developer. You have to join the developer platform from Apple.

F.4.3. AU by KDDI

Like Softbank, this provider uses a MIDP1.0/2.0 base with an additional API for accessing their phones. There is also CLDC1.0/1.1. The difficulty is to deploy the application on the mobile phone. This is necessary to use the AU by KDDI service to offer the application. Actually, there is not many information about deploying the application by another way. There is also a technology called BREW to develop application for their phones.

Documentation is available in Japanese version only. It does not seem some documentation in English will be available.

Their price plans are under NTT Docomo for about 30'000 Yen (about 300 CHF). For 4 – 5 months, the estimated total cost is about 80'000 to 95'000 Yen (800 to 950 CHF).

F.5. Final choice

The final choice is based on the price but also on the development capabilities. This last point is the most important. AU by KDDI does not offer an open platform to develop and deploy the mobile application. NTT Docomo is not so open too, but this is possible to host its application on its own webserver and after that, the application can only communicate with its own webserver via iMode. In the case of AU by KDDI, the application must to be signed by them and is provided by their webserver.

Finally, the choice is not easy to do and cannot pay attention only on the prices. Due to the development and requirements for the application, the needs are some easiest way to develop the client side.

The phone chosen is the SO906i. This mobile has GPS functions, a screen with a resolution of 480x864, motion sensor, bilingual and other functionalities. The price of the mobile phone is about 600 CHF (with accessories: AC Adapter, USB cable, dock). The monthly charges allow using iMode with unlimited packet.

heig-vd

Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

K-MEP

DoJa 5.1 Quick Start

Laurent Prévost

The logo for RITSUMEIKAN, featuring a large white letter 'R' on the left and the word 'RITSUMEIKAN' in white uppercase letters to its right, all set against a dark red rectangular background.

R RITSUMEIKAN

Table of Contents

G.1. INTRODUCTION	413
G.2. DOWNLOAD	413
G.3. INSTALLATION	413
G.4. CONFIGURATION	418
G.5. NETBEANS MODULE INSTALLATION	420
G.6. FIRST PROJECT IN DOJA 5.1 WITH NETBEANS 6.1	422
G.7. EMULATOR	426
G.8. SCREEN	431
G.9. CONCLUSION	432

Table of Figures

FIGURE 196 : LANGUAGE CHOICE.....	413
FIGURE 197 : INSTALLATION WELCOME SCREEN	414
FIGURE 198 : AGREEMENT LICENSE	414
FIGURE 199 : PATH CHOICE	415
FIGURE 200 : TYPE OF INSTALLATION	415
FIGURE 201 : COMPONENTS TO BE INSTALLED.....	416
FIGURE 202 : PATH OF NETBEANS & ECLIPSE MODULE	416
FIGURE 203 : ECLIPSE PATH CHOICE.....	417
FIGURE 204 : SUMMARY OF THE INSTALLATION	417
FIGURE 205 : INSTALLATION	418
FIGURE 206 : .JAM FILES ASSOCIATION	418
FIGURE 207 : FILES IN LIB/118N DIRECTORY	419
FIGURE 208 : DOJA 5.1 SDK WITH EMULATOR.....	419
FIGURE 209 : DOWNLOADED TAB FROM PLUGINS	420
FIGURE 210 : PLUGIN CHOICE	420
FIGURE 211 : DOJA 5.1 PLUGIN CHOICE.....	421
FIGURE 212 : INSTALLATION OF THE PLUGIN.....	421
FIGURE 213 : SIGNED WARNING	422
FIGURE 214 : NEW DOJA 5.1 PROJECT	422
FIGURE 215 : PROJECT CONFIGURATION.....	423
FIGURE 216 : HELLOWORLD CLASS CREATION	423
FIGURE 217 : DOJA PROPERTIES FILE	424
FIGURE 218 : DOJA 5.1 PROJECT CONFIGURATION (ADF CONFIGURATION).....	425
FIGURE 219 : EMULATOR VIEW FROM THE HELLOWORLD PROJECT.....	426
FIGURE 220 : CONSOLE OUTPUT OF THE HELLOWORLD PROJECT.....	426
FIGURE 221 : DEVICE CONFIGURATION TAB.....	427
FIGURE 222 : DEVICE CONFIGURATION CREATION	428
FIGURE 223 : DEVICE SELECTION.....	428
FIGURE 224 : NATIVE DATA EDIT SCREEN	429
FIGURE 225 : TRUSTED APPLICATION	430
FIGURE 226 : TRUSTED CONFIGURATION	430
FIGURE 227 : RUN SETUP CONFIGURATION.....	431
FIGURE 228 : SCREEN RESOLUTION CONFIGURATION	431

Table of Codes

CODE 24 : CODE OF THE CLASS HELLOWORLD	424
----------------------------------------------	-----

G.1. Introduction

This appendix explains how to install and use DoJa 5.1 from NTT Docomo for an iAppli development. There is also a part for integrate the DoJa SDK in NetBeans 4.0 (updated for NetBeans 6.1 or 6.5).

G.2. Download

The first step for this installation is to download the DoJa 5.1 from NTT Docomo website. Pay attention to the documentation only in Japanese but at this step, it is not a problem. A forum is available to find some information very helpful.

The link to download:

<http://www.nttdocomo.co.jp/english/service/imode/make/content/iappli/about/index.html#tool>

This link is more direct for the version 5.1:

http://www.nttdocomo.co.jp/binary/archive/service/imode/make/content/iappli/tool/doja51/download/emufordoja5_1_2_00.zip

G.3. Installation

In a first time, decompress the zip file on your hard disk and run the Disk1/Setup.exe. Choose the English language.

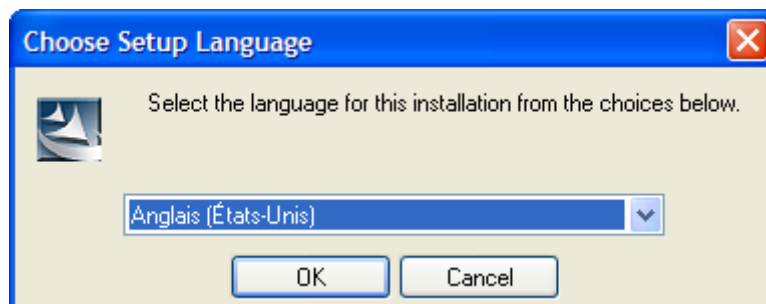


Figure 196 : Language choice

Wait the next step of the installation and click on the Next button.

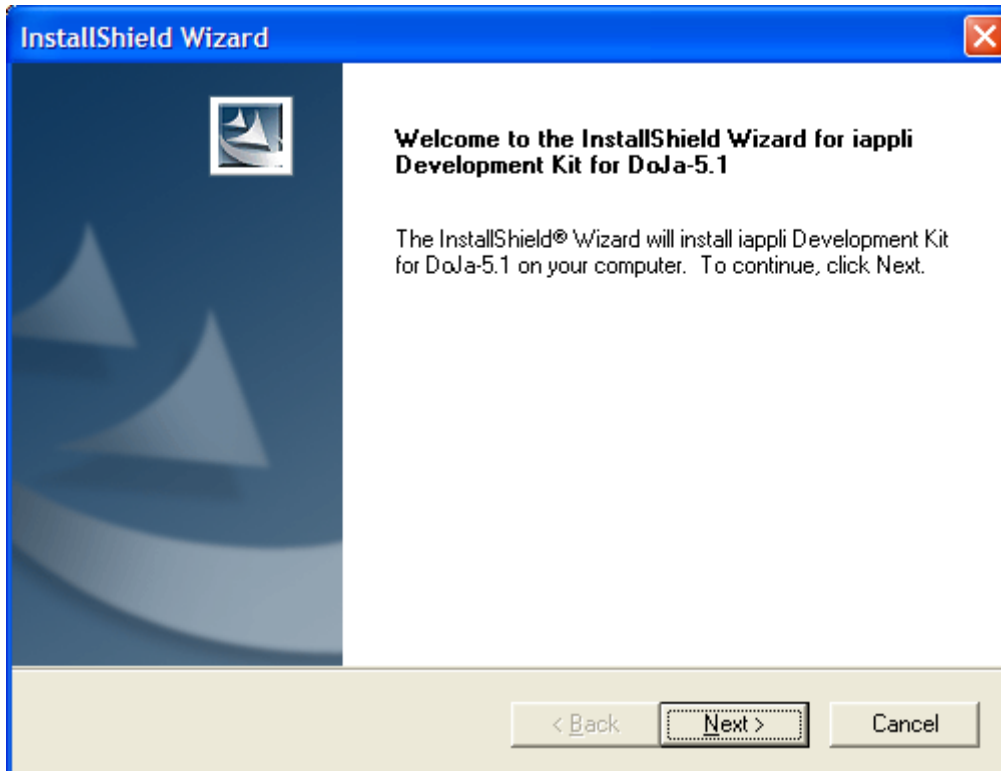


Figure 197 : Installation welcome screen

Read the agreement license and click on the Yes button.

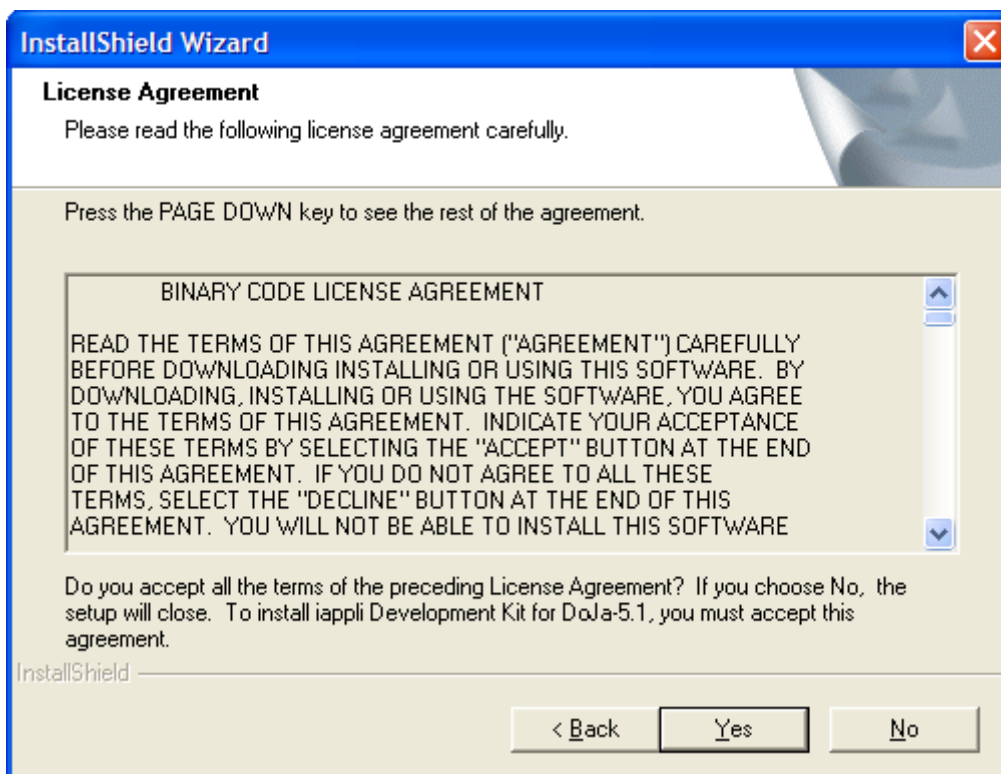


Figure 198 : Agreement license

Change the path if you want but by default, the path is directly at the root of C: after that click on Next button.

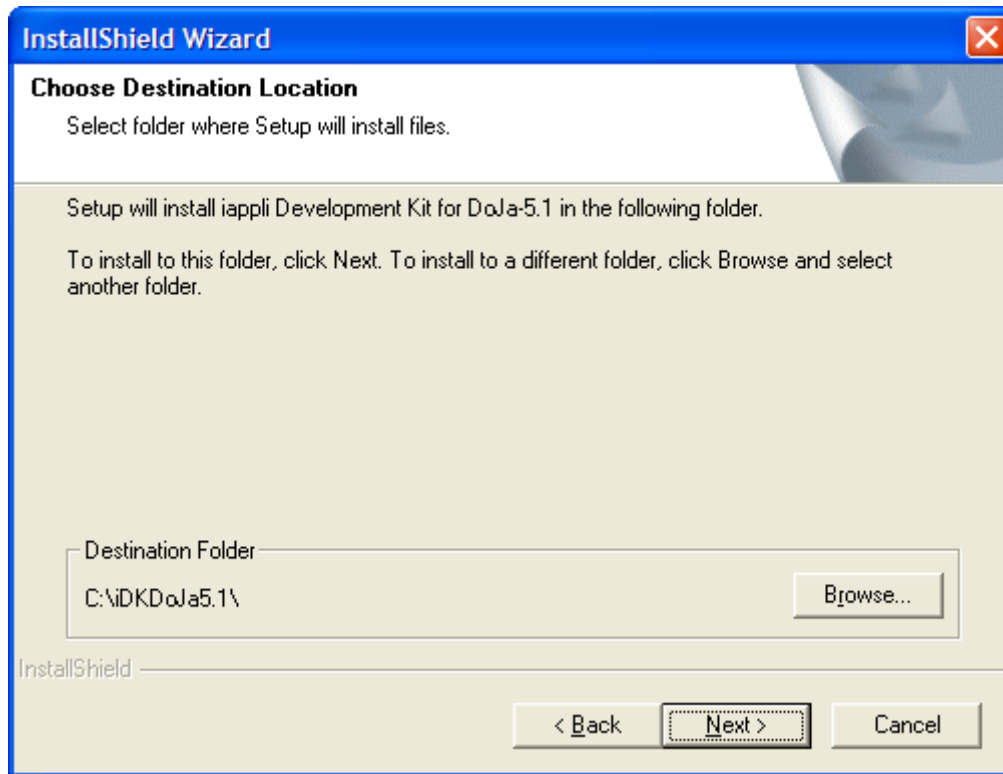


Figure 199 : Path choice

Choose the custom option to choose the different components to be installed.

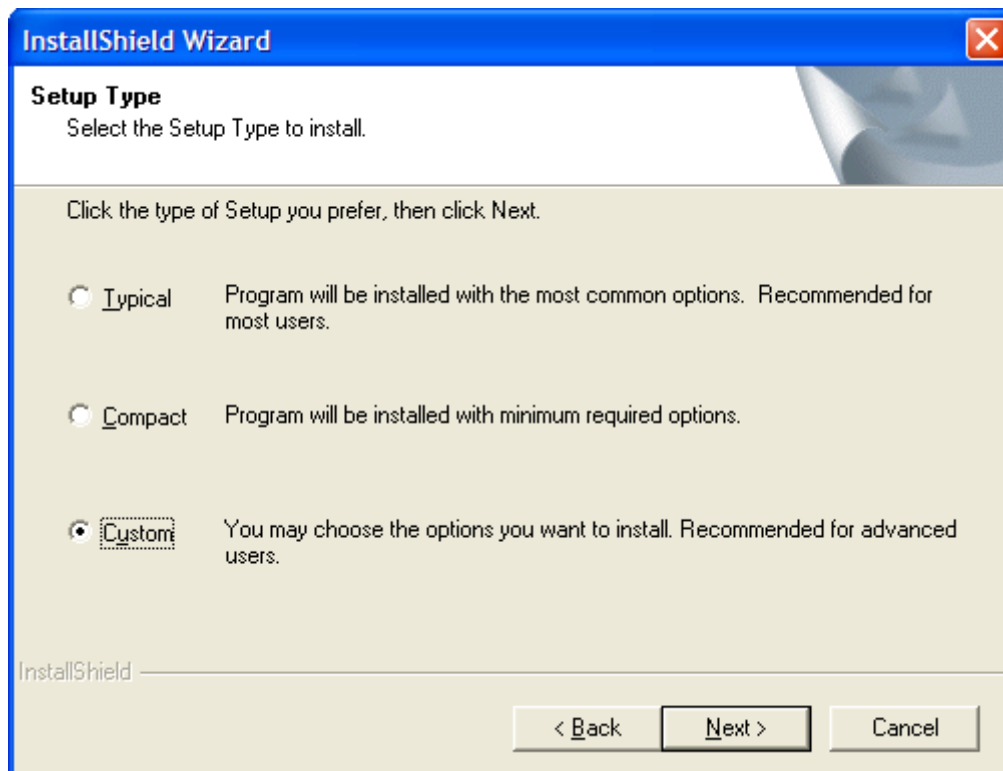


Figure 200 : Type of installation

Verify that all the components are checked for installation. Do not worry about the version of NetBeans module. It works also with NetBeans 6.1/6.5 with a little trick. Click on the Next button.

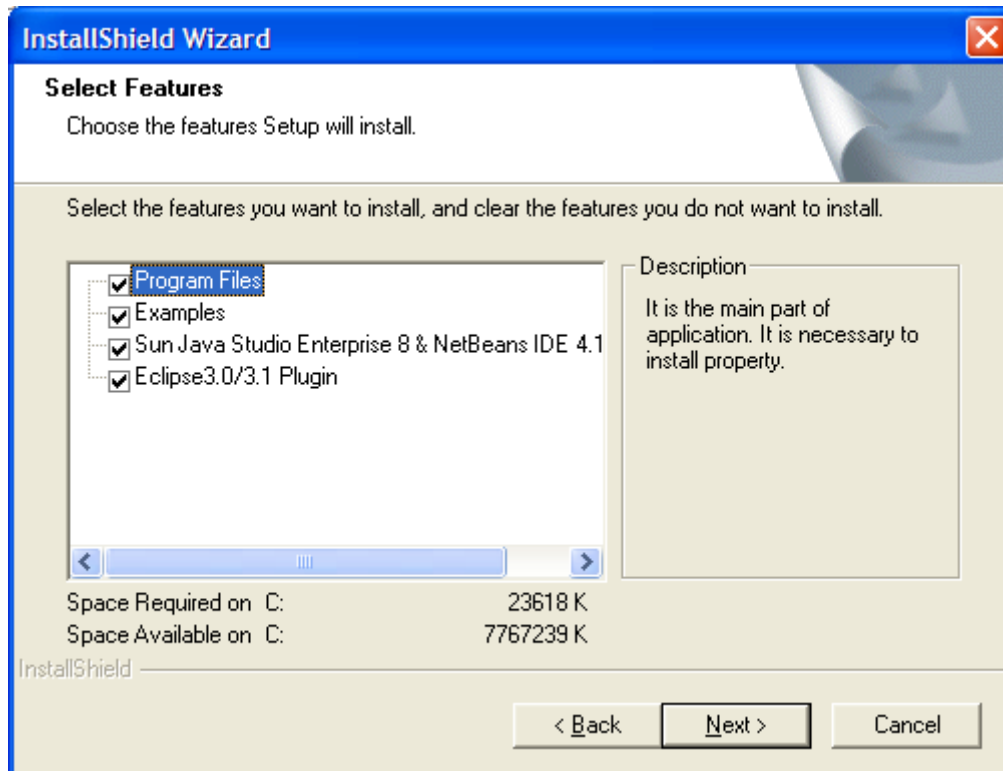


Figure 201 : Components to be installed

A message will be display to your attention to show where the component for Eclipse and NetBeans will be installed. Be carefully about this location, it is needed later. Click on the Ok button.

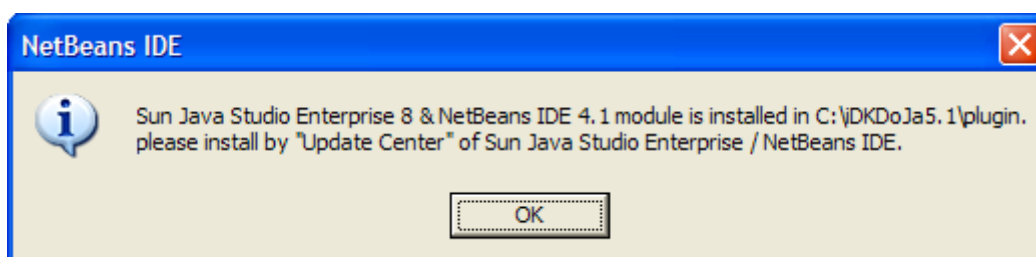


Figure 202 : Path of NetBeans & Eclipse module

For Eclipse, you have to specify the installation path of it. Choose the correct path and click on the Next button. If you do not have Eclipse, skip this step just by clicking Next button.

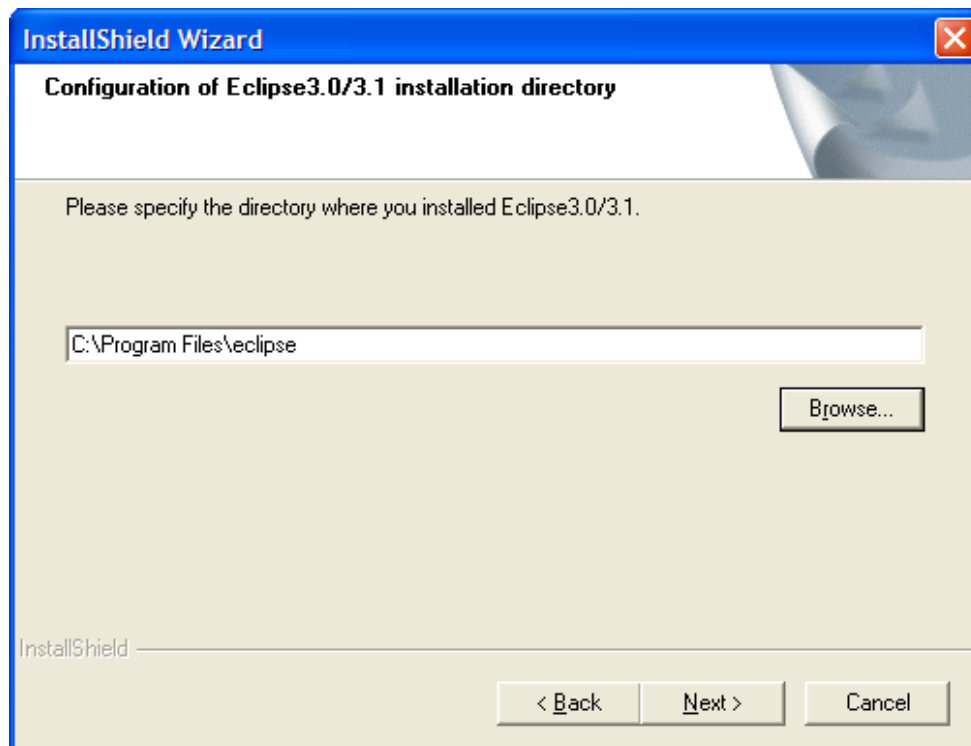


Figure 203 : Eclipse path choice

At this step, the setup summarizes all the information you configured before this step. You have just to click Next button to proceed of the installation.

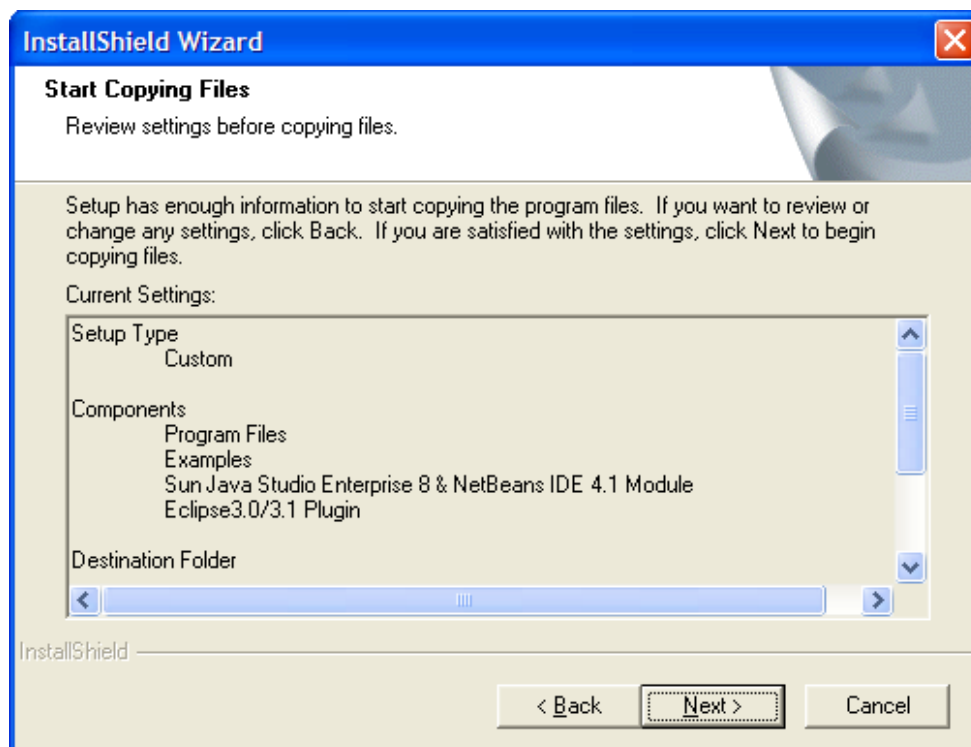


Figure 204 : Summary of the installation

Finally, wait until the end of the installation.

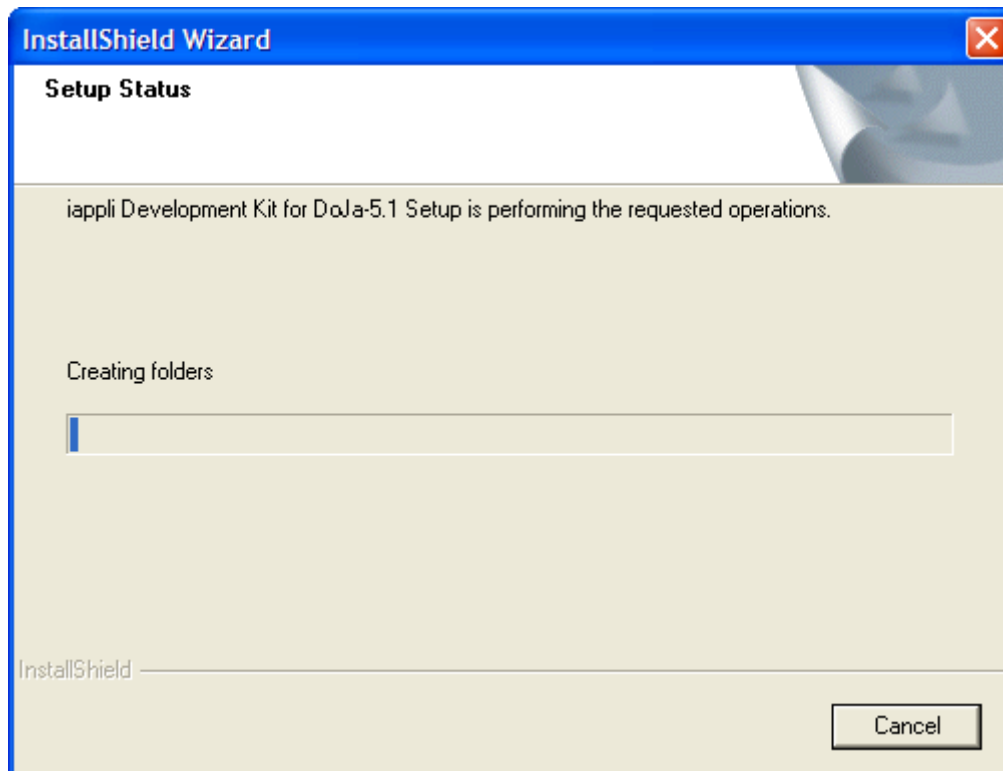


Figure 205 : Installation

After the installation, you will see a message to associate .jam files with the DoJa Emulator. Click on the Yes button to do it. After this message, the installation is finished. Just click Finish to end the installation.

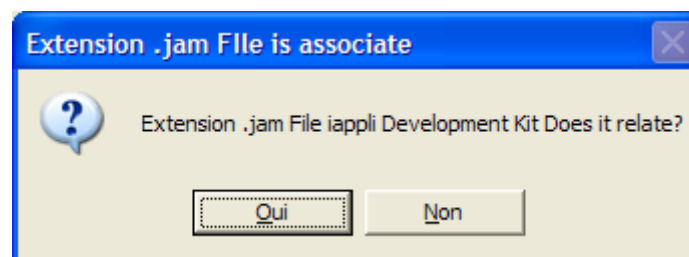


Figure 206 : .jam files association

G.4. Configuration

Actually and depending of you Operating System, the SDK could not work properly. The reason is that the SDK is ready for English OS or Japanese OS. To run the SDK in other Operating System Language, the name of three files has to be changed. Go to the DoJa SDK installation path and find the "lib/i18n" directory.

In this directory, you find these files:

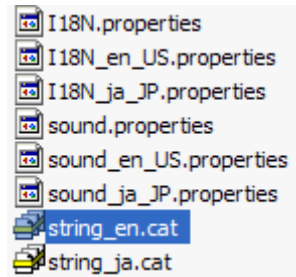


Figure 207 : Files in lib/i18n directory

Three files with “en” in the name have to be renamed. In this case, the Operating System is in French with Switzerland for the locale parameters.

The files are renamed like that:

- string_en.cat → string_fr.cat
- sound_en_US.properties → sound_fr_CH.properties
- I18N_en_US.properties → I18N_fr_CH.properties

Pay attention that the language is always English. This trick is only to allow running the DoJa SDK in another Operating System language.

To verify that the SDK is ok, you have just to go in Start menu and find the shortcut for “iappliTool for DoJa-5.1(FOMA)”. Click on it and you will be able to see the SDK environment with the emulator.

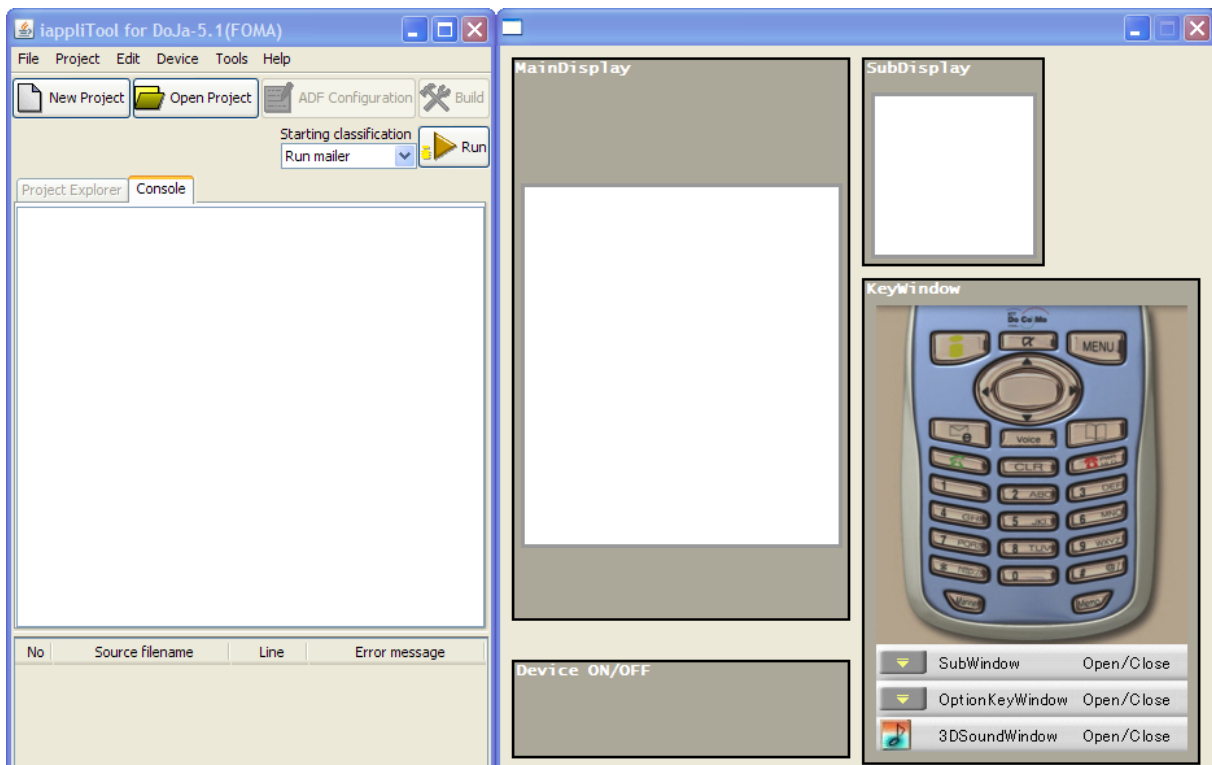


Figure 208 : DoJa 5.1 SDK with Emulator

G.5. NetBeans Module Installation

The DoJa 5.1 allows integrating a module directly in NetBeans. However, it does not work directly in NetBeans 6.1.

First, module in NetBeans must to be installed. For that, open your NetBeans IDE and go to menu “Tools → Plugins”. Go in the tab “Downloaded”.

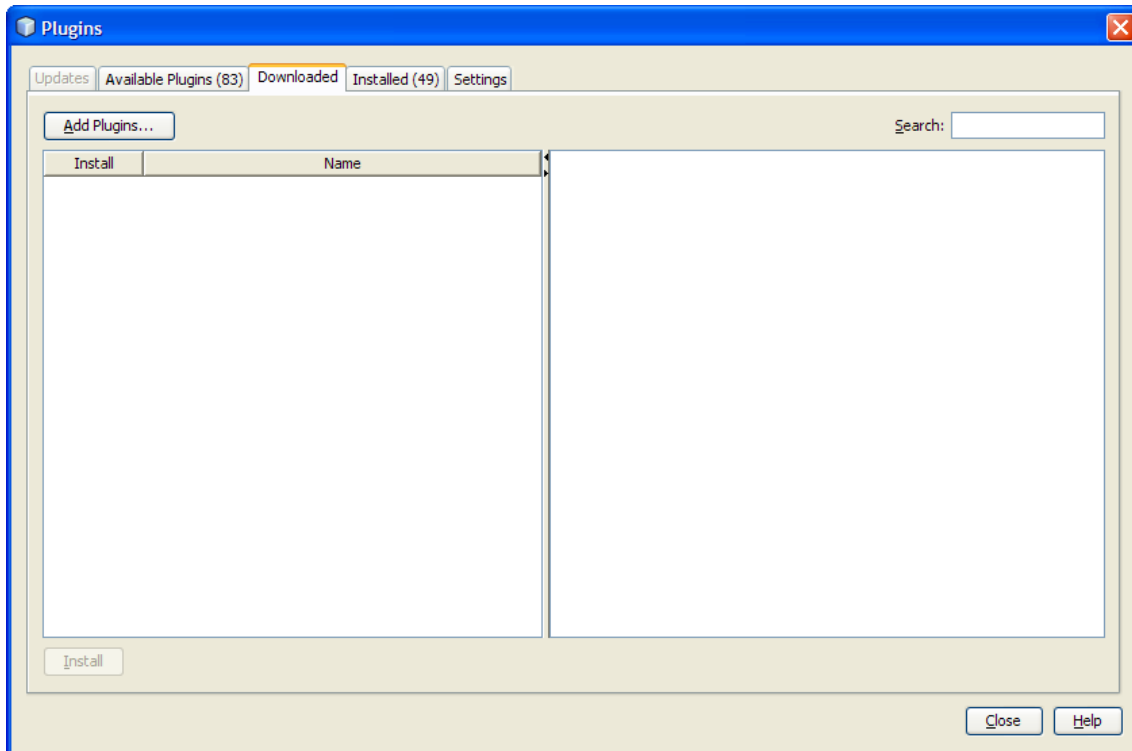


Figure 209 : Downloaded tab from Plugins

Click on the Add Plugins... and choose the path for the component to install. In this case, this path is “C:\iDKDoJa5.1\plugin”. After that, choose the plug-in and click on the Open button.

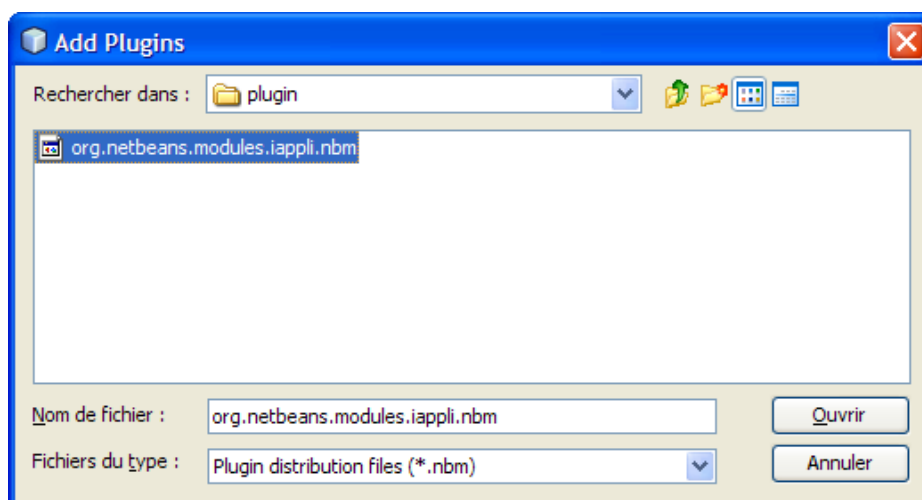


Figure 210 : Plugin choice

Verify that the check box is checked and click on the Install button. After that, click on the Next button.

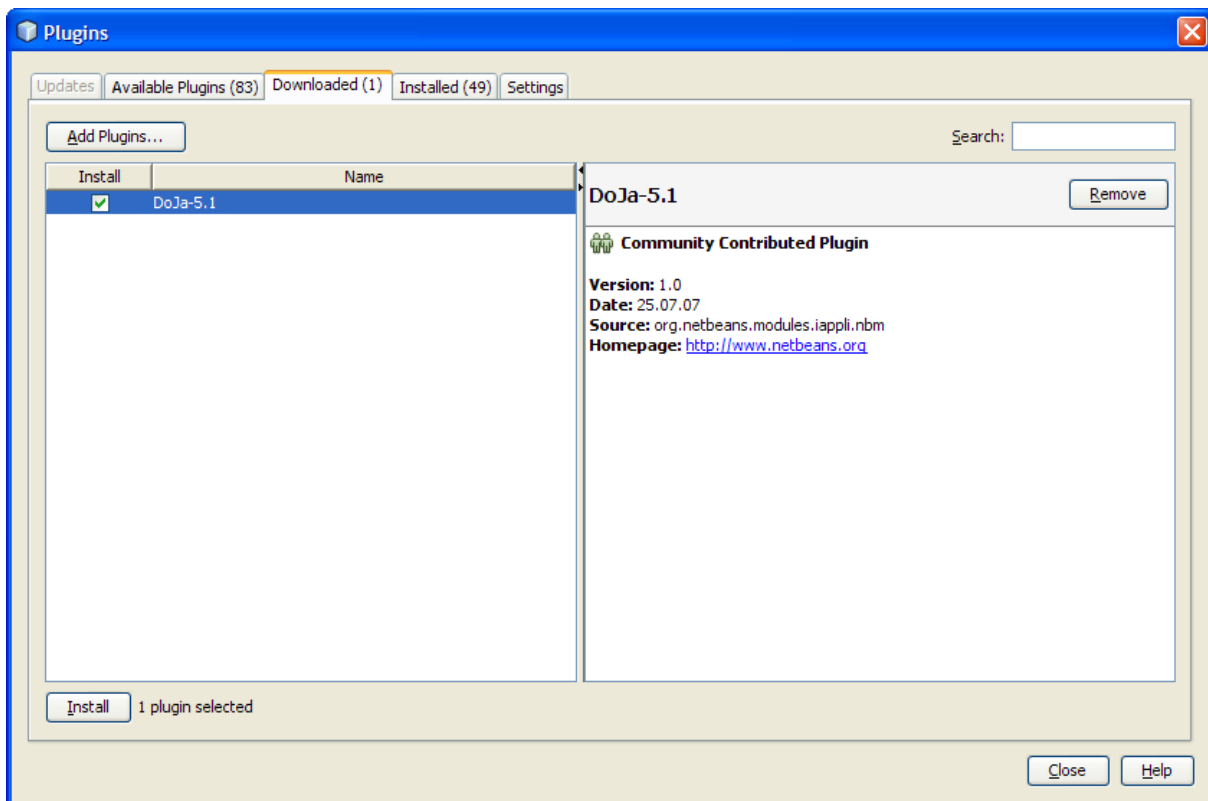


Figure 211 : DoJa 5.1 plugin choice

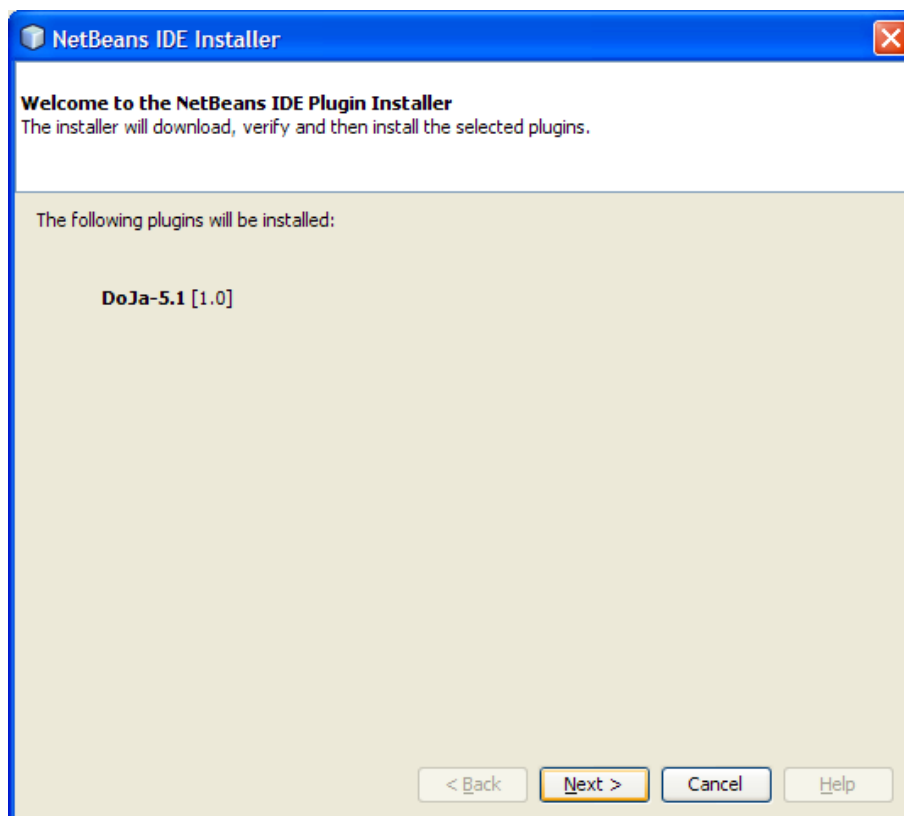


Figure 212 : Installation of the plugin

A warning will be show for warn you that the plug-in is not signed. It is not a problem. Click on the Continue Button.

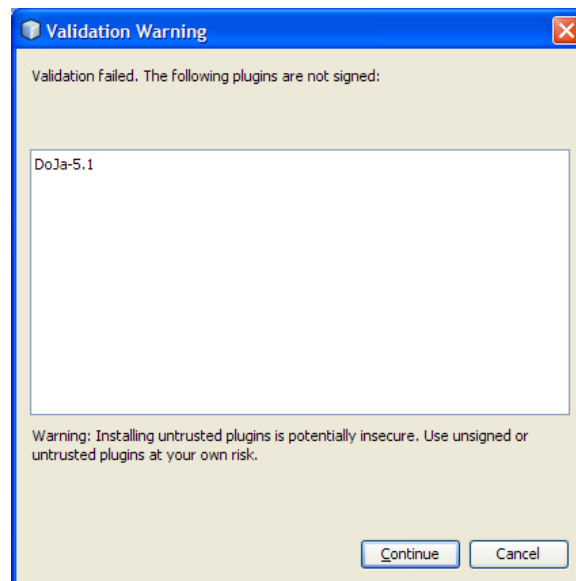


Figure 213 : Signed warning

You have to click on Finish button and Close button. The module is correctly installed. Now we can create new DoJa projects.

G.6. First project in DoJa 5.1 with NetBeans 6.1

Go in the File menu and choose "New Project...". Choose a DoJa 5.1 project. Click on the Next button.

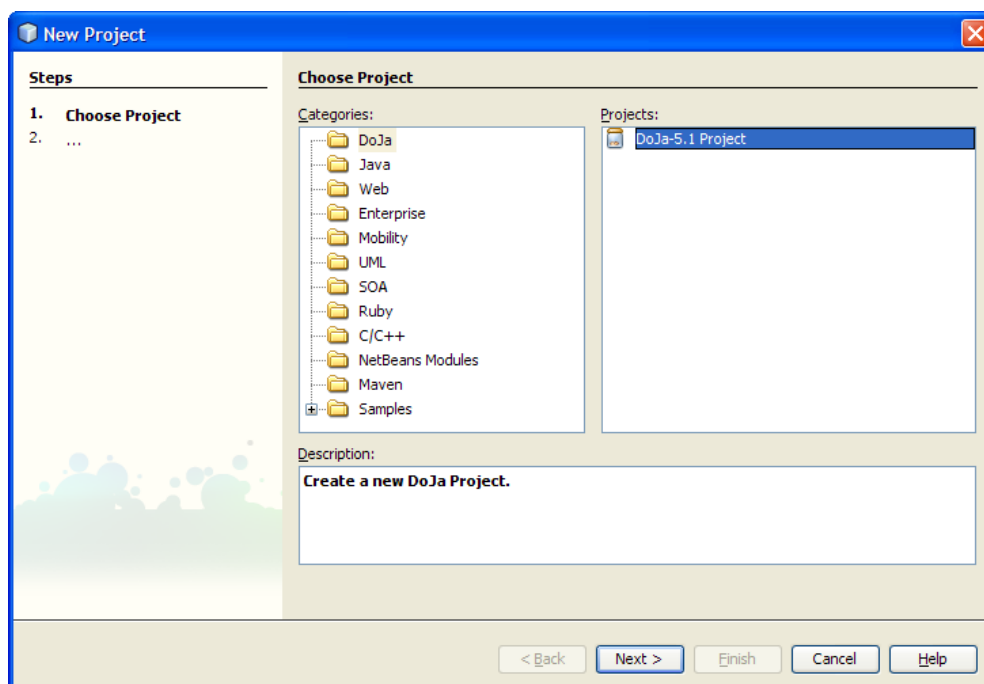


Figure 214 : New DoJa 5.1 project

Choose a path to store the project and a name for your project. In this case, the name of project is “DoJaTest” and stored in My Documents. Click on Finish button.

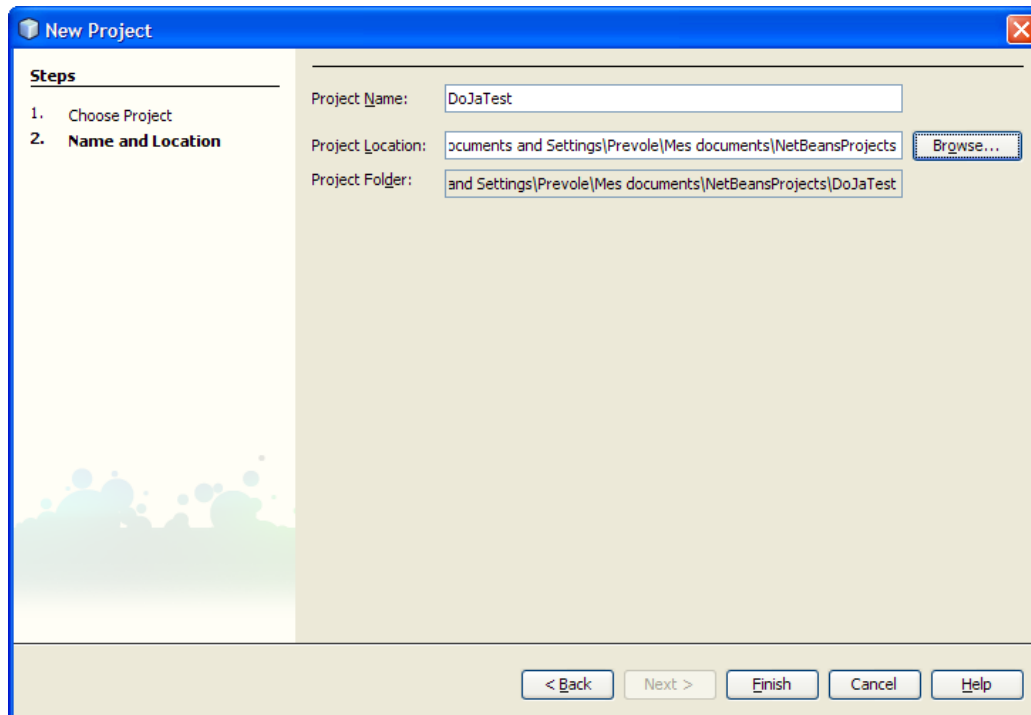


Figure 215 : Project configuration

You can see that the project is created and ready to go. Click on the project in the “Projects View” and choose “New → Java Class...”. Name your class “HelloWorld” and click on Finish button.

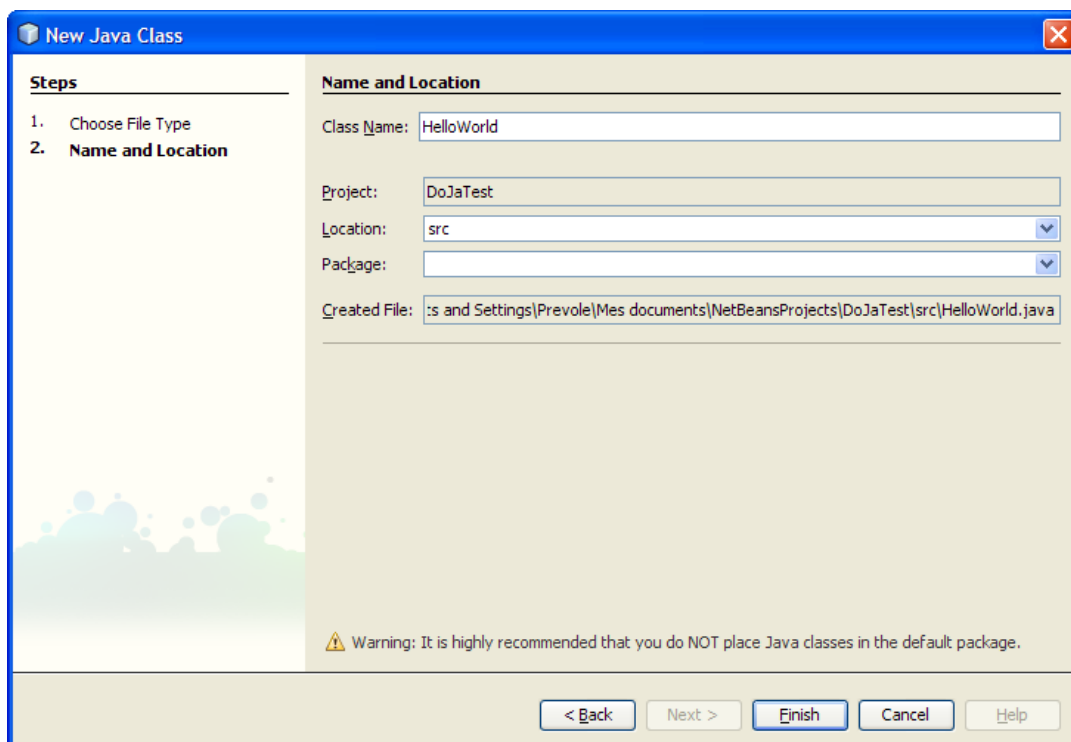


Figure 216 : HelloWorld class creation

Modify the base code by this:

```
import com.nttdocomo.ui.IApplication;

public class HelloWorld extends IApplication {
    public void start() {
        System.out.println("Hello World !");
    }
}
```

Code 24 : Code of the class HelloWorld

Now, try to compile and run the project. You will obtain this error:

```
C:\Documents and Settings\Prevole\Mes
documents\NetBeansProjects\DoJaTest\nbproject\build-impl.xml:34: taskdef
class org.netbeans.modules.iappli.ant.JamEdit cannot be found
BUILD FAILED (total time: 0 seconds)
```

This is the problem that the module is not ready for NetBeans 6.1. We will fix that now. Go to the view “Files” of your DoJa project, find the file “nbproject/doja.properties” and open it.

```
#
#Mon Aug 11 13:35:08 CEST 2008
doja.jar.name=${doja.project.name}.jar
doja.preverify.cmd=${doja.home}/bin/preverify.exe
doja.jpda.address=8000
doja.adfurl=
doja.home=C:\\iDkDoJa5.1
doja.build.classes.dir=${doja.build.dir}/classes
doja.build.classes.excludes=**/*.java,**/*.form
doja.dist.dir=bin
doja.build.dir=build
doja.debug.cmd=${doja.home}/bin/doja_g.exe
doja.jam.name=${doja.project.name}.jam
doja.run.cmd=${doja.home}/bin/doja.exe
doja.ant.classpath=${user.home}/.netbeans/4.1/modules/org.netbeans.modules.iap
doja.classpath=${doja.home}/lib/classes.zip;${doja.home}/lib/doja_classes.zip
doja.src.dir=src
doja.start.type=0
doja.project.name=DoJaTest
doja.debug=false
doja.preverify.output.dir=${doja.build.dir}/output
doja.device.type=device1
```

Figure 217 : Doja properties file

Add this line at the end of the line “doja.classpath” (or the correct path to NetBeans modules):

```
;%{user.home}/.netbeans/6.1/modules/org.netbeans.modules.iappli.jar
```

Save your file.

In addition, the configuration of DoJa applications has to be done to run in the emulator. Select your project in the “Project View” and go to the DoJa 5.1 menu. Choose “Configurations...”

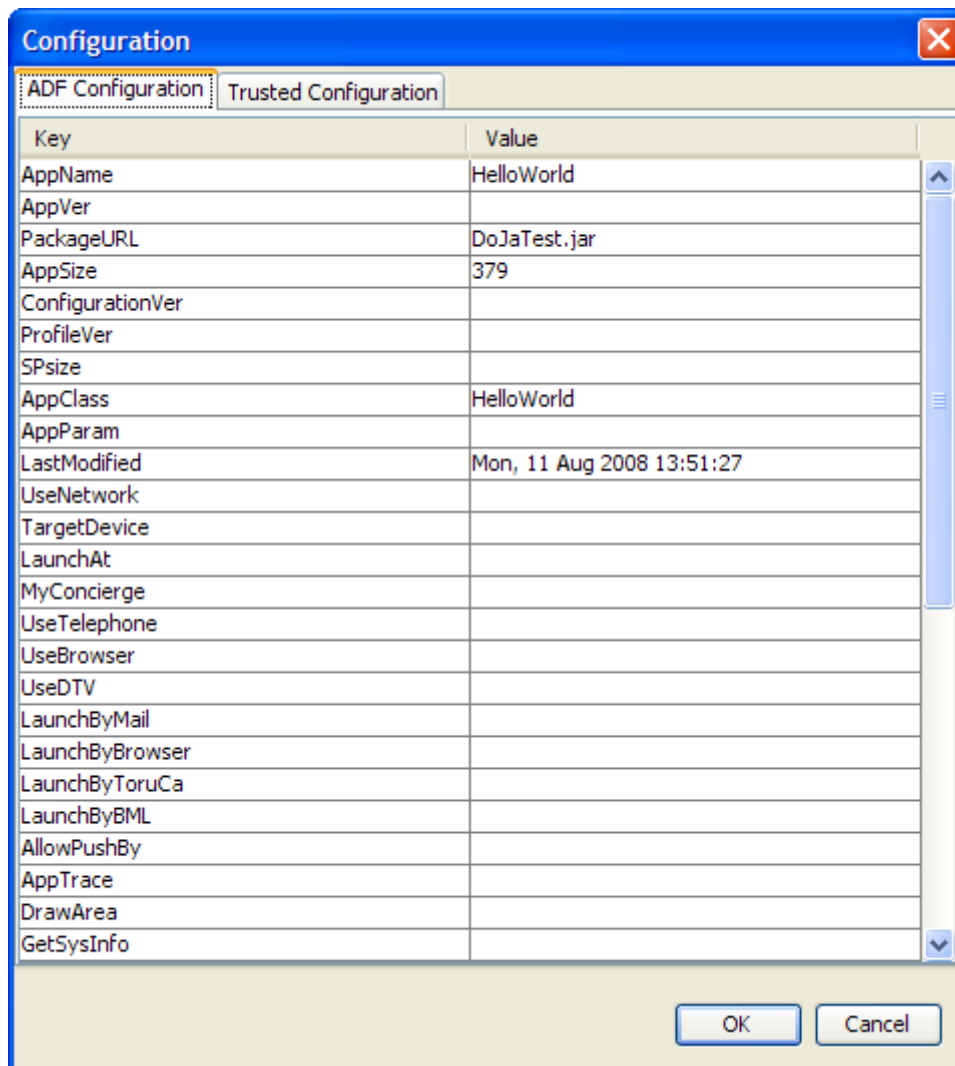


Figure 218 : DoJa 5.1 project configuration (ADF Configuration)

Find the line for AppClass and fill the blank with the class HelloWorld (do not write the extension, only the class name). After that find the AppName and fill the blank with the name of the application. In this case, the name is HelloWorld.

Try to run your project. Now you can view the emulator screen.



Figure 219 : Emulator view from the HelloWorld project

This view is not interesting now because nothing will appear in it. However, you can see the output of “HelloWorld !” in the NetBeans Console.

```
preverify:
jar:
Building zip: C:\Doc
jam:
run:
run-init-if:
run-exec-ex-if:
run-exec-if:
run-macro:
Hello World !
```

Figure 220 : Console output of the HelloWorld project

G.7. Emulator

After that, you know to create a project with NetBeans and use the emulator but sometime, you have to do some functionality like GPS... For this kind of functionality, you have to configure something.

Maybe an exception will be thrown the first time to use this kind of functions. There is not relevant documentation on the Internet. The reason is there is no predefined data for these functionalities. You have to create the data to avoid the exception.

First, you have to know that it is possible to create some “Device Profile”. You can create some devices for some specifics tests in example for resolution purpose.

Open the iAppli Tool for DoJa 5.1 and go to the menu “Edit → Emulator Environment Setup...” and after that go to the tab “Device Configuration”. You can see some configuration that created for test purpose.

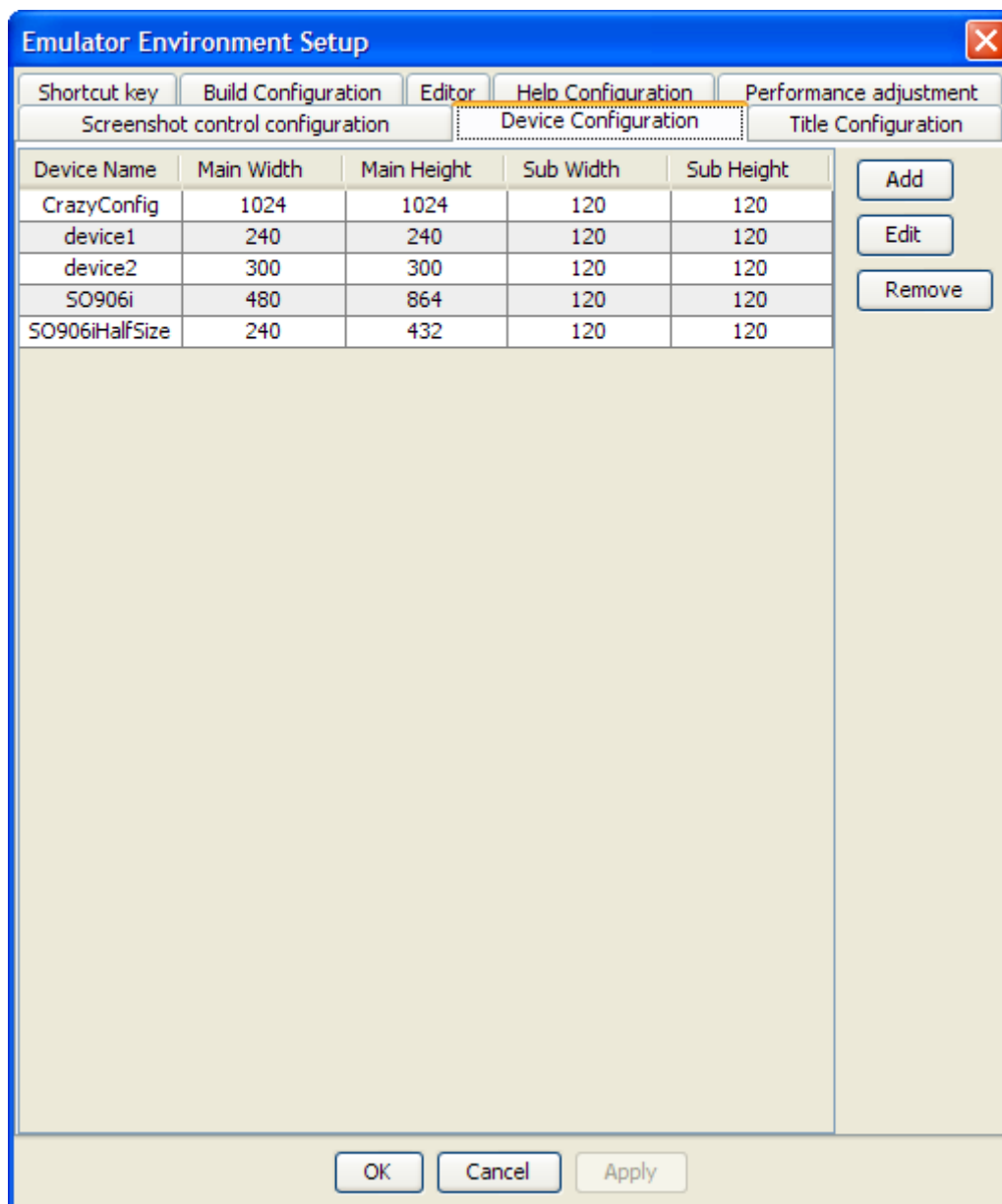


Figure 221 : Device configuration tab

After that, click on the “Add” button to create a new device configuration. Fill in the form. Pay attention that the max value for height and with is about 1024 pixels. Pay also attention to the name that it cannot contains spaces or special character (the name is directly used as a parameter when we run the application in the emulator). Then, click on “Ok” button. The new configuration will appear in list shown before. The most important thing is that you have to click on “Apply” or “Ok” button to validate your creation, if you do not do that, you will lose your creation.

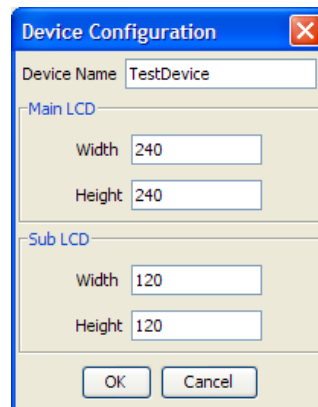


Figure 222 : Device configuration creation

Now, you are able to configure special data. Go to the “Device” menu and choose the configuration we just create.

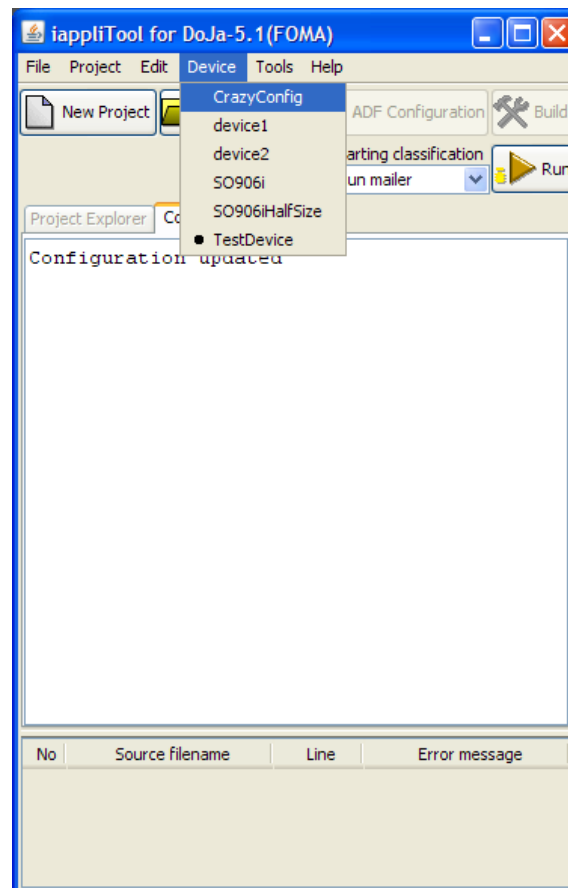


Figure 223 : Device selection

After that, go to “Edit → Native Data Setup...” menu. Now you can edit a lot of information for all the mobile devices. You can add GPS location, phonebooks address...

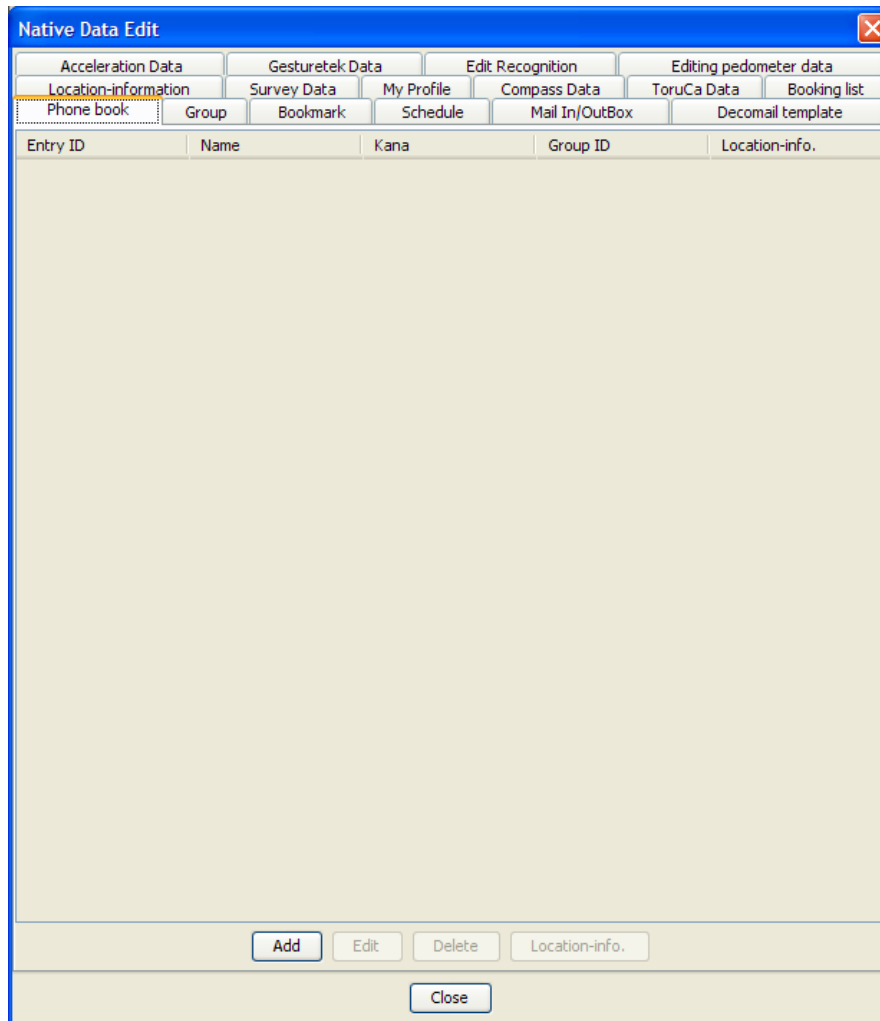


Figure 224 : Native Data Edit screen

After adding information, you can close this screen and run your application. Normally, there is no more exception without any explanation but we have to modify the ADF file to use some functions.

The iAppli have to be in the state of trusted application to use location for example. Go to your project configuration from “DoJa 5.1” menu in NetBeans and add eleven “1” in the blank of TrustedAPID (in “ADF Configuration” tab).

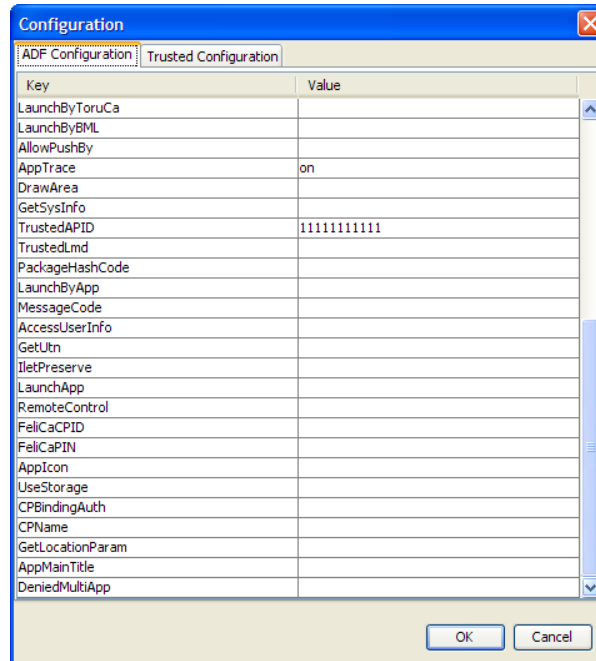


Figure 225 : Trusted application

Go to the “Trusted Configuration” to enable all functionalities you need. After that, click on “Ok” button. You can now run your application with the required functions. However, take care of the fact that a trusted iAppli must to be validated by NTT Docomo (or the provider of iMode service). It is not possible to run an iAppli on a real mobile phone without a true TrustedAPID and this ID is only available after an agreement with NTT Docomo.

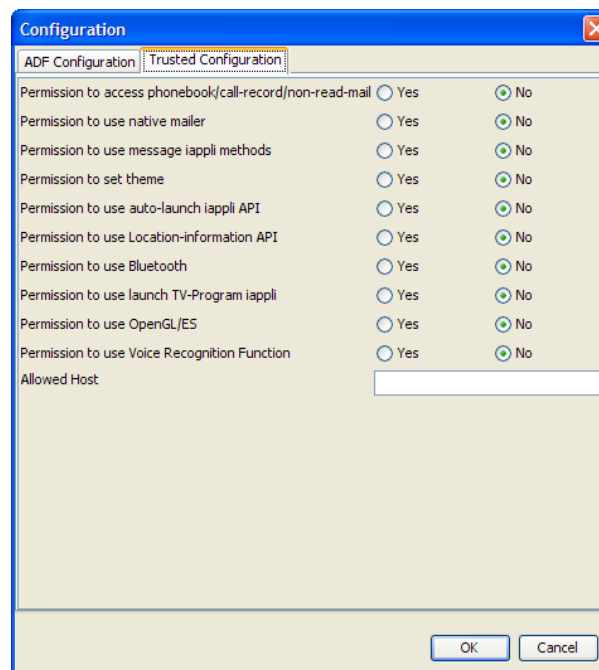


Figure 226 : Trusted configuration

You can run your application from NetBeans as usually but you can also change the device for the “DoJa 5.1” menu. Select the “Run Setup...” option from the menu and change the device you want. You can also change the mode to run the application.

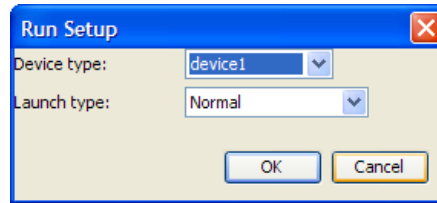


Figure 227 : Run Setup configuration

G.8. Screen

All real phones have different resolution. In that case, this is possible to develop for a specific resolution like the one discussed before in this appendix. However, actually, this is not possible to deploy an application correct for all resolution in the same time.

For example, if nothing special is done, the project application on a mobile phone comes in 240x240. However, if the configuration is done correctly for the ADF file, it is possible to manage a bigger resolution. You can prepare your code to a best rendering on devices but at least you have to modify the ADF file to match the phone you want deploy.

The value to modify in the ADF file is “DrawArea”. You can set a value like width cross height. So for example: 480x864.

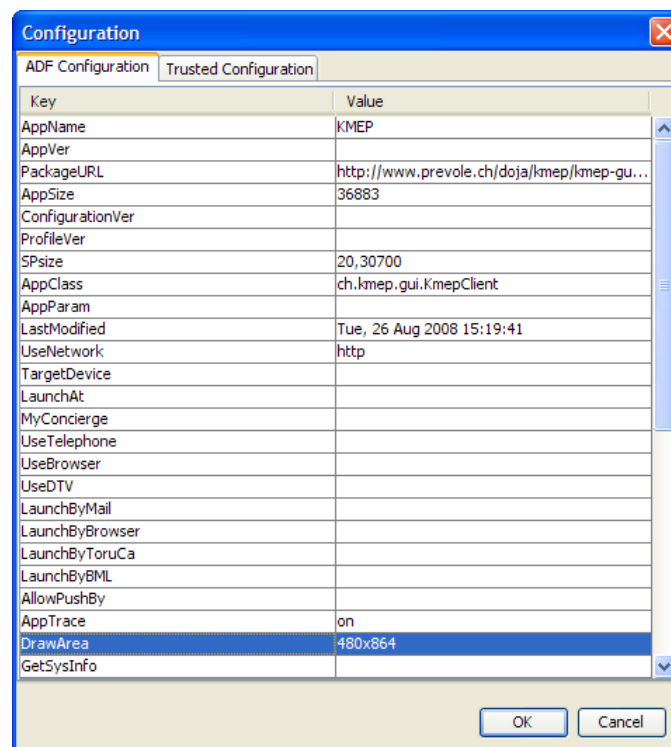


Figure 228 : Screen resolution configuration

If your code is good, you have just to change this parameter to deploy on specific phones without redo some code.

G.9. Conclusion

Now you are ready to develop some iAppli with DoJa 5.1 and NetBeans 6.1 / 6.5. The development environment is ready to go. However, you have to pay attention that some problems can appear in a further utilization of the environment. This document is just a base to begin with DoJa 5.1 and NetBeans 6.1 / 6.5. It offers some easy way to use the emulator too.

heig-vd

Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

K-MEP

Code Standards

Laurent Prévost

The logo for RITSUMEIKAN, featuring a large white letter 'R' on the left and the word 'RITSUMEIKAN' in a smaller, white, sans-serif font to its right, all set against a dark red rectangular background.

R RITSUMEIKAN

Table of Content

H.1. INTRODUCTION	437
H.2. NATURAL LANGUAGE	437
H.3. CODE STANDARDS	437
H.3.1. Documentation	437
H.3.2. Class naming	437
H.3.3. Interface naming	437
H.3.4. Enumeration naming	437
H.3.5. Variables naming	437
H.3.6. Constant naming	438
H.3.7. Method naming	438
H.3.8. Accessors naming	438
H.3.9. Annotations	438
H.3.10. Comments	438
H.3.11. File headers format	439
H.3.12. Other formats	439
H.3.13. Code presentation	439
H.3.14. Remarks	439
H.4. CONCLUSION	440

Table of Codes

CODE 25 : CLASS HEADER JAVADOC	439
CODE 26 : CLASS HEADER JAVADOC SAMPLE	439
CODE 27 : LONG CODE PRESENTATION	439

H.1. Introduction

This appendix brings the standards to write the Java code during the project. It explains how the code must be written for the comments, the conditions, the documentation inside the comments, the indentation, the naming and so on.

Usual habits of Java are used to code the application with some specificity. This appendix provides these specificities.

H.2. Natural language

The natural language used for the project is English. This language is used for the code and the comments.

H.3. Code standards

H.3.1. Documentation

The JavaDoc format is used to prepare the technical documentation for the code. In this situation, the comments and other code documentation inside the code must be in the correct JavaDoc format. This point is generally checked by the IDE.

H.3.2. Class naming

The Camel notation will be used for the class naming. The names must be relevant as much as possible.

H.3.3. Interface naming

Like the class naming, the Camel notation will be used too for the naming of interfaces. In addition, each interface has to begin with the prefix "I". This prefix indicates directly that it is an interface. Sample: "Buildable" becomes "IBuildable".

H.3.4. Enumeration naming

The same rules as Interfaces are available for the enumeration. In place of the prefix "I", we will use the prefix "E". Sample: "GameMode" becomes "EGameMode". This prefix allows seeing that it is an enumeration just based on the name.

H.3.5. Variables naming

There are no specific rules for the naming of variables but some principles to respect. One important point is that the variables' names must mean something relevant to the context of usage. In general, the names begin with a lower letter and the rest of the name follows the Camel notation.

There are some exceptions for the naming. Local variables (loop variables ...) could be named like “i”, “j” or “k”. In the community, these notations are very frequent and are a standard de facto.

H.3.6. Constant naming

The constant naming follows some of the variables rules. The principal difference is that the constants must to be only in capital letters and with underscore to separate the words in the constant name. This point is important to differentiate quickly a constant from a variable.

H.3.7. Method naming

In general, the names of a method will begin by a verb followed by the rest of the name. The verb means the action did by the method. The Camel notation will be used too. The names must to be relevant as usual.

H.3.8. Accessors naming

The names of accessors must to begin with the prefixes “get” or “set”. The rest of the name represents, in general, the name of the attribute to access. The names after the prefix will be in Camel notation. Some exceptions are necessary depending of some annotation and technologies (JAXB).

H.3.9. Annotations

In General, the annotations will be placed one line before the line of code that depends of the annotation.

H.3.10. Comments

Use many comments with a maximum of relevance. It will be good to comment each part of code to describe the functionality and the goal of the part code. Avoid using this kind of comments “if this value is bigger than this another value do that or do that”. This kind of comments means to rewrite the code in natural language and this is not the goal that the documentation will reach.

A good habit is to comment step by step. Try to avoid letting blank of comments some parts of code during the development. It is important to comment all the code during the project and not just at the end of the project. Some code will not be touch during the project and will be unclear at the end without some comments.

Pay attention to the copy/paste of the comments and to correct the comments when it is necessary (after modifications in the code or something else).

Only comment the top class/interface for the method and other documentation that produces copy/paste situation (like method in interfaces implemented in many classes).

H.3.11. File headers format

This is the class header format used in the project:

```
/**
 * <description of the class, interface, and enumeration>
 * @author <lastname, firstname>, <organization>, <country>
 * @date <month> <year> (of creation)
 * @version <major>.<minor>
 */
```

Code 25 : Class header javadoc

Sample:

```
/**
 * This service allows managing the kmep entities...
 * @author Laurent Prévost, HEIG-VD, Switzerland
 * @date August 2008
 * @version 1.0
 */
```

Code 26 : Class header javadoc sample

H.3.12. Other formats

For the method formats, the usual JavaDoc format is used as we already said before. Use the HTML tags when necessary to format the documentation.

H.3.13. Code presentation

Follow a maximum size of 80 characters for a line of code. If the line is bigger than this length, add a return carriage and one indentation in addition of the previous indentation. The size of indentation is of two spaces and composed by spaces (no tabulation!).

Depending of the code and the situation, we will format the code as best as possible. For example, a long call method with long arguments could be by one argument per line like the sample above.

```
longCallMethode (
    arg1,
    arg2,
    arg3,
    ...
);
```

Code 27 : Long code presentation

H.3.14. Remarks

All these rules are only applicable for the Java code and only for the code produce in the project. These rules are not applicable for the code received from other people not directly in the project.

H.4. Conclusion

These rules are important to keep a good quality of code. They offer some restriction but also a lot of liberty for the presentation. The goal is to have a great code to read when it is necessary. Another great point is if the rules are followed, it is possible to print the code when it can be useful (sometimes).

It is important to follow some rules during a long project to keep the coherence in the code and the documentation.

heig-vd

Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

K-MEP

Tools

Laurent Prévost

The logo for RITSUMEIKAN, featuring a large white letter 'R' on a dark red background, followed by the word 'RITSUMEIKAN' in white capital letters.

R RITSUMEIKAN

Table of Content

I.1. INTRODUCTION	445
I.2. COMMUNICATION	445
I.2.1. Instant messaging	445
I.2.2. Phone	445
I.2.3. E-mail	445
I.3. STORAGE	445
I.3.1. SVN	445
I.3.2. Backup	446
I.4. INTEGRATED DEVELOPMENT ENVIRONMENT	446
I.4.1.1. NetBeans	446
I.4.1.2. Maven	446
I.4.1.3. Hudson	446
I.4.1.4. Glassfish	446
I.4.1.5. MySQL	446
I.4.1.6. ToGether	447
I.4.1.7. XMLSpy	447
I.5. WORKING DIARY	447
I.6. TRAC	447
I.7. OTHERS	447
I.7.1. iMode HTML Simulator II	447
I.8. TOOLS AND VERSIONS	448
I.9. CONCLUSION	449

Table of Tables

TABLE 30 : TOOLS UNDER WINDOWS XP.....	448
TABLE 31 : TOOLS UNDER MAC OS X	449

I.1. Introduction

This appendix offer an overview of the tools used during the project. The tools cover the communication, the development, the design and so on.

At the end of the appendix, there is the list of tools with the version used for the project. Only the local development and known tools are listed. Some tools are away from our control. In addition, the relevant tools are listed.

I.2. Communication

Due to the particularity of the project, there is real need of communication tools to keep the contact with people in Switzerland.

I.2.1. Instant messaging

Some instant messaging clients are used in this project to get the ability to communicate and send documents easily. They allow a real time interaction between us. There is also a need to know when the people are available or due to jetlag.

I.2.2. Phone

The VoIP is also used to keep the oral contact between the actors of the project. For some situation, it will be better to use the oral conversation rather than the messages conversation. It is also useful to do the point of situation.

I.2.3. E-mail

If the two previous method of communication are not enough, there is the e-mail possibility to communicate. It is also useful to get a backlog of the discussion and to get a reminder for some purpose.

I.3. Storage

All the documents and the code are stored on the SVN based in Switzerland. In addition, this is the responsibility to everybody to keep sufficient backup of his or her work.

I.3.1. SVN

To use the Server SVN in Switzerland, there are some different ways. A client SVN can be used and/or Tortoise SVN (windows only), Smart SVN is an alternative multi-platform. The SVN plugins in NetBeans can be used too. This last point is particularly great to manage the code source directly in the IDE. For the documentation, the normal tools are sufficient.

I.3.2. Backup

It is very useful to do some backup in addition of the previous method of storage for the code and documentation. Some parts of the projects are not directly added to the SVN for some great reasons. In this situation, it is better to have a good backup solution. This point is of the responsibility of everybody.

I.4. Integrated Development Environment

I.4.1.1. NetBeans

The IDE NetBeans is used for this project with some additional features like GlassFish (the application server) and others plug-ins like maven2IDE (for the compilation, deployment ...). This tool is great for the features like generating code, auto format, and syntax checking and so on. The list of feature is too long to be written here.

I.4.1.2. Maven

This plug-in and tool is used to compile and prepare dependencies. It is also installed on the Swiss server to compile and prepare the project directly on the server. It allows deploying directly the application to GlassFish. The latest plug-in under windows is not stable and provides some crashes (freeze) during the compile or deploys process. It is recommended to do not update the plug-in.

I.4.1.3. Hudson

Hudson is a server side tool that allows to run some maven (not only) jobs periodically or manually to compile/deploys... This particularly useful to compile and deploy the application on the Swiss server. This is a tool for remote actions.

I.4.1.4. Glassfish

GlassFish is the application server included in NetBeans (depending on the version of NetBeans) and used to run the application in the project. There is also a GlassFish server on the Swiss server.

I.4.1.5. MySQL

This tool is not directly included in NetBeans and has to be installed manually. However, for the project, it is needed to run the persistence layer. A MySQL JDBC connector is also needed to allow the connection between GlassFish and MySQL. This connector has to be installed in GlassFish library (/lib of GlassFish home folder). Additional tools to manage the MySQL Server can be useful to have.

I.4.1.6. Together

Together is a fork of Eclipse done by Borland and is very useful for the design purpose. To do the UML diagrams, this tool is really better than NetBeans. In this project, this tool is used for the UML purpose but it allows to code in Java like Eclipse and NetBeans. This tool is not free but the HEIG-VD has some professors that have a license for the students.

I.4.1.7. XMLSpy

This tool is only oriented for the development of XML and XML Schema documents. It allows a better approach than NetBeans. The documents remain XML or XSD. It allows viewing graphical representation of the XML Schema. This is very great to debug purpose an easy way of visualization. This is a professional non-free tool. A trial version is available.

I.5. Working diary

A working diary is available under the Trac platform. This diary allows following the progression of the project day by day with some technical details and it allows planning some next activities (to-do list).

I.6. Trac

Trac is platform to manage a project with ticket system, wiki functionality for documentation, SVN code viewing and the time line progression. Other features are available. In this project, it allows to store and view data between each project participant (professors, assistants, coders).

I.7. Others

I.7.1. iMode HTML Simulator II

This tool is particularly useful to check and run the iMode application before to try on the mobile phone. This is a simulator of mobile phone that can show the iMode application and run the same functionalities than a real mobile phone. However, we have to test also in real phones to be sure the application runs correctly for the web user interface.

I.8. Tools and versions

The Table 30 shows some tools with their version for a windows installation and a remark if necessary.

Tool	Version	Remark
NetBeans	6.5 beta	There are many bugs (beta version) but not critic to develop in this version. For stable version, use the 6.1 (less functionalities).
ToGether	2006	HEIG School License
XMLSpy	2008	Evaluation version
MySQL	5.0.27	Bundle pack with EasyPHP (very useful for phpMyAdmin and Apache)
MySQL JDBC Connector	5.1.5	Recent ones cause troubles with GlassFish
SVN	1.4.6	More recent is possible
Subversion	1.5.2	More recent is possible
MavenIDE	3.1.1	Recent ones cause trouble in NetBeans
Java EE SDK	1.4	More recent is possible but pay attention with the real version on a production server. In this project, the real version to develop is the 1.5.
GlassFish	2ur2	The version 3 is available but on the server, there is the same version than on local.
Maven	2.0.9	This version comes from Apache foundation.

Table 30 : Tools under Windows XP

The Table 31 shows some tools with their version for a windows installation and a remark if necessary.

Tool	Version	Remark
NetBeans	6.5	This is the principal tool to develop KMEP.
ToGether	2008	HEIG School License
MySQL	5.0.67	Bundle pack with XAMP (very useful for phpMyAdmin and Apache)
MySQL JDBC Connector	5.1.5	Recent ones cause troubles with GlassFish
SVN	1.4.4	More recent is possible
Smart SVN	4.0.8	Smart SVN offer a free version very useful.
MavenIDE	4.0.5	Recent ones cause trouble in NetBeans
Java EE SDK	1.5	More recent is possible but pay attention with the real version on a production server. In this project, the real version to develop is the 1.5.
GlassFish	2ur2	The version 3 is available but on the server, there is the same version than on local.
Maven	2.0.9	This version comes from Apache Foundation.

Table 31 : Tools under Mac OS X

I.9. Conclusion

All the tools are very necessary due to this kind of projects. It is also a good habit to use many complete tools to improve the productivity and the efficiency of the development.

heig-vd

Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

K-MEP

Other

Laurent Prévost

The logo for RITSUMEIKAN, featuring a large white letter 'R' on the left and the word 'RITSUMEIKAN' in a smaller, white, uppercase sans-serif font to its right, all set against a dark red rectangular background.

R RITSUMEIKAN

Bibliography

1. Björk, S. et al. (2002) Designing Ubiquitous Computing Games - A Report from a Workshop Exploring Ubiquitous Computing Entertainment, Journal of Personal and Ubiquitous Computing, Volume 6, 6th issue: Special issue on Ubiquitous Gaming.

This document describes a Research Atelier about how to apply ubiquitous computing to fun entertainment. During five days, developers have imagined and prototyped some games. Two days were dedicated to game scenarios and three days to develop a prototype. The document describes three different games.

2. S. Lundgreen & S. Björk, Game Mechanics: Describing Computer-Augmented Games in Terms of Interaction,

In this document, we can find some definitions like “pervasive game” (very short description) and others. It brings a point of view about game mechanism that includes “pervasive game” as game mechanism.

3. F. Girardin, Pervasive Game Development Today, March 2005
<http://www.girardin.org/fabien/catchbob/pervasive/>
last visit: 12.2008

This document show a concrete game that use positioning information with WiFi (802.11b) technology. The game uses TabletPC or PocketPC. This game was developed at EPFL in Lausanne (CH).

4. P. Lankoski, S. Heliö, J. Nummela, J. Lahti, F. Mäyrä & L. Ermi, A Case Study in Pervasive Game Design: The Songs of North, NordiCHI '04, October 23-27, 2004 Tampere, Finland
<http://portal.acm.org/citation.cfm?doid=1028014.1028083>

The Songs of North is a pervasive game that illustrates this kind of game. The document offers an overview of game and discuss about implementation and game design. It also talks about persistent world.

5. L. Ermi & F. Mäyrä, Challenges for pervasive mobile game design: examining players' emotional responses, Valencia, Spain, 2005
<http://portal.acm.org/citation.cfm?doid=1178477.1178554>

Related to the precedent reference, this article is a study on feedback of The Song of North game. It shows some aspects like Joy, Anger and Fear. It is a good addition to previous article.

6. I. Smith, S. Consolvo & A. Lamarca, The Drop: pragmatic problems in the design of a compelling, pervasive game, ACM Computers in Entertainment, Vol. 3, No. 3, July 2005, Article 4C

<http://portal.acm.org/citation.cfm?doid=1077246.1077259>

This is another pervasive game called “The Drop”. The document describes some kind of problems that game encountered in game design. Game is built on a “capture the flag” principle. Players are organized into two groups. Geo-localization is used in this game.

7. J. Sinclair, P. Hingston & M. Masek, Considerations for the design of exergames, GRAPHITE 2007, Perth, Western Australia, December 1–4, 2007
<http://portal.acm.org/citation.cfm?doid=1321261.1321313>

This article gives to us some explanations about obesity. After that, we can discover some commercial systems and how to promote exercise with exergaming. The document shows some ways to follow when we are making an exergaming like success factors, attractiveness and effectiveness.

8. C. Magerkurth, A. D. Cheok, R. L. Mandryk & T. Nilsen, Pervasive Games: Bringing Computer Entertainment Back to the Real World, ACM Computers in Entertainment, Vol. 3, No. 3, July 2005. Article 4A
<http://portal.acm.org/citation.cfm?doid=1077246.1077257>

This document covers many different pervasive games. It talks about game board with augmented reality. We can find some interesting descriptions about localization technologies (GPS, GSM, WiFi...). In summary, this article covers general description of pervasive games.

9. I. Lindt, J. Ohlenburg, U. Pankoke-Babatz & S. Ghellal, A Report on the Crossmedia Game Epidemic Menace, ACM Computers in Entertainment, Vol. 5. No. 1, Article 8. Publication Date: April 2007.
<http://portal.acm.org/citation.cfm?doid=1236224.1236237>
<http://iperg.fit.fraunhofer.de/>

This document describes “Epidemic Menace” a pervasive game. Many components are used in this game and are described in this article. Player feedbacks are analyzed and shown how players like or dislike the game. It is a great overview of a concrete application of a pervasive game but due to numerous of systems used for this game. It could be especially difficult to develop this kind of game with reasonable equipment.

10. B. Joffe, Mogi – Location and presence in a pervasive game, Ubicomp 2005, September, Newt Games S.A.
<http://www.mogimogi.com/mogi.php?language=en>

This presentation shows Mogi a pervasive game based in Japan and developed by a French society. The game is currently in commercial usage. All aspects of this game are shown like positioning, community, game principles, used technologies, players’ feedback... The game is especially designed for the AU by KDDI provider.

11. Omer Rashid, Ian Mullins, Paul Coulton & Reuben Edwards, *Extending Cyberspace: Location Based Games Using Cellular Phones*, ACM Computers in Entertainment, Vol. 4, No. 1, January 2006. Article 3C.

This document provides a very large overview of a lot of localization based games. This is a great view to know what kind of games exists.

12. S. Matyas, C. Matyas, N. Crossley, S. Seypt, M. Kamata & H. Mitarai, *CityExplorer*, <http://www.kinf.wiai.uni-bamberg.de/cityexplorer>, 2008, last visit: 08.2008.

This website is a game with geo-localization services. It is based on the "Caracassone" game board. Actually, the game offers two versions: one in Bamberg (Germany) and one in Shonandai (Japan). In this situation, two version of the application exists. The Japanese version is developed for NTT Docomo phones, use iMode webpages, and DoJa application.

13. C. Schlieder, P. Kiefer & S. Matyas, *Geogames*, <http://www.kinf.wiai.uni-bamberg.de/geogames/index.php?site=6>, 2006-2008, last visit: 08.2008.

This website offers a view of the geo-games. Games designed around the geo-localization technologies. The website is in German but the geo-games description is in English. This page offers an overview of the evolution of research about geo-games.

14. <http://sites.google.com/site/mobiledevlab> and <http://groups.google.com/group/mobiledevlab?hl=en>, 2008, last visit: 11.2008.

This is a Google group about mobile phone development. There is a lot of information related to the mobile development in Japan. Some topics are about the DoJa development.

15. Takehiro Kanno, Wenxi Chen & Masumi Kitazawa, *Development of i-appli for Women's Health Information Management System*, Seventh International Conference on Computer and Information Technology, 2007.

This document describes a medical application developed for a specific device for NTT Docomo (FOMA900). This is a good example for an iMode development. Some explanations offer a great view of the architecture of the application and network communication.

16. Y.Kogure, H.Matsuoka, Y.Kinouchi & M.Akutagawa, *The Development of a Remote Patient Monitoring System using Java-enabled Mobile Phones*, Engineering in Medicine and Biology 27th Annual Conference, 2005.

This is another example of Java development on NTT Docomo mobile phone. This development is early than the previous one. It is also a medical application. They are some explanation of the limitation of mobiles phones (memory for the application, memory for data).

17. Akira Sasaki, Kazuaki Yamauchi, Wenxi Chen, Michael Cohen, Darning Wei & Zixue Cheng, *Innovative Mobile Phone Services Based on Next Generation Infrastructure in Japan - A Survey*, Seventh International Conference on Computer and Information Technology, 2007.

This document describes the actual services in the market of mobile phones and the future of these services and new services. A lot of information about the state of the actual market (provider, customer ...) gives a great overview.

18. NewtGames, *Mogi – item hunt*, 2003 [<http://www.mogimogi.com>].
Last visit: 08.2008

This game is specially designed for AU by KDDI and used Geo-Localization system. Only some terminals are supported (listed on the web site). Some explanations about the costs are provided. This is interesting to see that a choice was done about the provider.

- 19 Brent Ellison, Gamasutra, Defining Dialogue System,
http://www.gamasutra.com/view/feature/3719/defining_dialogue_systems.php?page=1
Last visit: 08.2008

This article provides a detailed description of “how to create a dialog system for a game”. Some ways to build a dialog system are explored with the advantages and inconvenients.

20. J2ME Polish, Mobile Phone Development Tool Suite,
<http://www.j2mepolish.org>
Last visit: 08.2008

J2ME Polish is a suite of tools to help into the development of mobile phone solution. This solution provides also the development for DoJa but oriented in DoJa for the European market.

Glossary

– A –

ADF Application Descriptor File. This a file that describes the application deployed on the mobile phone. For NTT Docomo phones, the file extension is “.jam” and includes the data to configure the application (application name, application size...).

– C –

CLDC Connected Limited Device Configuration. The Connected Limited Device Configuration is a specification of a framework for Java Mobile Edition applications targeted at devices with very limited resources such as pagers and mobile phones.

CSS Cascaded Style Sheet. This a mechanism to create styles for HTML pages to easily creating great rendering with a lot of reusability of code.

– D –

DoJa Docomo Java is the API to develop in Java for the NTT Docomo phones. It is very similar to the MIDP profile from Sun.

– E –

EAR Enterprise ARchive. This is a format like JAR specially developed to pack more than one module into only one archive in Java EE (See J2EE). It allows to be deployed on Application Servers.

EJB Enterprise Java Bean. This is abstract mechanism to allow creating rich application with a database support. There are no needs about the knowledge of the database for standard usage.

Entity Bean An Entity Bean corresponds to a business object in a persistent storage system. With EJB, the persistent system is a database.

– F –

FOMA Freedom of Mobile Multimedia Access. This technology is based on the W-CDMA and is used by NTT Docomo since 2001 in Japan. This is a 3G (third generation) technology for the mobile phone communication.

- G -

- GPRS** General Packet Radio Service. It is an evolution of the GSM technology. It is called 2.5G (G for generation) because this is just an evolution of the previous technology.
- GSM** Global System for Mobile Communications. This is the most used mobile phone network technology in the European countries.

- H -

- HEIG-VD** Haute Ecole d'Ingénierie et de Gestion du canton de Vaud. Western University of Applied Sciences and Management.
- HTML** HyperText Markup Language. This is the rendering language for documents in the Internet (principally). This language is approved by the W3.
- HTTP** HyperText Transfer Protocol. This is the well-known application protocol used on Internet. It allows the communication between browsers and web servers for example.

- I -

- iAppli** iAppli are the applications developed in DoJa for NTT Docomo phones. This is a commercial name.
- IDE** Integrated Development Environment. Actually, there are a lot of them like NetBeans, Visual Studio, Eclipse, ToGether... These tools allow an easy project management for the code part and many utilities to do the code.
- IICT** Institut des Technologies de l'Information et des Communications. Institute for Information and Communication Technologies. This is the institute which develop the inTrack platform and where the K-MEP is driven.
- iHTML** iMode HyperText Markup Language. This is the version of HTML from NTT Docomo for their mobile phones browser. The language defines a subset of normal tags.
- inTrack** inTrack is the localization platform used in association with MEP. It is also called inTrack middleware. This platform is developed in the institute of IICT at the HEIG-VD by professor Liechti and his staff.

iXHTML iMode eXtended HyperText Markup Language. This is an evolution of the iHTML based on XHTML. This is a subset of the main language approved by the W3.

– J –

J2EE Java 2 Enterprise Edition. This is the Java solution for the development of enterprise application with a lot of modules and high degree of abstraction.

JAR Java ARchive. Based on ZIP file, this format allows distributing many relative classes into a single file.

JavaScript JavaScript is client side (not only) scripting language to allow better interaction with the user and web pages. Originally, the name was LiteScript and was developed by Netscape Enterprise. Due to an agreement with Sun, Netscape Enterprise changes the name for JavaScript for marketing purpose only with the similitude to Java name.

JAXB Java Architecture for XML Binding. This is a solution from Sun to allow loading XML files into Java classes and vice-versa.

JNDI Java Naming and Directory Interface. This is a Java technology to allow finding objects in directories. It brings an interface to work with Domain Name System, Lightweight Directory Access Protocol and others directories type.

JPA Java Persistence API. This part is included in EJB technologies to do the persistence of the data in a database for example.

JSF Java Server Faces. This is a Java framework to develop web applications with no constraints about the rendering technology.

JSP Java Server Pages. This is a Java technology to build web pages. This is similar to other server side scripting technologies.

JVM Java Virtual Machine. This is the virtual machine to run Java byte codes. To run a Java application, a target computer must have the JVM installed.

– K –

K-MEP Kyoto Mobile Exergaming Project (Platform is also allowing). This project use MEP to implement a game in the context of the city of Kyoto in Japan.

– M –

Maven	<i>Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information.</i>
Mave IDE	Plugin for NetBeans that enables the support for maven projects.
MEP	Mobile Exergaming Platform. A platform to build exergaming mobile games with geo localization services
MIDP	Mobile Information Device Profile. This is a suite of mobile specifications for the development on mobile devices. It defines the way to access to the peripherals on the device.
MVC	Model View Controller. MVC is a well-known design pattern that allows a separation between task types in an execution flow. The View is in charge of the rendering, the Model of the data storing and processing and the Controller manage the communication between View and Model.

– N –

NPC	Non-Player Character. The player can interact in the game with these entities. The NPCs are controlled by the computer only.
NTT Docomo	The largest mobile phone provider in Japan. Actually (2008), it owns about 55% of the mobile phone market. This the provider used for this project.

– P –

PC	Against the NPC, PCs are Player Characters. It means this is human character of the game.
----	-------------------------------------------------------------------------------------------

– S –

SDK	Software Development Kit. This is a set of tools especially build to create a specific type of application.
Singleton	This is a design pattern allows building one and only one instance of a class.
SVN	Subversion. This is a system to store and manage multi version of file and keep the backlog of the changes.

- T -

- TLD** Tag Library Definition. This is configuration file dedicated to Java and especially JSP to define the HTML tags usable in JSP files. With this definition, this is possible to bind a HTML tag to a Java class that handles the tag.
- TO** Transfer Object. Defines object that can be used to transport data between two layers in an application.

- U -

- UML** Unified Modeling Language. This is a language to build Object elements in a graphical representation. There are many other possibilities with this language to represent graphically some abstract concept.
- URL** Uniform Resource Locator. In the common language, the term of web address is used.

- V -

- VoIP** Voice over IP (Internet Protocol). This the phone calls over the Internet network.

- W -

- W3** The W3 Consortium is the organization that tries to standardize all the languages relatives to the Internet like HTML, XML and so on. See www.w3.org for more details.
- WAR** Web ARchive. That is a file format based on JAR format that contains all the necessary files to run a Java Web Application on server that supports them.
- W-CDMA** Wideband Code Division Multiple Access Evaluation. This technology corresponds to the third generation of the mobile phone network communication technology.
- WUI** Web User Interface. This is an interface that the user can view and use to interact with an application. In this case, this interface is designed for browsers.

- X -

- XHTML** eXtended HyperText Markup language. This language is approved by the W3. This is the evolution of HTML language based on XML.

- XML eXtended Markup Language. The XML is a language based on tags to create other language especially useful to exchange data between different entities based on the same data definitions. It is approved by W3
- XSD XML Schema Definition. This is a recommendation from the W3 for the creation of XML document. The XML Schema is also a XML document. It allows fixing the rules for the correct construction of a XML document.

EOF
