Laboratory for Advanced Software Systems
University of Luxembourg

iCrash :
A Crisis Management Case Study
MESSIR Analysis Document
- v 1.4 -

(*Report type: Default*)

Wednesday 21st December, 2016 - 20:52

Contents

1 Introduction . . . **17**
1.1 Overview . . . 17
1.2 Purpose and recipients of the document . . . 17
1.3 Application Domain . . . 17
1.4 Definitions, acronyms and abbreviations . . . 17
1.5 Document structure . . . 18

2 General Description . . . **21**
2.1 Domain Stakeholders . . . 21
2.1.1 Communication Company . . . 21
2.1.2 Humans . . . 22
2.1.3 Coordinators . . . 22
2.1.4 Hospitals . . . 22
2.1.5 Corrupters . . . 22
2.1.6 Administrator . . . 23
2.1.7 Creator . . . 23
2.1.8 Activator . . . 23
2.2 System's Actors . . . 24
2.3 Use Cases Model . . . 24
2.3.1 Use Cases . . . 25
2.3.2 Use Case Instance(s) . . . 44

3 Environment Model . . . **49**
3.1 Local view 01 . . . 49
3.2 Local view 02 . . . 49
3.3 Local view 03 . . . 49
3.4 Local view 04 . . . 49
3.5 Local view 05 . . . 49
3.6 Local view 09 . . . 49
3.7 Global view 01 . . . 53
3.8 Actors and Interfaces Descriptions . . . 54
3.8.1 `actActivator` Actor . . . 54
3.8.2 `actAdministrator` Actor . . . 55
3.8.3 `actAuthenticated` Actor . . . 55
3.8.4 `actComCompany` Actor . . . 55
3.8.5 `actCoordinator` Actor . . . 56
3.8.6 `actHospital` Actor . . . 56
3.8.7 `actMsrCreator` Actor . . . 57

4 Concept Model . . . **59**

4.1 PrimaryTypes-Classes . . . 59
4.1.1 Local view 01 . . . 59
4.1.2 Local view 02 . . . 59
4.1.3 Local view 03 . . . 59
4.1.4 Local view 04 . . . 59
4.1.5 Global view 01 . . . 59
4.2 PrimaryTypes-Datatypes . . . 59
4.2.1 Local view 06 . . . 59
4.2.2 Local view 08 . . . 63
4.2.3 Global view 01 . . . 63
4.3 SecondaryTypes-Datatypes . . . 64
4.3.1 Local view 01 . . . 64
4.4 Concept Model Types Descriptions . . . 64
4.4.1 Primary types - Class types descriptions . . . 64
4.4.2 Primary types - Datatypes types descriptions . . . 67
4.4.3 Primary types - Association types descriptions . . . 69
4.4.4 Primary types - Aggregation types descriptions . . . 70
4.4.5 Secondary types - Class types descriptions . . . 70
4.4.6 Secondary types - Datatypes types descriptions . . . 70
4.4.7 Secondary types - Association types descriptions . . . 70
4.4.8 Secondary types - Aggregation types descriptions . . . 71
4.4.9 Secondary types - Composition types descriptions . . . 71

5 Operation Model . . . 73
5.1 Environment - Out Interface Operation Scheme for actActivator . . . 73
5.1.1 Operation Model for oeSetClock . . . 73
5.1.2 Operation Model for oeSollicitateCrisisHandling . . . 75
5.2 Environment - Out Interface Operation Scheme for actAdministrator . . . 78
5.2.1 Operation Model for oeAddCoordinator . . . 78
5.2.2 Operation Model for oeAddHospital . . . 82
5.2.3 Operation Model for oeDeleteCoordinator . . . 84
5.2.4 Operation Model for oeDeleteHospital . . . 87
5.3 Environment - Out Interface Operation Scheme for actAuthenticated . . . 88
5.3.1 Operation Model for oeLogin . . . 88
5.3.2 Operation Model for oeLogout . . . 91
5.4 Environment - Out Interface Operation Scheme for actComCompany . . . 94
5.4.1 Operation Model for oeAlert . . . 94
5.5 Environment - Out Interface Operation Scheme for actCoordinator . . . 100
5.5.1 Operation Model for oeAddVictim . . . 100
5.5.2 Operation Model for oeCloseCrisis . . . 103
5.5.3 Operation Model for oeDeleteInjury . . . 105
5.5.4 Operation Model for oeDeleteVictim . . . 106
5.5.5 Operation Model for oeGetAlertsSet . . . 107
5.5.6 Operation Model for oeGetCrisisSet . . . 109
5.5.7 Operation Model for oeGetInjurySet . . . 111
5.5.8 Operation Model for oeGetVictimSet . . . 112
5.5.9 Operation Model for oeInvalidateAlert . . . 113
5.5.10 Operation Model for oeReportOnCrisis . . . 115
5.5.11 Operation Model for oeSetCrisisHandler . . . 117

5.5.12 Operation Model for oeSetCrisisStatus . . . 119
5.5.13 Operation Model for oeSetCrisisType . . . 122
5.5.14 Operation Model for oeValidateAlert . . . 124
5.5.15 Operation Model for oeAddInjury . . . 126
5.6 Environment - Out Interface Operation Scheme for actHospital . . . 128
5.6.1 Operation Model for oeGetHospitalCrisisSet . . . 128
5.6.2 Operation Model for oeGetHospitalInjurySet . . . 129
5.6.3 Operation Model for oeGetHospitalVictimSet . . . 130
5.7 Environment - Out Interface Operation Scheme for actMsrCreator . . . 131
5.7.1 Operation Model for oeCreateSystemAndEnvironment . . . 131
5.8 Environment - Actor Operation Scheme for actMsrCreator . . . 136
5.8.1 Operation Model for init . . . 136
5.9 Primary Types - Operation Schemes for Class ctAdministrator . . . 136
5.9.1 Operation Model for init . . . 136
5.10 Primary Types - Operation Schemes for Class ctAlert . . . 137
5.10.1 Operation Model for init . . . 137
5.10.2 Operation Model for isSentToCoordinator . . . 139
5.11 Primary Types - Operation Schemes for Class ctAuthenticated . . . 140
5.11.1 Operation Model for init . . . 140
5.12 Primary Types - Operation Schemes for Class ctCoordinator . . . 141
5.12.1 Operation Model for init . . . 141
5.13 Primary Types - Operation Schemes for Class ctCrisis . . . 142
5.13.1 Operation Model for init . . . 142
5.13.2 Operation Model for handlingDelayPassed . . . 143
5.13.3 Operation Model for maxHandlingDelayPassed . . . 145
5.13.4 Operation Model for isSentToCoordinator . . . 146
5.13.5 Operation Model for isSentToHospital . . . 147
5.13.6 Operation Model for isAllocatedIfPossible . . . 148
5.14 Primary Types - Operation Schemes for Class ctHospital . . . 150
5.14.1 Operation Model for init . . . 150
5.15 Primary Types - Operation Schemes for Class ctHuman . . . 151
5.15.1 Operation Model for init . . . 151
5.15.2 Operation Model for isAcknowledged . . . 152
5.16 Primary Types - Operation Schemes for Class ctInjury . . . 153
5.16.1 Operation Model for init . . . 153
5.16.2 Operation Model for isSentToCoordinator . . . 154
5.16.3 Operation Model for isSentToHospital . . . 155
5.17 Primary Types - Operation Schemes for Class ctState . . . 155
5.17.1 Operation Model for init . . . 155
5.18 Primary Types - Operation Schemes for Class ctVictim . . . 157
5.18.1 Operation Model for init . . . 157
5.18.2 Operation Model for isSentToCoordinator . . . 158
5.18.3 Operation Model for isSentToHospital . . . 159
5.19 Primary Types - Operation Schemes for Datatype dtAlertID . . . 159
5.19.1 Operation Model for is . . . 159
5.20 Primary Types - Operation Schemes for Datatype dtComment . . . 161
5.20.1 Operation Model for is . . . 161
5.21 Primary Types - Operation Schemes for Datatype dtCoordinatorID . . . 162
5.21.1 Operation Model for is . . . 162

5.22 Primary Types - Operation Schemes for Datatype dtCrisisID . . . 163
5.22.1 Operation Model for is . . . 163
5.23 Primary Types - Operation Schemes for Datatype dtGPSLocation . . . 164
5.23.1 Operation Model for is . . . 164
5.23.2 Operation Model for isNearTo . . . 166
5.24 Primary Types - Operation Schemes for Datatype dtHospitalID . . . 167
5.24.1 Operation Model for is . . . 167
5.25 Primary Types - Operation Schemes for Datatype dtLatitude . . . 168
5.25.1 Operation Model for is . . . 168
5.26 Primary Types - Operation Schemes for Datatype dtLogin . . . 169
5.26.1 Operation Model for is . . . 169
5.27 Primary Types - Operation Schemes for Datatype dtLongitude . . . 170
5.27.1 Operation Model for is . . . 170
5.28 Primary Types - Operation Schemes for Datatype dtPassword . . . 171
5.28.1 Operation Model for is . . . 171
5.29 Primary Types - Operation Schemes for Datatype dtPhoneNumber . . . 173
5.29.1 Operation Model for is . . . 173
5.30 Primary Types - Operation Schemes for Enumeration etAlertStatus . . . 174
5.30.1 Operation Model for is . . . 174
5.31 Primary Types - Operation Schemes for Enumeration etCorruptionKind . . . 175
5.31.1 Operation Model for is . . . 175
5.32 Primary Types - Operation Schemes for Enumeration etCrisisStatus . . . 176
5.32.1 Operation Model for is . . . 176
5.33 Primary Types - Operation Schemes for Enumeration etCrisisType . . . 177
5.33.1 Operation Model for is . . . 177
5.34 Primary Types - Operation Schemes for Enumeration etHumanKind . . . 178
5.34.1 Operation Model for is . . . 178
5.35 Primary Types - Operation Schemes for Enumeration etInjuryKind . . . 179
5.35.1 Operation Model for is . . . 179
5.36 Secondary Types - Operation Schemes for Classes . . . 180
5.37 Secondary Types - Operation Schemes for Datatype dtSMS . . . 180
5.37.1 Operation Model for is . . . 180
5.38 Secondary Types - Operation Schemes for Enumerations . . . 181

6 Test Model(s) . . . 183
6.1 Test Model for testcase01 . . . 183
6.1.1 Test Steps Specification . . . 183
6.1.2 Test Case Instance - instance01 . . . 205
6.1.3 Test Case Instance - instance01Part01 . . . 205
6.1.4 Test Case Instance - instance01Part02 . . . 207

7 Additional Constraints . . . 209
7.1 Quality Constraints . . . 209
7.1.1 Functional suitability . . . 209
7.1.2 Performance efficiency . . . 209
7.1.3 Compatibility . . . 210
7.1.4 Usability . . . 210
7.1.5 Reliability . . . 211
7.1.6 Security . . . 212
7.1.7 Maintainability . . . 212

7.1.8 Portability . . . 213
7.2 Other Constraints . . . 214

A Undocumented Messir Specification Elements . . . 217
A.1 Undocumented Use Cases . . . 217
A.1.1 Undocumented Use Case Views . . . 217
A.2 Undocumented Use Case Instances . . . 217
A.2.1 Undocumented Use Case Instances - User-Goal Level . . . 217
A.2.2 Undocumented Use Case Instance Views . . . 217
A.3 Undocumented Environment Model Views . . . 217
A.4 Undocumented Primary Types . . . 217
A.4.1 Undocumented Primary Enumeration Types . . . 217
A.5 Undocumented Concept Model Views . . . 217
A.6 Undocumented Operation Specifications . . . 217
A.7 Undocumented Test-Case Instance Specifications . . . 218

B Specification project lu.uni.lassy.excalibur.examples.icrash . . . 219
B.1 Use Cases Model . . . 220
B.1.1 Use Cases . . . 220

C Messir Specification Files Listing . . . 221
C.1 File /src-gen/messir-spec/.views.msr . . . 221
C.2 File /src-gen/messir-spec/operations/concepts/secondarytypes-datatypes/dtSMS.msr 221
C.3 File /src-gen/messir-spec/operations.../environment-actActivator-oeSetClock.msr . . 222
C.4 File /src-gen.../environment-actActivator-oeSollicitateCrisisHandling.msr . . . 222
C.5 File /src-gen/messir-spec.../environment-actAdministrator-oeAddCoordinator.msr . 223
C.6 File /src-gen/messir-spec.../environment-actAdministrator-oeAddHospital.msr . . . 224
C.7 File /src-gen.../environment-actAdministrator-oeDeleteCoordinator.msr . . . 225
C.8 File /src-gen/messir-spec.../environment-actAdministrator-oeDeleteHospital.msr . . 226
C.9 File /src-gen/messir-spec/operations.../environment-actAuthenticated.msr . . . 227
C.10 File /src-gen/messir-spec/operations/environment/environment-actComCompany.msr 229
C.11 File /src-gen/messir-spec.../environment-actCoordinator-oeAddInjury.msr . . . 231
C.12 File /src-gen/messir-spec.../environment-actCoordinator-oeAddVictim.msr . . . 232
C.13 File /src-gen/messir-spec.../environment-actCoordinator-oeCloseCrisis.msr . . . 233
C.14 File /src-gen/messir-spec.../environment-actCoordinator-oeDeleteInjury.msr . . . 233
C.15 File /src-gen/messir-spec.../environment-actCoordinator-oeDeleteVictim.msr . . . 234
C.16 File /src-gen/messir-spec.../environment-actCoordinator-oeGetAlertsSet.msr . . . 235
C.17 File /src-gen/messir-spec.../environment-actCoordinator-oeGetCrisisSet.msr . . . 235
C.18 File /src-gen/messir-spec.../environment-actCoordinator-oeGetInjurySet.msr . . . 235
C.19 File /src-gen/messir-spec.../environment-actCoordinator-oeGetVictimSet.msr . . . 236
C.20 File /src-gen/messir-spec.../environment-actCoordinator-oeInvalidateAlert.msr . . . 237
C.21 File /src-gen/messir-spec.../environment-actCoordinator-oeReportOnCrisis.msr . . . 237
C.22 File /src-gen/messir-spec.../environment-actCoordinator-oeSetCrisisHandler.msr . . 237
C.23 File /src-gen/messir-spec.../environment-actCoordinator-oeSetCrisisStatus.msr . . . 238
C.24 File /src-gen/messir-spec.../environment-actCoordinator-oeSetCrisisType.msr . . . 238
C.25 File /src-gen/messir-spec.../environment-actCoordinator-oeValidateAlert.msr . . . 239
C.26 File /src-gen/messir-spec.../environment-actHospital-oeGetHospitalCrisisSet.msr . . 239
C.27 File /src-gen/messir-spec.../environment-actHospital-oeGetHospitalInjurySet.msr . . 240
C.28 File /src-gen/messir-spec.../environment-actHospital-oeGetHospitalVictimSet.msr . . 241
C.29 File /src-gen/messir-spec/operations.../environment-actMsrCreator-init.msr . . . 242

C.30 File /src-gen.../environment-actMsrCreator-oeCreateSystemAndEnvironment.msr . 242
C.31 File /src-gen/messir-spec/environment/environment.msr 243
C.32 File /src-gen/messir-spec.../evironment-actCoordinator-oeAddInjury.msr 246
C.33 File /src-gen/messir-spec/concepts/primarytypes-associations.msr 247
C.34 File /src-gen/messir-spec.../primarytypes-classes-ctAdministrator.msr 248
C.35 File /src-gen/messir-spec/operations.../primarytypes-classes-ctAlert.msr 248
C.36 File /src-gen/messir-spec.../primarytypes-classes-ctAuthenticated.msr 249
C.37 File /src-gen/messir-spec/operations.../primarytypes-classes-ctCoordinator.msr . . . 249
C.38 File /src-gen/messir-spec/operations.../primarytypes-classes-ctCrisis.msr 250
C.39 File /src-gen/messir-spec/operations.../primarytypes-classes-ctHuman.msr 252
C.40 File /src-gen/messir-spec/operations.../primarytypes-classes-ctInjury.msr 253
C.41 File /src-gen/messir-spec/operations.../primarytypes-classes-ctState.msr 254
C.42 File /src-gen/messir-spec/operations.../primarytypes-classes-ctVictim.msr 255
C.43 File /src-gen/messir-spec/concepts/primarytypes-classes.msr 256
C.44 File /src-gen/messir-spec/operations.../primarytypes-datatypes-dtAlertID.msr . . . 258
C.45 File /src-gen/messir-spec/operations.../primarytypes-datatypes-dtComment.msr . . 259
C.46 File /src-gen/messir-spec.../primarytypes-datatypes-dtCoordinatorID.msr 259
C.47 File /src-gen/messir-spec/operations.../primarytypes-datatypes-dtCrisisID.msr . . . 259
C.48 File /src-gen/messir-spec.../primarytypes-datatypes-dtGPSLocation.msr 260
C.49 File /src-gen/messir-spec/operations.../primarytypes-datatypes-dtHospitalID.msr . . 261
C.50 File /src-gen/messir-spec/operations.../primarytypes-datatypes-dtLogin.msr 262
C.51 File /src-gen/messir-spec/operations.../primarytypes-datatypes-dtPassword.msr . . 262
C.52 File /src-gen/messir-spec.../primarytypes-datatypes-dtPhoneNumber.msr 263
C.53 File /src-gen/messir-spec.../primarytypes-datatypes-etAlertStatus.msr 263
C.54 File /src-gen/messir-spec.../primarytypes-datatypes-etCorruptionKind.msr 264
C.55 File /src-gen/messir-spec.../primarytypes-datatypes-etCrisisStatus.msr 264
C.56 File /src-gen/messir-spec/operations.../primarytypes-datatypes-etCrisisType.msr . . 265
C.57 File /src-gen/messir-spec/operations.../primarytypes-datatypes-etHumanKind.msr . 265
C.58 File /src-gen/messir-spec/operations.../primarytypes-datatypes-etInjuryKind.msr . 265
C.59 File /src-gen/messir-spec/concepts/primarytypes-datatypes.msr 266
C.60 File /src-gen/messir-spec/operations.../ptimarytypes-classes-ctHospital.msr 267
C.61 File /src-gen/messir-spec/concepts/secondarytypes-associations.msr 268
C.62 File /src-gen/messir-spec/concepts/secondarytypes-classes.msr 268
C.63 File /src-gen/messir-spec/concepts/secondarytypes-datatypes.msr 268
C.64 File /src-gen/messir-spec/usecases/subfunctions-usecases.msr 269
C.65 File /src-gen/messir-spec/test/tc-testcase01.msr 272
C.66 File /src-gen/messir-spec/test/tci-testcase01-instance01.msr 280
C.67 File /src-gen/messir-spec/usecases/ugHospitalMonitor.msr 289
C.68 File /src-gen/messir-spec/usecases/usecase-suDeployAndRun.msr 290
C.69 File /src-gen/messir-spec/usecases/usecase-suGlobalCrisisHandling.msr 295
C.70 File /src-gen/messir-spec/usecases/usecase-suHospitalCrisisInteraction.msr 296
C.71 File /src-gen/messir-spec/usecases/usecase-ugAdministrateTheSystem.msr 296
C.72 File /src-gen/messir-spec/usecases/usecase-ugCrisisInteraction.msr 297
C.73 File /src-gen/messir-spec/usecases/usecase-ugManageCrisis.msr 298
C.74 File /src-gen/messir-spec/usecases/usecase-ugMonitor.msr 298
C.75 File /src-gen/messir-spec/usecases/usecase-ugSecurelyUseSystem.msr 299
C.76 File /src-gen.../usecaseinstance-ugSecurelyUseSystem-uciugSecurelyUseSystem.msr . 299

D Listing of the Prolog Files Referenced in the Operation Model Specification . . 301

D.1 File /src-gen/prolog-ref-spec/Operations.../outactActivator-oeSetClock.pl 301
D.2 File /src-gen/prolog-ref-spec.../outactActivator-oeSollicitateCrisisHandling.pl 302
D.3 File /src-gen/prolog-ref-spec.../outactAdministrator-oeAddCoordinator.pl 304
D.4 File /src-gen/prolog-ref-spec.../outactAdministrator-oeDeleteCoordinator.pl 305
D.5 File /src-gen/prolog-ref-spec/Operations.../outactAuthenticated-oeLogin.pl 306
D.6 File /src-gen/prolog-ref-spec/Operations.../outactAuthenticated-oeLogout.pl 308
D.7 File /src-gen/prolog-ref-spec/Operations.../outactComCompany-oeAlert.pl 309
D.8 File /src-gen/prolog-ref-spec/Operations.../outactCoordinator-oeCloseCrisis.pl . . . 313
D.9 File /src-gen/prolog-ref-spec/Operations.../outactCoordinator-oeGetAlertsSet.pl . . 314
D.10 File /src-gen/prolog-ref-spec/Operations.../outactCoordinator-oeGetCrisisSet.pl . . 315
D.11 File /src-gen/prolog-ref-spec.../outactCoordinator-oeInvalidateAlert.pl 316
D.12 File /src-gen/prolog-ref-spec.../outactCoordinator-oeReportOnCrisis.pl 318
D.13 File /src-gen/prolog-ref-spec.../outactCoordinator-oeSetCrisisHandler.pl 319
D.14 File /src-gen/prolog-ref-spec.../outactCoordinator-oeSetCrisisStatus.pl 321
D.15 File /src-gen/prolog-ref-spec.../outactCoordinator-oeSetCrisisType.pl 323
D.16 File /src-gen/prolog-ref-spec.../outactCoordinator-oeValidateAlert.pl 324
D.17 File /src-gen.../outactMsrCreator-oeCreateSystemAndEnvironment.pl 326
D.18 File /src-gen/prolog-ref-spec.../PrimaryTypesClasses-ctAdministrator-init.pl 328
D.19 File /src-gen/prolog-ref-spec/Operations.../PrimaryTypesClasses-ctAlert-init.pl . . . 328
D.20 File /src-gen.../PrimaryTypesClasses-ctAlert-isSentToCoordinator.pl 329
D.21 File /src-gen/prolog-ref-spec.../PrimaryTypesClasses-ctAuthenticated-init.pl 329
D.22 File /src-gen/prolog-ref-spec.../PrimaryTypesClasses-ctCoordinator-init.pl 330
D.23 File /src-gen.../PrimaryTypesClasses-ctCrisis-handlingDelayPassed.pl 330
D.24 File /src-gen/prolog-ref-spec.../PrimaryTypesClasses-ctCrisis-init.pl 331
D.25 File /src-gen.../PrimaryTypesClasses-ctCrisis-isAllocatedIfPossible.pl 331
D.26 File /src-gen.../PrimaryTypesClasses-ctCrisis-isSentToCoordinator.pl 332
D.27 File /src-gen.../PrimaryTypesClasses-ctCrisis-maxHandlingDelayPassed.pl 333
D.28 File /src-gen/prolog-ref-spec/Operations.../PrimaryTypesClasses-ctHuman-init.pl . 334
D.29 File /src-gen/prolog-ref-spec.../PrimaryTypesClasses-ctHuman-isAcknowledged.pl . 334
D.30 File /src-gen/prolog-ref-spec/Operations.../PrimaryTypesClasses-ctState-init.pl . . 334
D.31 File /src-gen/prolog-ref-spec.../PrimaryTypesDatatypes-dtAlertID-is.pl 335
D.32 File /src-gen/prolog-ref-spec.../PrimaryTypesDatatypes-dtComment-is.pl 336
D.33 File /src-gen/prolog-ref-spec.../PrimaryTypesDatatypes-dtCoordinatorID-is.pl . . . 336
D.34 File /src-gen/prolog-ref-spec.../PrimaryTypesDatatypes-dtCrisisID-is.pl 337
D.35 File /src-gen/prolog-ref-spec.../PrimaryTypesDatatypes-dtGPSLocation-is.pl 337
D.36 File /src-gen.../PrimaryTypesDatatypes-dtGPSLocation-isNearTo.pl 338
D.37 File /src-gen/prolog-ref-spec.../PrimaryTypesDatatypes-dtLatitude-is.pl 339
D.38 File /src-gen/prolog-ref-spec/Operations.../PrimaryTypesDatatypes-dtLogin-is.pl . 339
D.39 File /src-gen/prolog-ref-spec.../PrimaryTypesDatatypes-dtLongitude-is.pl 340
D.40 File /src-gen/prolog-ref-spec.../PrimaryTypesDatatypes-dtPassword-is.pl 340
D.41 File /src-gen/prolog-ref-spec.../PrimaryTypesDatatypes-dtPhoneNumber-is.pl 341
D.42 File /src-gen/prolog-ref-spec.../PrimaryTypesDatatypes-etAlertStatus-is.pl 341
D.43 File /src-gen/prolog-ref-spec.../PrimaryTypesDatatypes-etCrisisStatus-is.pl 342
D.44 File /src-gen/prolog-ref-spec.../PrimaryTypesDatatypes-etCrisisType-is.pl 342
D.45 File /src-gen/prolog-ref-spec.../PrimaryTypesDatatypes-etHumanKind-is.pl 343
D.46 File /src-gen/prolog-ref-spec/Operations.../SecondaryTypesDatatypes-dtSMS-is.pl . 343

Glossary **345**

List of Figures

2.1 lu.uni.lassy.excalibur.examples.icrash Use Case Diagram: uc-suDeployAndRun 27
2.2 lu.uni.lassy.excalibur.examples.icrash Use Case Diagram: uc-suGlobalCrisisHandling . . 31
2.3 lu.uni.lassy.excalibur.examples.icrash Use Case Diagram: uc-ugAdministrateTheSystem . 31
2.4 lu.uni.lassy.excalibur.examples.icrash Use Case Diagram: uc-ugManageCrisis 38
2.5 lu.uni.lassy.excalibur.examples.icrash Use Case Diagram: uc-ugMonitor 38
2.6 lu.uni.lassy.excalibur.examples.icrash Use Case Diagram: uc-ugSecurelyUseSystem . . 39
2.7 lu.uni.lassy.excalibur.examples.icrash Use Case Diagram: uc-oeDeleteHospital 39
2.8 lu.uni.lassy.excalibur.examples.icrash Use Case Diagram: uc-oeSetCrisisHandler 42
2.9 lu.uni.lassy.excalibur.examples.icrash Use Case Diagram: uc-oeSollicitateCrisisHandling 43
2.10 lu.uni.lassy.excalibur.examples.icrash Sequence Diagram: uci-suDeployAndRun-uciSimpleAndComplete-Pa
2.11 lu.uni.lassy.excalibur.examples.icrash Sequence Diagram: uci-suDeployAndRun-uciSimpleAndComplete-Pa
2.12 lu.uni.lassy.excalibur.examples.icrash Sequence Diagram: uci-uciugSecurelyUseSystem . 47

3.1 Environment Model - Local View 01 - environment model local view - Part 50
3.2 Environment Model - Local View 02 - environment model local view - Part 51
3.3 Environment Model - Local View 03 - administrator actor environment mode 51
3.4 Environment Model - Local View 04 - coordinator actor environment model 52
3.5 Environment Model - Local View 05 - authenticated actor environment mode 52
3.6 Environment Model - Local View 09 - 52
3.7 Environment Model - Global View 01 - em-gv-01 environment model global v 54

4.1 Concept Model - PrimaryTypes-Classes local view 01 - Local view of all the primary types 60
4.2 Concept Model - PrimaryTypes-Classes local view 02 - local view of the ctState primary ty 61
4.3 Concept Model - PrimaryTypes-Classes local view 03 - local view of the ctAlert primary ty 61
4.4 Concept Model - PrimaryTypes-Classes local view 04 - local view of the ctCrisis primary t 61
4.5 Concept Model - PrimaryTypes-Classes global view 01 - Primary types class types global vi 62
4.6 Concept Model - PrimaryTypes-Datatypes local view 06 - 62
4.7 Concept Model - PrimaryTypes-Datatypes local view 08 - 62
4.8 Concept Model - PrimaryTypes-Datatypes global view 01 - global view of primary types dataty 63
4.9 Concept Model - SecondaryTypes-Datatypes local view 01 - Local view of the secondary types da 64

5.1 lu.uni.lassy.excalibur.examples.icrash Operation Scope: operation-scope-outactActivator-oeSollicitateCrisis
5.2 lu.uni.lassy.excalibur.examples.icrash Operation Scope: operation-scope-outactComCompany-oeAlertv2 10
5.3 lu.uni.lassy.excalibur.examples.icrash Operation Scope: operation-scope-outactComCompany-oeAlertv3 10
5.4 lu.uni.lassy.excalibur.examples.icrash Operation Scope: operation-scope-outactMsrCreator-oeCreateSyster

6.1 lu.uni.lassy.excalibur.examples.icrash Sequence Diagram: tci-testcase01-instance01-Part01 206
6.2 lu.uni.lassy.excalibur.examples.icrash Sequence Diagram: tci-testcase01-instance01-Part02 208

B.1 lu.uni.lassy.excalibur.examples.icrash Use Case Diagram: uc-oeCloseCrisis 220

Listings

5.1 Messir (MCL-oriented) specification of the operation *oeSetClock*. 73
5.2 Messir (Prolog-oriented) implementation of the operation *oeSetClock*. 74
5.3 Messir (MCL-oriented) specification of the operation *oeSollicitateCrisisHandling*. . 75
5.4 Messir (Prolog-oriented) implementation of the operation *oeSollicitateCrisisHandling*. 76
5.5 Messir (MCL-oriented) specification of the operation *oeAddCoordinator*. 79
5.6 Messir (Prolog-oriented) implementation of the operation *oeAddCoordinator*. 81
5.7 Messir (MCL-oriented) specification of the operation *oeAddHospital*. 83
5.8 Messir (MCL-oriented) specification of the operation *oeDeleteCoordinator*. 84
5.9 Messir (Prolog-oriented) implementation of the operation *oeDeleteCoordinator*. . . 85
5.10 Messir (MCL-oriented) specification of the operation *oeDeleteHospital*. 87
5.11 Messir (MCL-oriented) specification of the operation *oeLogin*. 89
5.12 Messir (Prolog-oriented) implementation of the operation *oeLogin*. 90
5.13 Messir (MCL-oriented) specification of the operation *oeLogout*. 92
5.14 Messir (Prolog-oriented) implementation of the operation *oeLogout*. 93
5.15 Messir (MCL-oriented) specification of the operation *oeAlert*. 95
5.16 Messir (Prolog-oriented) implementation of the operation *oeAlert*. 96
5.17 Messir (MCL-oriented) specification of the operation *oeAddVictim*. 102
5.18 Messir (Prolog-oriented) implementation of the operation *oeCloseCrisis*. 103
5.19 Messir (MCL-oriented) specification of the operation *oeDeleteInjury*. 105
5.20 Messir (MCL-oriented) specification of the operation *oeDeleteVictim*. 107
5.21 Messir (Prolog-oriented) implementation of the operation *oeGetAlertsSet*. 108
5.22 Messir (Prolog-oriented) implementation of the operation *oeGetCrisisSet*. 109
5.23 Messir (MCL-oriented) specification of the operation *oeGetInjurySet*. 111
5.24 Messir (MCL-oriented) specification of the operation *oeGetVictimSet*. 112
5.25 Messir (Prolog-oriented) implementation of the operation *oeInvalidateAlert*. 113
5.26 Messir (Prolog-oriented) implementation of the operation *oeReportOnCrisis*. 115
5.27 Messir (Prolog-oriented) implementation of the operation *oeSetCrisisHandler*. . . . 118
5.28 Messir (Prolog-oriented) implementation of the operation *oeSetCrisisStatus*. 120
5.29 Messir (Prolog-oriented) implementation of the operation *oeSetCrisisType*. 122
5.30 Messir (MCL-oriented) specification of the operation *oeValidateAlert*. 124
5.31 Messir (Prolog-oriented) implementation of the operation *oeValidateAlert*. 125
5.32 Messir (MCL-oriented) specification of the operation *oeAddInjury*. 127
5.33 Messir (MCL-oriented) specification of the operation *oeGetHospitalCrisisSet*. . . . 128
5.34 Messir (MCL-oriented) specification of the operation *oeGetHospitalInjurySet*. . . . 129
5.35 Messir (MCL-oriented) specification of the operation *oeGetHospitalVictimSet*. . . . 130
5.36 Messir (MCL-oriented) specification of the operation *oeCreateSystemAndEnvironment*. 132
5.37 Messir (Prolog-oriented) implementation of the operation *oeCreateSystemAndEnvironment*. 133
5.38 Messir (MCL-oriented) specification of the operation *init*. 136
5.39 Messir (Prolog-oriented) implementation of the operation *init*. 137
5.40 Messir (MCL-oriented) specification of the operation *init*. 138

5.41 Messir (Prolog-oriented) implementation of the operation *init*. 138
5.42 Messir (MCL-oriented) specification of the operation *isSentToCoordinator*. 139
5.43 Messir (Prolog-oriented) implementation of the operation *isSentToCoordinator*. . . 139
5.44 Messir (Prolog-oriented) implementation of the operation *init*. 140
5.45 Messir (MCL-oriented) specification of the operation *init*. 141
5.46 Messir (Prolog-oriented) implementation of the operation *init*. 141
5.47 Messir (MCL-oriented) specification of the operation *init*. 142
5.48 Messir (Prolog-oriented) implementation of the operation *init*. 143
5.49 Messir (MCL-oriented) specification of the operation *handlingDelayPassed*. 144
5.50 Messir (Prolog-oriented) implementation of the operation *handlingDelayPassed*. . . 144
5.51 Messir (MCL-oriented) specification of the operation *maxHandlingDelayPassed*. . . 145
5.52 Messir (Prolog-oriented) implementation of the operation *maxHandlingDelayPassed*. 146
5.53 Messir (MCL-oriented) specification of the operation *isSentToCoordinator*. 147
5.54 Messir (Prolog-oriented) implementation of the operation *isSentToCoordinator*. . . 147
5.55 Messir (MCL-oriented) specification of the operation *isSentToHospital*. 148
5.56 Messir (MCL-oriented) specification of the operation *isAllocatedIfPossible*. 148
5.57 Messir (Prolog-oriented) implementation of the operation *isAllocatedIfPossible*. . . 149
5.58 Messir (MCL-oriented) specification of the operation *init*. 150
5.59 Messir (MCL-oriented) specification of the operation *init*. 151
5.60 Messir (Prolog-oriented) implementation of the operation *init*. 152
5.61 Messir (Prolog-oriented) implementation of the operation *isAcknowledged*. 152
5.62 Messir (MCL-oriented) specification of the operation *init*. 153
5.63 Messir (MCL-oriented) specification of the operation *isSentToCoordinator*. 154
5.64 Messir (Prolog-oriented) implementation of the operation *isSentToCoordinator*. . . 154
5.65 Messir (MCL-oriented) specification of the operation *isSentToHospital*. 155
5.66 Messir (MCL-oriented) specification of the operation *init*. 156
5.67 Messir (Prolog-oriented) implementation of the operation *init*. 156
5.68 Messir (MCL-oriented) specification of the operation *init*. 157
5.69 Messir (MCL-oriented) specification of the operation *isSentToCoordinator*. 158
5.70 Messir (Prolog-oriented) implementation of the operation *isSentToCoordinator*. . . 158
5.71 Messir (MCL-oriented) specification of the operation *isSentToHospital*. 159
5.72 Messir (MCL-oriented) specification of the operation *is*. 160
5.73 Messir (Prolog-oriented) implementation of the operation *is*. 160
5.74 Messir (MCL-oriented) specification of the operation *is*. 161
5.75 Messir (Prolog-oriented) implementation of the operation *is*. 161
5.76 Messir (MCL-oriented) specification of the operation *is*. 162
5.77 Messir (Prolog-oriented) implementation of the operation *is*. 163
5.78 Messir (MCL-oriented) specification of the operation *is*. 163
5.79 Messir (Prolog-oriented) implementation of the operation *is*. 164
5.80 Messir (MCL-oriented) specification of the operation *is*. 165
5.81 Messir (Prolog-oriented) implementation of the operation *is*. 165
5.82 Messir (MCL-oriented) specification of the operation *isNearTo*. 166
5.83 Messir (Prolog-oriented) implementation of the operation *isNearTo*. 167
5.84 Messir (MCL-oriented) specification of the operation *is*. 168
5.85 Messir (MCL-oriented) specification of the operation *is*. 168
5.86 Messir (Prolog-oriented) implementation of the operation *is*. 169
5.87 Messir (MCL-oriented) specification of the operation *is*. 169
5.88 Messir (Prolog-oriented) implementation of the operation *is*. 170
5.89 Messir (MCL-oriented) specification of the operation *is*. 171

5.90 Messir (Prolog-oriented) implementation of the operation *is*. 171
5.91 Messir (MCL-oriented) specification of the operation *is*. 172
5.92 Messir (Prolog-oriented) implementation of the operation *is*. 172
5.93 Messir (MCL-oriented) specification of the operation *is*. 173
5.94 Messir (Prolog-oriented) implementation of the operation *is*. 173
5.95 Messir (MCL-oriented) specification of the operation *is*. 174
5.96 Messir (Prolog-oriented) implementation of the operation *is*. 175
5.97 Messir (MCL-oriented) specification of the operation *is*. 175
5.98 Messir (MCL-oriented) specification of the operation *is*. 176
5.99 Messir (Prolog-oriented) implementation of the operation *is*. 176
5.100 Messir (MCL-oriented) specification of the operation *is*. 177
5.101 Messir (Prolog-oriented) implementation of the operation *is*. 177
5.102 Messir (MCL-oriented) specification of the operation *is*. 178
5.103 Messir (Prolog-oriented) implementation of the operation *is*. 179
5.104 Messir (MCL-oriented) specification of the operation *is*. 179
5.105 Messir (MCL-oriented) specification of the operation *is*. 180
5.106 Messir (Prolog-oriented) implementation of the operation *is*. 180
6.1 Messir (MCL-oriented) specification of the test step *testcase01-ts01oeCreateSystemAndEnvironm*
6.2 Messir (Prolog-oriented) implementation of the test step *testcase01-ts01oeCreateSystemAndEnvi*
6.3 Messir (MCL-oriented) specification of the test step *testcase01-ts02oeSetClock*. . . 186
6.4 Messir (MCL-oriented) specification of the test step *testcase01-ts03oeLogin*. 187
6.5 Messir (MCL-oriented) specification of the test step *testcase01-ts04oeAddCoordinator*. 188
6.6 Messir (MCL-oriented) specification of the test step *testcase01-ts05oeLogout*. . . . 189
6.7 Messir (MCL-oriented) specification of the test step *testcase01-ts06oeSetClock02*. . 190
6.8 Messir (MCL-oriented) specification of the test step *testcase01-ts07oeAlert1*. 191
6.9 Messir (MCL-oriented) specification of the test step *testcase01-ts08oeSetClock03*. . 192
6.10 Messir (MCL-oriented) specification of the test step *testcase01-ts09oeSollicitateCrisisHandling*. 1
6.11 Messir (MCL-oriented) specification of the test step *testcase01-ts10oeLogin02*. . . . 194
6.12 Messir (MCL-oriented) specification of the test step *testcase01-ts11oeGetCrisisSet*. 195
6.13 Messir (MCL-oriented) specification of the test step *testcase01-ts12oeSetCrisisHandler*. 197
6.14 Messir (MCL-oriented) specification of the test step *testcase01-ts13oeSetClock04*. . 198
6.15 Messir (MCL-oriented) specification of the test step *testcase01-ts14oeValidateAlert*. 199
6.16 Messir (MCL-oriented) specification of the test step *testcase01-ts15oeAlert2*. 200
6.17 Messir (MCL-oriented) specification of the test step *testcase01-ts16oeSetClock05*. . 202
6.18 Messir (MCL-oriented) specification of the test step *testcase01-ts17oeSetCrisisStatus*. 203
6.19 Messir (MCL-oriented) specification of the test step *testcase01-ts18oeReportOnCrisis*. 204
6.20 Messir (MCL-oriented) specification of the test step *testcase01-ts19oeCloseCrisis*. . 205
C.1 Messir Spec. file .views.msr. 221
C.2 Messir Spec. file dtSMS.msr. 221
C.3 Messir Spec. file environment-actActivator-oeSetClock.msr. 222
C.4 Messir Spec. file environment-actActivator-oeSollicitateCrisisHandling.msr. 222
C.5 Messir Spec. file environment-actAdministrator-oeAddCoordinator.msr. 223
C.6 Messir Spec. file environment-actAdministrator-oeAddHospital.msr. 224
C.7 Messir Spec. file environment-actAdministrator-oeDeleteCoordinator.msr. 225
C.8 Messir Spec. file environment-actAdministrator-oeDeleteHospital.msr. 226
C.9 Messir Spec. file environment-actAuthenticated.msr. 228
C.10 Messir Spec. file environment-actComCompany.msr. 229
C.11 Messir Spec. file environment-actCoordinator-oeAddInjury.msr. 231
C.12 Messir Spec. file environment-actCoordinator-oeAddVictim.msr. 232

C.13 Messir Spec. file environment-actCoordinator-oeCloseCrisis.msr. 233
C.14 Messir Spec. file environment-actCoordinator-oeDeleteInjury.msr. 233
C.15 Messir Spec. file environment-actCoordinator-oeDeleteVictim.msr. 234
C.16 Messir Spec. file environment-actCoordinator-oeGetAlertsSet.msr. 235
C.17 Messir Spec. file environment-actCoordinator-oeGetCrisisSet.msr. 235
C.18 Messir Spec. file environment-actCoordinator-oeGetInjurySet.msr. 235
C.19 Messir Spec. file environment-actCoordinator-oeGetVictimSet.msr. 236
C.20 Messir Spec. file environment-actCoordinator-oeInvalidateAlert.msr. 237
C.21 Messir Spec. file environment-actCoordinator-oeReportOnCrisis.msr. 237
C.22 Messir Spec. file environment-actCoordinator-oeSetCrisisHandler.msr. 237
C.23 Messir Spec. file environment-actCoordinator-oeSetCrisisStatus.msr. 238
C.24 Messir Spec. file environment-actCoordinator-oeSetCrisisType.msr. 238
C.25 Messir Spec. file environment-actCoordinator-oeValidateAlert.msr. 239
C.26 Messir Spec. file environment-actHospital-oeGetHospitalCrisisSet.msr. 239
C.27 Messir Spec. file environment-actHospital-oeGetHospitalInjurySet.msr. 240
C.28 Messir Spec. file environment-actHospital-oeGetHospitalVictimSet.msr. 241
C.29 Messir Spec. file environment-actMsrCreator-init.msr. 242
C.30 Messir Spec. file environment-actMsrCreator-oeCreateSystemAndEnvironment.msr. . . 242
C.31 Messir Spec. file environment.msr. 243
C.32 Messir Spec. file evironment-actCoordinator-oeAddInjury.msr. 246
C.33 Messir Spec. file primarytypes-associations.msr. 247
C.34 Messir Spec. file primarytypes-classes-ctAdministrator.msr. 248
C.35 Messir Spec. file primarytypes-classes-ctAlert.msr. 248
C.36 Messir Spec. file primarytypes-classes-ctAuthenticated.msr. 249
C.37 Messir Spec. file primarytypes-classes-ctCoordinator.msr. 249
C.38 Messir Spec. file primarytypes-classes-ctCrisis.msr. 250
C.39 Messir Spec. file primarytypes-classes-ctHuman.msr. 252
C.40 Messir Spec. file primarytypes-classes-ctInjury.msr. 253
C.41 Messir Spec. file primarytypes-classes-ctState.msr. 254
C.42 Messir Spec. file primarytypes-classes-ctVictim.msr. 255
C.43 Messir Spec. file primarytypes-classes.msr. 256
C.44 Messir Spec. file primarytypes-datatypes-dtAlertID.msr. 258
C.45 Messir Spec. file primarytypes-datatypes-dtComment.msr. 259
C.46 Messir Spec. file primarytypes-datatypes-dtCoordinatorID.msr. 259
C.47 Messir Spec. file primarytypes-datatypes-dtCrisisID.msr. 259
C.48 Messir Spec. file primarytypes-datatypes-dtGPSLocation.msr. 260
C.49 Messir Spec. file primarytypes-datatypes-dtHospitalID.msr. 261
C.50 Messir Spec. file primarytypes-datatypes-dtLogin.msr. 262
C.51 Messir Spec. file primarytypes-datatypes-dtPassword.msr. 262
C.52 Messir Spec. file primarytypes-datatypes-dtPhoneNumber.msr. 263
C.53 Messir Spec. file primarytypes-datatypes-etAlertStatus.msr. 263
C.54 Messir Spec. file primarytypes-datatypes-etCorruptionKind.msr. 264
C.55 Messir Spec. file primarytypes-datatypes-etCrisisStatus.msr. 264
C.56 Messir Spec. file primarytypes-datatypes-etCrisisType.msr. 265
C.57 Messir Spec. file primarytypes-datatypes-etHumanKind.msr. 265
C.58 Messir Spec. file primarytypes-datatypes-etInjuryKind.msr. 265
C.59 Messir Spec. file primarytypes-datatypes.msr. 266
C.60 Messir Spec. file ptimarytypes-classes-ctHospital.msr. 267
C.61 Messir Spec. file secondarytypes-associations.msr. 268

C.62 Messir Spec. file secondarytypes-classes.msr. . . . 268
C.63 Messir Spec. file secondarytypes-datatypes.msr. . . . 268
C.64 Messir Spec. file subfunctions-usecases.msr. . . . 269
C.65 Messir Spec. file tc-testcase01.msr. . . . 272
C.66 Messir Spec. file tci-testcase01-instance01.msr. . . . 280
C.67 Messir Spec. file ugHospitalMonitor.msr. . . . 289
C.68 Messir Spec. file usecase-suDeployAndRun.msr. . . . 290
C.69 Messir Spec. file usecase-suGlobalCrisisHandling.msr. . . . 295
C.70 Messir Spec. file usecase-suHospitalCrisisInteraction.msr. . . . 296
C.71 Messir Spec. file usecase-ugAdministrateTheSystem.msr. . . . 296
C.72 Messir Spec. file usecase-ugCrisisInteraction.msr. . . . 297
C.73 Messir Spec. file usecase-ugManageCrisis.msr. . . . 298
C.74 Messir Spec. file usecase-ugMonitor.msr. . . . 298
C.75 Messir Spec. file usecase-ugSecurelyUseSystem.msr. . . . 299
C.76 Messir Spec. file usecaseinstance-ugSecurelyUseSystem-uciugSecurelyUseSystem.msr. . . . 299
D.1 Prolog file outactActivator-oeSetClock.pl. . . . 301
D.2 Prolog file outactActivator-oeSollicitateCrisisHandling.pl. . . . 302
D.3 Prolog file outactAdministrator-oeAddCoordinator.pl. . . . 304
D.4 Prolog file outactAdministrator-oeDeleteCoordinator.pl. . . . 305
D.5 Prolog file outactAuthenticated-oeLogin.pl. . . . 306
D.6 Prolog file outactAuthenticated-oeLogout.pl. . . . 308
D.7 Prolog file outactComCompany-oeAlert.pl. . . . 309
D.8 Prolog file outactCoordinator-oeCloseCrisis.pl. . . . 313
D.9 Prolog file outactCoordinator-oeGetAlertsSet.pl. . . . 314
D.10 Prolog file outactCoordinator-oeGetCrisisSet.pl. . . . 315
D.11 Prolog file outactCoordinator-oeInvalidateAlert.pl. . . . 316
D.12 Prolog file outactCoordinator-oeReportOnCrisis.pl. . . . 318
D.13 Prolog file outactCoordinator-oeSetCrisisHandler.pl. . . . 319
D.14 Prolog file outactCoordinator-oeSetCrisisStatus.pl. . . . 321
D.15 Prolog file outactCoordinator-oeSetCrisisType.pl. . . . 323
D.16 Prolog file outactCoordinator-oeValidateAlert.pl. . . . 324
D.17 Prolog file outactMsrCreator-oeCreateSystemAndEnvironment.pl. . . . 326
D.18 Prolog file PrimaryTypesClasses-ctAdministrator-init.pl. . . . 328
D.19 Prolog file PrimaryTypesClasses-ctAlert-init.pl. . . . 328
D.20 Prolog file PrimaryTypesClasses-ctAlert-isSentToCoordinator.pl. . . . 329
D.21 Prolog file PrimaryTypesClasses-ctAuthenticated-init.pl. . . . 329
D.22 Prolog file PrimaryTypesClasses-ctCoordinator-init.pl. . . . 330
D.23 Prolog file PrimaryTypesClasses-ctCrisis-handlingDelayPassed.pl. . . . 330
D.24 Prolog file PrimaryTypesClasses-ctCrisis-init.pl. . . . 331
D.25 Prolog file PrimaryTypesClasses-ctCrisis-isAllocatedIfPossible.pl. . . . 331
D.26 Prolog file PrimaryTypesClasses-ctCrisis-isSentToCoordinator.pl. . . . 332
D.27 Prolog file PrimaryTypesClasses-ctCrisis-maxHandlingDelayPassed.pl. . . . 333
D.28 Prolog file PrimaryTypesClasses-ctHuman-init.pl. . . . 334
D.29 Prolog file PrimaryTypesClasses-ctHuman-isAcknowledged.pl. . . . 334
D.30 Prolog file PrimaryTypesClasses-ctState-init.pl. . . . 334
D.31 Prolog file PrimaryTypesDatatypes-dtAlertID-is.pl. . . . 335
D.32 Prolog file PrimaryTypesDatatypes-dtComment-is.pl. . . . 336
D.33 Prolog file PrimaryTypesDatatypes-dtCoordinatorID-is.pl. . . . 336
D.34 Prolog file PrimaryTypesDatatypes-dtCrisisID-is.pl. . . . 337

D.35 Prolog file PrimaryTypesDatatypes-dtGPSLocation-is.pl. 337
D.36 Prolog file PrimaryTypesDatatypes-dtGPSLocation-isNearTo.pl. 338
D.37 Prolog file PrimaryTypesDatatypes-dtLatitude-is.pl. 339
D.38 Prolog file PrimaryTypesDatatypes-dtLogin-is.pl. 339
D.39 Prolog file PrimaryTypesDatatypes-dtLongitude-is.pl. 340
D.40 Prolog file PrimaryTypesDatatypes-dtPassword-is.pl. 340
D.41 Prolog file PrimaryTypesDatatypes-dtPhoneNumber-is.pl. 341
D.42 Prolog file PrimaryTypesDatatypes-etAlertStatus-is.pl. 341
D.43 Prolog file PrimaryTypesDatatypes-etCrisisStatus-is.pl. 342
D.44 Prolog file PrimaryTypesDatatypes-etCrisisType-is.pl. 342
D.45 Prolog file PrimaryTypesDatatypes-etHumanKind-is.pl. 343
D.46 Prolog file SecondaryTypesDatatypes-dtSMS-is.pl. 343

Chapter 1

Introduction

1.1 Overview

iCrash is a simple system dedicated to any person who wants to inform of a car crash crisis situation in order to allow for crisis handling. At anytime and anywhere, anyone can be the witness or victim of a car crash and might be in a situation allowing for alerting this crisis. The *iCrash* system has for objectives to support crisis declaration and secure administration and crisis handling by the *iCrash* professional users.

1.2 Purpose and recipients of the document

This document is an analysis document complying with the Messir methodology [1]. Its intent is to provide an example of a precise specification of the functional properties of the *iCrash* system.

The recipients of this document are:

- the *iCrash* system's buyer company (ABC): this document is used as a contractual document jointly with any other document considered as useful (as requirement elicitation document, ...) in order to have a higher degree of precision in requirement description. It is also used as a basis document for the *iCrash* system validation using specification based testing.
- the *iCrash* system development company (ADC) is expected to use this document as the basis for development (mainly design, implementation, maintenance). It is also used for verification and validation using test plans defined using the analysis models described in this document and according to the Messir methodology.

1.3 Application Domain

The *iCrash* system belongs to the Crisis Management Systems Domain. It is a system dedicated to crisis professional and non professional end users. It has to be considered as an autonomous and external service for the society. It is not an institutional system certified and guaranteed by any governmental entity and thus, must be used with caution.

1.4 Definitions, acronyms and abbreviations

N.A.

1.5 Document structure

The document structure is designed to be coherent with the Messir methodology [1]. Section 2 provides a general description of the system purpose, its users, its environment and some general non functional requirements. A more detailed description of the non functional requirements, if any, are provided in section **??**. The system operation triggered by events sent by the external actors belonging to the environment are described in Section 3. The *iCrash* concepts used to represent the any persistent or transient information is given in Section 4. The precise specification of the system operations in term of system's state changes, events sent together with the constraints on the allowed sequences of system operations are described in Section 5.

Last Modification: @author Petrov Vladislav and Gusev Maxim @date 22.12.16

Chapter 2

General Description

In the context of the Messir method, the information provided in this section is intended to present the system for which the Messir analysis is provided. The content of this section is made accordingly to the requirements elicitation document that might have been done during the project but also adapted coherently in order to be an abstract introduction to the Messir analysis.

2.1 Domain Stakeholders

All stakeholders of the system are detailed in this section. After a brief description of a stakeholder, its objectives are first stated. Thereafter, the responsibilities of the stakeholder are detailed which help to achieve the stakeholder objectives to a certain degree. While the objectives characterize the general problems addressed by the *iCrash* system, the responsibilities describe concrete actions that are expected from a stakeholder. Some of these responsibilities can be traced looking at the use case described in Section B.1,and hence must be supported by the *iCrash* system. All stakeholders listed in this section have an interest in the system or are affected by the system in some way, but only a subset of the stakeholders are directly involved in the use cases described. Let us remind that use case diagrams or descriptions are not Messir analysis phase mandatory outputs. They are proposed as informal means to help understanding the semantics of the system specification made of the mandatory analysis models, which provide a complete executable specification.

2.1.1 Communication Company

A Communication Company is a company that has the capacity to ensure communication of information between its customers and the *iCrash* system. The objectives of a Communication Company are:

- to be able to deliver any SMS sent by any human to the *iCrash* 's phone number.
- to be able to transmit SMS messages from the ABC company that owns the *iCrash* system to any human having an SMS compatible device accessible using a phone number.

In order to achieve these objectives, the responsibilities of a Communication Company are:

- ensure confidentiality and integrity of the information sent by a human to the *iCrash* system or from the system to a human.
- to be always available and reliable.

2.1.2 Humans

A human is any person who considers himself related to a car crash either as a witness, a victim or an anonymous person. The objectives of a human are:

- inform the ***iCrash*** system about the crisis situation he detected.
- be sure that the ABC company has been informed about the situation.
- to be informed about the situation of the crisis he his related to as a victim or witness.

In order to achieve these objectives, the responsibilities of a human are:

- to provide has much details as possible concerning the crisis to the ABC company.
- to declare a crisis only if the crisis is real.
- to have access to the SMS compatible communication device he used to communication with the ***iCrash*** system.

2.1.3 Coordinators

A coordinator is a employee of the ABC company being responsible of handling one or several crisis. The objectives of a coordinator are:

- to securely monitor the existing alerts and crisis.
- to securely manage alerts and crisis until their termination.

In order to achieve these objectives, the responsibilities of a coordinator are:

- to be capable to determine how an alert received should be considered.
- to be available to react to requests to handle alerts and crisis.
- to be autonomous in handling crisis and to report on its handling.
- to be able to decide when a crisis or an alert can be closed.
- to know its system identification information for secure usage of the system.

2.1.4 Hospitals

A hospital is an employee of the Hospital being responsible of monitoring several crises. The objectives of a hospital is to securely monitor the existing crises, victims and their injuries.

In order to achieve these objectives, the responsibilities of a hospital is to know its system identification information for secure usage of the system.

2.1.5 Corrupters

Corrupter is an external fatal to data factor(earthquake, radiation, hacker or other). The objective of a corrupter is alert data corruption.

2.1.6 Administrator

An administrator is a employee of the ABC company being responsible of administrating the ***iCrash*** system. The objectives of an administrator are:

- to add or delete coordinator actors from the system and its environment.

In order to achieve these objectives, the responsibilities of a coordinator are:

- know the company employees that can be coordinators and that have access to the system.
- to know its system identification information for secure usage of the system.
- to know the security policy of the ABC company.
- to communicate the coordinators their identification information for secure system usage.

2.1.7 Creator

Any system has a `Creator` stakeholder which is a technician who is installing the ***iCrash*** system on the targeted deployment infrastructure.

The objectives of a `Creator` are:

- to install the ***iCrash*** system
- to define the values for the initial system's state
- to define the values for the initial system's environment
- to ensure the integration of the ***iCrash*** system with its initial environment

In order to achieve these objectives, the responsibilities of a `Creator` are:

- provide the necessary data to the ***iCrash*** system for its initialization.

2.1.8 Activator

An `activator` is a logical representation of the active part the ***iCrash*** system. It represents an implicit stakeholder belonging to the system's environment that interacts with the ***iCrash*** system autonomously without the need of a external entity. It is usually used for representing time triggered functionalities.

The objectives of a `activator` are:

- to communicate the current time to the system
- to notify the administrator that some crisis are still pending for a too long time.

In order to achieve these objectives, the responsibilities of a `activator` are:

- to know the current universal time
- to send the messages to the system according to the time constraints specifically defined for it.

2.2 System's Actors

The objective of this section is not to provide the full requirement elicitation document in this section but to reuse a part of this document to provide a informal introduction to the Messir specification of the system under development. The use case model is made of a use case diagrams modelling abstractly and informally the actors and their use cases together with a set of use cases descriptions. In addition, those diagrams and description tables are adapted to the Messir specification since actor and messages names together with parameters are partly adapted to be consistent with the specification identifiers (see [1] for more details).

Among all the stakeholders presented in the previous section, we can determine five types of direct actors[1]:

- `actComCompany`: for the Communication Company stakeholder.
- `actAdministrator`: for the Administrator stakeholder.
- `actCoordinator`: for the Coordinators stakeholders.
- `actHospital`: for the Hospitals stakeholders.
- `actCorrupterr`: for the Corrupters stakeholders.
- `actActivator`: for the Activator stakeholder.
- `actMsrCreator`: for the Creator stakeholder.

In addition to those system actors, we can add five other types of actors related to the system's ones. Those five actors are grouped into two categories:

- *Indirect actors*
  - *Witness*: for any human that is a witness of a car crash
  - *Victim*: for any human that is a victim of a car crash
  - *Anonymous*: for any human that want to inform about a car crash while staying anonymous.
- *Abstract actors*
  - `actHuman`: represent abstractly any kind of human being actor wanting to communicate with the ABC system in the context of a car crash.
  - `actAuthenticated`: for the logical Activator stakeholder.

2.3 Use Cases Model

This section contains the use cases elicited during the requirements elicitation phase. The use cases are textually described as suggested by the Messir method and inspired by the standard Cokburn template [2].

[1] The naming conventions in Messir propose to start each type name by lowercase letters indicating the meta model type used (i.e. act for actors, ct for class type, ... In addition to ease the reading it makes the translational semantics into Prolog code more straightforward.

2.3.1 Use Cases

2.3.1.1 summary-suDeployAndRun

The goal is to install the iCrash system on its infrastructure and to exploit its capacities related to the secure administration and efficient handling of car crash situations depending on alerts received.

Use-Case Description	
Name	suDeployAndRun
Scope	system
Level	summary
Primary actor(s)	
1	`actAdministrator[active]`
Secondary actor(s)	
1	`actMsrCreator[active]`
2	`actCoordinator[active, multiple]`
3	`actHospital[active, multiple]`
4	`actActivator[proactive]`
5	`actComCompany[active]`
6	`actCorrupter[active]`
Goal(s) description	
The goal is to install the iCrash system on its infrastructure and to exploit its capacities related to the secure administration and efficient handling of car crash situations depending on alerts received.	
Reuse	
1	`oeCreateSystemAndEnvironment [1..1]`
2	`ugAdministrateTheSystem [1..*]`
3	`suGlobalCrisisHandling [1..*]`
4	`suHospitalCrisisInteraction [1..*]`
5	`oeSetClock [1..*]`
6	`oeSollicitateCrisisHandling [0..*]`
7	`oeAlert [1..*]`
8	`oeCorruptAlert [0..*]`
Protocol condition(s)	
1	the iCrash system has never been deployed and used
Pre-condition(s)	
1	none
Main post-condition(s)	
1	the iCrash system has been created and has handled the crisis situations for which it received alerts through the communication company.
Main Steps	
a	the actor `actMsrCreator` executes the `oeCreateSystemAndEnvironment` use case
b	the actor `actAdministrator` executes the `ugAdministrateTheSystem` use case
c	the actor `actComCompany` executes the `oeAlert` use case
d	the actor `actActivator` executes the `oeSetClock` use case
e	the actor `actActivator` executes the `oeSollicitateCrisisHandling` use case
f	the actor `actCoordinator` executes the `suGlobalCrisisHandling` use case
g	the actor `actHospital` executes the `suHospitalCrisisInteraction` use case
h	the actor `actCorrupter` executes the `oeCorruptAlert` use case
Steps Ordering Constraints	
1	step (a) must be always the first step.
2	step (f) can be executed by different actCoordinator actors.
3	step (g) can be executed by different actHospital actors.
4	if (e) then previously (d).
5	if (h) then previously (c).

Figure 2.1 shows the use case diagram for the suDeployAndRun summary use case

Figure 2.1: suDeployAndRun summary use case

2.3.1.2 summary-suGlobalCrisisHandling

the actCoordinator's goal is to monitor the alerts received and the corresponding crisis in order to act as necessary to handle the crisis.

USE-CASE DESCRIPTION	
Name	suGlobalCrisisHandling
Scope	system
Level	summary
Primary actor(s)	

continues in next page . . .

. . . Use-Case Description table continuation

1	`actCoordinator[active]`
Goal(s) description	
the actCoordinator's goal is to monitor the alerts received and the corresponding crisis in order to act as necessary to handle the crisis.	
Reuse	
1	`ugSecurelyUseSystem [1..*]`
2	`ugMonitor [1..*]`
3	`ugManageCrisis [1..*]`
Protocol condition(s)	
1	the iCrash system has been deployed
2	the coordinator actor involded in the use case has been declared by the actor actAdministrator
Pre-condition(s)	
1	none
Main post-condition(s)	
1	modifications have been made by the coordinator on existing alerts or crisis OR the coordinator requested an updated status on existing alerts or crisis.
Main Steps	
a	the actor `actCoordinator` executes the `ugSecurelyUseSystem` use case
b	the actor `actCoordinator` executes the `ugMonitor` use case
c	the actor `actCoordinator` executes the `ugManageCrisis` use case
Steps Ordering Constraints	
1	steps (a) (b) and (c) executions are interleaved (steps (b) and (c) have their protocol constrained by steps of (a)).
2	steps (a) (b) and (c) can be executed multiple times.

Figure 2.2 shows the use case diagram for the suGlobalCrisisHandling user goal use case

2.3.1.3 summary-suHospitalCrisisInteraction

the actHospital's goal is to monitor the crisis in order to act as necessary to handle the victims.

USE-CASE DESCRIPTION	
Name	suHospitalCrisisInteraction
Scope	system
Level	summary
Primary actor(s)	
1	`actHospital[active]`
Goal(s) description	
the actHospital's goal is to monitor the crisis in order to act as necessary to handle the victims.	
Reuse	
1	`ugSecurelyUseSystem [1..*]`
2	`ugHospitalMonitor [1..*]`
Protocol condition(s)	
1	the iCrash system has been deployed

continues in next page . . .

. . . Use-Case Description table continuation

2	the coordinator actor involved in the use case has been declared by the actor actAdministrator
Pre-condition(s)	
1	none
Main post-condition(s)	
1	modifications have been made by the coordinator on existing alerts or crisis OR the coordinator requested an updated status on existing alerts or crisis.
Main Steps	
a	the actor `actHospital` executes the `ugSecurelyUseSystem` use case
b	the actor `actHospital` executes the `ugHospitalMonitor` use case
Steps Ordering Constraints	
1	steps (a) and (b) executions are interleaved (step (b) have their protocol constrained by steps of (a)).
2	steps (a) and (b) can be executed multiple times.

2.3.1.4 usergoal-ugAdministrateTheSystem

the actAdministrator's goal is to follow an identification procedure to be allowed to add or delete the necessary crisis coordinators that will be granted the responsibility to handle alerts and crisis.

Use-Case Description	
Name	ugAdministrateTheSystem
Scope	system
Level	usergoal
Primary actor(s)	
1	`actAdministrator[active]`
Goal(s) description	
the actAdministrator's goal is to follow an identification procedure to be allowed to add or delete the necessary crisis coordinators that will be granted the responsibility to handle alerts and crisis.	
Reuse	
1	`ugSecurelyUseSystem [1..*]`
2	`oeAddCoordinator [1..*]`
3	`oeDeleteCoordinator [0..*]`
4	`oeAddHospital [1..*]`
5	`oeDeleteHospital [0..*]`
Protocol condition(s)	
1	the iCrash system has been deployed
Pre-condition(s)	
1	none
Main post-condition(s)	
1	modifications have been made to the system and its environment concerning existing or new coordinators.
Main Steps	
a	the actor `actAdministrator` executes the `ugSecurelyUseSystem` use case
b	the actor `actAdministrator` executes the `oeAddCoordinator` use case
c	the actor `actAdministrator` executes the `oeDeleteCoordinator` use case
d	the actor `actAdministrator` executes the `oeAddHospital` use case

continues in next page . . .

. . . Use-Case Description table continuation

e	the actor `actAdministrator` executes the `oeDeleteHospital` use case
Steps Ordering Constraints	
1	steps (a) (b) and (c) executions are interleaved (steps (b) and (c) have their protocol constrained by steps of (a) as well as steps (d) and (e)).
2	steps (a) (b) (c) (d) and (e) can be executed multiple times.

Figure 2.3 shows the use case diagram for the ugAdministrateTheSystem user goal use case

2.3.1.5 usergoal-ugHospitalMonitor

the actHospital's goal is to get the detailed list of existing crisis, victims and injuries.

Use-Case Description	
Name	ugHospitalMonitor
Scope	system
Level	usergoal
Primary actor(s)	
1	`actHospital[active]`
Goal(s) description	
the actHospital's goal is to get the detailed list of existing crisis, victims and injuries.	
Reuse	
1	`oeGetHospitalCrisisSet [0..*]`
2	`oeGetHospitalVictimSet [0..*]`
3	`oeGetHospitalInjurySet [0..*]`
Protocol condition(s)	
1	the iCrash system has been deployed
Pre-condition(s)	
1	none
Main post-condition(s)	
1	none
Main Steps	
a	the actor `actHospital` executes the `oeGetHospitalVictimSet` use case
b	the actor `actHospital` executes the `oeGetHospitalInjurySet` use case
c	the actor `actHospital` executes the `oeGetHospitalCrisisSet` use case

2.3.1.6 usergoal-ugManageCrisis

The goal is to do an action that makes the handling of a crisis or an alert progress.

Use-Case Description	
Name	ugManageCrisis
Scope	system
Level	usergoal
Primary actor(s)	
1	`actCoordinator[active]`
Goal(s) description	

continues in next page . . .

. . . Use-Case Description table continuation

The goal is to do an action that makes the handling of a crisis or an alert progress.	
Reuse	
1	oeValidateAlert [0..*]
2	oeSetCrisisStatus [0..*]
3	oeSetCrisisHandler [0..*]
4	oeReportOnCrisis [0..*]
5	oeCloseCrisis [0..*]
6	oeInvalidateAlert [0..*]
7	oeAddVictim [0..*]
8	oeDeleteVictim [0..*]
9	oeAddInjury [0..*]
10	oeDeleteInjury [0..*]
Protocol condition(s)	
1	the iCrash system has been deployed
Pre-condition(s)	
1	none
Main post-condition(s)	
1	there exist one alert or one crisis whose related information has been changed.
Main Steps	
a	the actor actCoordinator executes the oeValidateAlert use case
b	the actor actCoordinator executes the oeSetCrisisStatus use case
c	the actor actCoordinator executes the oeSetCrisisHandler use case
d	the actor actCoordinator executes the oeReportOnCrisis use case
e	the actor actCoordinator executes the oeCloseCrisis use case
f	the actor actCoordinator executes the oeInvalidateAlert use case
g	the actor actCoordinator executes the oeAddVictim use case
h	the actor actCoordinator executes the oeDeleteVictim use case
i	the actor actCoordinator executes the oeAddInjury use case
j	the actor actCoordinator executes the oeDeleteInjury use case
Steps Ordering Constraints	
1	managing a crisis is doing one of the indicated use cases.

Figure 2.4 shows the use case diagram for the ugManageCrisis user goal use case

2.3.1.7 usergoal-ugMonitor

the actCoordinator's goal is to get the detailed list of existing crisis, alerts, victims and injuries to decide on next actions to undertake.

Use-Case Description	
Name	ugMonitor
Scope	system
Level	usergoal
Primary actor(s)	
1	actCoordinator[active]
Goal(s) description	

continues in next page . . .

. . . Use-Case Description table continuation

the actCoordinator's goal is to get the detailed list of existing crisis, alerts, victims and injuries to decide on next actions to undertake.	
Reuse	
1	`oeGetCrisisSet [0..*]`
2	`oeGetAlertsSet [0..*]`
3	`oeGetVictimSet [0..*]`
4	`oeGetInjurySet [0..*]`
Protocol condition(s)	
1	the iCrash system has been deployed
Pre-condition(s)	
1	none
Main post-condition(s)	
1	none
Main Steps	
a	the actor `actCoordinator` executes the `oeGetAlertsSet` use case
b	the actor `actCoordinator` executes the `oeGetCrisisSet` use case
c	the actor `actCoordinator` executes the `oeGetVictimSet` use case
d	the actor `actCoordinator` executes the `oeGetInjurySet` use case

Figure 2.5 shows the use case diagram for the ugMonitor user goal use case

2.3.1.8 usergoal-ugSecurelyUseSystem

the actAdministrator's goal is to follow an identification procedure to be allowed to add or delete the necessary crisis coordinators that will be granted the responsibility to handle alerts and crisis.

USE-CASE DESCRIPTION	
Name	ugSecurelyUseSystem
Scope	system
Level	usergoal
Primary actor(s)	
1	`actAuthenticated[active]`
Goal(s) description	
the actAdministrator's goal is to follow an identification procedure to be allowed to add or delete the necessary crisis coordinators that will be granted the responsibility to handle alerts and crisis.	
Reuse	
1	`oeLogin [1..1]`
2	`oeLogout [1..1]`
Protocol condition(s)	
1	the iCrash system has been deployed
Pre-condition(s)	
1	none
Main post-condition(s)	
1	the actAuthenticated is known by the system not to be logged.
Main Steps	
a	the actor `actAuthenticated` executes the `oeLogin` use case

continues in next page . . .

. . . Use-Case Description table continuation

b	the actor `actAuthenticated` executes the `oeLogout` use case
Steps Ordering Constraints	
1	step (a) must always precede step (b).

Figure 2.6 shows the use case diagram for the ugSecurelyUseSystem user goal use case

2.3.1.9 subfunction-oeAddInjury

goal is to add a new injury in the system's post state.

USE-CASE DESCRIPTION	
Name	oeAddInjury
Scope	system
Level	subfunction
Parameters	
AdtVictimID: dtVictimID 1	
Primary actor(s)	
1	`actCoordinator[active]`
Goal(s) description	
goal is to add a new injury in the system's post state.	
Protocol condition(s)	
1	
Pre-condition(s)	
1	
Main post-condition(s)	
1	
Additional Information	
none	

2.3.1.10 subfunction-oeAddVictim

goal is to add a new victim in the system's post state.

USE-CASE DESCRIPTION	
Name	oeAddVictim
Scope	system
Level	subfunction
Parameters	
AdtCrisisID: dtCrisisID 1	
Primary actor(s)	
1	`actCoordinator[active]`
Goal(s) description	
goal is to add a new victim in the system's post state.	
Protocol condition(s)	

continues in next page . . .

. . . Use-Case Description table continuation

1
Pre-condition(s)
1
Main post-condition(s)
1
Additional Information
none

2.3.1.11 subfunction-oeCorruptAlert

used to corrupt a specific alert.

Use-Case Description	
Name	oeCorruptAlert
Scope	system
Level	subfunction
Parameters	
ActAlert: ctAlert 1	
Primary actor(s)	
1	`actCorrupter[active]`
Goal(s) description	
used to corrupt a specific alert.	
Protocol condition(s)	
1	
Pre-condition(s)	
1	
Main post-condition(s)	
1	
Additional Information	
none	

2.3.1.12 subfunction-oeDeleteHospital

goal is to delete an existing hospital in the system's post state and environment's post state.

Use-Case Description	
Name	oeDeleteHospital
Scope	system
Level	subfunction
Parameters	
AdtHospitalID: dtHospitalID 1	
Primary actor(s)	
1	`actAdministrator[active]`
Goal(s) description	

continues in next page . . .

. . . Use-Case Description table continuation

goal is to delete an existing hospital in the system's post state and environment's post state.
Protocol condition(s)
1
Pre-condition(s)
1
Main post-condition(s)
1
Additional Information
none

Figure 2.7

2.3.1.13 subfunction-oeDeleteInjury

goal is to delete an existing injury in the system's post state.

USE-CASE DESCRIPTION	
Name	oeDeleteInjury
Scope	system
Level	subfunction
Parameters	
AdtInjuryID: dtInjuryID 1	
Primary actor(s)	
1	`actCoordinator[active]`
Goal(s) description	
goal is to delete an existing injury in the system's post state.	
Protocol condition(s)	
1	
Pre-condition(s)	
1	
Main post-condition(s)	
1	
Additional Information	
none	

2.3.1.14 subfunction-oeDeleteVictim

goal is to delete an existing victim in the system's post state.

USE-CASE DESCRIPTION	
Name	oeDeleteVictim
Scope	system
Level	subfunction
Parameters	

continues in next page . . .

. . . Use-Case Description table continuation

AdtVictimID: dtVictimID 1
Primary actor(s)
1 `actCoordinator[active]`
Goal(s) description
goal is to delete an existing victim in the system's post state.
Protocol condition(s)
1
Pre-condition(s)
1
Main post-condition(s)
1
Additional Information
none

2.3.1.15 subfunction-oeGetHospitalCrisisSet

goal is to request all the ctCrisis instances having a specific status.

USE-CASE DESCRIPTION	
Name	oeGetHospitalCrisisSet
Scope	system
Level	subfunction
Parameters	
AetCrisisStatus: etCrisisStatus 1	
Primary actor(s)	
1 `actHospital[active]`	
Goal(s) description	
goal is to request all the ctCrisis instances having a specific status.	
Protocol condition(s)	
1	
Pre-condition(s)	
1	
Main post-condition(s)	
1	
Additional Information	
none	

2.3.1.16 subfunction-oeGetHospitalInjurySet

goal is to request all the ctInjury instances related to given victim.

USE-CASE DESCRIPTION	
Name	oeGetHospitalInjurySet
Scope	system

continues in next page . . .

. . . Use-Case Description table continuation

Level	subfunction
Parameters	
AdtVictimID: dtVictimID 1	
Primary actor(s)	
1	`actHospital[active]`
Goal(s) description	
goal is to request all the ctInjury instances related to given victim.	
Protocol condition(s)	
1	
Pre-condition(s)	
1	
Main post-condition(s)	
1	
Additional Information	
none	

2.3.1.17 subfunction-oeGetHospitalVictimSet

goal is to request all the ctVictim instances related to given crisis.

USE-CASE DESCRIPTION	
Name	oeGetHospitalVictimSet
Scope	system
Level	subfunction
Parameters	
AdtCrisisID: dtCrisisID 1	
Primary actor(s)	
1	`actHospital[active]`
Goal(s) description	
goal is to request all the ctVictim instances related to given crisis.	
Protocol condition(s)	
1	
Pre-condition(s)	
1	
Main post-condition(s)	
1	
Additional Information	
none	

2.3.1.18 subfunction-oeGetInjurySet

goal is to request all the ctInjury instances related to given victim.

continues in next page . . .

. . . Use-Case Description table continuation

USE-CASE DESCRIPTION	
Name	oeGetInjurySet
Scope	system
Level	subfunction
Parameters	
AdtVictimID: dtVictimID 1	
Primary actor(s)	
1	`actCoordinator[active]`
Goal(s) description	
goal is to request all the ctInjury instances related to given victim.	
Protocol condition(s)	
1	
Pre-condition(s)	
1	
Main post-condition(s)	
1	
Additional Information	
none	

2.3.1.19 subfunction-oeGetVictimSet

goal is to request all the ctVictim instances related to given crisis.

USE-CASE DESCRIPTION	
Name	oeGetVictimSet
Scope	system
Level	subfunction
Parameters	
AdtCrisisID: dtCrisisID 1	
Primary actor(s)	
1	`actCoordinator[active]`
Goal(s) description	
goal is to request all the ctVictim instances related to given crisis.	
Protocol condition(s)	
1	
Pre-condition(s)	
1	
Main post-condition(s)	
1	
Additional Information	
none	

Figure 2.2: suGlobalCrisisHandling user goal use case

Figure 2.3: ugAdministrateTheSystem user goal use case

Figure 2.4: ugManageCrisis user goal use case

Figure 2.5: ugMonitor user goal use case

Figure 2.6: ugSecurelyUseSystem user goal use case

Figure 2.7:

2.3.1.20 subfunction-oeSetCrisisHandler

goal is to declare himself as been the handler of a crisis having the specified id.

Use-Case Description	
Name	oeSetCrisisHandler
Scope	system
Level	subfunction
Parameters	
AdtCrisisID: dtCrisisID 1	
Primary actor(s)	
1	`actCoordinator[active]`
Secondary actor(s)	
1	`actCoordinator[passive]`
2	`actComCompany[passive, multiple]`
Goal(s) description	
goal is to declare himself as been the handler of a crisis having the specified id.	
Protocol condition(s)	
1	
Pre-condition(s)	
1	
Main post-condition(s)	
1	
Additional Information	
none	

Figure 2.8 shows the use case diagram for the oeSetCrisisHandler subfunction use case

Figure 2.8: oeSetCrisisHandler subfunction use case

2.3.1.21 subfunction-oeSollicitateCrisisHandling

the actActivator's goal is to decrease the number of unhandled crisis.

Use-Case Description	
Name	oeSollicitateCrisisHandling
Scope	system
Level	subfunction
Primary actor(s)	
1	`actActivator[proactive]`
Secondary actor(s)	
1	`actCoordinator[passive, multiple]`
2	`actAdministrator[passive]`
Goal(s) description	
the actActivator's goal is to decrease the number of unhandled crisis.	
Protocol condition(s)	
1	the iCrash system has been deployed.
2	there exist some crisis still pending and for which no solicitation has been sent to the administrator and the coordinators for more than a predefined maximum delay.
Pre-condition(s)	
1	none
Main post-condition(s)	
1	a simple text message ieMessage('There are alerts not treated since more than the defined delay. Please REACT !') is sent to the system administrator and to all the coordinators of the environment for each crisis that is known to be not handled and for which no solicitation has been sent to the administrator and the coordinators for more than a predefined maximum delay.')
2	the reminder period for the concerned crisis is initialized.

Figure 2.9 shows the use case diagram for the oeSollicitateCrisisHandling subfunction use case

Figure 2.9: oeSollicitateCrisisHandling subfunction use case

2.3.2 Use Case Instance(s)

2.3.2.1 Use-Case Instance - uciSimpleAndCompletePart01:suDeployAndRun

First part of a use case instance for the summary use case `suDeployAndRun` illustrating a simple and complete interaction scenario primarily handled by an administrator in a concrete situation.

SUMMARY USE-CASE INSTANCE	
Instantiated Use Case	
suDeployAndRun	
Instance ID	
uciSimpleAndCompletePart01	
Remarks	
a	shows the system initilization and the first administrative tasks by the administrator.
b	The unique and always existing `actMsrCreator` actor instance (named here `theCreator`) requests the initialization of the system and its environment (made of one administrator identified here by `bill`), one activator actor (identified by `theClock`) and indicating that the number of communication company actor instances for the system's environment is 4 (one of them is identified here by `tango`)
c	the administrator logs in to initialize a coordinator
d	an alert is received. Time is goind one without having the coordinator handling the alert which let's the proactive actor trigger the automatic sollicitation of crisis handling.
e	this first part stops before the coordinator logs in the system.

Figure 2.10 shows the sequence diagram representing the first part of a simple and complete use case instance for the summary use case `suDeployAndRun`.

2.3.2.2 Use-Case Instance - uciSimpleAndCompletePart02:suDeployAndRun

Second part of a simple and complete use case instance for the summary use case `suDeployAndRun` illustrating a simple and complete interaction scenario primarily handled by an administrator in a concrete situation.

SUMMARY USE-CASE INSTANCE	
Instantiated Use Case	
suDeployAndRun	
Instance ID	
uciSimpleAndCompletePart02	
Remarks	
a	starts when the coordinator logs in the system until the full handling of all the existing crisis.
b	shows an instantiated case of handling of a crisis by a coordinator until its closure after reporting.

Figure 2.11 shows the sequence diagram representing the second part of a simple and complete use case instance for the summary use case `suDeployAndRun`.

2.3.2.3 Use-Case Instance - uciugSecurelyUseSystem:ugSecurelyUseSystem

Figure 2.10: uci-suDeployAndRun-uciSimpleAndComplete-Part01

Figure 2.11: uci-suDeployAndRun-uciSimpleAndComplete-Part02 use case instance sequence diagram

USERGOAL USE-CASE INSTANCE
Instantiated Use Case ugSecurelyUseSystem
Instance ID uciugSecurelyUseSystem

Figure 2.12

Figure 2.12:

Chapter 3

Environment Model

We provide below the view(s) defined for the Messir environment model (cf. [1]) of the system.

3.1 Local view 01

Figure 3.1 shows the local view giving the second part of the environment model of the system in term of its state class, actors with their input and output interfaces and all related associations.

3.2 Local view 02

Figure 3.2 shows the local view giving the second part the environment model of the system in term of its state class, actors with their input and output interfaces and all related associations.

3.3 Local view 03

Figure 3.3 shows the local view for the administrator actor and interfaces

3.4 Local view 04

Figure 3.4 shows the local view for the coordinator actor and interfaces

3.5 Local view 05

Figure 3.5 shows the local view for the authenticated actor and interfaces

3.6 Local view 09

Figure 3.6

Figure 3.1: Environment Model - Local View 01. environment model local view - Part 1.

Figure 3.2: Environment Model - Local View 02. environment model local view - Part 2.

Figure 3.3: Environment Model - Local View 03. administrator actor environment model view.

Figure 3.4: Environment Model - Local View 04. coordinator actor environment model view.

Figure 3.5: Environment Model - Local View 05. authenticated actor environment model local view.

Figure 3.6: Environment Model - Local View 09. .

3.7 Global view 01

Figure 3.7 shows a global view for all actors with their relationships with ctState

Figure 3.7: Environment Model - Global View 01. em-gv-01 environment model global view.

3.8 Actors and Interfaces Descriptions

We provide for the given views the description of the actors together with their associated input and output interface descriptions.

3.8.1 `actActivator` Actor

ACTOR	
actActivator	
represents a logical actor for time automatic message sending based on system's or environment status.	
OutputInterfaces	
OUT 1	**`[proactive] oeSollicitateCrisisHandling():ptBoolean`** used to avoid crisis to stay too long in an not handled status.
OUT 2	**`[proactive] oeSetClock(AcurrentClock:dtDateAndTime):ptBoolean`** used to update the system's time

3.8.2 `actAdministrator` Actor

ACTOR	
actAdministrator	
represents an actor responsible of administration tasks for the *iCrash* system.	
Extends	
icrash.environment.actAuthenticated	
OutputInterfaces	
OUT 1	**`oeAddCoordinator(AdtCoordinatorID:dtCoordinatorID, AdtLogin:dtLogin, AdtPassword:dtPassword):ptBoolean`** sent to add a new coordinator in the system's post state and environment's post state.
OUT 2	**`oeDeleteCoordinator(AdtCoordinatorID:dtCoordinatorID):ptBoolean`** sent to delete an existing coordinator in the system's post state and environment's post state.
InputInterfaces	
IN 1	**`ieCoordinatorAdded():ptBoolean`** its reception confirms the creation of the requested coodinator.
IN 2	**`ieCoordinatorDeleted():ptBoolean`** its reception confirms the deletion of the requested coodinator.

3.8.3 `actAuthenticated` Actor

ACTOR	
actAuthenticated	
abstract actor providing reusable input and output interfaces for actors that need to authenticate themselves.	
OutputInterfaces	
OUT 1	**`oeLogin(AdtLogin:dtLogin, AdtPassword:dtPassword):ptBoolean`** sent to request authorization to request access secured system operations.
OUT 2	**`oeLogout():ptBoolean`** sent to end the secured access to specific system operations.
InputInterfaces	
IN 1	**`ieMessage(AMessage:ptString):ptBoolean`** allows for receiving general textual messages.

3.8.4 `actComCompany` Actor

ACTOR	
actComCompany	
represents the communication company stakeholder ensuring the input/ouput of textual messages with humans having communicaiton devices.	
OutputInterfaces	
OUT 1	**`oeAlert(AetHumanKind:etHumanKind, AdtDate:dtDate, AdtTime:dtTime, AdtPhoneNumber:dtPhoneNumber, AdtGPSLocation:dtGPSLocation, AdtComment:dtComment):ptBoolean`** sent to alert of a potential crisis situation.
InputInterfaces	
IN 1	**`ieSmsSend(AdtPhoneNumber:dtPhoneNumber, AdtSMS:dtSMS):ptBoolean`**

continues in next page . . .

. . . Actor table continuation

	allows for receiving textual messages to be dispatched to the communication company customers having the provided phone number.

3.8.5 `actCoordinator` Actor

ACTOR	
actCoordinator	
represents actor responsible of handling one or several crisis for the *iCrash* system.	
Extends	
icrash.environment.actAuthenticated	
OutputInterfaces	
OUT 1	**`oeInvalidateAlert(AdtAlertID:dtAlertID):ptBoolean`** sent to indicate that an alert should be considered as closed.
OUT 2	**`oeCloseCrisis(AdtCrisisID:dtCrisisID):ptBoolean`** sent to indicate that a crisis should be considered as closed.
OUT 3	**`oeGetAlertsSet(AetAlertStatus:etAlertStatus):ptBoolean`** sent to request all the ctAlert instances having a specific status.
OUT 4	**`oeGetCrisisSet(AetCrisisStatus:etCrisisStatus):ptBoolean`** sent to request all the ctCrisis instances having a specific status.
OUT 5	**`oeSetCrisisHandler(AdtCrisisID:dtCrisisID):ptBoolean`** sent to declare himself as been the handler of a crisis having the specified id.
OUT 6	**`oeReportOnCrisis(AdtCrisisID:dtCrisisID, AdtComment:dtComment):ptBoo`** sent to update the textual information available for a specific handled crisis.
OUT 7	**`oeSetCrisisStatus(AdtCrisisID:dtCrisisID, AetCrisisStatus:etCrisisSta`** sent to define the handling status of a specific crisis.
OUT 8	**`oeSetCrisisType(AdtCrisisID:dtCrisisID, AetCrisisType:etCrisisType):p`** sent to define the gravity type of a specific crisis.
OUT 9	**`oeValidateAlert(AdtAlertID:dtAlertID):ptBoolean`** sent to indicate that a specific alert is not a fake.
InputInterfaces	
IN 1	**`ieSendAnAlert(ActAlert:ctAlert):ptBoolean`** allows for receiving a requested ctAlert instance.
IN 2	**`ieSendACrisis(ActCrisis:ctCrisis):ptBoolean`** allows for receiving a requested ctCrisis instance.

3.8.6 `actCorrupter` Actor

ACTOR	
actCorrupter	
OutputInterfaces	
OUT 1	**`[proactive] oeCorruptAlert(ActAlert:ctAlert):ptBoolean`**

3.8.7 `actHospital` Actor

ACTOR	
actHospital	
represents actor responsible of observing crisis with their victims and injuries for the ***iCrash*** system.	
Extends	
icrash.environment.actAuthenticated	
OutputInterfaces	
OUT 1	**`oeGetHospitalCrisisSet(AetCrisisStatus:etCrisisStatus):ptBoolean`** sent to request all the ctCrisis instances having a specific status.
OUT 2	**`oeGetHospitalVictimSet(AdtCrisisID:dtCrisisID):ptBoolean`** sent to request all the ctVictim instances for given Crisis.
OUT 3	**`oeGetHospitalInjurySet(AdtVictimID:dtVictimID):ptBoolean`** sent to request all the ctInjury instances for given Victim.
InputInterfaces	
IN 1	**`ieSendACrisisToHospital(ActCrisis:ctCrisis):ptBoolean`** allows for receiving a requested ctCrisis instance.
IN 2	**`ieSendAVictimToHospital(ActVictim:ctVictim):ptBoolean`** allows for receiving a requested ctVictim instance.
IN 3	**`ieSendAInjuryToHospital(ActInjury:ctInjury):ptBoolean`** allows for receiving a requested ctInjury instance.

3.8.8 `actMsrCreator` Actor

ACTOR	
actMsrCreator	
Represents the creator stakeholder in charge of state and environment initialization.	
OutputInterfaces	
OUT 1	**`oeCreateSystemAndEnvironment(AqtyComCompanies:ptInteger):ptBoolean`** sent to request the initialization of the system's class instances and the environment actors instances.

Chapter 4

Concept Model

4.1 PrimaryTypes-Classes

4.1.1 Local view 01

Figure 4.1 shows the local view on all the primary types class types.

4.1.2 Local view 02

Figure 4.2 shows the local view of the ctState primary type class type.

4.1.3 Local view 03

Figure 4.3 shows the local view of the ctAlert primary type class type.

4.1.4 Local view 04

Figure 4.4 shows the local view of the ctCrisis primary type class type.

4.1.5 Global view 01

Figure 4.5 shows the global view on primary types class types showing the association(s) types with the actor classes of the environment model.

4.2 PrimaryTypes-Datatypes

4.2.1 Local view 06

Figure 4.6

Figure 4.1: Concept Model - PrimaryTypes-Classes local view 01. Local view of all the primary types class types .

Figure 4.2: Concept Model - PrimaryTypes-Classes local view 02. local view of the ctState primary type.

Figure 4.3: Concept Model - PrimaryTypes-Classes local view 03. local view of the ctAlert primary type.

Figure 4.4: Concept Model - PrimaryTypes-Classes local view 04. local view of the ctCrisis primary type.

Figure 4.5: Concept Model - PrimaryTypes-Classes global view 01. Primary types class types global view - cm-pt-ct-gv-01 .

Figure 4.6: Concept Model - PrimaryTypes-Datatypes local view 06. .

Figure 4.7: Concept Model - PrimaryTypes-Datatypes local view 08. .

4.2.2 Local view 08

Figure 4.7

4.2.3 Global view 01

Figure 4.8 shows a global view on the *iCrash* primary types datatype types.

Figure 4.8: Concept Model - PrimaryTypes-Datatypes global view 01. global view of primary types datatype types - cm-pt-dt-gv-01 .

4.3 SecondaryTypes-Datatypes

4.3.1 Local view 01

Figure 4.9 shows the local view of the secondary types datatype types.

Figure 4.9: Concept Model - SecondaryTypes-Datatypes local view 01. Local view of the secondary types datatype types.

4.4 Concept Model Types Descriptions

This section provides the textual descriptions of all the types defined in the concept model and that can be part of the graphical views provided.

4.4.1 Primary types - Class types descriptions

The table below is providing comments on the graphical views given for the class types of the primary types. Type logical operations are precisely specified in the operation model.

CLASSES	
ctAdministrator	
used to caracterize internally the entity that is responsible of administrating the *iCrash* system.	
extends	icrash.concepts.primarytypes.classes.ctAuthenticated
operation	**`init(Alogin:dtLogin, Apwd:dtPassword):ptBoolean`** used to initialize the current object as a new instance of the ctAdministrator type.
ctAlert	
Used to model crisis alerts sent by any human having communication capability using communication companies belonging to the system's environment	
attribute	**`comment: dtComment`** a textual description providing unstructured information on the alert.
attribute	**`id: dtAlertID`** the alert unique identification information.
attribute	**`instant: dtDateAndTime`** the date and time at which the alert notification has been sent.
attribute	**`location: dtGPSLocation`** the position of the alert provided by the space-based satellite navigation system used by the human using the communication company to inform the *iCrash* system of a crisis.
attribute	**`status: etAlertStatus`** the alert validation status
operation	**`init(Aid:dtAlertID, Astatus:etAlertStatus, Alocation:dtGPSLocation, Ainstant:dtDateAndTime, Acomment:dtComment):ptBoolean`** used to initialize the current object as a new instance of the ctAlert type.
operation	**`isSentToCoordinator(AactCoordinator:actCoordinator):ptBoolean`** used to provide a given coordinator with current alert information.
ctAuthenticated	

continues in next page . . .

. . . Classes table continuation

used to model system's representation about actors that need to authenticate to access some specific functionalities.	
attribute	**`login: dtLogin`** an identifier for authentication.
attribute	**`pwd: dtPassword`** a key for authentication.
attribute	**`vpIsLogged: ptBoolean`** used to determine the access status.
operation	**`init(Alogin:dtLogin, Apwd:dtPassword):ptBoolean`** used to initialize the current object as a new instance of the ctAuthenticated type.
ctCoordinator used to model system's representation about the actors that have the responsibility to handle alerts and crisis.	
extends	icrash.concepts.primarytypes.classes.ctAuthenticated
attribute	**`id: dtCoordinatorID`** a unique identification information.
operation	**`init(Aid:dtCoordinatorID, Alogin:dtLogin, Apwd:dtPassword):ptBoolean`** used to initialize the current object as a new instance of the ctCoordinator type.
ctCrisis Used to model crisis that are infered from the reception of at least one alert message. Crisis aer entities that are handled by the *iCrash* system.	
attribute	**`comment: dtComment`** a textual description providing unstructured information on the crisis handling.
attribute	**`id: dtCrisisID`** the crisis unique identification information.
attribute	**`instant: dtDateAndTime`** the date and time at which the first related alert notification has been sent.
attribute	**`location: dtGPSLocation`** the position of the crisis equal by the one of the first alert received and associated to the crisis.
attribute	**`status: etCrisisStatus`** the crisis handling status.
attribute	**`type: etCrisisType`** an indication of the gravity of the crisis.
operation	**`handlingDelayPassed():ptBoolean`** used to determine if the crisis stood too longly in a pending status since last reminder.
operation	**`init(Aid:dtCrisisID, Atype:etCrisisType, Astatus:etCrisisStatus, Alocation:dtGPSLocation, Ainstant:dtDateAndTime, Acomment:dtComment):ptBoolean`** used to initialize the current object as a new instance of the ctAlert type.
operation	**`isAllocatedIfPossible():ptBoolean`** used to allocate a crisis to a coordinator if any or to alert the administrator of crisis waiting to be handled.
operation	**`isSentToCoordinator(AactCoordinator:actCoordinator):ptBoolean`** used to provide a given coordinator with current crisis information.
operation	**`maxHandlingDelayPassed():ptBoolean`** used to determine if the crisis stood too longly in a pending status since its creation.

continues in next page . . .

. . . Classes table continuation

ctHospital	
used to model system's representation about the actors that have the responsibility to monitor crisis and related victims with injuries.	
extends	icrash.concepts.primarytypes.classes.ctAuthenticated
attribute	**`id: dtHospitalID`** a unique identification information.
operation	**`init(Aid:dtHospitalID, Alogin:dtLogin, Apwd:dtPassword):ptBoolean`** used to initialize the current object as a new instance of the ctHospital type.
ctHuman	
used to model system's representation about the indirect actors that has alerted of potential crisis.	
attribute	**`id: dtPhoneNumber`** the number of the communication device used to send an alert to ***iCrash*** system.
attribute	**`kind: etHumanKind`** role with respect to the alert notified.
operation	**`init(Aid:dtPhoneNumber, Akind:etHumanKind):ptBoolean`** `init`: used to initialize the current object as a new instance of the ctHuman type.
ctInjury	
Used to model injury victim gets in a crisis	
attribute	**`id: dtInjuryID`** the injury unique identification information.
attribute	**`kind: etInjuryKind`** the type of injury describes body part
operation	**`init(Aid:dtInjuryID, AetInjuryKind:etInjuryKind):ptBoolean`** used to initialize the current object as a new instance of the ctInjury type.
operation	**`isSentToCoordinator(AactCoordinator:actCoordinator):ptBoolean`** used to provide a given coordinator with current injury information.
operation	**`isSentToHospital(AactHospital:actHospital):ptBoolean`** used to provide a given hospital with current injury information.
ctState	
used to model the system. Each system specified using Messir must include a ctState class for which there is only one instance at any state of the abstract machine after creation.	
attribute	**`clock: dtDateAndTime`** used to represent the system local time.
attribute	**`crisisReminderPeriod: dtSecond`** used to define the delay between two reminders after which a reminder must be sent to the administrator and to the known coordinators to encourage them to handle the crisis.
attribute	**`maxCrisisReminderPeriod: dtSecond`** used to define the maximum delay after which the crisis is ramdomly allocated to a coordinator if any or an alert message is sent to the administrator in order to encourage him to add coordinators.
attribute	**`nextValueForAlertID: dtInteger`** `nextValueForAlertID: dtInteger`: used to associate each alert declared with a unique idenitification value.
attribute	**`nextValueForCrisisID: dtInteger`** used to associate each crisis declared with a unique idenitification value.
attribute	**`vpLastReminder: dtDateAndTime`** date and time of the last reminder.

continues in next page . . .

. . . Classes table continuation

attribute	**vpStarted: ptBoolean** used to avoid reacting to an actor message if the system is not started (i.e. oeCreateSystemAndEnvironment not executed).
operation	**init(AnextValueForAlertID:dtInteger, AnextValueForCrisisID:dtInteger, AnextValueForVictimID:dtInteger, AnextValueForInjuryID:dtInteger, Aclock:dtDateAndTime, AcrisisReminderPeriod:dtSecond, AmaxCrisisReminderPeriod:dtSecond, AvpLastReminder:dtDateAndTime, AvpStarted:ptBoolean):ptBoolean** used to initialize the current object as a new instance of the ctState type.
ctVictim Used to model victim of a crisis	
attribute	**id: dtVictimID** the victim unique identification information.
operation	**init(Aid:dtVictimID):ptBoolean** used to initialize the current object as a new instance of the ctVictim type.
operation	**isSentToCoordinator(AactCoordinator:actCoordinator):ptBoolean** used to provide a given coordinator with current victim information.
operation	**isSentToHospital(AactHospital:actHospital):ptBoolean** used to provide a given hospital with current victim information.

4.4.2 Primary types - Datatypes types descriptions

The table below is providing comments on the graphical views given for the datatype types of the primary types.

DATATYPES	
dtAlertID A string used to identify alerts.	
extends	dtString
operation	**is():ptBoolean** used to determine which strings are considered as valid alert identifiers.
dtComment a datatype made of a string value used to receive,store and send textual information about crisis and alerts.	
extends	dtString
operation	**is():ptBoolean** used to determine which strings are considered as valid comments.
dtCoordinatorID A string used to identify coordinators.	
extends	dtString
operation	**is():ptBoolean** used to determine which strings are considered as valid coordinators identifiers.
dtCrisisID A string used to identify crisis.	
extends	dtString
operation	**is():ptBoolean** used to determine which strings are considered as valid crisis identifiers.

continues in next page . . .

. . . Datatypes table continuation

dtGPSLocation	
used to define coordinates of geograpical positions on earth. It is defined a couple made of a latitude and a longitude.	
attribute	**`latitude: dtLatitude`**
	for the latitude part of the coordinate.
attribute	**`longitude: dtLongitude`**
	for the longitude part of the coordinate.
operation	**`is():ptBoolean`**
	used to determine which couples are considered as valid dtGPSLocation values.
operation	**`isNearTo(AGPSLocation:dtGPSLocation):ptBoolean`**
	used to determine if locations are considered enough close to be treated as equivalent in the application domain context.
dtHospitalID	
A string used to identify hospitals.	
extends	dtString
operation	**`is():ptBoolean`**
	used to determine which strings are considered as valid hospitals identifiers.
dtInjuryID	
A string used to identify injury.	
extends	dtString
operation	**`is():ptBoolean`**
	used to determine which strings are considered as valid injury identifiers.
dtLatitude	
used to define a latitude value of a geograpical positions on earth.	
extends	dtReal
operation	**`is():ptBoolean`**
	used to determine which strings are considered as valid dtLatitude.
dtLogin	
a login string used to authentify an *iCrash* user	
extends	dtString
operation	**`is():ptBoolean`**
	used to determine which strings are considered as valid dtLogin.
dtLongitude	
used to define a longitude value of a geograpical positions on earth.	
extends	dtReal
operation	**`is():ptBoolean`**
	used to determine which strings are considered as valid dtLongitude.
dtPassword	
a password string used to authentify an *iCrash* user	
extends	dtString
operation	**`is():ptBoolean`**
	used to determine which strings are considered as valid dtPassword.
dtPhoneNumber	
a string used to store the phone number from the human declaring the crisis or the alert.	
extends	dtString
operation	**`is():ptBoolean`**
	used to determine which strings are considered as valid dtPhoneNumber.

continues in next page . . .

. . . Datatypes table continuation

dtVictimID	
A string used to identify victim.	
extends	dtString
operation	**`is():ptBoolean`** used to determine which strings are considered as valid victim identifiers.

ENUMERATIONS	
etAlertStatus	
this type is used to indicate the different validation status of an alert.	
operation	**`is():ptBoolean`** used to determine which litteral belongs to the enumeration.
etCorruptionKind	
operation	**`is():ptBoolean`**
etCrisisStatus	
this type is used to indicate the different handling status of a crisis.	
operation	**`is():ptBoolean`** used to determine which litteral belongs to the enumeration.
etCrisisType	
this type is used to indicate the different types of a crisis.	
operation	**`is():ptBoolean`** used to determine which litteral belongs to the enumeration.
etHumanKind	
this type is used to indicate the kind of human that informs about a car crash crisis.	
operation	**`is():ptBoolean`** used to determine which litteral belongs to the enumeration.
etInjuryKind	
this type is used to indicate the kind of injury victims gets in a crisis.	
operation	**`is():ptBoolean`** used to determine which litteral belongs to the enumeration.

4.4.3 Primary types - Association types descriptions

The table below is providing comments on the association types of the primary types.

UNDIRECTED ASSOCIATIONS
assctAlertctCrisis a crisis is related to one or more alerts as the alerts judged to concern all the same crisis due to their location. An alert alerts exactly one crisis.
assctAlertctHuman alerts are notified by human through the communication company. We need to keep an internal representation of those human to allow for communication of alert handling.
assctAuthenticatedactAuthenticated mainly used to determine if the login request of an authenticated actor can be granted based on the given credentials and the registered ones.

continues in next page . . .

... Undirected associations table continuation

assctCoordinatoractCoordinator frequent messages must be sent to coordinator especially in relation to crisis they handle.
assctCrisisctCoordinator at any point in time we need to know if a coordinator is handling existing crisis or not.
assctHospitalactHospital frequent messages we don't know anything about.
assctHumanactComCompany in order to communicate with humans who informed about potential crisis, we need to record the communication company to use to send them messages.
assctInjuryVictim at any point in time we need to know the crisis victim belongs to.
assctVictimCrisis at any point in time we need to know which victim has this injury.

4.4.4 Primary types - Aggregation types descriptions

There are no aggregation types for the primary types.

4.4.4.1 Primary types - Composition types descriptions

There are no composition types for the primary types.

4.4.5 Secondary types - Class types descriptions

There are no elements in this category in the system analysed.

4.4.6 Secondary types - Datatypes types descriptions

The table below is providing comments on the graphical views given for the datatype types of the secondary types.

DATATYPES	
dtSMS a datatype made of a string value used to send textual information to human mobile devices.	
attribute	**`value: ptString`** the textual information.
operation	**`is():ptBoolean`** used to determine which strings are considered as valid comments.

4.4.7 Secondary types - Association types descriptions

There are no association types for the secondary types.

4.4.8 Secondary types - Aggregation types descriptions

There are no aggregation types for the secondary types.

4.4.9 Secondary types - Composition types descriptions

There are no composition types for the secondary types.

Chapter 5

Operation Model

This section contains the operation schemes of each operation defined in either an actor, its output interface, in a primary or secondary type (class, datatype or enumeration types). The Messir OCL code listing is joined to the comment table.

5.1 Environment - Out Interface Operation Scheme for actActivator

5.1.1 Operation Model for oeSetClock

The `oeSetClock` operation has the following properties:

OPERATION	
oeSetClock[proactive] An active message used to statically set the date and time information in the system's state.	
Parameters	
1	**AcurrentClock: dtDateAndTime** the date and time to be considered as the actual one.
Return type	
ptBoolean	
Pre-Condition (protocol)	
PreP 1	the system is supposed to be created and initialized and the provided date and time value is greater than the one known by the system.
Pre-Condition (functional)	
PreF 1	none
Post-Condition (functional)	
PostF 1	the ctState instance post-state is updated to have its clock attribute equal to the given date and time.
Post-Condition (protocol)	
PostP 1	none

The listing 5.1 provides the Messir (MCL-oriented) specification of the operation.

```

/* Pre Protocol:*/
preP{let TheSystem: ctState in
```

```
 let AvpStarted: ptBoolean in

/* PreP01 */
 self.rnActor.rnSystem = TheSystem
 and self.rnActor.rnSystem.vpStarted = AvpStarted
 and AvpStarted = true
 and TheSystem.clock.lt(AcurrentClock)}

/* Pre Functional:*/
preF{true}

/* Post Functional:*/
postF{let TheSystem: ctState in
 self.rnActor.rnSystem = TheSystem

/* PostF01 */
 and TheSystem@post.clock = AcurrentClock}

/* Post Protocol:*/
postP{ true}
```

Listing 5.1: Messir (MCL-oriented) specification of the operation *oeSetClock*.

The listing 5.2 provides the Messir (Prolog-oriented) implementation of the operation.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%-------------------------------------------------------
msrop(outactActivator,
   oeSetClock,
   [preProtocol,Self,
    AcurrentClock
    ],
   []):-
/* Pre Protocol:*/
/* PreP01 */
msrVar(ctState,TheSystem),
msrVar(ptBoolean,AvpStarted),

msrNav([Self],[rnActor,rnSystem],[TheSystem]),

msrNav([Self],[rnActor,rnSystem,vpStarted],[AvpStarted]),
AvpStarted = [ptBoolean,true],

msrNav([TheSystem],
    [clock,lt,[AcurrentClock]],
    [[ptBoolean,true]])
.

msrop(outactActivator,
   oeSetClock,
   [preFunctional,Self,
    AcurrentClock
    ],
   []):-
/* Pre Functional:*/
/* PreF01 */
true.

msrop(outactActivator,
   oeSetClock,
   [post,Self,
    AcurrentClock
    ],
```

```
42     []):-
43
44 msrVar(ctState,TheSystem),
45
46/* Post Functional:*/
47
48 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
49
50/* PostF01 */
51 msrNav([TheSystem],
52     [msmAtPost,clock],
53     [AcurrentClock]),
54
55 /* Post Protocol:*/
56/* PostP01 */
57 true
58 .
```

Listing 5.2: Messir (Prolog-oriented) implementation of the operation *oeSetClock*.

5.1.2 Operation Model for oeSollicitateCrisisHandling

The `oeSollicitateCrisisHandling` operation has the following properties:

Operation	
oeSollicitateCrisisHandling[proactive]	
A proactive message (message of a pro-active actor with no parameter triggered automatically if the pre protocol condition is true) used to avoid crisis to stay too long in an not handled status.	
Return type	
ptBoolean	
Pre-Condition (protocol)	
PreP 1	the system is started
PreP 2	there exist some crisis that are in pending status and for which the duration between the current ctState clock information and the last reminder is greater than the crisis reminder period duration.
Pre-Condition (functional)	
PreF 1	none
Post-Condition (functional)	
PostF 1	if there exist coordinators and crisis who stood in a not handled status more than the maximum allowed time then those crisis are randomly allocated to the existing coordinators.
PostF 2	for all other crisis who stood too longly in a not handled status but not more than the maximum delay allowed then a reminder message is sent to the administrator and all coordinator actors of the environment to sollicitate handling of those crisis.
Post-Condition (protocol)	
PostP 1	the value of the last reminder known by the system at post state is the system's clock value.

The listing 5.3 provides the Messir (MCL-oriented) specification of the operation.

```
1
2 /* Pre Protocol:*/
3 preP{let TheSystem: ctState in
4  let AvpStarted: ptBoolean in
5  let ColctCrisisToHandle:
```

```
6      Bag(ctCrisis) in
7
8   self.rnActor.rnSystem = TheSystem
9
10 /* PreP01 */
11  and TheSystem.vpStarted
12
13 /* PreP02 */
14  and TheSystem.rnctCrisis->select(handlingDelayPassed())
15    = ColctCrisisToHandle
16  and ColctCrisisToHandle->size().geq(1)}
17
18 /* Pre Functional:*/
19 preF{true}
20
21 /* Post Functional:*/
22 postF{let TheSystem: ctState in
23  let AMessageForCrisisHandlers: dtComment in
24  let ColctCrisisToAllocateIfPossible:Bag(ctCrisis) in
25
26  self.rnActor.rnSystem = TheSystem
27 /* PostF01 */
28  and TheSystem.rnctCrisis->select(maxHandlingDelayPassed())
29    = ColctCrisisToAllocateIfPossible
30  and ColctCrisisToAllocateIfPossible->forAll(isAllocatedIfPossible())
31
32  /* PostF02 */
33  and TheSystem.rnctCrisis->select(handlingDelayPassed())
34  = ColctCrisisToHandle
35
36  and ColctCrisisToHandle->msrColSubtract(ColctCrisisToAllocateIfPossible)
37    = ColctCrisisToRemind
38
39  and if (ColctCrisisToRemind->size().geq(1))
40    then (AMessageForCrisisHandlers.value
41       ='There are alerts pending since more than the defined delay. Please REACT !'
42       and TheSystem.rnactAdministrator.
43         rnInterfaceIN^ieMessage(AMessageForCrisisHandlers)
44         and TheSystem.rnactCoordinator
45           ->forAll(rnInterfaceIN^ieMessage(AMessageForCrisisHandlers))
46       )
47    else true
48    endif}
49
50 /* Post Protocol:*/
51 postP{ let TheSystem: ctState in
52  let TheClock: dtDateAndTime in
53
54  self.rnActor.rnSystem = TheSystem
55  and TheSystem.clock = TheClock
56  and TheSystem@post.vpLastReminder = TheClock}
```

Listing 5.3: Messir (MCL-oriented) specification of the operation *oeSollicitateCrisisHandling*.

The listing 5.4 provides the Messir (Prolog-oriented) implementation of the operation.

```
1%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5%-------------------------------------------------------
6
7msrop(outactActivator,
8   oeSollicitateCrisisHandling,
9   [preProtocol,Self
10   ],
```

```
11    []):-
12/* Pre Protocol:*/
13 msrVar(ctState,TheSystem),
14 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
15
16 msrVarCol(ctCrisis,_,ColctCrisisToHandle),
17
18/* PreP01 */
19 msrNav([TheSystem],
20    [vpStarted],
21    [[ptBoolean,true]]),
22
23/* PreP02 */
24 msrNav([TheSystem],
25    [rnctCrisis,msrSelect,
26     handlingDelayPassed,[]
27    ],
28    ColctCrisisToHandle),
29
30 msrNav(ColctCrisisToHandle,
31    [msrSize,geq,[[ptInteger,1]]],
32    [[ptBoolean,true]])
33.
34
35msrop(outactActivator,
36   oeSollicitateCrisisHandling,
37   [preFunctional,Self
38    ],
39   []):-
40/* Pre Functional:*/
41/* PreF01 */
42true.
43
44msrop(outactActivator,
45   oeSollicitateCrisisHandling,
46   [post,Self
47    ],
48    []):-
49
50 msrVar(ctState,TheSystem),
51 msrVar(dtComment,AMessageForCrisisHandlers),
52 msrVar(dtDateAndTime, TheClock),
53 msrVarCol(ctCrisis,_,ColctCrisisToAllocateIfPossible),
54
55/* Post Functional:*/
56 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
57
58 /* PostF01 */
59 msrNav([TheSystem],
60    [rnctCrisis,msrSelect,
61     maxHandlingDelayPassed,[]
62    ],
63    ColctCrisisToAllocateIfPossible),
64
65msrNav(ColctCrisisToAllocateIfPossible,
66    [msrForAll,isAllocatedIfPossible,[]],
67    [[ptBoolean,true]]),
68
69 /* PostF02 */
70 msrNav([TheSystem],
71    [rnctCrisis,msrSelect,
72     handlingDelayPassed,[]
73    ],
74    ColctCrisisToHandle),
75
76 msrNav(ColctCrisisToHandle,
77     [msrColSubtract,[ColctCrisisToAllocateIfPossible]
78     ],
79     ColctCrisisToRemind),
80
```

```
81 (msrNav(ColctCrisisToRemind,
82     [msrSize,geq,[[ptInteger,1]]],
83     [[ptBoolean,true]])
84  -> (msrNav([AMessageForCrisisHandlers],
85       [value],
86       [[ptString,'There are alerts pending since more than the defined delay. Please REACT !']]),
87
88    msrNav([TheSystem],
89       [rnactAdministrator,rnInterfaceIN,
90        ieMessage,[AMessageForCrisisHandlers]
91       ],
92       [[ptBoolean,true]]),
93
94    msrNav([TheSystem],
95       [rnactCoordinator,msrForAll,rnInterfaceIN,
96        ieMessage,[AMessageForCrisisHandlers]
97       ],
98       [[ptBoolean,true]])
99   )
100 ; true
101 ),
102
103/* Post Protocol:*/
104/* PostP01 */
105 msrNav([TheSystem],
106    [clock],
107    [TheClock]),
108
109 msrNav([TheSystem],
110     [msmAtPost,vpLastReminder],
111     [TheClock])
112 .
```

Listing 5.4: Messir (Prolog-oriented) implementation of the operation *oeSollicitateCrisisHandling*.

Figure 5.1 shows concept model elements in the scope of the oeSollicitateCrisisHandling operation

Figure 5.1: oeSollicitateCrisisHandling operation scope

5.2 Environment - Out Interface Operation Scheme for actAdministrator

5.2.1 Operation Model for oeAddCoordinator

The `oeAddCoordinator` operation has the following properties:

OPERATION	
oeAddCoordinator	
sent to add a new coordinator in the system's post state and environment's post state.	
Parameters	
1	**AdtCoordinatorID: dtCoordinatorID** used to initialize the id field
2	**AdtLogin: dtLogin** used to initialize the login field
3	**AdtPassword: dtPassword** used to initialize the password field
Return type	
ptBoolean	
Pre-Condition (protocol)	
PreP 1	the system is started
PreP 2	the actor logged previously and did not log out ! (i.e. the associated ctAdministrator instance is considered logged)
Pre-Condition (functional)	
PreF 1	it is supposed that there cannot exist a ctCoordinator instance with the same `id` attribute as the one the administrator wants to delete.
Post-Condition (functional)	
PostF 1	the environment has a new instance of coordinator actor allowing for input/output message communication with the system.
PostF 2	the system's state has a new instance of ctCoordinator initialized with the given values.
PostF 3	the new actor instance and ctCoordinator instance are related.
PostF 4	the new actor instance and ctCoordinator instance are related according to the authenticated association.
PostF 5	the administrator actor is informed about the satisfaction of its request.
Post-Condition (protocol)	
PostP 1	none

The listing 5.5 provides the Messir (MCL-oriented) specification of the operation.

```

/* Pre Protocol:*/
preP{let TheSystem: ctState in
 let TheActor:actCoordinator in

 self.rnActor.rnSystem = TheSystem
 and self.rnActor = TheCoordinator

/* PreP01 */
 and TheSystem.vpStarted = true
```

```
11 /* PreP02 */
12  and TheActor.rnctAuthenticated.vpIsLogged = true}
13
14 /* Pre Functional:*/
15 preF{let TheSystem: ctState in
16  let TheActor:actAdministrator in
17  let ColctCoordinators:Bag(ctCoordinator) in
18
19  self.rnActor.rnSystem = TheSystem
20  and self.rnActor = TheActor
21 /* PreF01 */
22  and TheSystem.rnctCoordinator->select(id.eq(AdtCoordinatorID))
23    = ColctCoordinators
24  and ColctCoordinators->isEmpty() = true}
25
26 /* Post Functional:*/
27 postF{let TheSystem: ctState in
28  let TheactCoordinator:actCoordinator in
29  let ThectCoordinator:ctCoordinator in
30  self.rnActor.rnSystem = TheSystem
31  and self.rnActor = TheActor
32 /* PostF01 */
33  TheactCoordinator.init()
34 /* PostF02 */
35  and ThectCoordinator.init(AdtCoordinatorID,AdtLogin,AdtPassword)
36
37 /* PostF03 */
38  and TheactCoordinator@post.rnctCoordinator = ThectCoordinator
39
40 /* PostF04 */
41  and ThectCoordinator@post.rnactAuthenticated = TheactCoordinator
42
43 /* PostF05 */
44  and TheActor.rnInterfaceIN^ieCoordinatorAdded()}
45
46 /* Post Protocol:*/
47 postP{ true}
```

Listing 5.5: Messir (MCL-oriented) specification of the operation *oeAddCoordinator*.

The listing 5.6 provides the Messir (Prolog-oriented) implementation of the operation.

```
1%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5%-----------------------------------------------------
6msrop(outactAdministrator,
7   oeAddCoordinator,
8   [preProtocol,Self,
9    AdtCoordinatorID,
10   AdtLogin,
11   AdtPassword
12   ],
13   []):-
14/* Pre Protocol:*/
15 msrVar(ctState,TheSystem),
16 msrVar(actAdministrator,TheActor),
17 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
18 msrNav([Self],[rnActor],[TheActor]),
19
20/* PreP01 */
21 msrNav([TheSystem],
22    [vpStarted],
23    [[ptBoolean,true]]),
24
```

```
25/* PreP02 */
26 msrNav([TheActor],
27    [rnctAuthenticated,vpIsLogged],
28    [[ptBoolean,true]])
29
30 .
31
32msrop(outactAdministrator,
33   oeAddCoordinator,
34   [preFunctional,Self,
35    AdtCoordinatorID,
36    AdtLogin,
37    AdtPassword
38    ],
39   []):-
40/* Pre Functional:*/
41 msrVar(ctState,TheSystem),
42 msrVar(actAdministrator,TheActor),
43 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
44 msrNav([Self],[rnActor],[TheActor]),
45/* PreF01 */
46 msrNav([TheSystem],
47     [rnctCoordinator,
48     msrSelect,id,eq,[AdtCoordinatorID]],
49     ColctCoordinators),
50 msrNav(ColctCoordinators,
51     [msrIsEmpty],
52     [[ptBoolean,true]])
53 .
54
55msrop(outactAdministrator,
56   oeAddCoordinator,
57   [post,Self,
58    AdtCoordinatorID,
59    AdtLogin,
60    AdtPassword
61    ],
62    []):-
63
64/* Post Functional:*/
65 msrVar(ctState,TheSystem),
66 msrVar(actAdministrator,TheActor),
67 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
68 msrNav([Self],[rnActor],[TheActor]),
69
70 msrVar(actCoordinator,TheactCoordinator),
71 msrVar(ctCoordinator,ThectCoordinator),
72
73/* PostF01 */
74 msrNav([TheactCoordinator],
75     [init,[]],
76     [[ptBoolean,true]]),
77
78/* PostF02 */
79 msrNav([ThectCoordinator],
80     [init,[AdtCoordinatorID,AdtLogin,AdtPassword]],
81     [[ptBoolean,true]]),
82
83/* PostF03 */
84 msrNav([TheactCoordinator],
85     [msmAtPost,rnctCoordinator],
86     [ThectCoordinator]),
87
88/* PostF04 */
89  msrNav([ThectCoordinator],
90     [msmAtPost,rnactAuthenticated],
91     [TheactCoordinator]),
92
93/* PostF05 */
94  msrNav([TheActor],
```

```
95      [rnInterfaceIN,
96       ieCoordinatorAdded,[]],
97      [[ptBoolean,true]]),
98
99  /* Post Protocol:*/
100 /* PostP01 */
101 true
102 .
```

Listing 5.6: Messir (Prolog-oriented) implementation of the operation *oeAddCoordinator*.

5.2.2 Operation Model for oeAddHospital

The `oeAddHospital` operation has the following properties:

Operation	
oeAddHospital	
sent to add a new hospital in the system's post state and environment's post state.	
Parameters	
1	**AdtCoordinatorID: dtHospitalID** used to initialize the id field
2	**AdtLogin: dtLogin** used to initialize the login field
3	**AdtPassword: dtPassword** used to initialize the password field
Return type	
ptBoolean	
Pre-Condition (protocol)	
PreP 1	the system is started
PreP 2	the actor logged previously and did not log out ! (i.e. the associated ctAdministrator instance is considered logged)
Pre-Condition (functional)	
PreF 1	it is supposed that there cannot exist a ctHospital instance with the same `id` attribute as the one the administrator wants to delete.
Post-Condition (functional)	
PostF 1	the environment has a new instance of hospital actor allowing for input/output message communication with the system.
PostF 2	the system's state has a new instance of ctHospital initialized with the given values.
PostF 3	the new actor instance and ctHospital instance are related.
PostF 4	the new actor instance and ctHospital instance are related according to the authenticated association.
PostF 5	the administrator actor is informed about the satisfaction of its request.
Post-Condition (protocol)	
PostP 1	none

The listing 5.7 provides the Messir (MCL-oriented) specification of the operation.

```
1
2  /* Pre Protocol:*/
```

```
preP{let TheSystem: ctState in
 let TheActor:actAdministrator in

 self.rnActor.rnSystem = TheSystem
 and self.rnActor = TheActor

/* PreP01 */
 and TheSystem.vpStarted = true
/* PreP02 */
 and TheActor.rnctAuthenticated.vpIsLogged = true}

/* Pre Functional:*/
preF{let TheSystem: ctState in
 let TheActor:actAdministrator in

 let ColctHospitals:Bag(ctHospital) in

 self.rnActor.rnSystem = TheSystem
 and self.rnActor = TheActor
/* PreF01 */
 and TheSystem.rnctHospital->select(id.eq(AdtHospitalID))
   = ColctHospitals
 and ColctHospitals->isEmpty() = true}

/* Post Functional:*/
postF{let TheSystem: ctState in
 let TheactHospital:actHospital in
 let ThectHospital:ctHospital in
 self.rnActor.rnSystem = TheSystem
 and self.rnActor = TheActor
/* PostF01 */
 TheactHospital.init()
/* PostF02 */
 and ThectHospital.init(AdtHospitalID,AdtLogin,AdtPassword)

/* PostF03 */
 and TheactHospital@post.rnctHospital = ThectHospital

/* PostF04 */
 and ThectHospital@post.rnactAuthenticated = TheactHospital

/* PostF05 */
 and TheActor.rnInterfaceIN^ieHospitalAdded()}

/* Post Protocol:*/
postP{ true}
```

Listing 5.7: Messir (MCL-oriented) specification of the operation *oeAddHospital*.

5.2.3 Operation Model for oeDeleteCoordinator

The `oeDeleteCoordinator` operation has the following properties:

OPERATION	
oeDeleteCoordinator sent to delete an existing coordinator in the system's post state and environment's post state.	
Parameters	
1	**AdtCoordinatorID: dtCoordinatorID** used for ctCoordinator instance retrieval
Return type	
ptBoolean	
Pre-Condition (protocol)	

continues in next page . . .

. . . Operation table continuation

PreP 1	the system is started
PreP 2	the actor logged previously and did not log out ! (i.e. the associated ctAdministrator instance is considered logged)
Pre-Condition (functional)	
PreF 1	it is supposed that there exist one ctCoordinator instance with the same `id` attribute than the one the administrator wants to create.
Post-Condition (functional)	
PostF 1	the ctCoordinator class instance having the required id do not belong anymore to the post state as well as is related actCoordinator actor instance.
PostF 2	the administrator actor is informed about the satisfaction of its request.
Post-Condition (protocol)	
PostP 1	none

The listing 5.8 provides the Messir (MCL-oriented) specification of the operation.

```

/* Pre Protocol:*/
preP{let TheSystem: ctState in
 let TheActor:actAdministrator in

 self.rnActor.rnSystem = TheSystem
 and self.rnActor = TheActor

/* PreP01 */
 and TheSystem.vpStarted = true
/* PreP02 */
 and TheActor.rnctAuthenticated.vpIsLogged = true}

/* Pre Functional:*/
preF{let TheSystem: ctState in
 let TheActor:actAdministrator in

 self.rnActor.rnSystem = TheSystem
 and self.rnActor = TheActor
/* PreF01 */
 TheSystem.rnctCoordinator->select(id.eq(AdtCoordinatorID))
 = ColctCoordinators
 and ColctCoordinators->size().eq(1)}

/* Post Functional:*/
postF{let TheSystem: ctState in
 let TheActor:actAdministrator in
 let ThectCoordinator:ctCoordinator in
 self.rnActor.rnSystem = TheSystem
 and self.rnActor = TheActor
/* PostF01 */
 TheSystem.rnctCoordinator->select(id.eq(AdtCoordinatorID))
 = ThectCoordinator
 and ThectCoordinator.rnactCoordinator->forAll(msrIsKilled)
 and ThectCoordinator.msrIsKilled

 /* PostF02 */
 and TheActor.rnInterfaceIN^ieCoordinatorDeleted()

 /* Post Protocol:*/
/* PostP01 */
 and true}

/* Post Protocol:*/
```

```
postP{ true}
```

Listing 5.8: Messir (MCL-oriented) specification of the operation *oeDeleteCoordinator*.

The listing 5.9 provides the Messir (Prolog-oriented) implementation of the operation.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%-------------------------------------------------------
msrop(outactAdministrator,
   oeDeleteCoordinator,
   [preProtocol,Self,
    AdtCoordinatorID
    ],
   []):-
/* Pre Protocol:*/
 msrVar(ctState,TheSystem),
 msrVar(actAdministrator,TheActor),
 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
 msrNav([Self],[rnActor],[TheActor]),

/* PreP01 */
 msrNav([TheSystem],
    [vpStarted],
    [[ptBoolean,true]]),

 msrNav([TheActor],
    [rnctAuthenticated,vpIsLogged],
    [[ptBoolean,true]])
.

msrop(outactAdministrator,
   oeDeleteCoordinator,
   [preFunctional,Self,
    AdtCoordinatorID
    ],
   []):-
/* Pre Functional:*/
 msrVar(ctState,TheSystem),
 msrVar(actAdministrator,TheActor),
 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
 msrNav([Self],[rnActor],[TheActor]),

/* PreF01 */
 msrNav([TheSystem],
     [rnctCoordinator,
     msrSelect,id,eq,[AdtCoordinatorID]],
     ColctCoordinators),

 msrNav(ColctCoordinators,
     [msrSize,eq,[[ptInteger,1]]],
     [[ptBoolean,true]]).

msrop(outactAdministrator,
   oeDeleteCoordinator,
   [post,Self,
    AdtCoordinatorID
    ],
    []):-

/* Post Functional:*/
 msrVar(ctState,TheSystem),
 msrVar(actAdministrator,TheActor),
 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
```

```
61 msrNav([Self],[rnActor],[TheActor]),
62
63/* PostF01 */
64 msrNav([TheSystem],
65    [rnctCoordinator,
66     msrSelect,id,eq,[AdtCoordinatorID]],
67    [ThectCoordinator]),
68
69 msrNav([ThectCoordinator],
70     [rnactCoordinator,msrForAll,msrIsKilled],
71     [[ptBoolean,true]]),
72
73 msrNav([ThectCoordinator],
74     [msrIsKilled],
75     [[ptBoolean,true]]),
76
77 /* PostF02 */
78 msrNav([TheActor],
79     [rnInterfaceIN,
80     ieCoordinatorDeleted,[]
81     ],
82     [[ptBoolean,true]]),
83
84 /* Post Protocol:*/
85/* PostP01 */
86 true
87 .
```

Listing 5.9: Messir (Prolog-oriented) implementation of the operation *oeDeleteCoordinator*.

5.2.4 Operation Model for oeDeleteHospital

The `oeDeleteHospital` operation has the following properties:

OPERATION	
oeDeleteHospital	
sent to delete an existing hospital in the system's post state and environment's post state.	
Parameters	
1	**AdtHospitalID: dtHospitalID** used for ctHospital instance retrieval
Return type	
ptBoolean	
Pre-Condition (protocol)	
PreP 1	the system is started
Pre-Condition (functional)	
PreF 1	the actor logged previously and did not log out ! (i.e. the associated ctAdministrator instance is considered logged)
Post-Condition (functional)	
PostF 1	the ctHospital class instance having the required id do not belong anymore to the post state as well as is related actHospital actor instance.
PostF 2	the administrator actor is informed about the satisfaction of its request.
Post-Condition (protocol)	
PostP 1	none

The listing 5.10 provides the Messir (MCL-oriented) specification of the operation.

```

/* Pre Protocol:*/
preP{let TheSystem: ctState in
 let TheActor:actAdministrator in

 self.rnActor.rnSystem = TheSystem
 and self.rnActor = TheActor

/* PreP01 */
 and TheSystem.vpStarted = true
/* PreP02 */
 and TheActor.rnctAuthenticated.vpIsLogged = true}

/* Pre Functional:*/
preF{let TheSystem: ctState in
 let TheActor:actAdministrator in

 self.rnActor.rnSystem = TheSystem
 and self.rnActor = TheActor
/* PreF01 */
 TheSystem.rnctHospital->select(id.eq(AdtHospitalID))
 = ColctHospitals
 and ColctHospitals->size().eq(1)}

/* Post Functional:*/
postF{let TheSystem: ctState in
 let TheActor:actAdministrator in
 let ThectHospital:ctHospital in
 self.rnActor.rnSystem = TheSystem
 and self.rnActor = TheActor
/* PostF01 */
 TheSystem.rnctHospital->select(id.eq(AdtHospitalID))
 = ThectHospital
 and ThectHospital.rnactHospital->forAll(msrIsKilled)
 and ThectHospital.msrIsKilled

 /* PostF02 */
 and TheActor.rnInterfaceIN^ieHospitalDeleted()

 /* Post Protocol:*/
/* PostP01 */
 and true}

/* Post Protocol:*/
postP{ true}
```

Listing 5.10: Messir (MCL-oriented) specification of the operation *oeDeleteHospital*.

5.3 Environment - Out Interface Operation Scheme for actAuthenticated

5.3.1 Operation Model for oeLogin

The `oeLogin` operation has the following properties:

OPERATION	
oeLogin sent to request authorization to request access secured system operations.	
Parameters	
1	**AdtLogin: dtLogin** first information used to determine accessibility rights for the actual actor.
2	**AdtPassword: dtPassword**

continues in next page . . .

... Operation table continuation

	second information used to determine accessibility rights for the actual actor.
Return type	
ptBoolean	
Pre-Condition (protocol)	
PreP 1	the system is started
PreP 2	the actor is not already logged in ! (i.e. the associated ctAuthenticated instance is not considered logged)
Pre-Condition (functional)	
PreF 1	none
Post-Condition (functional)	
PostF 1	if the login and password provided by the actor correspond to the ones that belong to the ctAuthenticated instance he is related to then a welcome message is sent to the actor (n.b. the logged status is changed as a post-protocol condition); else the actor is notified that he gave incorrect data and all the administrator actors existing in the environement are notified of an intrusion temptative.
Post-Condition (protocol)	
PostP 1	if the authentication information is correct then the actor is known to be logged in ! (i.e. the associated ctAuthenticated instance with given login and password is considered logged)

The listing 5.11 provides the Messir (MCL-oriented) specification of the operation.

```
1
2  /* Pre Protocol:*/
3  preP{let TheSystem: ctState in
4   let TheActor:actAuthenticated in
5   self.rnActor.rnSystem = TheSystem
6   and self.rnActor = TheActor
7
8  /* PreP01 */
9   and TheSystem.vpStarted = true
10 /* PreP02 */
11  and TheActor.rnctAuthenticated.vpIsLogged = false}
12
13 /* Pre Functional:*/
14 preF{/* PreF01 */
15 true}
16
17 /* Post Functional:*/
18 postF{let TheSystem: ctState in
19  let TheactAuthenticated:actAuthenticated in
20
21  let AptStringMessageForTheactAuthenticated: ptString in
22  let AptStringMessageForTheactAdministrator:ptString in
23
24  self.rnActor.rnSystem = TheSystem
25  and self.rnActor = TheactAuthenticated
26
27  and /* PostF01 */
28    if (TheactAuthenticated.rnctAuthenticated.pwd
29      = AdtPassword
30      and TheactAuthenticated.rnctAuthenticated.login
31        = AdtLogin
32      )
33    then (AptStringMessageForTheactAuthenticated.eq('You are logged ! Welcome ...')
34       and TheactAuthenticated.rnInterfaceIN^ieMessage(AptStringMessageForTheactAuthenticated)
35       )
```

```
36     else (AptStringMessageForTheactAuthenticated
37        .eq('Wrong identification information ! Please try again ...')
38        and TheactAuthenticated.rnInterfaceIN^ieMessage(AptStringMessageForTheactAuthenticated)
39        and AptStringMessageForTheactAdministrator.eq('Intrusion tentative !')
40        and TheSystem.rnactAdministrator
41          .rnInterfaceIN^ieMessage(AptStringMessageForTheactAdministrator)
42        )
43     endif}
44
45  /* Post Protocol:*/
46  postP{ let TheSystem: ctState in
47   let TheactAuthenticated:actAuthenticated in
48
49   self.rnActor.rnSystem = TheSystem
50   and self.rnActor = TheactAuthenticated
51  /* PostP01 */
52   if (TheactAuthenticated.rnctAuthenticated.pwd = AdtPassword
53     and TheactAuthenticated.rnctAuthenticated.login = AdtLogin
54     )
55   then (TheactAuthenticated.rnctAuthenticated@post.vpIsLogged = true)
56   else true
57   endif}
```

Listing 5.11: Messir (MCL-oriented) specification of the operation *oeLogin*.

The listing 5.12 provides the Messir (Prolog-oriented) implementation of the operation.

```
1%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5%-------------------------------------------------------
6msrop(outactAuthenticated,
7   oeLogin,
8   [preProtocol,Self,
9    AdtLogin,
10   AdtPassword
11   ],
12   []):-
13/* Pre Protocol:*/
14 msrVar(ctState,TheSystem),
15 msrVar(actAuthenticated,TheactAuthenticated),
16 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
17 msrNav([Self],[rnActor],[TheactAuthenticated]),
18
19 /* PreP01 */
20 msrNav([TheSystem],
21    [vpStarted],
22    [[ptBoolean,true]]),
23
24 msrNav([TheactAuthenticated],
25    [rnctAuthenticated,vpIsLogged],
26    [[ptBoolean,false]])
27 .
28
29msrop(outactAuthenticated,
30   oeLogin,
31   [preFunctional,Self,
32    AdtLogin,
33    AdtPassword
34    ],
35   []):-
36/* Pre Functional:*/
37/* PreF01 */
38true
39.
```

```
40
41msrop(outactAuthenticated,
42   oeLogin,
43   [post,Self,
44    AdtLogin,
45    AdtPassword
46    ],
47    []):-
48
49 msrVar(ctState,TheSystem),
50 msrVar(actAuthenticated,TheactAuthenticated),
51
52 msrVar(ptString,AptStringMessageForTheactAuthenticated),
53 msrVar(ptString,AptStringMessageForTheactAdministrator),
54
55/* Post Functional:*/
56
57 msrNav([Self],[rnActor],[TheactAuthenticated]),
58 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
59
60/* PostF01 */
61
62 ( (msrNav([TheactAuthenticated],
63      [rnctAuthenticated,pwd],
64      [AdtPassword]),
65   msrNav([TheactAuthenticated],
66      [rnctAuthenticated,login],
67      [AdtLogin])
68  )
69 -> ( msrNav([AptStringMessageForTheactAuthenticated],
70       [eq,[[ptString,'You are logged ! Welcome ...']]],
71       [[ptBoolean,true]]),
72    msrNav([TheactAuthenticated],
73       [rnInterfaceIN,
74        ieMessage,[AptStringMessageForTheactAuthenticated]],
75       [[ptBoolean,true]])
76   )
77 ; ( msrNav([AptStringMessageForTheactAuthenticated],
78       [eq,[[ptString,'Wrong identification information ! Please try again ...']]],
79       [[ptBoolean,true]]),
80    msrNav([TheactAuthenticated],
81       [rnInterfaceIN,
82        ieMessage,[AptStringMessageForTheactAuthenticated]],
83       [[ptBoolean,true]]),
84
85    msrNav([AptStringMessageForTheactAdministrator],
86       [eq,[[ptString,'Intrusion tentative !']]],
87       [[ptBoolean,true]]),
88    msrNav([TheSystem],
89       [rnactAdministrator,rnInterfaceIN,
90        ieMessage,[AptStringMessageForTheactAdministrator]],
91       [[ptBoolean,true]])
92    )
93 ),
94
95 /* Post Protocol:*/
96/* PostP01 */
97 ( (msrNav([TheactAuthenticated],
98      [rnctAuthenticated,pwd],
99      [AdtPassword]),
100   msrNav([TheactAuthenticated],
101      [rnctAuthenticated,login],
102      [AdtLogin])
103  )
104 -> (msrNav([TheactAuthenticated],
105       [rnctAuthenticated,msmAtPost,vpIsLogged],
106       [[ptBoolean,true]])
107    )
108 ; true
109 )
```

```
110 .
```

Listing 5.12: Messir (Prolog-oriented) implementation of the operation *oeLogin*.

5.3.2 Operation Model for oeLogout

The `oeLogout` operation has the following properties:

OPERATION	
oeLogout sent to end the secured access to specific system operations.	
Return type	
ptBoolean	
Pre-Condition (protocol)	
PreP 1	the system is started
PreP 2	the actor is currently logged in ! (i.e. the associated ctAuthenticated instance is considered logged)
Pre-Condition (functional)	
PreF 1	
Post-Condition (functional)	
PostF 1	a logout confirmation message is sent to the actor (n.b. the logged status is changed as a post-protocol condition)
Post-Condition (protocol)	
PostP 1	the actor is known to be logged out ! (i.e. the associated ctAuthenticated instance with given login and password is considered logged out)

The listing 5.13 provides the Messir (MCL-oriented) specification of the operation.

```
1
2  /* Pre Protocol:*/
3  preP{let TheSystem: ctState in
4   let TheActor:actAdministrator in
5   self.rnActor.rnSystem = TheSystem
6   and self.rnActor = TheActor
7
8  /* PreP01 */
9   and TheSystem.vpStarted = true
10 /* PreP02 */
11  and TheActor.rnctAuthenticated.vpIsLogged = true}
12
13 /* Pre Functional:*/
14 preF{/* PreF01 */
15 true}
16
17 /* Post Functional:*/
18 postF{let TheSystem: ctState in
19  let TheactAuthenticated:actAuthenticated in
20  let AptStringMessageForTheactAuthenticated: ptString in
21
22  self.rnActor.rnSystem = TheSystem
23  and self.rnActor = TheactAuthenticated
24
25  /* PostF01 */
26  AptStringMessageForTheactAuthenticated.eq('You are logged out ! Good Bye ...')
```

```
27  and TheactAuthenticated.rnInterfaceIN^ieMessage(AptStringMessageForTheactAuthenticated)}
28
29 /* Post Protocol:*/
30 postP{ let TheSystem: ctState in
31  let TheactAuthenticated:actAuthenticated in
32
33  self.rnActor.rnSystem = TheSystem
34  and self.rnActor = TheactAuthenticated.asSet
35 /* PostP01 */
36  TheactAuthenticated.rnctAuthenticated@post.vpIsLogged = false}
```

Listing 5.13: Messir (MCL-oriented) specification of the operation *oeLogout*.

The listing 5.14 provides the Messir (Prolog-oriented) implementation of the operation.

```
1%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5%-------------------------------------------------------
6msrop(outactAuthenticated,
7   oeLogout,
8   [preProtocol,Self
9    ],
10   []):-
11/* Pre Protocol:*/
12 msrVar(ctState,TheSystem),
13 msrVar(actAuthenticated,TheActor),
14 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
15 msrNav([Self],[rnActor],[TheActor]),
16
17/* PreP01 */
18 msrNav([TheSystem],
19    [vpStarted],
20    [[ptBoolean,true]]),
21
22 msrNav([TheActor],
23    [rnctAuthenticated,vpIsLogged],
24    [[ptBoolean,true]])
25 .
26
27msrop(outactAuthenticated,
28   oeLogout,
29   [preFunctional,Self
30    ],
31   []):-
32/* Pre Functional:*/
33/* PreF01 */
34true
35.
36
37msrop(outactAuthenticated,
38   oeLogout,
39   [post,Self
40    ],
41    []):-
42
43 msrVar(ctState,TheSystem),
44 msrVar(actAuthenticated,TheactAuthenticated),
45
46 msrVar(ptString,AptStringMessageForTheactAuthenticated),
47
48/* Post Functional:*/
49 msrNav([Self],[rnActor],[TheactAuthenticated]),
50 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
51
```

```
/* PostF01 */
msrNav([AptStringMessageForTheactAuthenticated],
    [eq,[[ptString,'You are logged out ! Good Bye ...']]],
    [[ptBoolean,true]]),
msrNav([TheactAuthenticated],
    [rnInterfaceIN,
     ieMessage,[AptStringMessageForTheactAuthenticated]],
    [[ptBoolean,true]]),

 /* Post Protocol:*/
/* PostP01 */
msrNav([TheactAuthenticated],
    [rnctAuthenticated,msmAtPost,vpIsLogged],
    [[ptBoolean,false]])
.
```

Listing 5.14: Messir (Prolog-oriented) implementation of the operation *oeLogout*.

5.4 Environment - Out Interface Operation Scheme for actComCompany

5.4.1 Operation Model for oeAlert

The `oeAlert` operation has the following properties:

OPERATION	
oeAlert	
Any human having a phone able to connect to the communication companies using the *iCrash* system can send his company an sms message with structured information in order to declare an alert.	
Parameters	
1	**AetHumanKind: etHumanKind** the kind of human informing of an alert.
2	**AdtDate: dtDate** the date of the alert
3	**AdtTime: dtTime** the time of the alert
4	**AdtPhoneNumber: dtPhoneNumber** the phone number of the human sending the alert SMS message
5	**AdtGPSLocation: dtGPSLocation** the GPS position of the phone at the date and time the message was sent.
6	**AdtComment: dtComment** a free text message sent by the human providing information on the alert that he wants to declare
Return type	
ptBoolean	
Pre-Condition (protocol)	
PreP 1	the system is supposed to be created and initialized.
Pre-Condition (functional)	
PreF 1	the date and time the alert is declared is supposed to be in the past with respect to the current time known by the system.
Post-Condition (functional)	
PostF 1	the ctState attribute for the next value for alert IDs is incremented by one at post.

continues in next page . . .

. . . Operation table continuation

PostF 2	a new alert instance exists in the post state with status pending, instant information (resp. GPS location and comment) based on date and time provided (resp. position and comment); and with alert ID being a string conversion of the dtInteger value available in the pre state in the ctState instance.
PostF 3	if there exist no already registered alert near to the alert currently declared then a new crisis is added in the post state and initialized with: its ID being the one provided by the ctState instance (which is incremented by one in the post state), its type considered as small, its status being pending, its declared time being the same than the alert and a default comment indicating that a report will come later on. else the crisis to which the new alert must be related to is the one related to any alert nearby in the pre-state.
PostF 4	the post state relates the new alert to the previously characterized crisis.
PostF 5	if there is no ctHuman instance having same phone number and same kind in the pre-state then a new one is added in the post-state with given phone number and kind and is associated to the communication company actor used to declare the alert. else the pre-state one is chosen
PostF 6	and this specified ctHuman is related to the new alert thus indicating he has signled the alert.
Post-Condition (protocol)	
PostP 1	none

The listing 5.15 provides the Messir (MCL-oriented) specification of the operation.

```

/* Pre Protocol:*/
preP{let TheSystem: ctState in
 self.rnActor.rnSystem = TheSystem

/* PreP01 */
 and TheSystem.vpStarted = true}

/* Pre Functional:*/
preF{let TheSystem: ctState in
 self.rnActor.rnSystem = TheSystem

/* PreF01 */
 and (TheSystem.clock.date.gt(AdtDate)
    or (TheSystem.clock.date.eq(AdtDate)
      and TheSystem.clock.time.gt(AdtTime)
     )
   )}

/* Post Functional:*/
postF{let TheSystem: ctState in

 let ActHuman:ctHuman in
 let TheactComCompany:actComCompany in
 let ActAlert:ctAlert in
 let AAlertInstant:dtDateAndTime in
 let AetAlertStatus:etAlertStatus in
 let ActAlertNearBy:ctAlert in
 let ActCrisis:ctCrisis in
 let AdtCrisisID:dtCrisisID in
 let AetCrisisType:etCrisisType in
 let AetCrisisStatus:etCrisisStatus in
 let ACrisisInstant:dtDateAndTime in
 let ACrisisdtComment:dtComment in
```

```
 let AptStringMessage:ptString in
 let AdtSMS:dtSMS in
 let AdtAlertID:dtAlertID in

 self.rnActor.rnSystem = TheSystem
 and self.rnActor = TheactComCompany
/* PostF01 */
 TheSystem.nextValueForAlertID=PrenextValueForAlertID
 and PrenextValueForAlertID.add(1) = PostnextValueForAlertID // Inga
 and TheSystem@post.nextValueForAlertID = PostnextValueForAlertID

 /* PostF02 */
and AAlertInstant.date=AdtDate
and AAlertInstant.time=AdtTime

and AetAlertStatus=pending

and TheSystem.nextValueForAlertID.todtString().eq(AdtAlertID)

and ActAlert.init(AdtAlertID,
         AetAlertStatus,
         AdtGPSLocation,
         AAlertInstant,
         AdtComment)

 /* PostF03 */

and TheSystem.rnctAlert -> select(location.isNearTo(AdtGPSLocation)) = ColctAlertsNearBy
//(TheSystem.rnCtCrisis -> select(c:ctCrisis | c.id.value = 1)).rnctAlerts

and if (ColctAlertsNearBy->size()=0)
  then (TheSystem.nextValueForCrisisID = PrenextValueForCrisisID
      and PrenextValueForCrisisID.add(1) = PostnextValueForCrisisID
      and TheSystem@post.nextValueForCrisisID = PostnextValueForCrisisID
      and TheSystem.nextValueForCrisisID.todtString().eq(AdtCrisisID)
      and AdtCrisisType = small
      and AetCrisisStatus = pending
      and ACrisisInstant= AAlertInstant
      and ACrisisdtComment = 'no reporting yet defined'
      and ActCrisis.init( AdtCrisisID,
                AdtCrisisType,
                AetCrisisStatus,
                AdtGPSLocation,
                ACrisisInstant,
                ACrisisdtComment)
      )
 else (ColctAlertsNearBy.rnTheCrisis->msrAny(true) = ActCrisis)
 endif

 /* PostF04 */
and ActAlert@post.rnTheCrisis = ActCrisis

/* PostF05 */
and TheSystem.rnctHuman->select(id.eq(AdtPhoneNumber)) = HumanCol1

and HumanCol1->select(kind.etEq(AetHumanKind)) = HumanCol2
and if (HumanCol2->msrIsEmpty)
  then (ActHuman.init(AdtPhoneNumber,AetHumanKind)
      and ActHuman@post.rnactComCompany = TheactComCompany
      )
  else (HumanCol2->any(true) = ActHuman)
  endif

 and ActHuman.rnSignaled->msrIncluding(ActAlert) = ColAlerts

 and ActHuman@post.rnSignaled = ColAlerts

/* PostF06 */
AdtSMS.value = 'Your alert has been registered. We will handle it and keep you informed'
and TheactComCompany.rnInterfaceIN^ieSmsSend(AdtPhoneNumber,AdtSMS)}
```

```

/* Post Protocol:*/
postP{ true}
```

Listing 5.15: Messir (MCL-oriented) specification of the operation *oeAlert*.

The listing 5.16 provides the Messir (Prolog-oriented) implementation of the operation.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%-------------------------------------------------------
nico(A):-
 trace,
 write('here'),
 write('\n').

msrop(outactComCompany,
   oeAlert,
   [preProtocol,Self,
    AetHumanKind,
    AdtDate,
    AdtTime,
    AdtPhoneNumber,
    AdtGPSLocation,
    AdtComment
    ],
   []):-
/* Pre Protocol:*/
 msrVar(ctState,TheSystem),
 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
/* PreP01 */
 msrNav([TheSystem],
    [vpStarted],
    [[ptBoolean,true]])
 .

msrop(outactComCompany,
   oeAlert,
   [preFunctional,Self,
    AetHumanKind,
    AdtDate,
    AdtTime,
    AdtPhoneNumber,
    AdtGPSLocation,
    AdtComment
    ],
   []):-
/* Pre Functional:*/
/* PreF01 */
 msrVar(ctState,TheSystem),
 msrNav([Self],
     [msmAtPre,rnActor,rnSystem],
     [TheSystem]),

 ( msrNav([TheSystem],[clock,date,gt,[AdtDate]],[[ptBoolean,true]])
  ; (msrNav([TheSystem],[clock,date,eq,[AdtDate]],[[ptBoolean,true]])
   , msrNav([TheSystem],[clock,time,gt,[AdtTime]],[[ptBoolean,true]])
   )
 )
.

msrop(outactComCompany,
   oeAlert,
   [post,Self,
```

```
59     AetHumanKind,
60     AdtDate,
61     AdtTime,
62     AdtPhoneNumber,
63     AdtGPSLocation,
64     AdtComment
65     ],
66     []):-
67
68 msrVar(ctState,TheSystem),
69 msrVar(ctHuman,ActHuman),
70 msrVar(actComCompany,TheactComCompany),
71 msrVar(ctAlert,ActAlert),
72 msrVar(dtDateAndTime,AAlertInstant),
73 msrVar(etAlertStatus,AetAlertStatus),
74% msrVar(ctAlert,ActAlertNearBy),
75 msrVar(ctCrisis,ActCrisis),
76 msrVar(dtCrisisID,AdtCrisisID),
77% msrVar(etCrisisType,AetCrisisType),
78 msrVar(etCrisisStatus,AetCrisisStatus),
79 msrVar(dtDateAndTime,ACrisisInstant),
80 msrVar(dtComment,ACrisisdtComment),
81% msrVar(ptString,AptStringMessage),
82 msrVar(dtSMS,AdtSMS),
83 msrVar(dtAlertID,AdtAlertID),
84
85% msrVar(ptInteger,TheNextptIntegerValue),
86% msrVar(ptInteger,UpdatedNextptIntegerValue),
87% msrVar(inactComCompany,TheComCompanyIN),
88% msrVar(dtComment,TheCommentStored),
89% msrVar(dtString,TheCommentStoreddtString),
90
91/* Post Functional:*/
92
93 msrNav([Self],[rnActor],[TheactComCompany]),
94 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
95
96/* PostF01 */
97 msrNav([TheSystem],
98     [nextValueForAlertID],
99     [PrenextValueForAlertID]),
100 msrNav([PrenextValueForAlertID],
101     [add,[[dtInteger,[[value,[ptInteger,1]]],[]]]],
102     [PostnextValueForAlertID]),
103 msrNav([TheSystem],
104     [msmAtPost,nextValueForAlertID],
105     [PostnextValueForAlertID]),
106
107 /* PostF02 */
108 msrNav([AAlertInstant],[date],[AdtDate]),
109 msrNav([AAlertInstant],[time],[AdtTime]),
110
111 msrNav([AetAlertStatus],
112     [],
113     [[etAlertStatus,pending]]),
114
115 msrNav([TheSystem],
116     [nextValueForAlertID,
117     todtString,[],eq,[AdtAlertID]],
118     [[ptBoolean,true]]),
119
120 msrNav([ActAlert],
121     [init,[AdtAlertID,
122        AetAlertStatus,
123        AdtGPSLocation,
124        AAlertInstant,
125        AdtComment]],
126     [[ptBoolean,true]]),
127
128 /* PostF03 */
```

```
129 msrNav([TheSystem],
130     [rnctAlert,
131      msrSelect,location,isNearTo,[AdtGPSLocation]],
132      ColctAlertsNearBy),
133
134 ( (msrNav(ColctAlertsNearBy,
135      [msrIsEmpty],
136      [[ptBoolean,true]])
137  )
138 -> (
139   msrNav([TheSystem],
140       [nextValueForCrisisID],
141       [PrenextValueForCrisisID]),
142   msrNav([PrenextValueForCrisisID],
143       [add,[[dtInteger,[[value,[ptInteger,1]]],[]]]],
144       [PostnextValueForCrisisID]),
145   msrNav([TheSystem],
146       [msmAtPost,nextValueForCrisisID],
147       [PostnextValueForCrisisID]),
148
149  msrNav([TheSystem],
150       [nextValueForCrisisID,
151       todtString,[],eq,[AdtCrisisID]],
152       [[ptBoolean,true]]),
153
154  msrNav([AdtCrisisType],[],[[etCrisisType,small]]),
155  msrNav([AetCrisisStatus],[],[[etCrisisStatus,pending]]),
156  msrNav([ACrisisInstant],[],[AAlertInstant]),
157  msrNav([ACrisisdtComment],
158       [value],
159       [[ptString,'no reporting yet defined']]),
160  msrNav([ActCrisis],[init,[AdtCrisisID,
161                AdtCrisisType,
162                AetCrisisStatus,
163                AdtGPSLocation,
164                ACrisisInstant,
165                ACrisisdtComment]],
166       [[ptBoolean,true]])
167
168   )
169 ; (
170  msrNav(ColctAlertsNearBy,
171      [rnTheCrisis,msrAny,msrTrue],
172      [ActCrisis])
173  )
174 ),
175
176 /* PostF04 */
177
178 msrNav([ActAlert],
179     [msmAtPost,rnTheCrisis],
180     [ActCrisis]),
181
182/* PostF05 */
183
184 msrNav([TheSystem],
185    [rnctHuman,
186     msrSelect,id,eq,[AdtPhoneNumber]],
187    HumanCol1),
188
189 msrNav(HumanCol1,
190    [msrSelect,kind,etEq,[AetHumanKind]],
191    HumanCol2),
192
193 (msrNav(HumanCol2,[msrIsEmpty],[[ptBoolean,true]])
194 -> (msrNav([ActHuman],
195       [init,[AdtPhoneNumber,AetHumanKind]],
196       [[ptBoolean,true]]),
197  msrNav([ActHuman],
198       [msmAtPost,rnactComCompany],
```

```
199        [TheactComCompany])
200    )
201  ; msrNav(HumanCol2,
202      [msrAny],
203      [ActHuman])
204  ),
205
206msrNav([ActHuman],
207     [rnSignaled,msrIncluding,[ActAlert]],
208     ColAlerts),
209
210msrNav([ActHuman],
211     [msmAtPost,rnSignaled],
212     ColAlerts),
213
214/* PostF06 */
215msrNav([AdtSMS],
216     [value],
217     [[ptString,'Your alert has been registered. We will handle it and keep you informed']]),
218msrNav([TheactComCompany],
219     [rnInterfaceIN,
220     ieSmsSend,[AdtPhoneNumber,
221           AdtSMS]],[[ptBoolean,true]]),
222
223/*
224
225 */
226
227 /* Post Protocol:*/
228/* PostP01 */
229 true
230 .
```

Listing 5.16: Messir (Prolog-oriented) implementation of the operation *oeAlert*.

Figure 5.2 shows concept model elements in the scope of the oeAlert operation

Figure 5.3 shows concept model elements in the scope of the oeAlert operation

5.5 Environment - Out Interface Operation Scheme for actCoordinator

5.5.1 Operation Model for oeAddVictim

The `oeAddVictim` operation has the following properties:

OPERATION	
oeAddVictim	
sent to add a new victim in the system's post state.	
Parameters	
1	**AdtCrisisID: dtCrisisID** used to initialize the CrisisId field
Return type	
ptBoolean	
Pre-Condition (protocol)	
PreP 1	the system is started
PreP 2	the actor logged previously and did not log out !

continues in next page . . .

. . . Operation table continuation

Post-Condition (functional)	
PostF 1	the ctState attribute for the next value for victim IDs is incremented by one at post.
PostF 2	a new victim instance exists in the post state with victim ID being a string conversion of the dtInteger value available in the pre state in the ctState instance.
PostF 3	the post state relates the new victim to the crisis passed as a parameter.
PostF 4	the coordinator actor is informed about the satisfaction of its request.
Post-Condition (protocol)	
PostP 1	none

The listing 5.17 provides the Messir (MCL-oriented) specification of the operation.

```

/* Pre Protocol:*/
preP{let TheSystem: ctState in
 let TheActor:actCoordinator in
 let AdtVictimID:dtVictimID in

 self.rnActor.rnSystem = TheSystem
 and self.rnActor = TheCoordinator

/* PreP01 */
 and TheSystem.vpStarted = true
/* PreP02 */
 and TheActor.rnctAuthenticated.vpIsLogged = true}

/* Post Functional:*/
postF{let TheSystem: ctState in
 let TheActor:actCoordinator in
 let AdtVictimID:dtVictimID in

 self.rnActor.rnSystem = TheSystem
 and self.rnActor = TheActor

/* PostF01 */
 TheSystem.nextValueForVictimID=PrenextValueForVictimID
 and PrenextValueForVictimID.add(1) = PostnextValueForVictimID
 and TheSystem@post.nextValueForVictimID = PostnextValueForVictimID

/* PostF02 */
and TheSystem.nextValueForVictimID.todtString().eq(AdtVictimID)

and ActVictim.init(AdtVictimID)

and
TheSystem.rnCtCrisis@post.includes(ActVictim)

/* PostF03 */
and TheActor.rnInterfaceIN^ieVictimAdded()}

/* Post Protocol:*/
postP{ true}
```

Listing 5.17: Messir (MCL-oriented) specification of the operation *oeAddVictim*.

5.5.2 Operation Model for oeCloseCrisis

The `oeCloseCrisis` operation has the following properties:

Figure 5.2: oeAlert operation scope

Figure 5.3: oeAlert operation scope

OPERATION	
oeCloseCrisis	
sent to indicate that a crisis should be considered as closed.	
Parameters	
1	**AdtCrisisID: dtCrisisID** the identification information used to determine the crisis to close
Return type	
ptBoolean	
Pre-Condition (protocol)	
PreP 1	the system is started
PreP 2	the actor logged previously and did not log out ! (i.e. the associated ctCoordinator instance is considered logged)
Pre-Condition (functional)	
PreF 1	it is supposed that there exist one ctCrisis instance with the same `id` attribute value as the one provided by the coordinator actor who wants to close.
Post-Condition (functional)	
PostF 1	the ctCrisis class instance having the provided id is considered closed in the post state.
PostF 2	There is no handler declared in the system as associated to the crisis.
PostF 3	all the alert instances associated to this crisis do not belong any more to the system's post state.
PostF 4	the coordinator actor is informed about the satisfaction of its request.
Post-Condition (protocol)	
PostP 1	none

The listing 5.18 provides the Messir (Prolog-oriented) implementation of the operation.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%-------------------------------------------------------
msrop(outactCoordinator,
   oeCloseCrisis,
   [preProtocol,Self,
    AdtCrisisID
    ],
   []):-
/* Pre Protocol:*/
msrVar(ctState,TheSystem),
msrVar(actCoordinator,TheActor),
msrNav([Self],[rnActor,rnSystem],[TheSystem]),
msrNav([Self],[rnActor],[TheActor]),

/* PreP01 */
msrNav([TheSystem],
    [vpStarted],
    [[ptBoolean,true]]),

/* PreP02 */
msrNav([TheActor],
    [rnctAuthenticated,vpIsLogged],
    [[ptBoolean,true]])
.

```

```
29msrop(outactCoordinator,
30   oeCloseCrisis,
31   [preFunctional,Self,
32    AdtCrisisID
33    ],
34   []):-
35/* Pre Functional:*/
36 msrVar(ctState,TheSystem),
37 msrVar(actCoordinator,TheActor),
38
39 msrVar(dtCrisisID,AdtCrisisID),
40
41 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
42 msrNav([Self],[rnActor],[TheActor]),
43
44/* PreF01 */
45 msrNav([TheSystem],
46    [rnctCrisis,
47     msrSelect,
48     id,eq,[AdtCrisisID]
49    ],
50    ColCrisis),
51
52 msrNav(ColCrisis,
53     [msrSize,eq,[[ptInteger,1]]],
54     [[ptBoolean,true]])
55 .
56
57msrop(outactCoordinator,
58   oeCloseCrisis,
59   [post,Self,
60    AdtCrisisID
61    ],
62    []):-
63
64/* Post Functional:*/
65 msrVar(ctState,TheSystem),
66 msrVar(actCoordinator,TheActor),
67
68 msrVar(ctCrisis,TheCrisis),
69 msrVar(dtCrisisID,AdtCrisisID),
70
71 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
72 msrNav([Self],[rnActor],[TheActor]),
73
74/* PostF01 */
75 msrNav([TheSystem],
76    [rnctCrisis,
77     msrSelect,
78     id,eq,[AdtCrisisID]],
79    [TheCrisis]),
80
81 msrNav([TheCrisis],
82     [msmAtPost,status],
83     [[etCrisisStatus,closed]]),
84
85/* PostF02 */
86 msrNav([TheCrisis],
87     [msmAtPost,rnHandler],
88     []),
89
90 /* PostF03 */
91 msrNav([TheCrisis],
92     [rnAlerts,msrForAll,msrIsKilled],
93     [[ptBoolean,true]]),
94
95/* PostF04 */
96 msrNav([TheActor],
97     [rnInterfaceIN,
98     ieMessage,[[ptString,'The crisis is now closed !']]
```

```
    ],
    [[ptBoolean,true]]),

/* Post Protocol:*/
/* PostP01 */
true
.
```

Listing 5.18: Messir (Prolog-oriented) implementation of the operation *oeCloseCrisis*.

5.5.3 Operation Model for oeDeleteInjury

The `oeDeleteInjury` operation has the following properties:

OPERATION	
oeDeleteInjury sent to delete an existing Injury in the system's post.	
Parameters	
1	**AdtInjuryID: dtInjuryID** used for ctInjury instance retrieval
Return type	
ptBoolean	
Pre-Condition (protocol)	
PreP 1	the system is started
PreP 2	the actor logged previously and did not log out ! (i.e. the associated ctAdministrator instance is considered logged)
Pre-Condition (functional)	
PreF 1	it is supposed that there exist one ctInjury instance with the same `id` attribute that one of the coordinators want to delete.
Post-Condition (functional)	
PostF 1	the ctInjury class instance having the required id do not belong anymore to the post state.
PostF 2	the coordinator actor is informed about the satisfaction of its request.
Post-Condition (protocol)	
PostP 1	none

The listing 5.19 provides the Messir (MCL-oriented) specification of the operation.

```

/* Pre Protocol:*/
preP{let TheSystem: ctState in
 let TheActor:actCoordinator in
 let AdtInjuryID:dtInjuryID in

 self.rnActor.rnSystem = TheSystem
 and self.rnActor = TheCoordinator

/* PreP01 */
 and TheSystem.vpStarted = true
/* PreP02 */
 and TheActor.rnctAuthenticated.vpIsLogged = true}

/* Pre Functional:*/
preF{/* PreF01 */
```

```
17  TheSystem.rnctInjury->select(id.eq(AdtInjuryID))
18  = ColctInjury
19  and ColctInjury->size().eq(1)}
20
21 /* Post Functional:*/
22 postF{let TheSystem: ctState in
23  let TheActor:actCoordinator in
24
25  self.rnActor.rnSystem = TheSystem
26  and self.rnActor = TheActor
27
28 /* PostF01 */
29  TheSystem.rnctInjury->select(id.eq(AdtInjuryID))
30  = ColctInjury
31  and ColctInjury->size().eq(0)
32 /* PostF02 */
33  and TheActor.rnInterfaceIN^ieInjuryDeleted()}
34
35 /* Post Protocol:*/
36 postP{ true}
```

Listing 5.19: Messir (MCL-oriented) specification of the operation *oeDeleteInjury*.

5.5.4 Operation Model for oeDeleteVictim

The `oeDeleteVictim` operation has the following properties:

OPERATION	
oeDeleteVictim sent to delete an existing Victims in the system's post.	
Parameters	
1	**AdtVictimID: dtVictimID** used for ctVictim instance retrieval
Return type	
ptBoolean	
Pre-Condition (protocol)	
PreP 1	the system is started
PreP 2	the actor logged previously and did not log out ! (i.e. the associated ctAdministrator instance is considered logged)
Pre-Condition (functional)	
PreF 1	it is supposed that there exist one ctVictim instance with the same `id` attribute that one of the coordinators want to delete.
Post-Condition (functional)	
PostF 1	the ctVictim class instance having the required id do not belong anymore to the post state.
PostF 2	the coordinator actor is informed about the satisfaction of its request.
Post-Condition (protocol)	
PostP 1	none

The listing 5.20 provides the Messir (MCL-oriented) specification of the operation.

```
1
2 /* Pre Protocol:*/
3 preP{let TheSystem: ctState in
```

```
 let TheActor:actCoordinator in
 let AdtVictimID:dtVictimID in

 self.rnActor.rnSystem = TheSystem
 and self.rnActor = TheCoordinator

/* PreP01 */
 and TheSystem.vpStarted = true
/* PreP02 */
 and TheActor.rnctAuthenticated.vpIsLogged = true
/* PreF01 */
 TheSystem.rnctVictim->select(id.eq(AdtVictimID))
 = ColctVictims
 and ColctVictims->size().eq(1)}

/* Post Functional:*/
postF{let TheSystem: ctState in
 let TheActor:actCoordinator in

 self.rnActor.rnSystem = TheSystem
 and self.rnActor = TheActor

/* PostF01 */
 TheSystem.rnctVictim@post->select(id.eq(AdtVictimID))
 = ColctVictims
 and ColctVictims->size().eq(0)
/* PostF02 */
 and TheActor.rnInterfaceIN^ieVictimDeleted()}

/* Post Protocol:*/
postP{ true}
```

Listing 5.20: Messir (MCL-oriented) specification of the operation *oeDeleteVictim*.

5.5.5 Operation Model for oeGetAlertsSet

The `oeGetAlertsSet` operation has the following properties:

OPERATION	
oeGetAlertsSet	
sent to request all the ctAlert instances having a specific status.	
Parameters	
1	**AetAlertStatus: etAlertStatus** the criteria used to select the alerts to send back to the actor
Return type	
ptBoolean	
Pre-Condition (protocol)	
PreP 1	the system is started
PreP 2	the actor logged previously and did not log out ! (i.e. the associated ctCoordinator instance is considered logged)
Pre-Condition (functional)	
PreF 1	none
Post-Condition (functional)	
PostF 1	the post state is the one obtained by satisfying the `isSentToCoordinator` predicate for each alert having the provided status and for the actor sending the message. (cf. specification of `isSentToCoordinator` predicate given for the `ctAlert` type.
Post-Condition (protocol)	
PostP 1	none

The listing 5.21 provides the Messir (Prolog-oriented) implementation of the operation.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%-----------------------------------------------------
msrop(outactCoordinator,
   oeGetAlertsSet,
   [preProtocol,Self,
    AetAlertStatus
    ],
   []):-
/* Pre Protocol:*/
 msrVar(ctState,TheSystem),
 msrVar(actCoordinator,TheActor),
 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
 msrNav([Self],[rnActor],[TheActor]),

/* PreP01 */
 msrNav([TheSystem],
    [vpStarted],
    [[ptBoolean,true]]),

 msrNav([TheActor],
    [rnctAuthenticated,vpIsLogged],
    [[ptBoolean,true]])
.

msrop(outactCoordinator,
   oeGetAlertsSet,
   [preFunctional,Self,
    AetAlertStatus
    ],
   []):-
/* Pre Functional:*/
/* PreF01 */
true
 .

msrop(outactCoordinator,
   oeGetAlertsSet,
   [post,Self,
    AetAlertStatus
    ],
    []):-

/* Post Functional:*/
 msrVar(ctState,TheSystem),
 msrVar(actCoordinator,TheActor),
 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
 msrNav([Self],[rnActor],[TheActor]),

/* PostF01 */
 msrNav([TheSystem],
    [rnctAlert,
     msrSelect,
     status,etEq,[AetAlertStatus]],
    ColAlertSet),

 msrNav(ColAlertSet,
     [msrForAll,isSentToCoordinator,[TheActor]],
     [[ptBoolean,true]]),

 /* Post Protocol:*/
/* PostP01 */
 true
```

```
.
```

Listing 5.21: Messir (Prolog-oriented) implementation of the operation *oeGetAlertsSet*.

5.5.6 Operation Model for oeGetCrisisSet

The `oeGetCrisisSet` operation has the following properties:

Operation	
oeGetCrisisSet	
sent to request all the ctCrisis instances having a specific status.	
Parameters	
1	**AetCrisisStatus: etCrisisStatus** the status information used to determine the crisis to send back to the actor
Return type	
ptBoolean	
Pre-Condition (protocol)	
PreP 1	the system is started
PreP 2	the actor logged previously and did not log out ! (i.e. the associated ctCoordinator instance is considered logged)
Pre-Condition (functional)	
PreF 1	none
Post-Condition (functional)	
PostF 1	the post state is the one obtained by satisfying the `isSentToCoordinator` predicate for each crisis having the provided status and for the actor sending the message `ieSendACrisis`. (cf. specification of `isSentToCoordinator` predicate given for the `ctCrisis` type.
Post-Condition (protocol)	
PostP 1	none

The listing 5.22 provides the Messir (Prolog-oriented) implementation of the operation.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%-------------------------------------------------------
msrop(outactCoordinator,
   oeGetCrisisSet,
   [preProtocol,Self,
    AetCrisisStatus
    ],
   []):-
/* Pre Protocol:*/
 msrVar(ctState,TheSystem),
 msrVar(actCoordinator,TheActor),
 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
 msrNav([Self],[rnActor],[TheActor]),

/* PreP01 */
 msrNav([TheSystem],
    [vpStarted],
```

```
21     [[ptBoolean,true]]),
22
23 msrNav([TheActor],
24     [rnctAuthenticated,vpIsLogged],
25     [[ptBoolean,true]])
26 .
27
28msrop(outactCoordinator,
29    oeGetCrisisSet,
30    [preFunctional,Self,
31     AetCrisisStatus
32     ],
33    []):-
34/* Pre Functional:*/
35/* PreF01 */
36true
37 .
38
39msrop(outactCoordinator,
40    oeGetCrisisSet,
41    [post,Self,
42     AetCrisisStatus
43     ],
44     []):-
45
46/* Post Functional:*/
47 msrVar(ctState,TheSystem),
48 msrVar(actCoordinator,TheActor),
49 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
50 msrNav([Self],[rnActor],[TheActor]),
51
52/* PostF01 */
53 msrNav([TheSystem],
54     [rnctCrisis,
55      msrSelect,
56      status,etEq,[AetCrisisStatus]],
57     ColCrisisSet),
58
59 msrNav(ColCrisisSet,
60      [msrForAll,isSentToCoordinator,[TheActor]],
61      [[ptBoolean,true]]),
62
63 /* Post Protocol:*/
64/* PostP01 */
65 true
66 .
```

Listing 5.22: Messir (Prolog-oriented) implementation of the operation *oeGetCrisisSet*.

5.5.7 Operation Model for oeGetInjurySet

The `oeGetInjurySet` operation has the following properties:

OPERATION	
oeGetInjurySet sent to request all the ctInjury instances related to given victim.	
Parameters	
1	**AdtVictimID: dtVictimID** the victim id information used to determine the injuries to send back to the actor
Return type	
ptBoolean	
Pre-Condition (protocol)	

continues in next page . . .

. . . Operation table continuation

PreP 1	the system is started
PreP 2	the actor logged previously and did not log out ! (i.e. the associated ctCoordinator instance is considered logged)
Pre-Condition (functional)	
PreF 1	none
Post-Condition (functional)	
PostF 1	the post state is the one obtained by satisfying the `isSentToCoordinator` predicate for each Injury related to the provided victim id and for the actor sending the message `ieSendAInjuryToCoordinator`. (cf. specification of `isSentToCoordinator` predicate given for the `ctInjury` type.
Post-Condition (protocol)	
PostP 1	none

The listing 5.23 provides the Messir (MCL-oriented) specification of the operation.

```

/* Pre Protocol:*/
preP{let TheSystem: ctState in
 let TheActor:actCoordinator in

 self.rnActor.rnSystem = TheSystem
 and self.rnActor = TheCoordinator

/* PreP01 */
 and TheSystem.vpStarted = true
/* PreP02 */
 and TheActor.rnctAuthenticated.vpIsLogged = true}

/* Post Functional:*/
postF{let TheSystem: ctState in
 let TheActor:actCoordinator in

 self.rnActor.rnSystem = TheSystem
 and self.rnActor = TheActor

/* PostF01 */
TheSystem.rnCtInjury -> select(ctVictim.id = dtVictimID) = colVictimsInjury
and colVictimsInjury->forAll(T:ctInjury | T.isSentToACoordinator(TheActor).eq(true))}

/* Post Protocol:*/
postP{ true}
```

Listing 5.23: Messir (MCL-oriented) specification of the operation *oeGetInjurySet*.

5.5.8 Operation Model for oeGetVictimSet

The `oeGetVictimSet` operation has the following properties:

OPERATION
oeGetVictimSet sent to request all the ctVictim instances related to given crisis.
Parameters

continues in next page . . .

. . . Operation table continuation

1	**AdtCrisisID: dtCrisisID** the crisis id information used to determine the victims to send back to the actor
Return type	
ptBoolean	
Pre-Condition (protocol)	
PreP 1	the system is started
PreP 2	the actor logged previously and did not log out ! (i.e. the associated ctCoordinator instance is considered logged)
Pre-Condition (functional)	
PreF 1	none
Post-Condition (functional)	
PostF 1	the post state is the one obtained by satisfying the `isSentToCoordinator` predicate for each victim related to the provided crisis id and for the actor sending the message `ieSendAVictimToCoordinator`. (cf. specification of `isSentToCoordinator` predicate given for the `ctVictim` type.
Post-Condition (protocol)	
PostP 1	none

The listing 5.24 provides the Messir (MCL-oriented) specification of the operation.

```

/* Pre Protocol:*/
preP{let TheSystem: ctState in
 let TheActor:actCoordinator in

 self.rnActor.rnSystem = TheSystem
 and self.rnActor = TheCoordinator

/* PreP01 */
 and TheSystem.vpStarted = true
/* PreP02 */
 and TheActor.rnctAuthenticated.vpIsLogged = true}

/* Post Functional:*/
postF{let TheSystem: ctState in
 let TheActor:actCoordinator in

 self.rnActor.rnSystem = TheSystem
 and self.rnActor = TheActor

/* PostF01 */
TheSystem.rnCtVictim -> select(ctCrisis.id = AdtCrisisID) = colCrisisVictims
and colCrisisVictims->forAll(T:ctVictim | T.isSentToACoordinator(TheActor).eq(true))}

/* Post Protocol:*/
postP{ true}
```

Listing 5.24: Messir (MCL-oriented) specification of the operation *oeGetVictimSet*.

5.5.9 Operation Model for oeInvalidateAlert

The `oeInvalidateAlert` operation has the following properties:

OPERATION	
oeInvalidateAlert sent to indicate that an alert should be considered as closed.	
Parameters	
1	**AdtAlertID: dtAlertID** the identification information used to determine the alert to close
Return type	
ptBoolean	
Pre-Condition (protocol)	
PreP 1	the system is started
PreP 2	the actor logged previously and did not log out ! (i.e. the associated ctCoordinator instance is considered logged)
Pre-Condition (functional)	
PreF 1	it is supposed that there exist one ctAlert instance with the same `id` attribute value as the one provided by the coordinator actor who wants to close.
Post-Condition (functional)	
PostF 1	the ctAlert class instance having the provided id is considered closed in the post state.
PostF 2	the coordinator actor is informed about the satisfaction of its request.
Post-Condition (protocol)	
PostP 1	none

The listing 5.25 provides the Messir (Prolog-oriented) implementation of the operation.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%-------------------------------------------------------
msrop(outactCoordinator,
   oeInvalidateAlert,
   [preProtocol,Self,
    AdtAlertID
    ],
   []):-
/* Pre Protocol:*/
 msrVar(ctState,TheSystem),
 msrVar(actCoordinator,TheActor),
 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
 msrNav([Self],[rnActor],[TheActor]),

/* PreP01 */
 msrNav([TheSystem],
    [vpStarted],
    [[ptBoolean,true]]),

/* PreP02 */
 msrNav([TheActor],
    [rnctAuthenticated,vpIsLogged],
    [[ptBoolean,true]])
.

msrop(outactCoordinator,
   oeInvalidateAlert,
   [preFunctional,Self,
    AdtAlertID
    ],
```

```
34   []):-
35/* Pre Functional:*/
36 msrVar(ctState,TheSystem),
37 msrVar(actCoordinator,TheActor),
38
39 msrVar(dtAlertID,AdtAlertID),
40
41 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
42 msrNav([Self],[rnActor],[TheActor]),
43
44/* PreF01 */
45 msrNav([TheSystem],
46    [rnctAlert,
47     msrSelect,
48     id,eq,[AdtAlertID]
49    ],
50    ColAlert),
51
52 msrNav(ColAlert,
53     [msrSize,eq,[[ptInteger,1]]],
54     [[ptBoolean,true]])
55 .
56
57msrop(outactCoordinator,
58   oeInvalidateAlert,
59   [post,Self,
60    AdtAlertID
61    ],
62    []):-
63
64/* Post Functional:*/
65 msrVar(ctState,TheSystem),
66 msrVar(actCoordinator,TheActor),
67
68 msrVar(ctAlert,TheAlert),
69 msrVar(dtAlertID,AdtAlertID),
70
71 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
72 msrNav([Self],[rnActor],[TheActor]),
73
74/* PostF01 */
75 msrNav([TheSystem],
76    [rnctAlert,
77     msrSelect,
78     id,eq,[AdtAlertID]],
79    [TheAlert]),
80
81 msrNav([TheAlert],
82     [msmAtPost,status],
83     [[etAlertStatus,invalid]]),
84
85/* PostF02 */
86 msrNav([TheActor],
87     [rnInterfaceIN,
88     ieMessage,[[ptString,'The alert is now declared as invalid !']]
89     ],
90     [[ptBoolean,true]]),
91
92/* Post Protocol:*/
93/* PostP01 */
94 true
95 .
```

Listing 5.25: Messir (Prolog-oriented) implementation of the operation *oeInvalidateAlert*.

5.5.10 Operation Model for oeReportOnCrisis

The `oeReportOnCrisis` operation has the following properties:

OPERATION	
oeReportOnCrisis	
sent to update the textual information available for a specific handled crisis.	
Parameters	
1	**AdtCrisisID: dtCrisisID** the identification information used to determine the crisis to report on
2	**AdtComment: dtComment** the textual information commenting the crisis
Return type	
ptBoolean	
Pre-Condition (protocol)	
PreP 1	the system is started
PreP 2	the actor logged previously and did not log out ! (i.e. the associated ctCoordinator instance is considered logged)
Pre-Condition (functional)	
PreF 1	it is supposed that there exist one crisis in the pre state having the given id.
Post-Condition (functional)	
PostF 1	the comment attribute of the crisis instance having the given id is replaced by the given one and the requesting actor is notified of this update.
Post-Condition (protocol)	
PostP 1	none

The listing 5.26 provides the Messir (Prolog-oriented) implementation of the operation.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%-------------------------------------------------------
msrop(outactCoordinator,
   oeReportOnCrisis,
   [preProtocol,Self,
    AdtCrisisID,
    AdtComment
    ],
   []):-
/* Pre Protocol:*/
 msrVar(ctState,TheSystem),
 msrVar(actCoordinator,TheActor),
 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
 msrNav([Self],[rnActor],[TheActor]),

/* PreP01 */
 msrNav([TheSystem],
    [vpStarted],
    [[ptBoolean,true]]),

 msrNav([TheActor],
    [rnctAuthenticated,vpIsLogged],
    [[ptBoolean,true]])
.

msrop(outactCoordinator,
   oeReportOnCrisis,
   [preFunctional,Self,
```

```
32    AdtCrisisID,
33    AdtComment
34    ],
35   []):-
36/* Pre Functional:*/
37 msrVar(ctState,TheSystem),
38 msrVar(actCoordinator,TheActor),
39
40 msrVar(dtCrisisID,AdtCrisisID),
41
42 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
43 msrNav([Self],[rnActor],[TheActor]),
44
45/* PreF01 */
46 msrNav([TheSystem],
47    [rnctCrisis,
48     msrSelect,
49     id,eq,[AdtCrisisID]
50    ],
51    ColCrisis),
52
53 msrNav(ColCrisis,
54     [msrSize,eq,[[ptInteger,1]]],
55     [[ptBoolean,true]])
56 .
57
58msrop(outactCoordinator,
59   oeReportOnCrisis,
60   [post,Self,
61    AdtCrisisID,
62    AdtComment
63    ],
64    []):-
65
66/* Post Functional:*/
67 msrVar(ctState,TheSystem),
68 msrVar(actCoordinator,TheActor),
69
70 msrVar(ctCrisis,TheCrisis),
71 msrVar(dtCrisisID,AdtCrisisID),
72 msrVar(dtComment,AdtComment),
73
74 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
75 msrNav([Self],[rnActor],[TheActor]),
76
77/* PostF01 */
78 msrNav([TheSystem],
79    [rnctCrisis,
80     msrSelect,
81     id,eq,[AdtCrisisID]],
82    [TheCrisis]),
83
84 msrNav([TheCrisis],
85     [msmAtPost,comment],
86     [AdtComment]),
87
88 msrNav([TheActor],
89     [rnInterfaceIN,
90     ieMessage,[[ptString,'The crisis comment has been updated !']]
91     ],
92     [[ptBoolean,true]]),
93
94/* Post Protocol:*/
95/* PostP01 */
96 true
97 .
```

Listing 5.26: Messir (Prolog-oriented) implementation of the operation *oeReportOnCrisis*.

5.5.11 Operation Model for oeSetCrisisHandler

The `oeSetCrisisHandler` operation has the following properties:

OPERATION	
oeSetCrisisHandler	
sent to declare himself as been the handler of a crisis having the specified id.	
Parameters	
1	**AdtCrisisID: dtCrisisID** the identification information used to determine the crisis
Return type	
ptBoolean	
Pre-Condition (protocol)	
PreP 1	the system is started
PreP 2	the actor logged previously and did not log out ! (i.e. the associated ctCoordinator instance is considered logged)
Pre-Condition (functional)	
PreF 1	there exist one crisis having the given id in the pre-state.
Post-Condition (functional)	
PostF 1	the ctCrisis instance having the provided id is in handled status at poststate and is associated to the actor that sends the message (which himself is notified with a textual message as confirmation).
PostF 2	All the alerts related to this crisis are sent to the actor such that he can decide how to handle them.
PostF 3	if the crisis was already handled at pre-sate then the associated handler actor is notified about the change of handler for one of his crisis (n.b. it might be the same even if not relevant).
PostF 4	a message is sent to the communication company for any human related to an alert associated to the crisis. A human will receive as many messages as alerts he sent despite the fact that they might relate to the same crisis (i.e. one alert, one acknoledgement).
Post-Condition (protocol)	
PostP 1	none

The listing 5.27 provides the Messir (Prolog-oriented) implementation of the operation.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%-------------------------------------------------------
msrop(outactCoordinator,
   oeSetCrisisHandler,
   [preProtocol,Self,
    AdtCrisisID
    ],
   []):-
/* Pre Protocol:*/
 msrVar(ctState,TheSystem),
 msrVar(actCoordinator,TheActor),
 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
 msrNav([Self],[rnActor],[TheActor]),
```

```
17
18/* PreP01 */
19 msrNav([TheSystem],
20    [vpStarted],
21    [[ptBoolean,true]]),
22
23 msrNav([TheActor],
24    [rnctAuthenticated,vpIsLogged],
25    [[ptBoolean,true]])
26.
27
28msrop(outactCoordinator,
29   oeSetCrisisHandler,
30   [preFunctional,Self,
31    AdtCrisisID
32    ],
33   []):-
34/* Pre Functional:*/
35 msrVar(ctState,TheSystem),
36 msrVar(actCoordinator,TheActor),
37
38 msrVar(dtCrisisID,AdtCrisisID),
39
40 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
41 msrNav([Self],[rnActor],[TheActor]),
42
43/* PreF01 */
44 msrNav([TheSystem],
45    [rnctCrisis,
46     msrSelect,
47     id,eq,[AdtCrisisID]
48    ],
49    ColCrisis),
50
51 msrNav(ColCrisis,
52     [msrSize,eq,[[ptInteger,1]]],
53     [[ptBoolean,true]])
54 .
55
56msrop(outactCoordinator,
57   oeSetCrisisHandler,
58   [post,Self,
59    AdtCrisisID
60    ],
61    []):-
62
63/* Post Functional:*/
64 msrVar(ctState,TheSystem),
65 msrVar(actCoordinator,TheActor),
66 msrVar(ctCoordinator,TheCoordinator),
67 msrVar(ctCoordinator,TheCurrentHandler),
68
69 msrVar(ctCrisis,TheCrisis),
70 msrVar(dtCrisisID,AdtCrisisID),
71
72 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
73 msrNav([Self],[rnActor],[TheActor]),
74
75/* PostF01 */
76 msrNav([TheSystem],
77    [rnctCrisis,
78     msrSelect,
79     id,eq,[AdtCrisisID]],
80    [TheCrisis]),
81
82 msrNav([TheCrisis],
83     [msmAtPost,status],
84     [[etCrisisStatus,handled]]),
85
86 msrNav([TheActor],
```

```
    [rnctCoordinator],
    [TheCoordinator]),
msrNav([TheCrisis],
    [msmAtPost,rnHandler],
    [TheCoordinator]),

msrNav([TheActor],
    [rnInterfaceIN,
    ieMessage,[[ptString,'You are now considered as handling the crisis !']]
    ],
    [[ptBoolean,true]]),

/* PostF02 */
msrNav([TheCrisis],
    [rnAlerts,msrForAll,isSentToCoordinator,[TheActor]],
    [[ptBoolean,true]]),

/* PostF03 */
( msrNav([TheCrisis],
     [rnHandler,msrSize,eq,[[ptInteger,1]]],
     [[ptBoolean,true]])
 -> (msrNav([TheCrisis],
        [rnHandler],
        [TheCurrentHandler]),
   msrNav([TheCurrentHandler],
        [rnactCoordinator,rnInterfaceIN,
        ieMessage,[[ptString,'One of the crisis you were handling is now handled by one of your
    colleagues!']]
        ],
        [[ptBoolean,true]])
   )
 ; true
),

 /* PostF04 */
msrNav([TheCrisis],
    [rnAlerts,rnSignaler,msrForAll,isAcknowledged,[]],
    [[ptBoolean,true]]),

/* Post Protocol:*/
/* PostP01 */
true
.
```

Listing 5.27: Messir (Prolog-oriented) implementation of the operation *oeSetCrisisHandler*.

5.5.12 Operation Model for oeSetCrisisStatus

The `oeSetCrisisStatus` operation has the following properties:

OPERATION	
oeSetCrisisStatus sent to define the handling status of a specific crisis.	
Parameters	
1	**AdtCrisisID: dtCrisisID** the identification information used to determine the crisis
2	**AetCrisisStatus: etCrisisStatus** the new status value
Return type	
ptBoolean	
Pre-Condition (protocol)	

continues in next page . . .

. . . Operation table continuation

PreP 1	the system is started
PreP 2	the actor logged previously and did not log out ! (i.e. the associated ctCoordinator instance is considered logged)
Pre-Condition (functional)	
PreF 1	it is supposed that there exist one crisis in the pre state having the given id.
Post-Condition (functional)	
PostF 1	the crisis status attribute of the crisis instance having the given id is replaced by the given one and the requesting actor is notified of this update.
Post-Condition (protocol)	
PostP 1	none

The listing 5.28 provides the Messir (Prolog-oriented) implementation of the operation.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%-------------------------------------------------------
msrop(outactCoordinator,
   oeSetCrisisStatus,
   [preProtocol,Self,
    AdtCrisisID,
    AetCrisisStatus
    ],
   []):-
/* Pre Protocol:*/
 msrVar(ctState,TheSystem),
 msrVar(actCoordinator,TheActor),
 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
 msrNav([Self],[rnActor],[TheActor]),

/* PreP01 */
 msrNav([TheSystem],
    [vpStarted],
    [[ptBoolean,true]]),

 msrNav([TheActor],
    [rnctAuthenticated,vpIsLogged],
    [[ptBoolean,true]])
.

msrop(outactCoordinator,
   oeSetCrisisStatus,
   [preFunctional,Self,
    AdtCrisisID,
    AetCrisisStatus
    ],
   []):-
/* Pre Functional:*/
 msrVar(ctState,TheSystem),
 msrVar(actCoordinator,TheActor),

 msrVar(dtCrisisID,AdtCrisisID),

 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
 msrNav([Self],[rnActor],[TheActor]),

/* PreF01 */
 msrNav([TheSystem],
```

```
47     [rnctCrisis,
48      msrSelect,
49      id,eq,[AdtCrisisID]
50     ],
51     ColCrisis),
52
53 msrNav(ColCrisis,
54      [msrSize,eq,[[ptInteger,1]]],
55      [[ptBoolean,true]])
56 .
57
58msrop(outactCoordinator,
59   oeSetCrisisStatus,
60   [post,Self,
61    AdtCrisisID,
62    AetCrisisStatus
63    ],
64    []):-
65
66/* Post Functional:*/
67 msrVar(ctState,TheSystem),
68 msrVar(actCoordinator,TheActor),
69
70 msrVar(ctCrisis,TheCrisis),
71 msrVar(dtCrisisID,AdtCrisisID),
72 msrVar(etCrisisStatus,AetCrisisStatus),
73
74 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
75 msrNav([Self],[rnActor],[TheActor]),
76
77/* PostF01 */
78 msrNav([TheSystem],
79     [rnctCrisis,
80      msrSelect,
81      id,eq,[AdtCrisisID]],
82     [TheCrisis]),
83
84 msrNav([TheCrisis],
85      [msmAtPost,status],
86      [AetCrisisStatus]),
87
88 msrNav([TheActor],
89      [rnInterfaceIN,
90      ieMessage,[[ptString,'The crisis status has been updated !']]
91      ],
92      [[ptBoolean,true]]),
93
94/* Post Protocol:*/
95/* PostP01 */
96 true
97 .
```

Listing 5.28: Messir (Prolog-oriented) implementation of the operation *oeSetCrisisStatus*.

5.5.13 Operation Model for oeSetCrisisType

The `oeSetCrisisType` operation has the following properties:

OPERATION	
oeSetCrisisType sent to define the gravity type of a specific crisis.	
Parameters	
1	**AdtCrisisID: dtCrisisID** the identification information used to determine the crisis

continues in next page . . .

. . . Operation table continuation

2	**AetCrisisType: etCrisisType** the new type value
Return type	
ptBoolean	
Pre-Condition (protocol)	
PreP 1	the system is started
PreP 2	the actor logged previously and did not log out ! (i.e. the associated ctCoordinator instance is considered logged)
Pre-Condition (functional)	
PreF 1	it is supposed that there exist one crisis in the pre state having the given id.
Post-Condition (functional)	
PostF 1	the crisis type attribute of the crisis instance having the given id is replaced by the given one and the requesting actor is notified of this update.
Post-Condition (protocol)	
PostP 1	none

The listing 5.29 provides the Messir (Prolog-oriented) implementation of the operation.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%-------------------------------------------------------
msrop(outactCoordinator,
   oeSetCrisisType,
   [preProtocol,Self,
    AdtCrisisID,
    AetCrisisType
    ],
   []):-
/* Pre Protocol:*/
 msrVar(ctState,TheSystem),
 msrVar(actCoordinator,TheActor),
 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
 msrNav([Self],[rnActor],[TheActor]),

/* PreP01 */
 msrNav([TheSystem],
    [vpStarted],
    [[ptBoolean,true]]),

 msrNav([TheActor],
    [rnctAuthenticated,vpIsLogged],
    [[ptBoolean,true]])
.

msrop(outactCoordinator,
   oeSetCrisisType,
   [preFunctional,Self,
    AdtCrisisID,
    AetCrisisType
    ],
   []):-
/* Pre Functional:*/
 msrVar(ctState,TheSystem),
 msrVar(actCoordinator,TheActor),

```

```
msrVar(dtCrisisID,AdtCrisisID),

msrNav([Self],[rnActor,rnSystem],[TheSystem]),
msrNav([Self],[rnActor],[TheActor]),

/* PreF01 */
msrNav([TheSystem],
   [rnctCrisis,
    msrSelect,
    id,eq,[AdtCrisisID]
   ],
   ColCrisis),

msrNav(ColCrisis,
    [msrSize,eq,[[ptInteger,1]]],
    [[ptBoolean,true]])
.

msrop(outactCoordinator,
  oeSetCrisisType,
  [post,Self,
   AdtCrisisID,
   AetCrisisType
   ],
   []):-

/* Post Functional:*/
msrVar(ctState,TheSystem),
msrVar(actCoordinator,TheActor),

msrVar(ctCrisis,TheCrisis),
msrVar(dtCrisisID,AdtCrisisID),
msrVar(etCrisisType,AetCrisisType),

msrNav([Self],[rnActor,rnSystem],[TheSystem]),
msrNav([Self],[rnActor],[TheActor]),

/* PostF01 */
msrNav([TheSystem],
   [rnctCrisis,
    msrSelect,
    id,eq,[AdtCrisisID]],
   [TheCrisis]),

msrNav([TheCrisis],
    [msmAtPost,type],
    [AetCrisisType]),

msrNav([TheActor],
    [rnInterfaceIN,
    ieMessage,[[ptString,'The crisis type has been updated !']]
    ],
    [[ptBoolean,true]]),

/* Post Protocol:*/
/* PostP01 */
true
.
```

Listing 5.29: Messir (Prolog-oriented) implementation of the operation *oeSetCrisisType*.

5.5.14 Operation Model for oeValidateAlert

The `oeValidateAlert` operation has the following properties:

OPERATION

continues in next page . . .

. . . Operation table continuation

oeValidateAlert	
sent to indicate that a specific alert is not a fake.	
Parameters	
1	**AdtAlertID: dtAlertID** the identification information used to determine the alert instance
Return type	
ptBoolean	
Pre-Condition (protocol)	
PreP 1	the system is started
PreP 2	the actor logged previously and did not log out ! (i.e. the associated ctCoordinator instance is considered logged)
Pre-Condition (functional)	
PreF 1	it is supposed that there exist one ctAlert instance with the same `id` attribute value as the one provided by the coordinator actor who wants to validate.
Post-Condition (functional)	
PostF 1	the ctAlert class instance having the provided id is considered as valid in the post state and the coordinator actor is informed about the satisfaction of its request.
Post-Condition (protocol)	
PostP 1	none

The listing 5.30 provides the Messir (MCL-oriented) specification of the operation.

```
1
2  /* Pre Protocol:*/
3  preP{let TheSystem: ctState in
4   let TheActor:actCoordinator in
5
6   self.rnActor.rnSystem = TheSystem
7   and self.rnActor = TheCoordinator
8
9  /* PreP01 */
10  and TheSystem.vpStarted = true
11 /* PreP02 */
12  and TheActor.rnctAuthenticated.vpIsLogged = true}
13
14 /* Pre Functional:*/
15 preF{/* PreF01 */
16  TheSystem.rnctAlert->select(id.eq(AdtAlertID))
17  = ColctAlert
18  and ColctAlert->size().eq(1)}
19
20 /* Post Functional:*/
21 postF{let TheSystem: ctState in
22  let TheActor:actAuthenticated in
23
24  self.rnActor.rnSystem = TheSystem
25  and self.rnActor = TheActor
26
27  and TheSystem.rnctAlert->select(id.eq(AdtAlertID))
28  = ColctAlert
29
30  /* PostF01 */
31  and if (ColctAlert->forAll(T:ctAlert | T.corruption=regular or T.corruption=restored ))
32  then(
33  TheActor.rnInterfaceIN^ieMessage('The Alert is now declared as valid !')
```

```
34  true
35  )
36  else(
37  TheActor.rnInterfaceIN^ieMessage('Can\'t validate corrupted alert !')
38  )
39  endif}
40
41 /* Post Protocol:*/
42 postP{ let TheSystem: ctState in
43
44  self.rnActor.rnSystem = TheSystem
45
46  TheSystem.rnctAlert->select(id.eq(AdtAlertID))
47  = ColctAlert
48
49  /* PostP01 */
50  and if (ColctAlert->forAll(T:ctAlert | T.corruption=regular or T.corruption=restored ))
51  then(
52  ColctAlert->forAll(T:ctAlert | T.status@post=valid)
53  )
54  else true
55  endif}
```

Listing 5.30: Messir (MCL-oriented) specification of the operation *oeValidateAlert*.

The listing 5.31 provides the Messir (Prolog-oriented) implementation of the operation.

```
1%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5%-------------------------------------------------------
6msrop(outactCoordinator,
7   oeValidateAlert,
8   [preProtocol,Self,
9    AdtAlertID
10   ],
11  []):-
12/* Pre Protocol:*/
13 msrVar(ctState,TheSystem),
14 msrVar(actCoordinator,TheActor),
15 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
16 msrNav([Self],[rnActor],[TheActor]),
17
18/* PreP01 */
19 msrNav([TheSystem],
20    [vpStarted],
21    [[ptBoolean,true]]),
22
23 msrNav([TheActor],
24    [rnctAuthenticated,vpIsLogged],
25    [[ptBoolean,true]])
26.
27
28msrop(outactCoordinator,
29   oeValidateAlert,
30   [preFunctional,Self,
31    AdtAlertID
32   ],
33  []):-
34/* Pre Functional:*/
35 msrVar(ctState,TheSystem),
36 msrVar(actCoordinator,TheActor),
37
38 msrVar(dtAlertID,AdtAlertID),
39
```

```
40 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
41 msrNav([Self],[rnActor],[TheActor]),
42
43/* PreF01 */
44 msrNav([TheSystem],
45    [rnctAlert,
46     msrSelect,
47     id,eq,[AdtAlertID]
48    ],
49    ColAlerts),
50
51 msrNav(ColAlerts,
52     [msrSize,eq,[[ptInteger,1]]],
53     [[ptBoolean,true]])
54 .
55
56msrop(outactCoordinator,
57   oeValidateAlert,
58   [post,Self,
59    AdtAlertID
60    ],
61    []):-
62
63/* Post Functional:*/
64 msrVar(ctState,TheSystem),
65 msrVar(actCoordinator,TheActor),
66
67 msrVar(ctAlert,TheAlert),
68 msrVar(dtAlertID,AdtAlertID),
69
70 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
71 msrNav([Self],[rnActor],[TheActor]),
72
73/* PostF01 */
74 msrNav([TheSystem],
75    [rnctAlert,
76     msrSelect,
77     id,eq,[AdtAlertID]],
78    [TheAlert]),
79
80 msrNav([TheAlert],
81     [msmAtPost,status],
82     [[etAlertStatus,valid]]),
83
84 msrNav([TheActor],
85     [rnInterfaceIN,
86     ieMessage,[[ptString,'The Alert is now declared as valid !']]
87     ],
88     [[ptBoolean,true]]),
89
90 /* Post Protocol:*/
91/* PostP01 */
92 true
93 .
```

Listing 5.31: Messir (Prolog-oriented) implementation of the operation *oeValidateAlert*.

5.5.15 Operation Model for oeAddInjury

The `oeAddInjury` operation has the following properties:

OPERATION
oeAddInjury   sent to add a new injury in the system's post state and environment's post state.
Parameters

continues in next page . . .

. . . Operation table continuation

1	**AdtVictimID: dtVictimID** used to initialize the VictimId field
2	**AetInjuryKind: etInjuryKind** used to initialize the InjuryKind field
Return type	
ptBoolean	
Pre-Condition (protocol)	
PreP 1	the system is started
PreP 2	the actor logged previously and did not log out !
Post-Condition (functional)	
PostF 1	the ctState attribute for the next value for injury IDs is incremented by one at post.
PostF 2	a new injury instance exists in the post state with etInjuryKind and injury ID being a string conversion of the dtInteger value available in the pre state in the ctState instance.
PostF 3	the post state relates the new injury to the victim passed as a parameter.
PostF 4	the coordinator actor is informed about the satisfaction of its request.
Post-Condition (protocol)	
PostP 1	none

The listing 5.32 provides the Messir (MCL-oriented) specification of the operation.

```

/* Pre Protocol:*/
preP{let TheSystem: ctState in
 let TheActor:actCoordinator in
 let AdtInjuryID:dtInjuryID in

 self.rnActor.rnSystem = TheSystem
 and self.rnActor = TheCoordinator

/* PreP01 */
 and TheSystem.vpStarted = true
/* PreP02 */
 and TheActor.rnctAuthenticated.vpIsLogged = true}

/* Post Functional:*/
postF{let TheSystem: ctState in
 let TheActor:actCoordinator in

 self.rnActor.rnSystem = TheSystem
 and self.rnActor = TheActor

/* PostF01 */
 TheSystem.nextValueForInjuryID=PrenextValueForInjuryID
 and PrenextValueForInjuryID.add(1) = PostnextValueForInjuryID
 and TheSystem@post.nextValueForInjuryID = PostnextValueForInjuryID

/* PostF02 */
and TheSystem.nextValueForInjuryID.todtString().eq(AdtInjuryID)
and ActInjury.init(AdtInjuryID, AetInjuryKind)

/* PostF03 */
TheSystem.rnCtVictim -> select(ctVictim.id = dtVictimID) = TEMP
TEMP.rnTheInjury->msrAny(true) = ActInjury

/* PostF04 */
and TheActor.rnInterfaceIN^ieInjuryAdded()}
```

```

/* Post Protocol:*/
postP{ true}
```

Listing 5.32: Messir (MCL-oriented) specification of the operation *oeAddInjury*.

5.6 Environment - Out Interface Operation Scheme for actCorrupter

5.6.1 Operation Model for oeCorruptAlert

The `oeCorruptAlert` operation has the following properties:

OPERATION	
oeCorruptAlert[proactive]	
used to corrupt alert data and set its corruption status to corrupted	
Parameters	
1	**ActAlert: ctAlert** actAlert corrupted by corrupter
Return type	
ptBoolean	
Pre-Condition (protocol)	
PreP 1	the system is started
PreP 2	the actor logged previously and did not log out ! (i.e. the associated ctCoordinator instance is considered logged)
Pre-Condition (functional)	
PreF 1	none
Post-Condition (functional)	
PostF 1	in systems post state ctAlert with given dtAlertID is now corrupted and its state is set to etCorruptionKind.corrupted
Post-Condition (protocol)	
PostP 1	none

The listing ?? provides the Messir (MCL-oriented) specification of the operation.

```

/* Pre Protocol:*/
preP{let TheSystem: ctState in
     self.rnActor.rnSystem = TheSystem

    /* PreP01 */
    and TheSystem.vpStarted = true
    /* PreP02 */
    TheSystem.rnctAlert->select(id.eq(AdtAlertID))
     = ColctAlert
     and ColctAlert->size().eq(1)}

/* Post Functional:*/
postF{let TheSystem: ctState in
     let CorruptedAlert: ctAlert in
     self.rnActor.rnSystem = TheSystem

     /* PostF02 */
```

```
19        and CorruptedAlert.id=ActAlert.id
20        and CorruptedAlert.status=ActAlert.status
21        and CorruptedAlert.location=ActAlert.location
22        and CorruptedAlert.instant=ActAlert.instant
23        and CorruptedAlert.comment=ActAlert.comment
24        and CorruptedAlert.corruption=ActAlert.etCorruptionKind.corrupted
25        and TheSystem.rnctAlert@post->select(id.eq(ActAlert.id)) = CorruptedAlert}
```

Listing 5.33: Messir (MCL-oriented) specification of the operation *oeCorruptAlert*.

5.7 Environment - Out Interface Operation Scheme for actHospital

5.7.1 Operation Model for oeGetHospitalCrisisSet

The `oeGetHospitalCrisisSet` operation has the following properties:

OPERATION	
oeGetHospitalCrisisSet sent to request all the ctCrisis instances having a specific status.	
Parameters	
1	**AetCrisisStatus: etCrisisStatus** the status information used to determine the crisis to send back to the actor
Return type	
ptBoolean	
Pre-Condition (protocol)	
PreP 1	the system is started
PreP 2	the actor logged previously and did not log out ! (i.e. the associated ctCoordinator instance is considered logged)
Pre-Condition (functional)	
PreF 1	none
Post-Condition (functional)	
PostF 1	for each crisis having the provided status, a message `ieSendACrisisToHospital` has been sent with the crisis instance to the calling actor instance.
Post-Condition (protocol)	
PostP 1	none

The listing 5.33 provides the Messir (MCL-oriented) specification of the operation.

```
1
2  /* Pre Protocol:*/
3  preP{let TheSystem: ctState in
4   let TheActor:actHospital in
5
6   self.rnActor.rnSystem = TheSystem
7   and self.rnActor = actHospital
8
9  /* PreP01 */
10  and TheSystem.vpStarted = true
11 /* PreP02 */
12  and TheActor.rnctAuthenticated.vpIsLogged = true}
13
14 /* Post Functional:*/
```

```
postF{let TheSystem: ctState in
 let TheActor:actHospital in
 let ThectHospital:ctHospital in

 self.rnActor.rnSystem = TheSystem
 and self.rnActor = TheActor
 and TheActor.rnctHospital = ThectHospital

/* PostF01 */
 TheSystem.rnCtCrisis -> select(ctCrisis.status = AetCrisisStatus) = colCrisis
 and colCrisis->forAll(T:ctCrisis | T.isSentToAHospital(TheActor).eq(true))}

/* Post Protocol:*/
postP{ true}
```

Listing 5.34: Messir (MCL-oriented) specification of the operation *oeGetHospitalCrisisSet*.

5.7.2 Operation Model for oeGetHospitalInjurySet

The `oeGetHospitalInjurySet` operation has the following properties:

OPERATION	
oeGetHospitalInjurySet	
sent to request all the ctInjury instances related to given victim.	
Parameters	
1	**AdtVictimID: dtVictimID** the victim id information used to determine the injuries to send back to the actor
Return type	
ptBoolean	
Pre-Condition (protocol)	
PreP 1	the system is started
PreP 2	the actor logged previously and did not log out !
Pre-Condition (functional)	
PreF 1	none
Post-Condition (functional)	
PostF 1	the post state is the one obtained by satisfying the `isSentToHospital` predicate for each Injury related to the provided victim id and for the actor sending the message `ieSendAInjuryToHospital`. (cf. specification of `isSentToHospital` predicate given for the `ctInjury` type.
Post-Condition (protocol)	
PostP 1	none

The listing 5.34 provides the Messir (MCL-oriented) specification of the operation.

```

/* Pre Protocol:*/
preP{let TheSystem: ctState in
 let TheActor:actHospital in

 self.rnActor.rnSystem = TheSystem
 and self.rnActor = actHospital

/* PreP01 */
```

```
10  and TheSystem.vpStarted = true
11 /* PreP02 */
12  and TheActor.rnctAuthenticated.vpIsLogged = true}
13
14 /* Post Functional:*/
15 postF{let TheSystem: ctState in
16  let TheActor:actHospital in
17
18  self.rnActor.rnSystem = TheSystem
19  and self.rnActor = TheActor
20
21 /* PostF01 */
22 TheSystem.rnCtInjury -> select(ctVictim.id = dtVictimID) = colVictimInjuries
23 and colVictimInjuries->forAll(T:ctInjury | T.isSentToAHospital(TheActor).eq(true))}
24
25 /* Post Protocol:*/
26 postP{ true}
```

Listing 5.35: Messir (MCL-oriented) specification of the operation *oeGetHospitalInjurySet*.

5.7.3 Operation Model for oeGetHospitalVictimSet

The `oeGetHospitalVictimSet` operation has the following properties:

Operation	
oeGetHospitalVictimSet	
sent to request all the ctVictim instances related to given crisis.	
Parameters	
1	**AdtCrisisID: dtCrisisID** the crisis id information used to determine the victims to send back to the actor
Return type	
ptBoolean	
Pre-Condition (protocol)	
PreP 1	the system is started
PreP 2	the actor logged previously and did not log out !
Pre-Condition (functional)	
PreF 1	none
Post-Condition (functional)	
PostF 1	the post state is the one obtained by satisfying the `isSentToHospital` predicate for each victim related to the provided crisis id and for the actor sending the message `ieSendAVictimToHospital`. (cf. specification of `isSentToHospital` predicate given for the `ctVictim` type.
Post-Condition (protocol)	
PostP 1	none

The listing 5.35 provides the Messir (MCL-oriented) specification of the operation.

```
1
2 /* Pre Protocol:*/
3 preP{let TheSystem: ctState in
4  let TheActor:actHospital in
5
6  self.rnActor.rnSystem = TheSystem
```

```
 and self.rnActor = actHospital

/* PreP01 */
 and TheSystem.vpStarted = true
/* PreP02 */
 and TheActor.rnctAuthenticated.vpIsLogged = true}

/* Post Functional:*/
postF{let TheSystem: ctState in
 let TheActor:actHospital in

 self.rnActor.rnSystem = TheSystem
 and self.rnActor = TheActor

/* PostF01 */
TheSystem.rnCtVictim -> select(ctCrisis.id = dtCrisisID) = colCrisisVictims
and colCrisisVictims->forAll(T:ctVictim | T.isSentToAHospital(TheActor).eq(true))}

/* Post Protocol:*/
postP{ true}
```

Listing 5.36: Messir (MCL-oriented) specification of the operation *oeGetHospitalVictimSet*.

5.8 Environment - Out Interface Operation Scheme for actMsrCreator

5.8.1 Operation Model for oeCreateSystemAndEnvironment

The `oeCreateSystemAndEnvironment` operation has the following properties:

OPERATION	
oeCreateSystemAndEnvironment	
sent to request the initialization of the system's class instances and the environment actors instances.	
Parameters	
1	**AqtyComCompanies: ptInteger** the quantity of communication companies to create in the environment
Return type	
ptBoolean	
Pre-Condition (protocol)	
PreP 1	none
Pre-Condition (functional)	
PreF 1	none
Post-Condition (functional)	
PostF 1	the ctState instance is initialized with the integer 1 for the crisis and alert counters used for their identifications, a value for the clock corresponding to a default inital time (i.e. January 1st, 1970) the crisis reminder period is set to 300 seconds, the maximum crisis reminder period is fixed to 1200 seconds (i.e. 20 minutes), an initial value for the automatic reminder period equal to the current date and time and the system is considered in a started state. **Those predicates must be satisfied first since all the other depend on the existence of a ctState instance !**
PostF 2	the `actMsrCreator` actor instance is initiated (remember that since the `oeCreateSystemAndEnvironment` is a special event it role is to make consistent the post state thus creating the actor and its interfaces is required even though the sending of this message logically would need the actor and its interfaces to already exist ...).

continues in next page . . .

. . . Operation table continuation

PostF 3	the environment for communication company actors, in the post state, is made of `AqtyComCompanies` instances allowing for receiving and sending messages to humans.
PostF 4	the environment for administrator actors, in the post state, is made of one instance.
PostF 5	the environment for activator actors, in the post state, is made of one instance allowing for automatic message sending based on current system's and environment state'.
PostF 6	the set of ctAdministrator instances at post is made of one instance initialized with 'icrashadmin' (resp. '7WXC1359') for login (resp. password) values.
PostF 7	the association between ctAdministrator and actAdministrator is made of one couple made of the conjointly specified instances.
Post-Condition (protocol)	
PostP 1	none is given since the only protocol variable to be modified in the post state is the one initialized with the ctState instance (i.e. vpStarted).

The listing 5.36 provides the Messir (MCL-oriented) specification of the operation.

```

/* Pre Protocol:*/
preP{true}

/* Pre Functional:*/
preF{true}

/* Post Functional:*/
postF{let TheSystem: ctState in
 let AactMsrCreator: actMsrCreator in
 let AactAdministrator: actAdministrator in
 let AnextValueForAlertID: dtInteger in
 let AnextValueForCrisisID: dtInteger in
 let Aclock: dtDateAndTime in
 let AcrisisReminderPeriod: dtSecond in
 let AmaxCrisisReminderPeriod: dtSecond in
 let AvpStarted: ptBoolean in

 /* PostF01 -- MUST ALWAYS BE MADE FIRST -- */
 AnextValueForAlertID.value.eq(1)
 and AnextValueForCrisisID.value.eq(1)
 and Aclock.date.year.value = 1970
 and Aclock.date.month.value = 01
 and Aclock.date.day.value = 01
 and Aclock.time.hour.value = 00
 and Aclock.time.minute.value = 00
 and Aclock.time.second.value = 00

 and AcrisisReminderPeriod.value.eq(300)
 and AmaxCrisisReminderPeriod.value.eq(1200)
 and AvpStarted = true
 and TheSystem.init(AnextValueForAlertID,
           AnextValueForCrisisID,
           Aclock,
           AcrisisReminderPeriod,
           AmaxCrisisReminderPeriod,
           Aclock,
           AvpStarted
          )
 /* PostF02*/
 and AactMsrCreator.init()
 /* PostF03 */
 and let AactComCompanyCol: Bag(actComCompany) in
 AactComCompanyCol->size() = AqtyComCompanies
```

```
45  AactComCompanyCol-> forAll(init())
46  /* PostF04*/
47  and AactAdministrator.init()
48  /* PostF05*/
49  and let AactActivator:actActivator in
50  AactActivator.init()
51 /* PostF06 */
52  and let ActAdministrator:ctAdministrator in
53    let AdtLogin:dtLogin in
54    let AdtPassword:dtPassword in
55    AdtLogin.value.eq('icrashadmin')
56    and AdtPassword.value.eq('7WXC1359')
57    and ActAdministrator.init(AdtLogin,AdtPassword)
58  /* PostF07*/
59  and ActAdministrator@post.rnactAuthenticated = AactAdministrator}
60
61 /* Post Protocol:*/
62 postP{ true}
```

Listing 5.37: Messir (MCL-oriented) specification of the operation *oeCreateSystemAndEnvironment*.

The listing 5.37 provides the Messir (Prolog-oriented) implementation of the operation.

```
1%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5/*
6/****************************************
7MSRCreatorActor
8****************************************/
9
10/*** createSystemAndEnvironment ***/
11
12msrop(outactMsrCreator,
13   oeCreateSystemAndEnvironment,
14   [preFunctional,_Self,_AqtyComCompanies],
15   []):-
16 true.
17
18msrop(outactMsrCreator,
19   oeCreateSystemAndEnvironment,
20   [preProtocol,_Self,_AqtyComCompanies],
21   []):-
22 true.
23
24msrop(outactMsrCreator,
25   oeCreateSystemAndEnvironment,
26   [post,_Self,AqtyComCompanies],
27   []):-
28
29 msrVar(ctState,TheSystem),
30 msrVar(actMsrCreator,AactMsrCreator),
31 msrVar(actAdministrator,AactAdministrator),
32
33 msrVar(dtInteger, AnextValueForAlertID),
34 msrVar(dtInteger, AnextValueForCrisisID),
35 msrVar(dtDateAndTime, Aclock),
36 msrVar(dtSecond, AcrisisReminderPeriod),
37 msrVar(dtSecond, AmaxCrisisReminderPeriod),
38 msrVar(ptBoolean, AvpStarted),
39
40 /* PostF01 -- MUST ALWAYS BE MADE FIRST -- */
41 msrNav([AnextValueForAlertID],
42     [value,eq,[[ptInteger,1]]],
```

```
43      [[ptBoolean,true]]),
44
45  msrNav([AnextValueForCrisisID],
46      [value,eq,[[ptInteger,1]]],
47      [[ptBoolean,true]]),
48
49msrNav([Aclock],
50     [date,year,value],
51     [[ptInteger,1970]]),
52msrNav([Aclock],
53     [date,month,value],
54     [[ptInteger,01]]),
55msrNav([Aclock],
56     [date,day,value],
57     [[ptInteger,01]]),
58
59msrNav([Aclock],
60     [time,hour,value],
61     [[ptInteger,00]]),
62msrNav([Aclock],
63     [time,minute,value],
64     [[ptInteger,00]]),
65msrNav([Aclock],
66     [time,second,value],
67     [[ptInteger,00]]),
68
69  msrNav([AcrisisReminderPeriod],
70      [value,eq,[[ptInteger,300]]],
71      [[ptBoolean,true]]),
72
73  msrNav([AmaxCrisisReminderPeriod],
74      [value,eq,[[ptInteger,1200]]],
75      [[ptBoolean,true]]),
76
77  msrNav([AvpStarted],
78      [],
79      [[ptBoolean,true]]),
80
81  msrNav([TheSystem],
82      [init,[AnextValueForAlertID,
83         AnextValueForCrisisID,
84         Aclock,
85         AcrisisReminderPeriod,
86         AmaxCrisisReminderPeriod,
87         Aclock,
88         AvpStarted]
89         ],
90      [[ptBoolean,true]]),
91
92/* PostF02*/
93  msrNav([AactMsrCreator],
94      [init,[]],
95      [[ptBoolean,true]]),
96
97  /* PostF03 */
98  msrVarCol(actComCompany,AqtyComCompanies,AactComCompanyCol),
99
100 msrNav(AactComCompanyCol,
101     [msrForAll,init,[]],
102     [[ptBoolean,true]]),
103
104 /* PostF04*/
105 msrNav([AactAdministrator],
106     [init,[]],
107     [[ptBoolean,true]]),
108
109 /* PostF05*/
110 msrVar(actActivator,AactActivator),
111 msrNav([AactActivator],
112     [init,[]],
```

```
113     [[ptBoolean,true]]),
114
115/* PostF06 */
116 msrVar(ctAdministrator,ActAdministrator),
117 msrVar(dtLogin,AdtLogin),
118 msrVar(dtPassword,AdtPassword),
119
120 msrNav([AdtLogin],
121     [value,eq,[[ptString,'icrashadmin']]],
122     [[ptBoolean,true]]),
123
124 msrNav([AdtPassword],
125     [value,eq,[[ptString,'7WXC1359']]],
126     [[ptBoolean,true]]),
127
128 msrNav([ActAdministrator],
129     [init,[AdtLogin,AdtPassword]],
130     [[ptBoolean,true]]),
131
132 /* PostF07*/
133 msrNav([ActAdministrator],
134     [msmAtPost,rnactAuthenticated],
135     [AactAdministrator]),
136
137/* Post Protocol:*/
138/* PostP01 */
139true
140.
```

Listing 5.38: Messir (Prolog-oriented) implementation of the operation *oeCreateSystemAndEnvironment*.

Figure 5.4 shows all the concept model elements in the scope of the oeCreateSystemAndEnvironment operation

5.9 Environment - Actor Operation Scheme for actMsrCreator

5.9.1 Operation Model for init

The `init` operation has the following properties:

OPERATION
init used to create an instance of the actor together with its interface instances and update the assocations with the `ctState` instance.
Return type
ptBoolean

5.10 Primary Types - Operation Schemes for Class ctAdministrator

5.10.1 Operation Model for init

The `init` operation has the following properties:

OPERATION
init used to initialize the current object as a new instance of the ctAdministrator type.

continues in next page . . .

. . . Operation table continuation

Parameters	
1	**Alogin: dtLogin** used to initialize the login field
2	**Apwd: dtPassword** used to initialize the password field
Return type	
ptBoolean	
Post-Condition (functional)	
PostF 1	true iff the system poststate includes the current object as a new ctAdministrator instance having its login and password attributes equal to the one provided as parameters and its vpIsLogged attribute equal to false.

The listing 5.38 provides the Messir (MCL-oriented) specification of the operation.

```

/* Post Functional:*/
postF{if
(
let Self:ctAdministrator in
/* Post F01 */
Self.login(Alogin)
and Self.pwd = Apwd
and Self.vpIsLogged = false

/* Post F02 */
and (Self.oclIsNew and self = Self)
)
then (result = true)
else (result = false)
endif}
```

Listing 5.39: Messir (MCL-oriented) specification of the operation *init*.

The listing 5.39 provides the Messir (Prolog-oriented) implementation of the operation.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

msrop(ctAdministrator,init,[Self,
          Alogin,
          Apwd],
   Result):-
(
msrVar(ctAdministrator,Self),

/* Post F01 */
msrNav([Self],[login],[Alogin]),
msrNav([Self],[pwd],[Apwd]),
msrNav([Self],[vpIsLogged],[[ptBoolean,false]]),

/* Post F02 */
 msrNav([Self],[msrIsNew],[Self])
)
```

Figure 5.4: oeCreateSystemAndEnvironment operation scope

```
-> Result = [ptBoolean,true]
; Result = [ptBoolean,false]
.
```

Listing 5.40: Messir (Prolog-oriented) implementation of the operation *init*.

5.11 Primary Types - Operation Schemes for Class ctAlert

5.11.1 Operation Model for init

The `init` operation has the following properties:

OPERATION	
init	
used to initialize the current object as a new instance of the ctAlert type.	
Parameters	
1	**Aid: dtAlertID** used to initialize the id field
2	**Astatus: etAlertStatus** used to initialize the status field
3	**Alocation: dtGPSLocation** used to initialize the location field
4	**Ainstant: dtDateAndTime** used to initialize the instant field
5	**Acomment: dtComment** used to initialize the comment field
Return type	
ptBoolean	
Post-Condition (functional)	
PostF 1	true iff the system poststate includes the current object as a new ctAlert instance having its attributes equal to the ones provided as parameters.

The listing 5.40 provides the Messir (MCL-oriented) specification of the operation.

```

/* Post Functional:*/
postF{let AetCorruptionKind:etCorruptionKind in
AetCorruptionKind = regular

if
(
/* Post F01 */
let Self:ctAlert in
Self.id = Aid
and Self.status = Astatus
and Self.location = Alocation
and Self.instant = Ainstant
and Self.comment = Acomment
and Self.corruption = AetCorruptionKind
/* Post F02 */
and (Self.oclIsNew and self = Self)
)
then (result = true)
```

```
else (result = false)
endif}
```

Listing 5.41: Messir (MCL-oriented) specification of the operation *init*.

The listing 5.41 provides the Messir (Prolog-oriented) implementation of the operation.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

msrop(ctAlert,init,[Self,
          Aid,
          Astatus,
          Alocation,
          Ainstant,
          Acomment],
   Result):-

/* Post F01 */
(
msrVar(ctAlert,Self),

msrNav([Self],[id],[Aid]),
msrNav([Self],[status],[Astatus]),
msrNav([Self],[location],[Alocation]),
msrNav([Self],[instant],[Ainstant]),
msrNav([Self],[comment],[Acomment]),

/* Post F02 */
 msrNav([Self],[msrIsNew],[Self])
)
-> Result = [ptBoolean,true]
; Result = [ptBoolean,false]
.
```

Listing 5.42: Messir (Prolog-oriented) implementation of the operation *init*.

5.11.2 Operation Model for isSentToCoordinator

The `isSentToCoordinator` operation has the following properties:

Operation	
isSentToCoordinator	
used to provide a given coordinator with current alert information.	
Parameters	
1	**AactCoordinator: actCoordinator** the message destination
Return type	
ptBoolean	
Post-Condition (functional)	
PostF 1	true iff the message ieSendAnAlert is sent to the input interface of the given coordinator actor with the current alert as parameter value.

The listing 5.42 provides the Messir (MCL-oriented) specification of the operation.

```
1
2 /* Post Functional:*/
3 postF{if
4 (
5 /* Post F01 */
6 AactCoordinator.rnInterfaceIN.ieSendAnAlert(self)
7 )
8 then (result = true)
9 else (result = false)
10 endif}
```

Listing 5.43: Messir (MCL-oriented) specification of the operation *isSentToCoordinator*.

The listing 5.43 provides the Messir (Prolog-oriented) implementation of the operation.

```
1%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5
6msrop(ctAlert,isSentToCoordinator,[Self,AactCoordinator],
7   Result):-
8
9/* Post F01 */
10(
11 msrNav([AactCoordinator],
12     [rnInterfaceIN,ieSendAnAlert,[Self]],
13     [[ptBoolean,true]])
14)
15-> Result = [ptBoolean,true]
16; Result = [ptBoolean,false]
17.
```

Listing 5.44: Messir (Prolog-oriented) implementation of the operation *isSentToCoordinator*.

5.12 Primary Types - Operation Schemes for Class ctAuthenticated

5.12.1 Operation Model for init

The `init` operation has the following properties:

OPERATION	
init	
used to initialize the current object as a new instance of the ctAuthenticated type.	
Parameters	
1	**Alogin: dtLogin** used to initialize the login field
2	**Apwd: dtPassword** used to initialize the password field
Return type	
ptBoolean	
Post-Condition (functional)	

continues in next page . . .

. . . Operation table continuation

PostF 1	true iff the system poststate includes the current object as a new ctAuthenticated instance having its attributes equal to the ones provided as parameters.

The listing 5.44 provides the Messir (Prolog-oriented) implementation of the operation.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

msrop(ctAuthenticated,init,[Self,
              Alogin,
              Apwd],
   Result):-

/* Post F01 */
(
msrVar(ctAuthenticated,Self),

msrNav([Self],[login],[Alogin]),
msrNav([Self],[pwd],[Apwd]),
msrNav([Self],[vpIsLogged],[[ptBoolean,false]]),

/* Post F02 */
 msrNav([Self],[msrIsNew],[Self])
)
-> Result = [ptBoolean,true]
; Result = [ptBoolean,false]
.
```

Listing 5.45: Messir (Prolog-oriented) implementation of the operation *init*.

5.13 Primary Types - Operation Schemes for Class ctCoordinator

5.13.1 Operation Model for init

The `init` operation has the following properties:

OPERATION	
init	
used to initialize the current object as a new instance of the ctCoordinator type.	
Parameters	
1	**Aid: dtCoordinatorID** used to initialize the id field
2	**Alogin: dtLogin** used to initialize the login field
3	**Apwd: dtPassword** used to initialize the password field
Return type	
ptBoolean	
Post-Condition (functional)	

continues in next page . . .

. . . Operation table continuation

PostF 1	true iff the system poststate includes the current object as a new ctCoordinator instance having its attributes equal to the ones provided as parameters.

The listing 5.45 provides the Messir (MCL-oriented) specification of the operation.

```

/* Post Functional:*/
postF{if
(
/* Post F01 */
let Self:ctCoordinator in
Self.id = Aid
and Self.login = Alogin
and Self.pwd = Apwd
and Self.vpIsLogged = false
/* Post F02 */
and (Self.oclIsNew and self = Self)
)
then (result = true)
else (result = false)
endif}
```

Listing 5.46: Messir (MCL-oriented) specification of the operation *init*.

The listing 5.46 provides the Messir (Prolog-oriented) implementation of the operation.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

msrop(ctCoordinator,init,[Self,
          Aid,
          Alogin,
          Apwd],
   Result):-

/* Post F01 */
(
msrVar(ctCoordinator,Self),

msrNav([Self],[id],[Aid]),
msrNav([Self],[login],[Alogin]),
msrNav([Self],[pwd],[Apwd]),
msrNav([Self],[vpIsLogged],[[ptBoolean,false]]),

/* Post F02 */
 msrNav([Self],[msrIsNew],[Self])
)
-> Result = [ptBoolean,true]
; Result = [ptBoolean,false]
.
```

Listing 5.47: Messir (Prolog-oriented) implementation of the operation *init*.

5.14 Primary Types - Operation Schemes for Class ctCrisis

5.14.1 Operation Model for init

The `init` operation has the following properties:

OPERATION	
init	
used to initialize the current object as a new instance of the ctCrisis type.	
Parameters	
1	**Aid: dtCrisisID** used to initialize the id field
2	**Atype: etCrisisType** used to initialize the type field
3	**Astatus: etCrisisStatus** used to initialize the status field
4	**Alocation: dtGPSLocation** used to initialize the location field
5	**Ainstant: dtDateAndTime** used to initialize the instant field
6	**Acomment: dtComment** used to initialize the comment field
Return type	
ptBoolean	
Post-Condition (functional)	
PostF 1	true iff the system poststate includes the current object as a new ctCrisis instance having its attributes equal to the ones provided as parameters.

The listing 5.47 provides the Messir (MCL-oriented) specification of the operation.

```

/* Post Functional:*/
postF{if
(
/* Post F01 */
let Self:ctCrisis in
Self.id = Aid
and Self.type = Atype
and Self.status = Astatus
and Self.location = Alocation
and Self.instant = Ainstant
and Self.comment = Acomment
/* Post F02 */
and (Self.oclIsNew and self = Self)
)
then (result = true)
else (result = false)
endif}
```

Listing 5.48: Messir (MCL-oriented) specification of the operation *init*.

The listing 5.48 provides the Messir (Prolog-oriented) implementation of the operation.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

msrop(ctCrisis,init,[Self,
          Aid,
          Atype,
          Astatus,
          Alocation,
          Ainstant,
          Acomment],
   Result):-

/* Post F01 */
(
msrVar(ctCrisis,Self),

msrNav([Self],[id],[Aid]),
msrNav([Self],[type],[Atype]),
msrNav([Self],[status],[Astatus]),
msrNav([Self],[location],[Alocation]),
msrNav([Self],[instant],[Ainstant]),
msrNav([Self],[comment],[Acomment]),

/* Post F02 */
 msrNav([Self],[msrIsNew],[Self])
)
-> Result = [ptBoolean,true]
; Result = [ptBoolean,false]
.
```

Listing 5.49: Messir (Prolog-oriented) implementation of the operation *init*.

5.14.2 Operation Model for handlingDelayPassed

The `handlingDelayPassed` operation has the following properties:

OPERATION	
handlingDelayPassed used to determine if the crisis stood too longly in a pending status since last reminder.	
Return type	
ptBoolean	
Post-Condition (functional)	
PostF 1	true iff the crisis is in pending status and if the duration between the current ctState clock information and the last reminder is greater than the crisis reminder period duration.

The listing 5.49 provides the Messir (MCL-oriented) specification of the operation.

```

 /* Post Functional:*/
 postF{let TheSystem:ctState in
 let CurrentClockSecondsQty:dtInteger in
 let vpLastReminderSecondsQty:dtInteger in
 let CrisisReminderPeriod:dtSecond in
```

```
7  if
8  ( /* Post F01 */
9   self.rnSystem = TheSystem
10  and self.status = pending
11  and TheSystem.clock.toSecondsQty() = CurrentClockSecondsQty
12  and TheSystem.vpLastReminder.toSecondsQty() = vpLastReminderSecondsQty
13  and TheSystem.crisisReminderPeriod = CrisisReminderPeriod
14  and CurrentClockSecondsQty.sub(vpLastReminderSecondsQty).gt(CrisisReminderPeriod) = true
15 )
16 then (result = true)
17 else (result = false)
18 endif}
```

Listing 5.50: Messir (MCL-oriented) specification of the operation *handlingDelayPassed*.

The listing 5.50 provides the Messir (Prolog-oriented) implementation of the operation.

```
1%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5
6msrop(ctCrisis,handlingDelayPassed,[Self],
7   Result):-
8
9/* Post F01 */
10(
11 msrVar(ctState,TheSystem),
12 msrVar(dtInteger,CurrentClockSecondsQty),
13 msrVar(dtInteger,LastReminderSecondsQty),
14 msrVar(dtSecond,CrisisReminderPeriod),
15
16 msrNav([Self],[rnSystem],[TheSystem]),
17
18 msrNav([Self],
19     [status],
20     [[etCrisisStatus,pending]]),
21
22 msrNav([TheSystem],
23    [clock,toSecondsQty,[]],
24    [CurrentClockSecondsQty]),
25
26 msrNav([TheSystem],
27    [vpLastReminder,toSecondsQty,[]],
28    [LastReminderSecondsQty]),
29
30 msrNav([TheSystem],
31    [crisisReminderPeriod],
32    [CrisisReminderPeriod]),
33
34 msrNav([CurrentClockSecondsQty],
35    [sub,[LastReminderSecondsQty],
36     gt, [CrisisReminderPeriod]
37     ],
38    [[ptBoolean,true]])
39
40)
41-> Result = [ptBoolean,true]
42; Result = [ptBoolean,false]
43.
```

Listing 5.51: Messir (Prolog-oriented) implementation of the operation *handlingDelayPassed*.

5.14.3 Operation Model for maxHandlingDelayPassed

The `maxHandlingDelayPassed` operation has the following properties:

OPERATION	
maxHandlingDelayPassed used to determine if the crisis stood too longly in a pending status since its creation.	
Return type	
ptBoolean	
Post-Condition (functional)	
PostF 1	true iff the crisis is in pending status and if the duration between the current ctState clock information and the crisis instant is greater than the maximum reminder period duration.

The listing 5.51 provides the Messir (MCL-oriented) specification of the operation.

```

/* Post Functional:*/
postF{let TheSystem:ctState in
let CurrentClockSecondsQty:dtInteger in
let CrisisInstantSecondsQty:dtInteger in
let MaxCrisisReminderPeriod:dtSecond in
if
( /* Post F01 */
 self.rnSystem = TheSystem
 and self.status = pending
 and TheSystem.clock.toSecondsQty() = CurrentClockSecondsQty
 and Self.instant.toSecondsQty() = CrisisInstantSecondsQty
 and TheSystem.maxCrisisReminderPeriod = MaxCrisisReminderPeriod
 and CurrentClockSecondsQty.sub(CrisisInstantSecondsQty)
              .gt(MaxCrisisReminderPeriod)
)
then (result = true)
else (result = false)
endif}
```

Listing 5.52: Messir (MCL-oriented) specification of the operation *maxHandlingDelayPassed*.

The listing 5.52 provides the Messir (Prolog-oriented) implementation of the operation.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

msrop(ctCrisis,maxHandlingDelayPassed,[Self],
   Result):-

/* Post F01 */
(
 msrVar(ctState,TheSystem),
 msrVar(dtInteger,CurrentClockSecondsQty),
 msrVar(dtInteger,CrisisInstantSecondsQty),
 msrVar(dtSecond,MaxCrisisReminderPeriod),

 msrNav([Self],[rnSystem],[TheSystem]),

```

```
18 msrNav([Self],
19     [status],
20     [[etCrisisStatus,pending]]),
21
22 msrNav([TheSystem],
23    [clock,toSecondsQty,[]],
24    [CurrentClockSecondsQty]),
25
26 msrNav([Self],
27    [instant,toSecondsQty,[]],
28    [CrisisInstantSecondsQty]),
29
30 msrNav([TheSystem],
31    [maxCrisisReminderPeriod],
32    [MaxCrisisReminderPeriod]),
33
34 msrNav([CurrentClockSecondsQty],
35    [sub,[CrisisInstantSecondsQty],
36     gt, [MaxCrisisReminderPeriod]
37     ],
38    [[ptBoolean,true]])
39
40)
41-> Result = [ptBoolean,true]
42; Result = [ptBoolean,false]
43.
```

Listing 5.53: Messir (Prolog-oriented) implementation of the operation *maxHandlingDelayPassed*.

5.14.4 Operation Model for isSentToCoordinator

The `isSentToCoordinator` operation has the following properties:

OPERATION	
isSentToCoordinator	
used to provide a given coordinator with current crisis information.	
Parameters	
1	**AactCoordinator: actCoordinator** the message destination actor
Return type	
ptBoolean	
Post-Condition (functional)	
PostF 1	true iff the message ieSendACrisis is sent by the simulator to the input interface of the given coordinator actor with the current crisis as parameter value.

The listing 5.53 provides the Messir (MCL-oriented) specification of the operation.

```
1
2 /* Post Functional:*/
3 postF{if
4 (
5 /* Post F01 */
6  AactCoordinator.rnInterfaceIN.ieSendACrisis(self)
7 )
8 then (result = true)
9 else (result = false)
```

```
endif}
```

Listing 5.54: Messir (MCL-oriented) specification of the operation *isSentToCoordinator*.

The listing 5.54 provides the Messir (Prolog-oriented) implementation of the operation.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

msrop(ctCrisis,isSentToCoordinator,[Self,AactCoordinator],
   Result):-

/* Post F01 */
(
 msrNav([AactCoordinator],
     [rnInterfaceIN,ieSendACrisis,[Self]],
     [[ptBoolean,true]])
)
-> Result = [ptBoolean,true]
; Result = [ptBoolean,false]
.
```

Listing 5.55: Messir (Prolog-oriented) implementation of the operation *isSentToCoordinator*.

5.14.5 Operation Model for isSentToHospital

The `isSentToHospital` operation has the following properties:

OPERATION	
isSentToHospital	
used to provide a given hospital with current crisis information.	
Parameters	
1	**AactHospital: actHospital** the message destination actor
Return type	
ptBoolean	
Post-Condition (functional)	
PostF 1	true if the message ieSendACrisisToHospital is sent by the simulator to the input interface of the given hospital actor with the current crisis as parameter value.

The listing 5.55 provides the Messir (MCL-oriented) specification of the operation.

```

/* Post Functional:*/
postF{if
 (
 /* Post F01 */
  AactHospital.rnInterfaceIN^ieSendACrisis(self)
 )
 then (result = true)
 else (result = false)
```

```
endif}
```

Listing 5.56: Messir (MCL-oriented) specification of the operation *isSentToHospital*.

5.14.6 Operation Model for isAllocatedIfPossible

The `isAllocatedIfPossible` operation has the following properties:

Operation	
isAllocatedIfPossible	
used to allocate a crisis to a coordinator if any or to alert the administrator of crisis waiting to be handled.	
Return type	
ptBoolean	
Post-Condition (functional)	
PostF 1	true iff the duration between the crisis creation and the system's clock is greater than the maximum delay defined and
PostF 2	if there exist at least one coordinator then (a) the post state associates to the crisis any of the existing coordinators and (b) the coordinator is informed that he is now the handlers of the crisis whose ID is communicated
PostF 3	else a message is sent to all known administrators to request creation of new coordinators.

The listing 5.56 provides the Messir (MCL-oriented) specification of the operation.

```

/* Post Functional:*/
postF{if (
/* Post F01 */
self.maxHandlingDelayPassed()
and
 if (TheSystem.rnactCoordinator->msrIsEmpty = false)
 then (
   /* Post F02 */
   TheSystem.rnactCoordinator->msrAny(true) = TheCoordinatorActor
   and TheCoordinatorActor.rnctCoordinator = TheCoordinator
   and self@post.rnHandler = TheCoordinator
   and self@post.status = handled
   and self.id.value = TheCrisisIDptString
   and 'You are now considered as handling the crisis having ID: '
     .ptStringConcat(TheCrisisIDptString) = TheMessage
    and TheCoordinatorActor.rnInterfaceIN^ieMessage(TheMessage)
  )
 else ( /* Post F03 */
    TheSystem.rnactAdministrator
    ->forAll(rnInterfaceIN.ieMessage('Please add new coordinators to handle pending crisis !'))
  )
 endif
 )
then (result = true)
else (result = false)
endif}
```

Listing 5.57: Messir (MCL-oriented) specification of the operation *isAllocatedIfPossible*.

The listing 5.57 provides the Messir (Prolog-oriented) implementation of the operation.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

msrop(ctCrisis,isAllocatedIfPossible,[Self],
   Result):-
(
 msrVar(ctState,TheSystem),
 msrNav([Self],[rnSystem],[TheSystem]),

 msrVar(actCoordinator,TheCoordinatorActor),
 msrVar(ctCoordinator,TheCoordinator),
 msrVar(ptString,TheMessage),
 msrVar(ptString,TheCrisisIDptString),

 (
 /* Post F01 */
 msrNav([Self],
     [maxHandlingDelayPassed,[]],
     [[ptBoolean,true]]),

 ( msrNav([TheSystem],
      [rnactCoordinator,msrIsEmpty],
      [[ptBoolean,false]])
  -> (
     /* Post F02 */
     msrNav([TheSystem],
        [rnactCoordinator,msrAny,msrTrue],
        [TheCoordinatorActor]),

     msrNav([TheCoordinatorActor],
         [rnctCoordinator],
         [TheCoordinator]),

     msrNav([Self],
         [msmAtPost,rnHandler],
         [TheCoordinator]),

      msrNav([Self],
         [msmAtPost,status],
         [[etCrisisStatus,handled]]),

      msrNav([Self],
         [id,value],
         [TheCrisisIDptString]),

     msrNav([[ptString,'You are now considered as handling the crisis having ID: ']],
         [ptStringConcat,[TheCrisisIDptString]],
         [TheMessage]),

     msrNav([TheCoordinatorActor],
         [rnInterfaceIN,
         ieMessage,[TheMessage]
         ],
         [[ptBoolean,true]])
    )
 ; ( /* Post F03 */
    msrNav([TheSystem],
       [rnactAdministrator,msrForAll,rnInterfaceIN,
       ieMessage,[[ptString,'Please add new coordinators to handle pending crisis !']]],
       [[ptBoolean,true]])
  )
 )
 )
)
-> Result = [ptBoolean,true]
```

```
; Result = [ptBoolean,false]
.
```

Listing 5.58: Messir (Prolog-oriented) implementation of the operation *isAllocatedIfPossible*.

5.15 Primary Types - Operation Schemes for Class ctHospital

5.15.1 Operation Model for init

The `init` operation has the following properties:

OPERATION	
init	
used to initialize the current object as a new instance of the ctHospital type.	
Parameters	
1	**Aid: dtHospitalID**
2	**Alogin: dtLogin**
3	**Apwd: dtPassword**
Return type	
ptBoolean	
Post-Condition (functional)	
PostF 1	true iff the system poststate includes the current object as a new ctHospital instance having its attributes equal to the ones provided as parameters.

The listing 5.58 provides the Messir (MCL-oriented) specification of the operation.

```

/* Post Functional:*/
postF{if
(
/* Post F01 */
let Self:ctHospital in
Self.id = Aid
and Self.login = Alogin
and Self.pwd = Apwd
and Self.vpIsLogged = false
/* Post F02 */
and (Self.oclIsNew and self = Self)
)
then (result = true)
else (result = false)
endif}
```

Listing 5.59: Messir (MCL-oriented) specification of the operation *init*.

5.16 Primary Types - Operation Schemes for Class ctHuman

5.16.1 Operation Model for init

The `init` operation has the following properties:

OPERATION	
init	
used to initialize the current object as a new instance of the ctHuman type.	
Parameters	
1	**Aid: dtPhoneNumber** used to initialize the id field
2	**Akind: etHumanKind** used to initialize the kind field
Return type	
ptBoolean	
Post-Condition (functional)	
PostF 1	true iff the system poststate includes the current object as a new ctHuman instance having its attributes equal to the ones provided as parameters.

The listing 5.59 provides the Messir (MCL-oriented) specification of the operation.

```

/* Post Functional:*/
postF{if
(
/* Post F01 */
let Self:ctHuman in

Self.id = Aid
and Self.kind = Akind

/* Post F02 */
and (Self.oclIsNew and self = Self)
)
then (result = true)
else (result = false)
endif}
```

Listing 5.60: Messir (MCL-oriented) specification of the operation *init*.

The listing 5.60 provides the Messir (Prolog-oriented) implementation of the operation.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

msrop(ctHuman,init,[Self,
          Aid,
          Akind],
   Result):-

```

```
11/* Post F01 */
12(
13msrVar(ctHuman,Self),
14
15msrNav([Self],[id],[Aid]),
16msrNav([Self],[kind],[Akind]),
17
18/* Post F02 */
19 msrNav([Self],[msrIsNew],[Self])
20)
21-> Result = [ptBoolean,true]
22; Result = [ptBoolean,false]
23.
```

Listing 5.61: Messir (Prolog-oriented) implementation of the operation *init*.

5.16.2 Operation Model for isAcknowledged

The `isAcknowledged` operation has the following properties:

OPERATION	
isAcknowledged used to specify the property of having sent an alert acknowledge message to the human having declared the alert through its own communication company.	
Return type	
ptBoolean	
Post-Condition (functional)	
PostF 1	true iff the message ieSmsSend is sent to the related input interface of the related communication company actor with the human phone number and the generic message 'The handling of your alert by our services is in progress !'

The listing 5.61 provides the Messir (Prolog-oriented) implementation of the operation.

```
1%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5
6msrop(ctHuman,isAcknowledged,[Self],Result):-
7
8/* Post F01 */
9(msrVar(dtPhoneNumber,AdtPhoneNumber),
10 msrVar(dtSMS,AdtSMS),
11
12 msrNav([Self],
13     [id,eq,[AdtPhoneNumber]],
14     [[ptBoolean,true]]),
15 msrNav([AdtSMS],
16     [value,eq,[[ptString,'The handling of your alert by our services is in progress !']]],
17     [[ptBoolean,true]]),
18 msrNav([Self],
19     [rnactComCompany,rnInterfaceIN,ieSmsSend,[AdtPhoneNumber,AdtSMS]],
20     [[ptBoolean,true]])
21)
22-> Result = [ptBoolean,true]
23; Result = [ptBoolean,false]
```

```
24.
```

Listing 5.62: Messir (Prolog-oriented) implementation of the operation *isAcknowledged*.

5.17 Primary Types - Operation Schemes for Class ctInjury

5.17.1 Operation Model for init

The `init` operation has the following properties:

OPERATION	
init	
used to initialize the current object as a new instance of the ctInjury type.	
Parameters	
1	**Aid: dtInjuryID**
Return type	
ptBoolean	
Post-Condition (functional)	
PostF 1	true if the system poststate includes the current object as a new ctInjury instance having its attributes equal to the ones provided as parameters.

The listing 5.62 provides the Messir (MCL-oriented) specification of the operation.

```
1
2  /* Post Functional:*/
3  postF{if
4  (
5  /* Post F01 */
6  let Self:ctInjury in
7  Self.id = Aid
8  and Self.victimID = AvictimID
9  /* Post F02 */
10 and (Self.oclIsNew and self = Self)
11 )
12 then (result = true)
13 else (result = false)
14 endif}
```

Listing 5.63: Messir (MCL-oriented) specification of the operation *init*.

5.17.2 Operation Model for isSentToCoordinator

The `isSentToCoordinator` operation has the following properties:

OPERATION	
isSentToCoordinator	
used to provide a given coordinator with current injury information.	
Parameters	
1	**AactCoordinator: actCoordinator** the message destination actor

continues in next page . . .

. . . Operation table continuation

Return type	
ptBoolean	
Post-Condition (functional)	
PostF 1	true if the message ieSendAInjury is sent by the simulator to the input interface of the given coordinator actor with the current injury as parameter value.

The listing 5.63 provides the Messir (MCL-oriented) specification of the operation.

```

/* Post Functional:*/
postF{if
(
/* Post F01 */
 AactCoordinator.rnInterfaceIN.ieSendAInjury(self)
)
then (result = true)
else (result = false)
endif}
```

Listing 5.64: Messir (MCL-oriented) specification of the operation *isSentToCoordinator*.

The listing 5.64 provides the Messir (Prolog-oriented) implementation of the operation.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

msrop(ctCrisis,isSentToCoordinator,[Self,AactCoordinator],
   Result):-

/* Post F01 */
(
msrNav([AactCoordinator],
     [rnInterfaceIN,ieSendACrisis,[Self]],
     [[ptBoolean,true]])
)
-> Result = [ptBoolean,true]
; Result = [ptBoolean,false]
.
```

Listing 5.65: Messir (Prolog-oriented) implementation of the operation *isSentToCoordinator*.

5.17.3 Operation Model for isSentToHospital

The `isSentToHospital` operation has the following properties:

OPERATION
isSentToHospital used to provide a given hospital with current injury information.
Parameters

continues in next page . . .

. . . Operation table continuation

1	**AactHospital: actHospital** the message destination actor
Return type	
ptBoolean	
Post-Condition (functional)	
PostF 1	true if the message ieSendAInjuryToHospital is sent by the simulator to the input interface of the given hospital actor with the current injury as parameter value.

The listing 5.65 provides the Messir (MCL-oriented) specification of the operation.

```

/* Post Functional:*/
postF{if
(
/* Post F01 */
 AactHospital.rnInterfaceIN.ieSendAInjury(self)
)
then (result = true)
else (result = false)
endif}
```

Listing 5.66: Messir (MCL-oriented) specification of the operation *isSentToHospital*.

5.18 Primary Types - Operation Schemes for Class ctState

5.18.1 Operation Model for init

The `init` operation has the following properties:

OPERATION	
init used to initialize the current object as a new instance of the ctState type.	
Parameters	
1	**AnextValueForAlertID: dtInteger** used to initialize the nextValueForAlertID field
2	**AnextValueForCrisisID: dtInteger** used to initialize the nextValueForCrisisID field
3	**Aclock: dtDateAndTime** used to initialize the clock field
4	**AcrisisReminderPeriod: dtSecond** used to initialize the crisisReminderPeriod field
5	**AmaxCrisisReminderPeriod: dtSecond** used to initialize the maxCrisisReminderPeriod field
6	**AvpLastReminder: dtDateAndTime** used to initialize the vpLastReminder field
7	**AvpStarted: ptBoolean** used to initialize the vpStarted field

continues in next page . . .

. . . Operation table continuation

Return type	
ptBoolean	
Post-Condition (functional)	
PostF 1	true iff the system poststate includes the current object as a new ctState instance having its attributes equal to the ones provided as parameters.

The listing 5.66 provides the Messir (MCL-oriented) specification of the operation.

```

/* Post Functional:*/
postF{if
(
/* Post F01 */
let Self:ctState in

Self.nextValueForAlertID = AnextValueForAlertID
and Self.nextValueForCrisisID = AnextValueForCrisisID
and Self.nextValueForVictimID = AnextValueForVictimID
and Self.nextValueForInjuryID = AnextValueForInjuryID
and Self.clock = Aclock
and Self.crisisReminderPeriod = AcrisisReminderPeriod
and Self.maxCrisisReminderPeriod = AmaxCrisisReminderPeriod
and Self.vpLastReminder = AvpLastReminder
and Self.vpStarted = AvpStarted

and (Self.oclIsNew and self = Self)
)
then (result = true)
else (result = false)
endif}
```

Listing 5.67: Messir (MCL-oriented) specification of the operation *init*.

The listing 5.67 provides the Messir (Prolog-oriented) implementation of the operation.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

msrop(ctState,init,[Self,
          AnextValueForAlertID,
          AnextValueForCrisisID,
          Aclock,
          AcrisisReminderPeriod,
          AmaxCrisisReminderPeriod,
          AvpLastReminder,
          AvpStarted],
   Result):-

/* Post F01 */
(
msrVar(ctState,Self),

msrNav([Self],[nextValueForAlertID],[AnextValueForAlertID]),
msrNav([Self],[nextValueForCrisisID],[AnextValueForCrisisID]),
msrNav([Self],[clock],[Aclock]),
```

```
23 msrNav([Self],[crisisReminderPeriod],[AcrisisReminderPeriod]),
24 msrNav([Self],[maxCrisisReminderPeriod],[AmaxCrisisReminderPeriod]),
25 msrNav([Self],[vpLastReminder],[AvpLastReminder]),
26 msrNav([Self],[vpStarted],[AvpStarted]),
27
28 msrNav([Self],[msrIsNew],[Self])
29)
30-> Result = [ptBoolean,true]
31; Result = [ptBoolean,false]
32.
```

Listing 5.68: Messir (Prolog-oriented) implementation of the operation *init*.

5.19 Primary Types - Operation Schemes for Class ctVictim

5.19.1 Operation Model for init

The `init` operation has the following properties:

OPERATION	
init	
used to initialize the current object as a new instance of the ctVictim type.	
Parameters	
1	**Aid: dtVictimID**
Return type	
ptBoolean	
Post-Condition (functional)	
PostF 1	true iff the system poststate includes the current object as a new ctVictim instance having its attributes equal to the ones provided as parameters.

The listing 5.68 provides the Messir (MCL-oriented) specification of the operation.

```
1
2 /* Post Functional:*/
3 postF{if
4 (
5 /* Post F01 */
6 let Self:ctVictim in
7 Self.id = Aid
8 and Self.crisisID = AcrisisID
9 /* Post F02 */
10 and (Self.oclIsNew and self = Self)
11 )
12 then (result = true)
13 else (result = false)
14 endif}
```

Listing 5.69: Messir (MCL-oriented) specification of the operation *init*.

5.19.2 Operation Model for isSentToCoordinator

The `isSentToCoordinator` operation has the following properties:

OPERATION	
isSentToCoordinator	
used to provide a given coordinator with current victim information.	
Parameters	
1	**AactCoordinator: actCoordinator** the message destination actor
Return type	
ptBoolean	
Post-Condition (functional)	
PostF 1	true if the message ieSendAVictim is sent by the simulator to the input interface of the given coordinator actor with the current victim as parameter value.

The listing 5.69 provides the Messir (MCL-oriented) specification of the operation.

```

/* Post Functional:*/
postF{if
(
/* Post F01 */
 AactCoordinator.rnInterfaceIN.ieSendAVictim(self)
)
then (result = true)
else (result = false)
endif}
```

Listing 5.70: Messir (MCL-oriented) specification of the operation *isSentToCoordinator*.

The listing 5.70 provides the Messir (Prolog-oriented) implementation of the operation.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

msrop(ctCrisis,isSentToCoordinator,[Self,AactCoordinator],
   Result):-

/* Post F01 */
(
msrNav([AactCoordinator],
     [rnInterfaceIN,ieSendACrisis,[Self]],
     [[ptBoolean,true]])
)
-> Result = [ptBoolean,true]
; Result = [ptBoolean,false]
.
```

Listing 5.71: Messir (Prolog-oriented) implementation of the operation *isSentToCoordinator*.

5.19.3 Operation Model for isSentToHospital

The `isSentToHospital` operation has the following properties:

OPERATION	
isSentToHospital	
used to provide a given hospital with current victim information.	
Parameters	
1	**AactHospital: actHospital** the message destination actor
Return type	
ptBoolean	
Post-Condition (functional)	
PostF 1	true if the message ieSendAVictimToHospital is sent by the simulator to the input interface of the given hospital actor with the current victim as parameter value.

The listing 5.71 provides the Messir (MCL-oriented) specification of the operation.

```
1
2 /* Post Functional:*/
3 postF{if
4 (
5 /* Post F01 */
6  AactHospital.rnInterfaceIN.ieSendAVictim(self)
7 )
8 then (result = true)
9 else (result = false)
10 endif}
```

Listing 5.72: Messir (MCL-oriented) specification of the operation *isSentToHospital*.

5.20 Primary Types - Operation Schemes for Datatype dtAlertID

5.20.1 Operation Model for is

The `is` operation has the following properties:

OPERATION	
is	
used to determine which strings are considered as valid alert identifiers.	
Return type	
ptBoolean	
Post-Condition (functional)	
PostF 1	if the length of the value attribute of a dtAlertID is a ptInteger greater than zero and lower or equal to 20 then the operation returns the ptBoolean true, else the ptBoolean false.

The listing 5.72 provides the Messir (MCL-oriented) specification of the operation.

```
1
2 /* Post Functional:*/
3 postF{let TheResult: ptBoolean in
```

```
4    ( if
5     ( AdtValue.value.length().gt(0)
6      and AdtValue.value.length().leq(20)
7     )
8     then (TheResult = true)
9     else (TheResult = false)
10    endif
11    result = TheResult
12   )}
```

Listing 5.73: Messir (MCL-oriented) specification of the operation *is*.

The listing 5.73 provides the Messir (Prolog-oriented) implementation of the operation.

```
1%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5
6msrop(dtAlertID,is,[AdtValue],Result):-
7% msd01
8msrVar(ptBoolean,TheResult),
9(
10 ( msrNav([AdtValue],
11    [value,length,[],gt,[[ptInteger,0]]],
12    [[ptBoolean,true]]),
13   msrNav([AdtValue],
14    [value,length,[],leq,[[ptInteger,20]]],
15    [[ptBoolean,true]])
16 )
17 -> (TheResult = [ptBoolean,true])
18 ; (TheResult = [ptBoolean,false])
19),
20TheResult = Result
21.
22
23/*
24| ?- X = [dtAlertID,[],[[dtString,[[value,[ptString,'0123456789']]],[]]]],
25msrNav([X],[is,[]],[Result]).
26
27X = [dtAlertID,[],[[dtString,[[value,[ptString,'0123456789']]],[]]]],
28Result = [ptBoolean,true] ?
29
30yes
31
32| ?- X = [dtAlertID,[],[[dtString,[[value,[ptString,'0123456789012345678901234567 89']]],[]]]],
33msrNav([X],[is,[]],[Result]).
34
35X = [dtAlertID,[],[[dtString,[[value,[ptString,'0123456789012345678901234567 89']]],[]]]],
36Result = [ptBoolean,false] ?
37
38yes
39*/
```

Listing 5.74: Messir (Prolog-oriented) implementation of the operation *is*.

5.21 Primary Types - Operation Schemes for Datatype dtComment

5.21.1 Operation Model for is

The `is` operation has the following properties:

OPERATION
is
used to determine which strings are considered as valid comments.
Return type
ptBoolean
Post-Condition (functional)
PostF 1 true iff the length of the string value is not more than 160 characters.

The listing 5.74 provides the Messir (MCL-oriented) specification of the operation.

```

/* Post Functional:*/
postF{let TheResult: ptBoolean in
    ( if
     ( MaxLength = 160
      and AdtValue.value.length().leq(MaxLength)
     )
     then (TheResult = true)
     else (TheResult = false)
     endif
     result = TheResult
    )}
```

Listing 5.75: Messir (MCL-oriented) specification of the operation *is*.

The listing 5.75 provides the Messir (Prolog-oriented) implementation of the operation.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% dtComment

%msd01
msrop(dtComment,is,[AdtValue],Result):-
 msrVar(ptBoolean,TheResult),
 msrVar(ptInteger,MaxLength),
(
  (
    (
     MaxLength = [ptInteger,160],
     msrNav([AdtValue],
        [value,length,[],leq,[MaxLength]],
        [[ptBoolean,true]])
    )
     -> TheResult = [ptBoolean,true]
     ; TheResult = [ptBoolean,false]
  )
),
 Result = TheResult
.

/*
| ?- X = [dtComment,[],[[dtString,[[value,[ptString,'I broke my leg ! Please help ...']]],[]]]],
msrNav([X],[is,[]],[Result]).
X = [dtComment,[],[[dtString,[[value,[ptString,'I broke my leg ! Please help ...']]],[]]]],
Result = [ptBoolean,true] ?
```

```
yes

| ?- X = [dtComment,[],[[dtString,[[value,[ptString,'I broke my leg when I was running with my dog to go to the skate park because my friends called me on my mobile phone and told me that a skate star was doing triple back flips.']]],[]]]],
msrNav([X],[is,[]],[Result]).
X = [dtComment,[],[[dtString,[[value,[ptString,'I broke my leg when I was running with my dog to go to the skate park because my friends called me on my mobile phone and told me that a skate star was doing triple back flips.']]],[]]]],
Result = [ptBoolean,false] ?
yes
*/
```

Listing 5.76: Messir (Prolog-oriented) implementation of the operation *is*.

5.22 Primary Types - Operation Schemes for Datatype dtCoordinatorID

5.22.1 Operation Model for is

The `is` operation has the following properties:

OPERATION	
is	
used to determine which string are considered as valid alert identifiers.	
Return type	
ptBoolean	
Post-Condition (functional)	
PostF 1	if the length of the value attribute of a dtCoordinatorID is a ptInteger greater than zero and lower or equal to 5 than the operation returns the ptBoolean true, else the ptBoolean false.

The listing 5.76 provides the Messir (MCL-oriented) specification of the operation.

```

/* Post Functional:*/
postF{let TheResult: ptBoolean in
    ( if
     ( AdtValue.value.length().gt(0)
      and AdtValue.value.length().leq(5)
     )
     then (TheResult = true)
     else (TheResult = false)
     endif
     result = TheResult
    )}
```

Listing 5.77: Messir (MCL-oriented) specification of the operation *is*.

The listing 5.77 provides the Messir (Prolog-oriented) implementation of the operation.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
```

```
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

msrop(dtCoordinatorID,is,[AdtValue],Result):-
% msd01
 msrVar(ptBoolean,TheResult),
(
 ( msrNav([AdtValue],
    [value,length,[],gt,[[ptInteger,0]]],
    [[ptBoolean,true]]),
   msrNav([AdtValue],
    [value,length,[],leq,[[ptInteger,5]]],
    [[ptBoolean,true]])
 )
 -> (TheResult = [ptBoolean,true])
 ; (TheResult = [ptBoolean,false])
),
TheResult = Result
.
```

Listing 5.78: Messir (Prolog-oriented) implementation of the operation *is*.

5.23 Primary Types - Operation Schemes for Datatype dtCrisisID

5.23.1 Operation Model for is

The `is` operation has the following properties:

OPERATION	
is	
used to determine which strings are considered as valid crisis identifiers.	
Return type	
ptBoolean	
Post-Condition (functional)	
PostF 1	if the length of the value attribute of a dtCrisisID is a ptInteger greater than zero and lower or equal to 10 than the operation returns the ptBoolean true, else the ptBoolean false.

The listing 5.78 provides the Messir (MCL-oriented) specification of the operation.

```

/* Post Functional:*/
postF{let TheResult: ptBoolean in
    ( if
     ( AdtValue.value.length().gt(0)
      and AdtValue.value.length().leq(10)
     )
     then (TheResult = true)
     else (TheResult = false)
     endif
     result = TheResult
    )}
```

Listing 5.79: Messir (MCL-oriented) specification of the operation *is*.

The listing 5.79 provides the Messir (Prolog-oriented) implementation of the operation.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

msrop(dtCrisisID,is,[AdtValue],Result):-
% msd01
 msrVar(ptBoolean,TheResult),
(
 ( msrNav([AdtValue],
    [value,length,[],gt,[[ptInteger,0]]],
    [[ptBoolean,true]]),
   msrNav([AdtValue],
    [value,length,[],leq,[[ptInteger,10]]],
    [[ptBoolean,true]])
 )
 -> (TheResult = [ptBoolean,true])
 ; (TheResult = [ptBoolean,false])
),
TheResult = Result
.
/*
| ?- X = [dtCrisisID,[],[[dtString,[[value,[ptString,'0123456789']]],[]]]],
msrNav([X],[is,[]],[Result]).
X = [dtCrisisID,[],[[dtString,[[value,[ptString,'0123456789']]],[]]]],
Result = [ptBoolean,true] ?
yes

| ?- X = [dtCrisisID,[],[[dtString,[[value,[ptString,'0123456789a']]],[]]]],
msrNav([X],[is,[]],[Result]).
X = [dtCrisisID,[],[[dtString,[[value,[ptString,'0123456789a']]],[]]]],
Result = [ptBoolean,false] ?
yes
*/
```

Listing 5.80: Messir (Prolog-oriented) implementation of the operation *is*.

5.24 Primary Types - Operation Schemes for Datatype dtGPSLocation

5.24.1 Operation Model for is

The `is` operation has the following properties:

OPERATION
is used to determine which couples are considered as valid dtGPSLocation values.
Return type
ptBoolean
Post-Condition (functional)
PostF 1 true if both latitude and longitude are valid values according to their is operation.

The listing 5.80 provides the Messir (MCL-oriented) specification of the operation.

```

/* Post Functional:*/
postF{let TheResult: ptBoolean in
```

```
4      ( if
5       ( AdtValue.latitude.is()
6        and AdtValue.longitude.is
7       )
8       then (TheResult = true)
9       else (TheResult = false)
10      endif
11      result = TheResult
12     )}
```

Listing 5.81: Messir (MCL-oriented) specification of the operation *is*.

The listing 5.81 provides the Messir (Prolog-oriented) implementation of the operation.

```
1%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5
6%% dtPhoneNumber
7
8% msd01
9msrop(dtGPSLocation,is,[AdtValue],Result):-
10msrVar(ptBoolean,TheResult),
11(
12  (
13     msrNav([AdtValue],
14        [latitude,is,[]],
15        [[ptBoolean,true]]),
16     msrNav([AdtValue],
17        [longitude,is,[]],
18        [[ptBoolean,true]])
19  )
20 -> TheResult = [ptBoolean,true]
21 ; TheResult = [ptBoolean,false]
22),
23
24 Result = TheResult
25.
```

Listing 5.82: Messir (Prolog-oriented) implementation of the operation *is*.

5.24.2 Operation Model for isNearTo

The `isNearTo` operation has the following properties:

OPERATION
isNearTo
used to determine if locations are considered enough close to be treated as equivalent in the application domain context. In the context of the iCrash system, we compute the distance between two GPS locations using the following Haversine formula. (more details can be found at: http://www.movable-type.co.uk/scripts/latlong.html and http://www.gpsvisualizer.com/calculators#distance)
Parameters
1 **AGPSLocation: dtGPSLocation** the GPS location to be compared to.
Return type
ptBoolean

continues in next page . . .

. . . Operation table continuation

Post-Condition (functional)	
PostF 1	if the Haversine formula (ACOS(SIN(lat1)*SIN(lat2)+COS(lat1)*COS(lat2)*COS(lon2-lon1))*6371, in which latitudes and longitudes are in radiansapplied to the two dtGPS coordinates is lower to 100 meters) then the predicate is true and false otherwise.

The listing 5.82 provides the Messir (MCL-oriented) specification of the operation.

```

/* Post Functional:*/
postF{let TheResult: ptBoolean in true
    let EarthRadius: dtReal in
    let MaxDistance: dtReal in
    let ComparedLatitude: dtLatitude in
    let ComparedLongitude: dtLongitude in
    let R1: dtReal in let R1a: dtReal in
    let R2: dtReal in let R2a: dtReal in

    ( if
     ( EarthRadius.value = 6371
      and MaxDistance.value = 100

      and AdtValue.latitude = ComparedLatitude
      and AdtValue.longitude = ComparedLongitude
      and Self.latitude.sin() = R1a
      and AdtValue.latitude.sin().mul(R1a) = R1
      and Self.latitude.cos() = R2a
      and AdtValue.latitude.cos().mul(R2a) = R2

      and AdtValue.longitude = ComparedLongitude
      and Self.longitude.sub(ComparedLongitude).cos().mul(R2)
          .add(R1).acos().mul(EarthRadius).sub(MaxDistance)
          .value.leq(0)
     )
     then (TheResult = true)
     else (TheResult = false)
     endif
     result = TheResult
    )}
```

Listing 5.83: Messir (MCL-oriented) specification of the operation *isNearTo*.

The listing 5.83 provides the Messir (Prolog-oriented) implementation of the operation.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% dtGPSLocation

msrop(dtGPSLocation,isNearTo,[Self,AdtValue],Result):-
msrVar(ptBoolean,TheResult),
msrVar(dtReal,EarthRadius),
msrVar(dtReal,MaxDistance),

msrVar(dtLatitude,ComparedLatitude),
msrVar(dtLongitude,ComparedLongitude),

```

```
msrVar(dtReal,R1),msrVar(dtReal,R1a),
msrVar(dtReal,R2),msrVar(dtReal,R2a),

(
  (
    (
     % msd01
     msrNav([EarthRadius],[value],[[ptReal,6371]]),
     msrNav([MaxDistance],[value],[[ptReal,100]]),

     msrNav([AdtValue],[latitude],[ComparedLatitude]),
     msrNav([AdtValue],[longitude],[ComparedLongitude]),

     msrNav([Self],[latitude,sin,[]],[R1a]),
     msrNav([AdtValue],[latitude,sin,[],mul,[R1a]],[R1]),

     msrNav([Self],[latitude,cos,[]],[R2a]),
     msrNav([AdtValue],[latitude,cos,[],mul,[R2a]],[R2]),

     msrNav([AdtValue],[longitude],[ComparedLongitude]),
     msrNav([Self],[longitude,sub,[ComparedLongitude],cos,[],mul,[R2],
            add,[R1],
            acos,[],
            mul,[EarthRadius],
            sub,[MaxDistance],
            value,leq,[[ptReal,0]]],
        [[ptBoolean,true]])
    )
     -> TheResult = [ptBoolean,true]
     ; TheResult = [ptBoolean,false]
  )
),
 Result = TheResult
.
```

Listing 5.84: Messir (Prolog-oriented) implementation of the operation *isNearTo*.

5.25 Primary Types - Operation Schemes for Datatype dtHospitalID

5.25.1 Operation Model for is

The `is` operation has the following properties:

OPERATION	
is	
used to determine which string are considered as valid hospital identifiers.	
Return type	
ptBoolean	
Post-Condition (functional)	
PostF 1	if the length of the value attribute of a dtHospitalID is a ptInteger greater than zero and lower or equal to 5 than the operation returns the ptBoolean true, else the ptBoolean false.

The listing 5.84 provides the Messir (MCL-oriented) specification of the operation.

```

/* Post Functional:*/
postF{let TheResult: ptBoolean in
    ( if
```

```
      ( AdtValue.value.length().gt(0)
       and AdtValue.value.length().leq(5)
      )
      then (TheResult = true)
      else (TheResult = false)
      endif
      result = TheResult
     )}
```

Listing 5.85: Messir (MCL-oriented) specification of the operation *is*.

5.26 Primary Types - Operation Schemes for Datatype dtLatitude

5.26.1 Operation Model for is

The `is` operation has the following properties:

OPERATION	
is	
used to determine which strings are considered as valid dtLatitude.	
Return type	
ptBoolean	
Post-Condition (functional)	
PostF 1	is true if the value is a real in the interval [-90.0 , +90.0].

The listing 5.85 provides the Messir (MCL-oriented) specification of the operation.

```

/* Post Functional:*/
postF{let TheResult: ptBoolean in
     ( if
      ( AdtValue.value.geq(-90.0)
       and AdtValue.value.leq(+90.0)
      )
      then (TheResult = true)
      else (TheResult = false)
      endif
      result = TheResult
     )}
```

Listing 5.86: Messir (MCL-oriented) specification of the operation *is*.

The listing 5.86 provides the Messir (Prolog-oriented) implementation of the operation.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% msd01
msrop(dtLatitude,is,[AdtValue],Result):-
msrVar(ptBoolean,TheResult),
(
 ( msrNav([AdtValue],
```

```
    [value,geq,[[ptReal,-90.0]]],
    [[ptBoolean,true]]),
  msrNav([AdtValue],
    [value,leq,[[ptReal,+90.0]]],
    [[ptBoolean,true]])
)
-> (TheResult = [ptBoolean,true])
; (TheResult = [ptBoolean,false])
),
Result = TheResult
.
```

Listing 5.87: Messir (Prolog-oriented) implementation of the operation *is*.

5.27 Primary Types - Operation Schemes for Datatype dtLogin

5.27.1 Operation Model for is

The `is` operation has the following properties:

OPERATION
is
used to determine which strings are considered as valid dtLogin.
Return type
ptBoolean
Post-Condition (functional)
PostF 1 is true of the length of the string value is not more than 20 characters.

The listing 5.87 provides the Messir (MCL-oriented) specification of the operation.

```

/* Post Functional:*/
postF{let TheResult: ptBoolean in
    let MaxLength: ptInteger in
    ( if
     ( MaxLength = 20
      and AdtValue.value.length().leq(MaxLength)
     )
     then (TheResult = true)
     else (TheResult = false)
     endif
     result = TheResult
    )}
```

Listing 5.88: Messir (MCL-oriented) specification of the operation *is*.

The listing 5.88 provides the Messir (Prolog-oriented) implementation of the operation.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% dtComment

```

```
%msd01
msrop(dtLogin,is,[AdtValue],Result):-
 msrVar(ptBoolean,TheResult),
 msrVar(ptInteger,MaxLength),
(
  (
    (
     MaxLength = [ptInteger,20],
     msrNav([AdtValue],
        [value,length,[],leq,[MaxLength]],
        [[ptBoolean,true]])
    )
     -> TheResult = [ptBoolean,true]
     ; TheResult = [ptBoolean,false]
  )
),
 Result = TheResult
.
/*
| ?- X = [dtLogin,[],[[dtString,[[value,[ptString,'01234567']]],[]]]],
msrNav([X],[is,[]],[Result]).
X = [dtLogin,[],[[dtString,[[value,[ptString,'01234567']]],[]]]],
Result = [ptBoolean,true] ?
yes

| ?- X = [dtLogin,[],[[dtString,[[value,[ptString,'01234567a']]],[]]]],
msrNav([X],[is,[]],[Result]).
X = [dtLogin,[],[[dtString,[[value,[ptString,'01234567a']]],[]]]],
Result = [ptBoolean,false] ?
yes
*/
```

Listing 5.89: Messir (Prolog-oriented) implementation of the operation *is*.

5.28 Primary Types - Operation Schemes for Datatype dtLongitude

5.28.1 Operation Model for is

The `is` operation has the following properties:

OPERATION
is used to determine which strings are considered as valid dtLongitude.
Return type
ptBoolean
Post-Condition (functional)
PostF 1 is true if the value is a real in the interval [-180.0 , +180.0].

The listing 5.89 provides the Messir (MCL-oriented) specification of the operation.

```

/* Post Functional:*/
postF{let TheResult: ptBoolean in
    ( if
     ( AdtValue.value.geq(-180.0)
      and AdtValue.value.leq(+180.0)
     )
     then (TheResult = true)
```

```
      else (TheResult = false)
      endif
      result = TheResult
     )}
```

Listing 5.90: Messir (MCL-oriented) specification of the operation *is*.

The listing 5.90 provides the Messir (Prolog-oriented) implementation of the operation.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% dtPhoneNumber

% msd01
msrop(dtLongitude,is,[AdtValue],Result):-
msrVar(ptBoolean,TheResult),
(
 ( msrNav([AdtValue],
    [value,geq,[[ptReal,-180.0]]],
    [[ptBoolean,true]]),
   msrNav([AdtValue],
    [value,leq,[[ptReal,+180.0]]],
    [[ptBoolean,true]])
 )
 -> (TheResult = [ptBoolean,true])
 ; (TheResult = [ptBoolean,false])
),

 Result = TheResult
.
```

Listing 5.91: Messir (Prolog-oriented) implementation of the operation *is*.

5.29 Primary Types - Operation Schemes for Datatype dtPassword

5.29.1 Operation Model for is

The `is` operation has the following properties:

OPERATION
is used to determine which strings are considered as valid dtPassword.
Return type
ptBoolean
Post-Condition (functional)
PostF 1 is true of the length of the string value is at least 6 characters long.

The listing 5.91 provides the Messir (MCL-oriented) specification of the operation.

```

/* Post Functional:*/
```

```
postF{let TheResult: ptBoolean in
    let MinLength: ptInteger in
    ( if
     ( MinLength = 6
      and AdtValue.value.length().geq(MinLength)
     )
     then (TheResult = true)
     else (TheResult = false)
     endif
     result = TheResult
    )}
```

Listing 5.92: Messir (MCL-oriented) specification of the operation *is*.

The listing 5.92 provides the Messir (Prolog-oriented) implementation of the operation.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% dtComment

%msd01
msrop(dtPassword,is,[AdtValue],Result):-
 msrVar(ptBoolean,TheResult),
 msrVar(ptInteger,MinLength),
(
  (
    (
     MinLength = [ptInteger,6],
     msrNav([AdtValue],
        [value,length,[],geq,[MinLength]],
        [[ptBoolean,true]])
    )
     -> TheResult = [ptBoolean,true]
     ; TheResult = [ptBoolean,false]
  )
),
 Result = TheResult
.
/*
| ?- X = [dtPassword,[],[[dtString,[[value,[ptString,'012345']]],[]]]],
msrNav([X],[is,[]],[Result]).
X = [dtPassword,[],[[dtString,[[value,[ptString,'012345']]],[]]]],
Result = [ptBoolean,true] ?
yes

| ?- X = [dtPassword,[],[[dtString,[[value,[ptString,'01234']]],[]]]],
msrNav([X],[is,[]],[Result]).
X = [dtPassword,[],[[dtString,[[value,[ptString,'01234']]],[]]]],
Result = [ptBoolean,false] ?
yes
*/
```

Listing 5.93: Messir (Prolog-oriented) implementation of the operation *is*.

5.30 Primary Types - Operation Schemes for Datatype dtPhoneNumber

5.30.1 Operation Model for is

The `is` operation has the following properties:

OPERATION
is
used to determine which strings are considered as valid dtPhoneNumber.
Return type
ptBoolean
Post-Condition (functional)
PostF 1 is true of the length of the string value is from 4 to 30 characters. No standard is applied !

The listing 5.93 provides the Messir (MCL-oriented) specification of the operation.

```

/* Post Functional:*/
postF{let TheResult: ptBoolean in
  ( if
   ( AdtValue.value.length().gt(4)
    and AdtValue.value.length().leq(30)
   )
   then (TheResult = true)
   else (TheResult = false)
   endif
   result = TheResult
  )}
```

Listing 5.94: Messir (MCL-oriented) specification of the operation *is*.

The listing 5.94 provides the Messir (Prolog-oriented) implementation of the operation.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% dtPhoneNumber

% msd01
msrop(dtPhoneNumber,is,[AdtValue],Result):-
msrVar(ptBoolean,TheResult),
(
 ( msrNav([AdtValue],
   [value,length,[],gt,[[ptInteger,4]]],
   [[ptBoolean,true]]),
  msrNav([AdtValue],
   [value,length,[],leq,[[ptInteger,30]]],
   [[ptBoolean,true]])
 )

 -> TheResult = [ptBoolean,true]
 ; TheResult = [ptBoolean,false]
),
 Result = TheResult
.
/*
| ?- X = [dtPhoneNumber,[],[[dtString,[[value,[ptString,'(+352) 46 66 44 60 00']]],[]]]],
msrNav([X],[is,[]],[Result]).

X = [dtPhoneNumber,[],[[dtString,[[value,[ptString,'(+352) 46 66 44 60 00']]],[]]]],

```

```
Result = [ptBoolean,true] ?

yes
*/
```

Listing 5.95: Messir (Prolog-oriented) implementation of the operation *is*.

5.31 Primary Types - Operation Schemes for Enumeration etAlertStatus

5.31.1 Operation Model for is

The `is` operation has the following properties:

OPERATION	
is	
used to determine which litteral belongs to the enumeration.	
Return type	
ptBoolean	
Post-Condition (functional)	
PostF 1	true iff the value is equal to one of the following values: `pending, valid, invalid`

The listing 5.95 provides the Messir (MCL-oriented) specification of the operation.

```

/* Post Functional:*/
postF{let TheResult: ptBoolean in
    ( if
     ( self = pending
      or self = valid
      or self = invalid
     )
     then (TheResult = true)
     else (TheResult = false)
     endif
     result = TheResult
    )}
```

Listing 5.96: Messir (MCL-oriented) specification of the operation *is*.

The listing 5.96 provides the Messir (Prolog-oriented) implementation of the operation.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% etAlertStatus

% msd01
msrop(etAlertStatus,is,[AdtValue],Result):-
msrVar(ptBoolean,TheResult),
(
```

```
12  (
13     member(AdtValue,[pending, valid, invalid])
14  )
15 -> TheResult = [ptBoolean,true]
16 ; TheResult = [ptBoolean,false]
17),
18 Result = TheResult
19.
```

Listing 5.97: Messir (Prolog-oriented) implementation of the operation *is*.

5.32 Primary Types - Operation Schemes for Enumeration etCorruptionKind

5.32.1 Operation Model for is

The `is` operation has the following properties:

OPERATION
is
Return type
ptBoolean
Post-Condition (functional)
PostF 1

The listing 5.97 provides the Messir (MCL-oriented) specification of the operation.

```
1
2  /* Post Functional:*/
3  postF{let TheResult: ptBoolean in
4      ( if
5       ( self = regular
6        or self = corrupted
7        or self = restored
8       )
9       then (TheResult = true)
10      else (TheResult = false)
11      endif
12      result = TheResult
13     )}
```

Listing 5.98: Messir (MCL-oriented) specification of the operation *is*.

5.33 Primary Types - Operation Schemes for Enumeration etCrisisStatus

5.33.1 Operation Model for is

The `is` operation has the following properties:

OPERATION	
is	
used to determine which litteral belongs to the enumeration.	
Return type	
ptBoolean	
Post-Condition (functional)	
PostF 1	true iff the value is equal to one of the following values: `pending`, `handled`, `solved`, `closed`.

The listing 5.98 provides the Messir (MCL-oriented) specification of the operation.

```

/* Post Functional:*/
postF{let TheResult: ptBoolean in
    ( if
     ( self = pending
      or self = handled
      or self = solved
      or self = closed
     )
     then (TheResult = true)
     else (TheResult = false)
     endif
     result = TheResult
    )}
```

Listing 5.99: Messir (MCL-oriented) specification of the operation *is*.

The listing 5.99 provides the Messir (Prolog-oriented) implementation of the operation.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% etCrisisStatus

% msd01
msrop(etCrisisStatus,is,[AdtValue],Result):-
msrVar(ptBoolean,TheResult),
(
  (
     member(AdtValue,[pending, handled, solved, closed])
  )
 -> TheResult = [ptBoolean,true]
 ; TheResult = [ptBoolean,false]
),
 Result = TheResult
.
```

Listing 5.100: Messir (Prolog-oriented) implementation of the operation *is*.

5.34 Primary Types - Operation Schemes for Enumeration etCrisisType

5.34.1 Operation Model for is

The `is` operation has the following properties:

OPERATION
is used to determine which litteral belongs to the enumeration.
Return type
ptBoolean
Post-Condition (functional)
PostF 1 true iff the value is equal to one of the following values: `small, medium, huge`

The listing 5.100 provides the Messir (MCL-oriented) specification of the operation.

```

/* Post Functional:*/
postF{let TheResult: ptBoolean in
    ( if
     ( self = small
      or self = medium
      or self = huge
     )
     then (TheResult = true)
     else (TheResult = false)
     endif
     result = TheResult
    )}
```

Listing 5.101: Messir (MCL-oriented) specification of the operation *is*.

The listing 5.101 provides the Messir (Prolog-oriented) implementation of the operation.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% etCrisisType

% msd01
msrop(etCrisisType,is,[AdtValue],Result):-
msrVar(ptBoolean,TheResult),
(
  (
    member(AdtValue,[small, medium, huge])
  )
 -> TheResult = [ptBoolean,true]
 ; TheResult = [ptBoolean,false]
),
 Result = TheResult
.
```

Listing 5.102: Messir (Prolog-oriented) implementation of the operation *is*.

5.35 Primary Types - Operation Schemes for Enumeration etHumanKind

5.35.1 Operation Model for is

The `is` operation has the following properties:

OPERATION	
is	
used to determine which litteral belongs to the enumeration.	
Return type	
ptBoolean	
Post-Condition (functional)	
PostF 1	true iff the value is equal to one of the following values: `witness`,`victim`,`anonym`

The listing 5.102 provides the Messir (MCL-oriented) specification of the operation.

```

/* Post Functional:*/
postF{let TheResult: ptBoolean in
    ( if
     ( self = witness
      or self = victim
      or self = anonymous
     )
     then (TheResult = true)
     else (TheResult = false)
     endif
     result = TheResult
    )}
```

Listing 5.103: Messir (MCL-oriented) specification of the operation *is*.

The listing 5.103 provides the Messir (Prolog-oriented) implementation of the operation.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% etHumanKind

% msd01
msrop(etHumanKind,is,[AdtValue],Result):-
msrVar(ptBoolean,TheResult),
(
  (
     member(AdtValue,[witness,victim,anonymous])
  )
 -> TheResult = [ptBoolean,true]
 ; TheResult = [ptBoolean,false]
),
 Result = TheResult
.
```

Listing 5.104: Messir (Prolog-oriented) implementation of the operation *is*.

5.36 Primary Types - Operation Schemes for Enumeration etInjuryKind

5.36.1 Operation Model for is

The `is` operation has the following properties:

OPERATION
is used to determine which litteral belongs to the enumeration.
Return type
ptBoolean
Post-Condition (functional)
PostF 1 true if the value is equal to one of the following values: `arm`,`head`,`leg`,`chest`

The listing 5.104 provides the Messir (MCL-oriented) specification of the operation.

```
1
2  /* Post Functional:*/
3  postF{let TheResult: ptBoolean in
4      ( if
5       ( self = arm
6        or self = leg
7        or self = chest
8        or self = head
9       )
10      then (TheResult = true)
11      else (TheResult = false)
12      endif
13      result = TheResult
14     )}
```

Listing 5.105: Messir (MCL-oriented) specification of the operation *is*.

5.37 Secondary Types - Operation Schemes for Classes

There are no elements in this category in the system analysed.

5.38 Secondary Types - Operation Schemes for Datatype dtSMS

5.38.1 Operation Model for is

The `is` operation has the following properties:

OPERATION
is used to determine which strings are considered as valid comments
Return type
ptBoolean
Post-Condition (functional)

continues in next page . . .

. . . Operation table continuation

PostF 1	true iff the length of the string value is not more than 160 characters.

The listing 5.105 provides the Messir (MCL-oriented) specification of the operation.

```

/* Post Functional:*/
postF{let TheResult: ptBoolean in
    let MaxLength: ptInteger in
    ( if
     ( MaxLength = 160
      and AdtValue.value.length().leq(MaxLength)
     )
     then (TheResult = true)
     else (TheResult = false)
     endif
     result = TheResult
    )}
```

Listing 5.106: Messir (MCL-oriented) specification of the operation *is*.

The listing 5.106 provides the Messir (Prolog-oriented) implementation of the operation.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% dtComment

%msd01
msrop(dtSMS,is,[AdtValue],Result):-
 msrVar(ptBoolean,TheResult),
 msrVar(ptInteger,MaxLength),
(
  (
    (
     MaxLength = [ptInteger,160],
     msrNav([AdtValue],
        [value,length,[],leq,[MaxLength]],
        [[ptBoolean,true]])
    )
     -> TheResult = [ptBoolean,true]
     ; TheResult = [ptBoolean,false]
  )
),
 Result = TheResult
.
```

Listing 5.107: Messir (Prolog-oriented) implementation of the operation *is*.

5.39 Secondary Types - Operation Schemes for Enumerations

There are no elements in this category in the system analysed.

Chapter 6

Test Model(s)

6.1 Test Model for testcase01

this positive test case intends to verify the correctness of the execution of a simple instance of the `suDeployAndRun` use case.

6.1.1 Test Steps Specification

6.1.1.1 testcase01-ts01oeCreateSystemAndEnvironment-actMsrCreator.outactMsrCreator.oeCreat

The `testcase01-ts01oeCreateSystemAndEnvironment-actMsrCreator.outactMsrCreator.oeCre` has the following properties:

Test Step	
ts01oeCreateSystemAndEnvironment This test step initializes the system state and environment.	
Test Sent Message	
TSM 1	**out:Creator** **sends to system** **actMsrCreator.outactMsrCreator.oeCreateSystemAndEnvironment (AqtyComCompanies)**
Variables	
V 1	**Creator:icrash.environment.actMsrCreator** only actMsrtCreator actors can trigger the system and environment creation and initialization.
Constraints	
C 1	the number of communication company actor instances present in the environment is equal to four to represent all the communication companies available in Luxembourg.
Oracle Constraints	
OC 1	true for testing only the executability (is available and can be triggered) of the operation.

The listing 6.1 provides the Messir (MCL-oriented) specification of the test step.

```
1
2 variables{
3  Creator:actMsrCreator
4  AqtyComCompanies: ptInteger
5 }
6
7 constraints{
8  AqtyComCompanies = 4
9 }
10
11 oracle{
12  constraints{
13   true
14  }
15 }
```

Listing 6.1: **Messir** (MCL-oriented) specification of the test step *testcase01-ts01oeCreateSystemAndEnvironment*.

The listing 6.2 provides the **Messir** (Prolog-oriented) implementation of the test step.

```
1%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrTest/1.
4%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5
6%-----------------------------------------------
7:-msrTestAddStep([system,sim,1,1]).
8%-----------------------------------------------
9msrTest([[system,sim,1,1],
10      [[target,oeCreateSystemAndEnvironment],
11      [context,Context],
12      [inputParameters,InputParameters],
13      [outputParameters,OutputParameters],
14      [comments,Comments],
15      TestResult]
16     ]):-
17%%-----------------------------------------------
18(
19 (
20% Step 0
21
22%% Context Declaration
23%% N.A.
24
25%% Input Parameters Declaration
26msrVar(ptInteger,AqtyComCompanies),
27
28%% Output Parameters Declaration
29%% N.A.
30
31%% Context Specification
32%% N.A.
33
34%% Input Parameters Specification
35AqtyComCompanies = [ptInteger,4],
36
37%% Output Parameters Specification
38%% N.A.
39
40%% Test Specification
41Target = launchCreateSystemAndEnvironment,
42ParametersList =
43[ [AqtyComCompanies],
44 Result
45], !,
```

```
46GoalGet=..[Target | ParametersList],
47
48%% Oracle specification
49OracleGet=..[true]
50)
51->
52%% Test Interpretation
53((GoalGet,!)
54-> ((OracleGet,!)
55  -> TestResult = [success]
56  ; TestResult = [failedAtOracle])
57; TestResult = [failedAtGoal]
58)
59; TestResult = [failedAtTestDeclarationOrSpecification]
60),
61%% Test Outcome
62Context = [],
63InputParameters = ['AqtyComCompanies',AqtyComCompanies],
64OutputParameters = [],
65Comments = 'System launch ! '
66.
```

Listing 6.2: Messir (Prolog-oriented) implementation of the test step *testcase01-ts01oeCreateSystemAndEnvironment*.

6.1.1.2 testcase01-ts02oeSetClock-actActivator.outactActivator.oeSetClock

The `testcase01-ts02oeSetClock-actActivator.outactActivator.oeSetClock` has the following properties:

Test Step	
ts02oeSetClock test the update of the current time.	
Test Sent Message	
TSM 1	**out:TheActor** **sends to system** **actActivator.outactActivator.oeSetClock (ACurrentClock)**
Variables	
V 1	**TheActor:actActivator** proactive actor responsible of requesting the update of the system's clock.
Constraints	
C 1 C 2	TheActor is any instance existing in the current environment status. ACurrentClock is a fixed date equal to the 24th November 2017 at 15:20:00 using a 24-hours notation [1].
Oracle Constraints	
OC 1	true for testing only the executability (is available and can be triggered) of the operation.

The listing 6.3 provides the Messir (MCL-oriented) specification of the test step.

[1] for more details see the ISO 8601 Data elements and interchange formats Information interchange Representation of dates and times - http://www.iso.org/iso/home/standards/iso8601.htm

```
1
2  variables{
3   TheActor:actActivator
4   ACurrentClock:dtDateAndTime
5  }
6
7  constraints{
8   TheActor=TheSystem.rnactActivator->any2(true)
9   ACurrentClock.date.year.value = 2017
10  ACurrentClock.date.month.value = 11
11  ACurrentClock.date.day.value = 24
12  ACurrentClock.time.hour.value = 15
13  ACurrentClock.time.minute.value = 20
14  ACurrentClock.time.second.value = 00
15 }
16
17 oracle{
18  constraints{
19   true
20  }
21 }
```

Listing 6.3: Messir (MCL-oriented) specification of the test step *testcase01-ts02oeSetClock*.

6.1.1.3 testcase01-ts03oeLogin-actAdministrator.outactAdministrator.oeLogin

The `testcase01-ts03oeLogin-actAdministrator.outactAdministrator.oeLogin` has the following properties:

Test Step	
ts03oeLogin test the authentified access of the administrator	
Test Sent Message	
TSM 1	**out:TheActor** **sends to system** **actAdministrator.outactAdministrator.oeLogin (AdtLogin, AdtPassword)**
Variables	
V 1	**TheActor:actAdministrator** an actAdministor actor as subtype of actAuthenticated can send oeLogin messages to the system.
Constraints	
C 1	TheActor is any `actAdministrator` instance existing in the environment. It is thus expected that there exist at least one.
C 2	AdtLogin has its value attribute equal to the primitive string 'icrashadmin' (which is the correct administrator login known by the system after the step one.)
C 3	AdtPassword has its value attribute equal to the primitive string '7WXC1359' (which is the correct administrator password known by the system after the step one.)
Oracle Constraints	
OC 1	the `AMessage` value is expected to be equal to the primitive string 'You are logged ! Welcome ...'
OC 2	TheActor receives from system ieMessage(AMessage)

The listing 6.4 provides the Messir (MCL-oriented) specification of the test step.

```
1
2  variables{
3   TheActor : actAdministrator
4   AdtLogin:dtLogin
5   AdtPassword:dtPassword
6  }
7
8  constraints{
9   TheActor=TheSystem.rnactAdministrator->any2(true)
10  AdtLogin.value.eq('icrashadmin')
11  AdtPassword.value.eq('7WXC1359')
12 }
13
14 oracle{
15  variables{
16   AMessage:ptString
17  }
18  constraints{
19   AMessage = 'You are logged ! Welcome ...'
20   TheActor.inactAdministrator.ieMessage(AMessage)
21  }
22 }
```

Listing 6.4: Messir (MCL-oriented) specification of the test step *testcase01-ts03oeLogin*.

6.1.1.4 testcase01-ts04oeAddCoordinator-actAdministrator.outactAdministrator.oeAddCoordinato

The `testcase01-ts04oeAddCoordinator-actAdministrator.outactAdministrator.oeAddCoordi` has the following properties:

Test Step	
ts04oeAddCoordinator to test the add of a new coordinator by an administrator.	
Test Sent Message	
TSM 1	**out:TheActor** **sends to system** **actAdministrator.outactAdministrator.oeAddCoordinator (AdtCoordinatorID, AdtLogin, AdtPassword)**
Variables	
V 1	**TheActor:actAdministrator** actAdministor actors as being the only one allowed to add coordinators.
Constraints	
C 1	TheActor is any `actAdministrator` instance existing in the environment. It is expected that there exists at least one which is the same during all the test case.
C 2	AdtCoordinatorID is equal to 1 to set the new coordinator ID
C 3	AdtLogin has its value attribute equal to the primitive string 'steve' which is the ID defined for the new coordinator.
C 4	AdtPassword has its value attribute equal to the primitive string 'pwdMessirExcalibur2017' which is the password to be set for steve.
Oracle Constraints	

continues in next page . . .

. . . Test Step table continuation

OC 1	the administrator should have been acknowledged for the adding of the new coordinator.

The listing 6.5 provides the Messir (MCL-oriented) specification of the test step.

```

variables{
 TheActor : actAdministrator
 AdtCoordinatorID : dtCoordinatorID
 AdtLogin:dtLogin
 AdtPassword:dtPassword
}

constraints{
 TheActor = TheSystem.rnactAdministrator->any2(true)
 AdtCoordinatorID.value.eq('1')
 AdtLogin.value.eq('steve')
 AdtPassword.value.eq('pwdMessirExcalibur2017')
}

oracle{
 constraints{
  TheActor.inactAdministrator.ieCoordinatorAdded()
 }
}
```

Listing 6.5: Messir (MCL-oriented) specification of the test step *testcase01-ts04oeAddCoordinator*.

6.1.1.5 testcase01-ts05oeLogout-actAdministrator.outactAdministrator.oeLogout

The `testcase01-ts05oeLogout-actAdministrator.outactAdministrator.oeLogout` has the following properties:

TEST STEP	
ts05oeLogout to test the loggout of a connected administrator.	
Test Sent Message	
TSM 1	**out:TheActor** **sends to system** **actAdministrator.outactAdministrator.oeLogout** ()
Variables	
V 1	**TheActor:actAdministrator** an actAdministor actor as subtype of actAuthenticated can send oeLogout messages to the system.
Constraints	
C 1	TheActor is any `actAdministrator` instance existing in the environment. It is expected that there exists at least one which is the same during all the test case.
Oracle Constraints	

continues in next page . . .

. . . Test Step table continuation

OC 1	the `AMessage` value is expected to be equal to the primitive string 'You are logged out ! Good Bye ...'
OC 2	the administrator should have received the messahe AMessage.

The listing 6.6 provides the Messir (MCL-oriented) specification of the test step.

```
1
2  variables{
3   TheActor : actAdministrator
4  }
5
6  constraints{
7   TheActor = TheSystem.rnactAdministrator->any2(true)
8  }
9
10 oracle{
11  variables{
12   AMessage:ptString
13  }
14  constraints{
15   AMessage = 'You are logged out ! Good Bye ...'
16   TheActor.inactAdministrator.ieMessage(AMessage)
17  }
18 }
```

Listing 6.6: Messir (MCL-oriented) specification of the test step *testcase01-ts05oeLogout*.

6.1.1.6 testcase01-ts06oeSetClock02-actActivator.outactActivator.oeSetClock

The `testcase01-ts06oeSetClock02-actActivator.outactActivator.oeSetClock` has the following properties:

Test Step	
ts06oeSetClock02 test the update of the current time.	
Test Sent Message	
TSM 1	**out:TheActor** **sends to system** **actActivator.outactActivator.oeSetClock (ACurrentClock)**
Variables	
V 1	**TheActor:icrash.environment.actActivator** proactive actors responsible of requesting the update of the system's clock.
Constraints	
C 1 C 2	TheActor is any instance existing in the current environment status. ACurrentClock is a fixed date equal to the 26th November 2017 at 10:15:00 using a 24-hours notation.
Oracle Constraints	
OC 1	true for testing only the executability (is available and can be triggered) of the operation.

The listing 6.7 provides the Messir (MCL-oriented) specification of the test step.

```

variables{
 TheActor:actActivator
 ACurrentClock:dtDateAndTime
}

constraints{
 TheActor=TheSystem.rnactActivator->any2(true)
 ACurrentClock.date.year.value = 2017
 ACurrentClock.date.month.value = 11
 ACurrentClock.date.day.value = 26
 ACurrentClock.time.hour.value = 10
 ACurrentClock.time.minute.value = 15
 ACurrentClock.time.second.value = 00
}

oracle{
 constraints{
  true
 }
}
```

Listing 6.7: Messir (MCL-oriented) specification of the test step *testcase01-ts06oeSetClock02*.

6.1.1.7 testcase01-ts07oeAlert1-actComCompany.outactComCompany.oeAlert

The `testcase01-ts07oeAlert1-actComCompany.outactComCompany.oeAlert` has the following properties:

Test Step	
ts07oeAlert1 tests the declaration of a new alert functionality.	
Test Sent Message	
TSM 1	**out:TheActor** **sends to system** **actComCompany.outactComCompany.oeAlert (AetHumanKind, AdtDate, AdtTime, AdtPhoneNumber, AdtGPSLocation, AdtComment)**
Variables	
V 1	**TheActor:actComCompany** actComCompany actors transfer alert declaration messages.
Constraints	
C 1	TheActor is any instance existing in the current environment status. It is expected to exist at least one.
C 2	AetHumanKind is equal to witness
C 3	AdtDate is equal to the 26th of November 2017
C 4	AdtTime is equal to 10:10:16 using a 24-hours.
C 5	AdtPhoneNumber is equal to the ptString value '+3524666445252'.
C 6	AdtGPSLocation is equal to (49.627675 , 6.159590).
C 7	AdtComment is equal to '3 cars involved in an accident.'

continues in next page . . .

. . . Test Step table continuation

Oracle Constraints	
OC 1	AdtSMS is equal to the ptString 'Your alert has been registered. We will handle it and keep you informed'.
OC 2	AdtSMS is sent to the phone number AdtPhoneNumber using the communication company having sent the alert using its ieSmsSend input message.

The listing 6.8 provides the Messir (MCL-oriented) specification of the test step.

```

variables{
 TheActor : actComCompany
 AetHumanKind:etHumanKind
 AdtDate:dtDate
 AdtTime:dtTime
 AdtPhoneNumber:dtPhoneNumber
 AdtGPSLocation:dtGPSLocation
 AdtComment:dtComment
}

constraints{
 TheActor = TheSystem.rnactComCompany->any2(true)
 AetHumanKind = witness
 AdtDate.year.value = 2017
 AdtDate.month.value = 11
 AdtDate.day.value = 26
 AdtTime.hour.value = 10
 AdtTime.minute.value = 10
 AdtTime.second.value = 16
 AdtPhoneNumber.value = '+3524666445252'
 AdtGPSLocation.latitude.value = 49.627675
 AdtGPSLocation.longitude.value = 6.159590
 AdtComment.value = '3 cars involved in an accident.'
}

oracle{
 variables{
  AdtSMS:dtSMS
 }
 constraints{
  AdtSMS.value = 'Your alert has been registered. We will handle it and keep you informed'
  TheActor.inactComCompany.ieSmsSend(AdtPhoneNumber,AdtSMS)
 }
}
```

Listing 6.8: Messir (MCL-oriented) specification of the test step *testcase01-ts07oeAlert1*.

6.1.1.8 testcase01-ts08oeSetClock03-actActivator.outactActivator.oeSetClock

The `testcase01-ts08oeSetClock03-actActivator.outactActivator.oeSetClock` has the following properties:

TEST STEP
ts08oeSetClock03 test the update of the current time.
Test Sent Message

continues in next page . . .

. . . Test Step table continuation

TSM 1	**out:TheActor** **sends to system** **actActivator.outactActivator.oeSetClock (ACurrentClock)**
Variables	
V 1	**TheActor:actActivator** proactive actor responsible of requesting the update of the system's clock.
Constraints	
C 1 C 2	TheActor is any instance existing in the current environment status. ACurrentClock is a fixed date equal to the 26th November 2017 at 10:30:00 using a 24-hours notation.
Oracle Constraints	
OC 1	true for testing only the executability (is available and can be triggered) of the operation.

The listing 6.9 provides the Messir (MCL-oriented) specification of the test step.

```
1
2  variables{
3   TheActor:actActivator
4   ACurrentClock:dtDateAndTime
5  }
6
7  constraints{
8   TheActor=TheSystem.rnactActivator->any2(true)
9   ACurrentClock.date.year.value = 2017
10  ACurrentClock.date.month.value = 11
11  ACurrentClock.date.day.value = 26
12  ACurrentClock.time.hour.value = 10
13  ACurrentClock.time.minute.value = 30
14  ACurrentClock.time.second.value = 00
15 }
16
17 oracle{
18  constraints{
19   true
20  }
21 }
```

Listing 6.9: Messir (MCL-oriented) specification of the test step *testcase01-ts08oeSetClock03*.

6.1.1.9 testcase01-ts09oeSollicitateCrisisHandling-actActivator.outactActivator.oeSollicitateCr

The `testcase01-ts09oeSollicitateCrisisHandling-actActivator.outactActivator.oeSo` has the following properties:

TEST STEP
ts09oeSollicitateCrisisHandling test the proactive sollication to handle an alert.
Test Sent Message

continues in next page . . .

. . . Test Step table continuation

TSM 1	**out:TheActor** **sends to system** **actActivator.outactActivator.oeSollicitateCrisisHandling** ()
Variables	
V 1	**TheActor:icrash.environment.actActivator** proactive actor responsible of triggering sollicitation functionality.
Constraints	
C 1	TheActor is any instance existing in the current environment status. It is expected to exist at least one.
Oracle Variables	
OV 1	**TheAdministrator:actAdministrator** actAdministrator actors can be sollicitated to handle alerts.
OV 2	**TheCoordinator:actCoordinator** actCoordinator actors can be sollicitated to handle alerts.
OV 3	**AMessageForCrisisHandlers:ptString** messages sent to sollicitated actors are of type ptString.
Oracle Constraints	
OC 1	TheAdministrator is any instance existing in the current environment status. It is expected to exist at least one.
OC 2	TheCoordinator is any instance existing in the current environment status. It is expected to exist at least one.
OC 3	AMessageForCrisisHandlers is equal to the ptString 'There are alerts pending since more than the defined delay. Please REACT !'
OC 4	TheCoordinator and TheAdministrator have received the message AMessageForCrisisHandlers.

The listing 6.10 provides the Messir (MCL-oriented) specification of the test step.

```
1
2  variables{
3   TheActor : actActivator
4  }
5
6  constraints{
7   TheActor = TheSystem.rnactActivator->any2(true)
8  }
9
10 oracle{
11  variables{
12   TheAdministrator:actAdministrator
13   TheCoordinator:actCoordinator
14   AMessageForCrisisHandlers:ptString
15  }
16  constraints{
17   TheAdministrator = TheSystem.rnactAdministrator->any2(true)
18   TheCoordinator = TheSystem.rnactCoordinator->any2(true)
19   AMessageForCrisisHandlers = 'There are alerts pending since more than the defined delay. Please
        REACT !'
20   TheAdministrator.inactAdministrator.ieMessage(AMessageForCrisisHandlers)
```

```
21   TheCoordinator.inactAdministrator.ieMessage(AMessageForCrisisHandlers)
22  }
23 }
```

Listing 6.10: Messir (MCL-oriented) specification of the test step *testcase01-ts09oeSollicitateCrisisHandling*.

6.1.1.10 testcase01-ts10oeLogin02-actAuthenticated.outactAuthenticated.oeLogin

The `testcase01-ts10oeLogin02-actAuthenticated.outactAuthenticated.oeLogin` has the following properties:

Test Step	
ts10oeLogin02	
test the authentified access of the coordinator	
Test Sent Message	
TSM 1	**out:TheActor** **sends to system** **actAuthenticated.outactAuthenticated.oeLogin (AdtLogin, AdtPassword)**
Variables	
V 1	**TheActor:actCoordinator** an actCoordinator actor as subtype of actAuthenticated can send oeLogin messages to the system.
Constraints	
C 1	TheActor is any `actAdministrator` instance existing in the environment. It is thus expected that there exist at least one.
C 2	AdtLogin has its value attribute equal to the primitive string 'icrashadmin' (which is the correct administrator login known by the system after the step one.)
C 3	AdtPassword has its value attribute equal to the primitive string '7WXC1359' (which is the correct administrator password known by the system after the step one.)
Oracle Constraints	
OC 1	the `AMessage` value is expected to be equal to the primitive string 'You are logged ! Welcome ...'

The listing 6.11 provides the Messir (MCL-oriented) specification of the test step.

```
1
2 variables{
3  TheActor : actCoordinator
4  AdtLogin:dtLogin
5  AdtPassword:dtPassword
6 }
7
8 constraints{
9  TheActor = TheSystem.rnactCoordinator->select(a | a.rnctCoordinator.login.value.eq('steve'))->any2
       (true)
10  AdtLogin.value.eq('steve')
11  AdtPassword.value.eq('pwdMessirExcalibur2017')
```

```
12 }
13
14 oracle{
15  variables{
16   AMessage:ptString
17  }
18  constraints{
19   AMessage = 'You are logged ! Welcome ...'
20   TheActor.inactAuthenticated.ieMessage(AMessage)
21  }
22 }
```

Listing 6.11: Messir (MCL-oriented) specification of the test step *testcase01-ts10oeLogin02*.

6.1.1.11 testcase01-ts11oeGetCrisisSet-actCoordinator.outactCoordinator.oeGetCrisisSet

The `testcase01-ts11oeGetCrisisSet-actCoordinator.outactCoordinator.oeGetCrisisSet` has the following properties:

Test Step	
ts11oeGetCrisisSet cf. actor documentation	
Test Sent Message	
TSM 1	**out:TheActor** **sends to system** **actCoordinator.outactCoordinator.oeGetCrisisSet (AetCrisisStatus)**
Variables	
V 1	**TheActor:icrash.environment.actCoordinator** cf. actor documentation
V 2	**AetCrisisStatus:icrash.concepts.primarytypes.datatypes.etCrisisStatus** cf. actor documentation
V 3	**ActCrisis:icrash.concepts.primarytypes.classes.ctCrisis** cf. actor documentation
Constraints	
C 1	TheActor is the coordinator actor related to a coordinator in the system's state having steve as login value
C 2	AetCrisisStatus value is pending
Oracle Constraints	
OC 1	ActCrisis is any ctCrisis instance that has been sent to TheActor.

The listing 6.12 provides the Messir (MCL-oriented) specification of the test step.

```
1
2 variables{
3  TheActor : actCoordinator
4  AetCrisisStatus : etCrisisStatus
5 }
6
7 constraints{
```

```
 8   TheActor=TheSystem.rnactCoordinator
 9         ->select(a | a.rnctCoordinator.login.value.eq('steve'))
10         ->any2(true)
11   AetCrisisStatus = pending
12  }
13
14 oracle{
15  variables{
16   ActCrisis:ctCrisis
17  }
18  constraints{
19   TheActor.inactCoordinator.ieSendACrisis(ActCrisis)
20  }
21 }
```

Listing 6.12: Messir (MCL-oriented) specification of the test step *testcase01-ts11oeGetCrisisSet*.

6.1.1.12 testcase01-ts12oeSetCrisisHandler-actCoordinator.outactCoordinator.oeSetCrisisHan

The `testcase01-ts12oeSetCrisisHandler-actCoordinator.outactCoordinator.oeSetCris` has the following properties:

Test Step	
ts12oeSetCrisisHandler cf. actor documentation	
Test Sent Message	
TSM 1	**out:TheActor** **sends to system** **actCoordinator.outactCoordinator.oeSetCrisisHandler (AdtCrisisID)**
Variables	
V 1	**TheActor:icrash.environment.actCoordinator** cf. actor documentation
V 2	**TheComCompany:icrash.environment.actComCompany** cf. actor documentation
V 3	**TheCoordinator:icrash.environment.actCoordinator** cf. actor documentation
V 4	**AdtCrisisID:icrash.concepts.primarytypes.datatypes.dtCrisisID** cf. actor documentation
V 5	**AMessage:lu.uni.lassy.messir.libraries.primitives.ptString** cf. actor documentation
V 6	**AdtPhoneNumber:icrash.concepts.primarytypes.datatypes.dtPhoneNumber** cf. actor documentation
V 7	**AdtSMS:icrash.concepts.secondarytypes.datatypes.dtSMS** cf. actor documentation
V 8	**ActAlert:icrash.concepts.primarytypes.classes.ctAlert** cf. actor documentation
Constraints	
C 1	TheActor is the coordinator actor related to a coordinator in the system's state having steve as login value
C 2	AdtCrisisID as a value of 1

continues in next page . . .

. . . Test Step table continuation

C 3	AMessage is the string 'You are now considered as handling the crisis !'
C 4	AdtPhoneNumber
C 5	AdtSMS has for value the string 'The handling of your alert by our services is in progress !'
Oracle Constraints	
OC 1	there is a communication company actor that received the message ieSmsSend(AdtPhoneNumber,AdtSMS)
OC 2	there is a coordinator actor that received an alert using the message ieSendAnAlert(ActAlert)

The listing 6.13 provides the Messir (MCL-oriented) specification of the test step.

```
1
2  variables{
3   TheActor : actCoordinator
4   AdtCrisisID : dtCrisisID
5  }
6
7  constraints{
8   TheActor=TheSystem.rnactCoordinator
9         ->select(a | a.rnctCoordinator.login.value.eq('steve'))
10        ->any2(true)
11 }
12
13 oracle{
14  variables{
15   AMessage:ptString
16   AdtPhoneNumber:dtPhoneNumber
17   AdtSMS:dtSMS
18   ActAlert:ctAlert
19   TheComCompany: actComCompany
20   TheCoordinator:actCoordinator
21  }
22  constraints{
23   AMessage = 'You are now considered as handling the crisis !'
24   AdtSMS.value = 'The handling of your alert by our services is in progress !'
25   TheComCompany.inactComCompany.ieSmsSend(AdtPhoneNumber,AdtSMS)
26   TheCoordinator.inactCoordinator.ieSendAnAlert(ActAlert)
27   TheActor.inactAuthenticated.ieMessage(AMessage)
28  }
29 }
```

Listing 6.13: Messir (MCL-oriented) specification of the test step *testcase01-ts12oeSetCrisisHandler*.

6.1.1.13 testcase01-ts13oeSetClock04-actActivator.outactActivator.oeSetClock

The `testcase01-ts13oeSetClock04-actActivator.outactActivator.oeSetClock` has the following properties:

TEST STEP
ts13oeSetClock04 cf. actor documentation
Test Sent Message

continues in next page . . .

. . . Test Step table continuation

TSM 1	**out:TheActor** **sends to system** **actActivator.outactActivator.oeSetClock (ACurrentClock)**
Variables	
V 1	**TheActor:icrash.environment.actActivator** cf. actor documentation
V 2	**ACurrentClock:lu.uni.lassy.messir.libraries.calendar.dtDateAndTime** cf. actor documentation
Constraints	
C 1	TheActor
C 2	ACurrentClock

The listing 6.14 provides the Messir (MCL-oriented) specification of the test step.

```

variables{
 TheActor:actActivator
 ACurrentClock:dtDateAndTime
}

constraints{
 TheActor=TheSystem.rnactActivator->any2(true)
 ACurrentClock.date.year.value = 2017
 ACurrentClock.date.month.value = 11
 ACurrentClock.date.day.value = 26
 ACurrentClock.time.hour.value = 10
 ACurrentClock.time.minute.value = 45
 ACurrentClock.time.second.value = 00
}

oracle{
 constraints{
  true
 }
}
```

Listing 6.14: Messir (MCL-oriented) specification of the test step *testcase01-ts13oeSetClock04*.

6.1.1.14 testcase01-ts14oeValidateAlert-actCoordinator.outactCoordinator.oeValidateAlert

The `testcase01-ts14oeValidateAlert-actCoordinator.outactCoordinator.oeValidateAl` has the following properties:

TEST STEP
ts14oeValidateAlert cf. actor documentation
Test Sent Message

continues in next page . . .

. . . Test Step table continuation

TSM 1	**out:TheActor** **sends to system** **actCoordinator.outactCoordinator.oeValidateAlert (AdtAlertID)**
Variables	
V 1	**TheActor:icrash.environment.actCoordinator** cf. actor documentation
V 2	**AdtAlertID:icrash.concepts.primarytypes.datatypes.dtAlertID** cf. actor documentation
V 3	**AMessage:lu.uni.lassy.messir.libraries.primitives.ptString** cf. actor documentation
Constraints	
C 1	TheActor is the coordinator actor related to a coordinator in the system's state having steve as login value
C 2	AdtAlertID
C 3	AMessage
Oracle Constraints	
OC 1	

The listing 6.15 provides the Messir (MCL-oriented) specification of the test step.

```
1
2  variables{
3   TheActor : actCoordinator
4   AdtAlertID : dtAlertID
5  }
6
7  constraints{
8   TheActor=TheSystem.rnactCoordinator
9         ->select(a | a.rnctCoordinator.login.value.eq('steve'))
10        ->any2(true)
11 }
12
13 oracle{
14  variables{
15   AMessage:ptString
16  }
17  constraints{
18   AMessage = 'The Alert is now declared as valid !'
19   TheActor.actAuthenticated.inactAuthenticated.ieMessage(AMessage)
20  }
21 }
```

Listing 6.15: Messir (MCL-oriented) specification of the test step *testcase01-ts14oeValidateAlert*.

6.1.1.15 testcase01-ts15oeAlert2-actComCompany.outactComCompany.oeAlert

The `testcase01-ts15oeAlert2-actComCompany.outactComCompany.oeAlert` has the following properties:

Test Step	
ts15oeAlert2 cf. actor documentation	
Test Sent Message	
TSM 1	**out:TheActor** **sends to system** **actComCompany.outactComCompany.oeAlert (AetHumanKind, AdtDate, AdtTime, AdtPhoneNumber, AdtGPSLocation, AdtComment)**
Variables	
V 1	**TheActor:icrash.environment.actComCompany** cf. actor documentation
V 2	**AetHumanKind:icrash.concepts.primarytypes.datatypes.etHumanKind** cf. actor documentation
V 3	**AdtDate:lu.uni.lassy.messir.libraries.calendar.dtDate** cf. actor documentation
V 4	**AdtTime:lu.uni.lassy.messir.libraries.calendar.dtTime** cf. actor documentation
V 5	**AdtPhoneNumber:icrash.concepts.primarytypes.datatypes.dtPhoneNumber** cf. actor documentation
V 6	**AdtGPSLocation:icrash.concepts.primarytypes.datatypes.dtGPSLocation** cf. actor documentation
V 7	**AdtComment:icrash.concepts.primarytypes.datatypes.dtComment** cf. actor documentation
V 8	**AdtSMS:icrash.concepts.secondarytypes.datatypes.dtSMS** cf. actor documentation
Constraints	
C 1	TheActor
C 2	AetHumanKind
C 3	AdtDate
C 4	AdtTime
C 5	AdtPhoneNumber
C 6	AdtGPSLocation
C 7	AdtComment
C 8	AdtSMS
Oracle Constraints	
OC 1	

The listing 6.16 provides the Messir (MCL-oriented) specification of the test step.

```

variables{
 TheActor : actComCompany
 AetHumanKind:etHumanKind
 AdtDate:dtDate
 AdtTime:dtTime
```

```
 7   AdtPhoneNumber:dtPhoneNumber
 8   AdtGPSLocation:dtGPSLocation
 9   AdtComment:dtComment
10  }
11
12  constraints{
13   TheActor = TheSystem.rnactComCompany->any2(true)
14   AetHumanKind = witness
15   AdtDate.year.value = 2017
16   AdtDate.month.value = 11
17   AdtDate.day.value = 26
18   AdtTime.hour.value = 10
19   AdtTime.minute.value = 20
20   AdtTime.second.value = 00
21   AdtPhoneNumber.value = '+3524666445000'
22   AdtGPSLocation.latitude.value = 49.627095
23   AdtGPSLocation.longitude.value = 6.160251
24   AdtComment.value = 'A car crash just happened.'
25  }
26
27  oracle{
28   variables{
29    AdtSMS:dtSMS
30   }
31   constraints{
32    AdtSMS.value = 'Your alert has been registered. We will handle it and keep you informed'
33    TheActor.actComCompany.inactComCompany.ieSmsSend(AdtPhoneNumber,AdtSMS)
34   }
35  }
```

Listing 6.16: Messir (MCL-oriented) specification of the test step *testcase01-ts15oeAlert2*.

6.1.1.16 testcase01-ts16oeSetClock05-actActivator.outactActivator.oeSetClock

The `testcase01-ts16oeSetClock05-actActivator.outactActivator.oeSetClock` has the following properties:

Test Step	
ts16oeSetClock05 cf. actor documentation	
Test Sent Message	
TSM 1	**out:TheActor** **sends to system** **actActivator.outactActivator.oeSetClock (ACurrentClock)**
Variables	
V 1	**TheActor:icrash.environment.actActivator** cf. actor documentation
V 2	**ACurrentClock:lu.uni.lassy.messir.libraries.calendar.dtDateAndTime** cf. actor documentation
Constraints	
C 1	TheActor
C 2	ACurrentClock

The listing 6.17 provides the Messir (MCL-oriented) specification of the test step.

```

variables{
 TheActor:actActivator
 ACurrentClock:dtDateAndTime
}

constraints{
 TheActor=TheSystem.rnactActivator->any2(true)
 ACurrentClock.date.year.value = 2017
 ACurrentClock.date.month.value = 11
 ACurrentClock.date.day.value = 26
 ACurrentClock.time.hour.value = 12
 ACurrentClock.time.minute.value = 45
 ACurrentClock.time.second.value = 00
}

oracle{
 constraints{
  true
 }
}
```

Listing 6.17: Messir (MCL-oriented) specification of the test step *testcase01-ts16oeSetClock05*.

6.1.1.17 testcase01-ts17oeSetCrisisStatus-actCoordinator.outactCoordinator.oeSetCrisisStatus

The `testcase01-ts17oeSetCrisisStatus-actCoordinator.outactCoordinator.oeSetCris` has the following properties:

Test Step	
ts17oeSetCrisisStatus cf. actor documentation	
Test Sent Message	
TSM 1	**out:TheActor** **sends to system** **actCoordinator.outactCoordinator.oeSetCrisisStatus (AdtCrisisID, AetCrisisStatus)**
Variables	
V 1	**TheActor:icrash.environment.actCoordinator** cf. actor documentation
V 2	**AdtCrisisID:icrash.concepts.primarytypes.datatypes.dtCrisisID** cf. actor documentation
V 3	**AetCrisisStatus:icrash.concepts.primarytypes.datatypes.etCrisisStatus** cf. actor documentation
V 4	**AMessage:lu.uni.lassy.messir.libraries.primitives.ptString** cf. actor documentation
Constraints	
C 1	TheActor is the coordinator actor related to a coordinator in the system's state having steve as login value
C 2	AdtCrisisID

continues in next page . . .

. . . Test Step table continuation

C 3	AetCrisisStatus
C 4	AMessage
Oracle Constraints	
OC 1	

The listing 6.18 provides the Messir (MCL-oriented) specification of the test step.

```
1
2  variables{
3   TheActor : actCoordinator
4   AdtCrisisID : dtCrisisID
5   AetCrisisStatus : etCrisisStatus
6  }
7
8  constraints{
9   TheActor=TheSystem.rnactCoordinator
10        ->select(a | a.rnctCoordinator.login.value.eq('steve'))
11        ->any2(true)
12 }
13
14 oracle{
15  variables{
16   AMessage:ptString
17  }
18  constraints{
19   AMessage = 'The crisis status has been updated !'
20   TheActor.inactAuthenticated.ieMessage(AMessage)
21  }
22 }
```

Listing 6.18: Messir (MCL-oriented) specification of the test step *testcase01-ts17oeSetCrisisStatus*.

6.1.1.18 testcase01-ts18oeReportOnCrisis-actCoordinator.outactCoordinator.oeReportOnCrisis

The `testcase01-ts18oeReportOnCrisis-actCoordinator.outactCoordinator.oeReportOnCris` has the following properties:

TEST STEP

ts18oeReportOnCrisis cf. actor documentation	
Test Sent Message	
TSM 1	**out:TheActor** **sends to system** **actCoordinator.outactCoordinator.oeReportOnCrisis (AdtCrisisID, AdtComment)**
Variables	
V 1	**TheActor:icrash.environment.actCoordinator** cf. actor documentation
V 2	**AdtCrisisID:icrash.concepts.primarytypes.datatypes.dtCrisisID**

continues in next page . . .

. . . Test Step table continuation

	cf. actor documentation
V 3	**AdtComment:icrash.concepts.primarytypes.datatypes.dtComment**
	cf. actor documentation
V 4	**AMessage:lu.uni.lassy.messir.libraries.primitives.ptString**
	cf. actor documentation
Constraints	
C 1	TheActor is the coordinator actor related to a coordinator in the system's state having steve as login value
C 2	AdtCrisisID
C 3	AdtComment
C 4	AMessage
Oracle Constraints	
OC 1	

The listing 6.19 provides the Messir (MCL-oriented) specification of the test step.

```

variables{
 TheActor : actCoordinator
 AdtCrisisID : dtCrisisID
 AdtComment : dtComment
}

constraints{
 TheActor=TheSystem.rnactCoordinator
       ->select(a | a.rnctCoordinator.login.value.eq('steve'))
       ->any2(true)
}

oracle{
 variables{
  AMessage:ptString
 }
 constraints{
  AMessage = 'The crisis comment has been updated !'
  TheActor.inactAuthenticated.ieMessage(AMessage)
 }
}
```

Listing 6.19: Messir (MCL-oriented) specification of the test step *testcase01-ts18oeReportOnCrisis*.

6.1.1.19 testcase01-ts19oeCloseCrisis-actCoordinator.outactCoordinator.oeCloseCrisis

The `testcase01-ts19oeCloseCrisis-actCoordinator.outactCoordinator.oeCloseCrisis` has the following properties:

TEST STEP

ts19oeCloseCrisis
cf. actor documentation
Test Sent Message

continues in next page . . .

. . . Test Step table continuation

TSM 1	**out:TheActor** **sends to system** **actCoordinator.outactCoordinator.oeCloseCrisis (AdtCrisisID)**
Variables	
V 1	**TheActor:icrash.environment.actCoordinator** cf. actor documentation
V 2	**AdtCrisisID:icrash.concepts.primarytypes.datatypes.dtCrisisID** cf. actor documentation
V 3	**AMessage:lu.uni.lassy.messir.libraries.primitives.ptString** cf. actor documentation
Constraints	
C 1	TheActor is the coordinator actor related to a coordinator in the system's state having steve as login value
C 2	AdtCrisisID
C 3	AMessage
Oracle Constraints	
OC 1	

The listing 6.20 provides the Messir (MCL-oriented) specification of the test step.

```

variables{
 TheActor : actCoordinator
 AdtCrisisID : dtCrisisID
}

constraints{
 TheActor=TheSystem.rnactCoordinator
       ->select(a | a.rnctCoordinator.login.value.eq('steve'))
       ->any2(true)
}

oracle{
 variables{
  AMessage:ptString
 }
 constraints{
  AMessage = 'The crisis is now closed !'
  TheActor.inactAuthenticated.ieMessage(AMessage)
 }
}
```

Listing 6.20: Messir (MCL-oriented) specification of the test step *testcase01-ts19oeCloseCrisis*.

6.1.2 Test Case Instance - instance01

6.1.3 Test Case Instance - instance01Part01

Figure 6.1 Sequence diagram representing the first part of a simple and complete testcase instance for *iCrash* .

Figure 6.1: tci-testcase01-instance01-Part01 testcase instance sequence diagram

6.1.4 Test Case Instance - instance01Part02

Figure 6.2 Sequence diagram representing the second part of a simple and complete testcase instance for *iCrash* .

Figure 6.2: tci-testcase01-instance01-Part02 testcase instance sequence diagram

Chapter 7

Additional Constraints

7.1 Quality Constraints

Description of all the constraints that concern the required quality criteria according to their ISO definition [3].

7.1.1 Functional suitability

Constraints on the degree to which the product provides functions that meet stated and implied needs when the product is used under specified conditions.

7.1.1.1 Functional completeness

List of requirements on the degree to which the set of functions covers all the specified tasks and user objectives.

1. (to be filled)

7.1.1.2 Functional correctness

List of requirements on the degree to which the set of functions covers all the specified tasks and user objectives.

1. (to be filled)

7.1.1.3 Functional appropriateness

List of requirements on the degree to which the functions facilitate the accomplishment of specified tasks and objectives.

1. (to be filled)

7.1.2 Performance efficiency

Constraints on the performance relative to the amount of resources used under stated conditions

7.1.2.1 Time behaviour

List of requirements on the degree to which the response and processing times and throughput rates of a product or system, when performing its functions, meet requirements.

1. (to be filled)

7.1.2.2 Resource utilization

List of requirements on the degree to which the amounts and types of resources used by a product or system, when performing its functions, meet requirements.

1. (to be filled)

7.1.2.3 Capacity

List of requirements on the degree to which the maximum limits of a product or system parameter meet requirements.

1. (to be filled)

7.1.3 Compatibility

Constraints on the degree to which a product, system or component can exchange information with other products, systems or components, and/or perform its required functions, while sharing the same hardware or software environment.

7.1.3.1 Co-existence

List of requirements on the degree to which a product can perform its required functions efficiently while sharing a common environment and resources with other products, without detrimental impact on any other product.

1. (to be filled)

7.1.3.2 Interoperability

List of requirements on the degree to which two or more systems, products or components can exchange information and use the information that has been exchanged.

1. (to be filled)

7.1.4 Usability

Constraints on the usability degree to which a product or system can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use.

7.1.4.1 Appropriateness recognizability

List of requirements on the degree to which users can recognize whether a product or system is appropriate for their needs.

1. (to be filled)

7.1.4.2 Learnability

List of requirements on the degree to which a product or system can be used by specified users to achieve specified goals of learning to use the product or system with effectiveness, efficiency, freedom from risk and satisfaction in a specified context of use.

1. (to be filled)

7.1.4.3 Operability

List of requirements on the degree to which a product or system has attributes that make it easy to operate and control.

1. (to be filled)

7.1.4.4 User error protection

List of requirements on the degree to which a system protects users against making errors.

1. (to be filled)

7.1.4.5 User interface aesthetics

List of requirements on the degree to which a user interface enables pleasing and satisfying interaction for the user.

1. (to be filled)

7.1.4.6 Accessibility

List of requirements on the degree to which a product or system can be used by people with the widest range of characteristics and capabilities to achieve a specified goal in a specified context of use.

1. (to be filled)

7.1.5 Reliability

Contraints on the degree to which a system, product or component performs specified functions under specified conditions for a specified period of time.

7.1.5.1 Maturity

List of requirements on the degree to which a system, product or component meets needs for reliability under normal operation.

1. (to be filled)

7.1.5.2 Availability

List of requirements on the degree to which a system, product or component is operational and accessible when required for use.

1. (to be filled)

7.1.5.3 Fault tolerance

List of requirements on the degree to which a system, product or component operates as intended despite the presence of hardware or software faults.

1. (to be filled)

7.1.5.4 Recoverability

List of requirements on the degree to which, in the event of an interruption or a failure, a product or system can recover the data directly affected and re-establish the desired state of the system.

1. (to be filled)

7.1.6 Security

Contraints on the degree to which a product or system protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorization.

7.1.6.1 Confidentiality

List of requirements on the degree to which a product or system ensures that data are accessible only to those authorized to have access.

1. (to be filled)

7.1.6.2 Integrity

List of requirements on the degree to which a system, product or component prevents unauthorized access to, or modification of, computer programs or data.

1. (to be filled)

7.1.6.3 Non-repudiation

List of requirements on the degree to which actions or events can be proven to have taken place, so that the events or actions cannot be repudiated later.

1. (to be filled)

7.1.6.4 Accountability

List of requirements on the degree to which the actions of an entity can be traced uniquely to the entity.

1. (to be filled)

7.1.6.5 Authenticity

List of requirements on the degree to which the identity of a subject or resource can be proved to be the one claimed.

1. (to be filled)

7.1.7 Maintainability

Contraints on the degree of effectiveness and efficiency with which a product or system can be modified by the intended maintainers.

7.1.7.1 Modularity

List of requirements on the degree to which a system or computer program is composed of discrete components such that a change to one component has minimal impact on other components.

1. (to be filled)

7.1.7.2 Reusability

List of requirements on the degree to which an asset can be used in more than one system, or in building other assets.

1. (to be filled)

7.1.7.3 Analysability

List of requirements on the degree of effectiveness and efficiency with which it is possible to assess the impact on a product or system of an intended change to one or more of its parts, or to diagnose a product for deficiencies or causes of failures, or to identify parts to be modified.

1. (to be filled)

7.1.7.4 Modifiability

List of requirements on the degree to which a product or system can be effectively and efficiently modified without introducing defects or degrading existing product quality.

1. (to be filled)

7.1.7.5 Testability

List of requirements on the degree of effectiveness and efficiency with which test criteria can be established for a system, product or component and tests can be performed to determine whether those criteria have been met.

1. (to be filled)

7.1.8 Portability

Contraints on the degree of effectiveness and efficiency with which a system, product or component can be transferred from one hardware, software or other operational or usage environment to another.

7.1.8.1 Adaptability

List of requirements on the degree to which a product or system can effectively and efficiently be adapted for different or evolving hardware, software or other operational or usage environments.

1. (to be filled)

7.1.8.2 Installability

List of requirements on the degree of effectiveness and efficiency with which a product or system can be successfully installed and/or uninstalled in a specified environment.

1. (to be filled)

7.1.8.3 Replaceability

List of requirements on the degree to which a product can replace another specified software product for the same purpose in the same environment.

1. (to be filled)

7.2 Other Constraints

Any other unclassified constraints judged as required for the product under development.

Last Modification: @author Petrov Vladislav and Gusev Maxim @date 22.12.16

Appendix A

Undocumented Messir Specification Elements

A.1 Undocumented Use Cases

A.1.1 Undocumented Use Case Views

- uc-oeDeleteHospital

A.2 Undocumented Use Case Instances

A.2.1 Undocumented User-Goal Level Use Case Instances

- usecases.uciugSecurelyUseSystem.uciugSecurelyUseSystem

A.2.2 Undocumented Use Case Instance Views

- uci-uciugSecurelyUseSystem

A.3 Undocumented Actors

- icrash.environment.actCorrupter

A.4 Undocumented Environment Model Views

- em-view9

A.5 Undocumented Primary Types

A.5.1 Undocumented Primary Enumeration Types

- icrash.concepts.primarytypes.datatypes.etCorruptionKind

A.6 Undocumented Concept Model Views

- cm-pt-dt-lv-02-dtGPSLocation
- cm-view8

A.7 Undocumented Operation Specifications

- icrash.concepts.primarytypes.datatypes.etCorruptionKind.is

A.8 Undocumented Test-Case Instance Specifications

- lu.uni.lassy.excalibur.examples.icrash.tests.testcase01.instance01.instance01
- lu.uni.lassy.excalibur.examples.icrash.tests.testcase01.instance01.instance01Part01
- lu.uni.lassy.excalibur.examples.icrash.tests.testcase01.instance01.instance01Part02

Appendix B

Specification project lu.uni.lassy.excalibur.examples.icrash

B.1 Use Cases Model

This section contains the use cases elicited during the requirements elicitation phase. The use cases are textually described as suggested by the Messir method and inspired by the standard Cokburn template [2].

B.1.1 Use Cases

B.1.1.1 subfunction-oeCloseCrisis

the `actCoordinator`'s goal is to declare a crisis as closed.

USE-CASE DESCRIPTION

Name	oeCloseCrisis
Scope	system
Level	subfunction
Primary actor(s)	
1	`actCoordinator[active]`
Goal(s) description	
the `actCoordinator`'s goal is to declare a crisis as closed.	
Protocol condition(s)	
1	the iCrash system has been deployed.
Pre-condition(s)	
1	none
Main post-condition(s)	
1	the crisis is know by the system to be closed.
2	a message `ieMessage(AMessage)` is sent to the `actCoordinator` to inform him that his crisis is now considered as closed.

Figure B.1 shows the use case diagram for the oeCloseCrisis subfunction use case

Figure B.1: oeCloseCrisis subfunction use case

Appendix C

Messir Specification Files Listing

C.1 File ./src-gen/messir-spec/.views.msr

```
//
//DON'T TOUCH THIS FILE !!!
//
package uuid7e0d382938204f3c9036c123484468fb {
 Concept Model {}
}
```

Listing C.1: Messir Spec. file .views.msr.

C.2 File ./src-gen/messir-spec/operations/concepts/secondarytypes-datatypes/dtSMS.msr

```
package icrash.operations.concepts.secondarytypes.datatypes.dtSMS{

import lu.uni.lassy.messir.libraries.primitives
import lu.uni.lassy.messir.libraries.calendar
import lu.uni.lassy.messir.libraries.math

import icrash.concepts.primarytypes.datatypes
import icrash.concepts.primarytypes.classes
import icrash.concepts.secondarytypes.datatypes
import icrash.concepts.secondarytypes.classes

Operation Model {
operation: icrash.concepts.secondarytypes.datatypes.dtSMS.is():ptBoolean{
  postF{
    let TheResult: ptBoolean in
    let MaxLength: ptInteger in
    ( if
     ( MaxLength = 160
      and AdtValue.value.length().leq(MaxLength)
     )
     then (TheResult = true)
     else (TheResult = false)
     endif
     result = TheResult
    )}
prolog{ "src/Operations/Concepts/SecondaryTypesDatatypes/SecondaryTypesDatatypes-dtSMS-is.pl"}
}
}
}
```

Listing C.2: Messir Spec. file dtSMS.msr.

C.3 File ./src-gen/messir-spec/operations/environment/enviroment-actCorrupter-oeCorruptAlert.msr

```
1 package icrash.operations.enviroment.actCorrupter.oeCorruptAlert {
2
3 import lu.uni.lassy.messir.libraries.primitives
4
5 import icrash.concepts.primarytypes.datatypes
6 import icrash.concepts.primarytypes.classes
7 import icrash.environment
8
9  Operation Model {
10   operation: actCorrupter.outactCorrupter.oeCorruptAlert(ActAlert:ctAlert):ptBoolean{
11    preP{
12      let TheSystem: ctState in
13      self.rnActor.rnSystem = TheSystem
14
15     /* PreP01 */
16     and TheSystem.vpStarted = true
17     /* PreP02 */
18     TheSystem.rnctAlert->select(id.eq(AdtAlertID))
19      = ColctAlert
20      and ColctAlert->size().eq(1)
21     }
22
23     postF{
24      let TheSystem: ctState in
25      let CorruptedAlert: ctAlert in
26      self.rnActor.rnSystem = TheSystem
27
28      /* PostF02 */
29       and CorruptedAlert.id=ActAlert.id
30       and CorruptedAlert.status=ActAlert.status
31       and CorruptedAlert.location=ActAlert.location
32       and CorruptedAlert.instant=ActAlert.instant
33       and CorruptedAlert.comment=ActAlert.comment
34       and CorruptedAlert.corruption=ActAlert.etCorruptionKind.corrupted
35       and TheSystem.rnctAlert@post->select(id.eq(ActAlert.id)) = CorruptedAlert
36     }
37
38   }
39  }
40
41 }
```

Listing C.3: Messir Spec. file enviroment-actCorrupter-oeCorruptAlert.msr.

C.4 File ./src-gen/messir-spec/operations/environment/environment-actActivator-oeSetClock.msr

```
1 package icrash.operations.environment.actActivator.oeSetClock {
2
3 import icrash.environment
4
5 import lu.uni.lassy.messir.libraries.primitives
6 import lu.uni.lassy.messir.libraries.calendar
7 import lu.uni.lassy.messir.libraries.math
8
9 import icrash.concepts.primarytypes.datatypes
10 import icrash.concepts.primarytypes.classes
11
12  Operation Model {
13
14 operation: actActivator.outactActivator.oeSetClock(AcurrentClock:dtDateAndTime):ptBoolean
15 {
16 preP{
17  let TheSystem: ctState in
```

```
 let AvpStarted: ptBoolean in

/* PreP01 */
 self.rnActor.rnSystem = TheSystem
 and self.rnActor.rnSystem.vpStarted = AvpStarted
 and AvpStarted = true
 and TheSystem.clock.lt(AcurrentClock)
}
preF{true}

postF{
 let TheSystem: ctState in
 self.rnActor.rnSystem = TheSystem

/* PostF01 */
 and TheSystem@post.clock = AcurrentClock
}
postP{true}

prolog{"src/Operations/Environment/OUT/outactActivator-oeSetClock.pl"}

}
}
}
```

Listing C.4: Messir Spec. file environment-actActivator-oeSetClock.msr.

C.5 File ./src-gen/messir-spec/operations/environment/environment-actActivator-oeSollicitateCrisisHandling.msr

```
package icrash.operations.environment.actActivator.oeSollicitateCrisisHandling {

import lu.uni.lassy.messir.libraries.primitives
import lu.uni.lassy.messir.libraries.math
import lu.uni.lassy.messir.libraries.string
import lu.uni.lassy.messir.libraries.calendar

import icrash.concepts.primarytypes.datatypes
import icrash.concepts.primarytypes.classes
import icrash.environment

Operation Model {

operation: actActivator.outactActivator.oeSollicitateCrisisHandling():ptBoolean
{
preP{
 let TheSystem: ctState in
 let AvpStarted: ptBoolean in
 let ColctCrisisToHandle:
    Bag(ctCrisis) in

 self.rnActor.rnSystem = TheSystem

/* PreP01 */
 and TheSystem.vpStarted

/* PreP02 */
 and TheSystem.rnctCrisis->select(handlingDelayPassed())
   = ColctCrisisToHandle
 and ColctCrisisToHandle->size().geq(1)
}
preF{true}

postF{
 let TheSystem: ctState in
 let AMessageForCrisisHandlers: dtComment in
 let ColctCrisisToAllocateIfPossible:Bag(ctCrisis) in

```

```
39 self.rnActor.rnSystem = TheSystem
40 /* PostF01 */
41 and TheSystem.rnctCrisis->select(maxHandlingDelayPassed())
42   = ColctCrisisToAllocateIfPossible
43 and ColctCrisisToAllocateIfPossible->forAll(isAllocatedIfPossible())
44
45 /* PostF02 */
46 and TheSystem.rnctCrisis->select(handlingDelayPassed())
47 = ColctCrisisToHandle
48
49 and ColctCrisisToHandle->msrColSubtract(ColctCrisisToAllocateIfPossible)
50   = ColctCrisisToRemind
51
52 and if (ColctCrisisToRemind->size().geq(1))
53   then (AMessageForCrisisHandlers.value
54      ='There are alerts pending since more than the defined delay. Please REACT !'
55      and TheSystem.rnactAdministrator.
56        rnInterfaceIN^ieMessage(AMessageForCrisisHandlers)
57        and TheSystem.rnactCoordinator
58          ->forAll(rnInterfaceIN^ieMessage(AMessageForCrisisHandlers))
59      )
60   else true
61   endif
62 }
63 postP{
64 let TheSystem: ctState in
65 let TheClock: dtDateAndTime in
66
67 self.rnActor.rnSystem = TheSystem
68 and TheSystem.clock = TheClock
69 and TheSystem@post.vpLastReminder = TheClock
70 }
71
72 prolog{"src/Operations/Environment/OUT/outactActivator-oeSollicitateCrisisHandling.pl"}
73 }
74 }
75 }
```

Listing C.5: Messir Spec. file environment-actActivator-oeSollicitateCrisisHandling.msr.

C.6 File ./src-gen/messir-spec/operations/environment/environment-actAdministrator-oeAddCoordinator.msr

```
1 package icrash.operations.environment.actAdministrator.oeAddCoordinator {
2
3 import lu.uni.lassy.messir.libraries.primitives
4
5 import icrash.concepts.primarytypes.datatypes
6 import icrash.concepts.primarytypes.classes
7 import icrash.environment
8
9 Operation Model {
10
11 operation: actAdministrator.outactAdministrator.oeAddCoordinator(AdtCoordinatorID:dtCoordinatorID,
      AdtLogin:dtLogin, AdtPassword:dtPassword):ptBoolean
12 {
13 preP{
14 let TheSystem: ctState in
15 let TheActor:actCoordinator in
16
17 self.rnActor.rnSystem = TheSystem
18 and self.rnActor = TheCoordinator
19
20 /* PreP01 */
21 and TheSystem.vpStarted = true
22 /* PreP02 */
23 and TheActor.rnctAuthenticated.vpIsLogged = true
24 }
```

```
preF{
 let TheSystem: ctState in
 let TheActor:actAdministrator in
 let ColctCoordinators:Bag(ctCoordinator) in

 self.rnActor.rnSystem = TheSystem
 and self.rnActor = TheActor
/* PreF01 */
 and TheSystem.rnctCoordinator->select(id.eq(AdtCoordinatorID))
   = ColctCoordinators
 and ColctCoordinators->isEmpty() = true
  }
postF{
 let TheSystem: ctState in
 let TheactCoordinator:actCoordinator in
 let ThectCoordinator:ctCoordinator in
 self.rnActor.rnSystem = TheSystem
 and self.rnActor = TheActor
/* PostF01 */
 TheactCoordinator.init()
/* PostF02 */
 and ThectCoordinator.init(AdtCoordinatorID,AdtLogin,AdtPassword)

/* PostF03 */
 and TheactCoordinator@post.rnctCoordinator = ThectCoordinator

/* PostF04 */
 and ThectCoordinator@post.rnactAuthenticated = TheactCoordinator

/* PostF05 */
 and TheActor.rnInterfaceIN^ieCoordinatorAdded()
}
postP{true}

prolog{"src/Operations/Environment/OUT/outactAdministrator-oeAddCoordinator.pl"}
}
}
}
```

Listing C.6: Messir Spec. file environment-actAdministrator-oeAddCoordinator.msr.

C.7 File ./src-gen/messir-spec/operations/environment/environment-actAdministrator-oeAddHospital.msr

```
package icrash.operations.environment.actAdministrator.oeAddHospital {

import lu.uni.lassy.messir.libraries.primitives

import icrash.concepts.primarytypes.datatypes
import icrash.concepts.primarytypes.classes
import icrash.environment

Operation Model {

operation: actAdministrator.outactAdministrator.oeAddHospital(AdtHospitalID:dtHospitalID, AdtLogin:
    dtLogin, AdtPassword:dtPassword):ptBoolean
{
preP{
 let TheSystem: ctState in
 let TheActor:actAdministrator in

 self.rnActor.rnSystem = TheSystem
 and self.rnActor = TheActor

/* PreP01 */
 and TheSystem.vpStarted = true
/* PreP02 */
 and TheActor.rnctAuthenticated.vpIsLogged = true
```

```
24  }
25 preF{
26  let TheSystem: ctState in
27  let TheActor:actAdministrator in
28
29  let ColctHospitals:Bag(ctHospital) in
30
31  self.rnActor.rnSystem = TheSystem
32  and self.rnActor = TheActor
33 /* PreF01 */
34  and TheSystem.rnctHospital->select(id.eq(AdtHospitalID))
35    = ColctHospitals
36  and ColctHospitals->isEmpty() = true
37   }
38 postF{
39  let TheSystem: ctState in
40  let TheactHospital:actHospital in
41  let ThectHospital:ctHospital in
42  self.rnActor.rnSystem = TheSystem
43  and self.rnActor = TheActor
44 /* PostF01 */
45  TheactHospital.init()
46 /* PostF02 */
47  and ThectHospital.init(AdtHospitalID,AdtLogin,AdtPassword)
48
49 /* PostF03 */
50  and TheactHospital@post.rnctHospital = ThectHospital
51
52 /* PostF04 */
53  and ThectHospital@post.rnactAuthenticated = TheactHospital
54
55 /* PostF05 */
56  and TheActor.rnInterfaceIN^ieHospitalAdded()
57 }
58 postP{true}
59 }
60 }
61 }
```

Listing C.7: Messir Spec. file environment-actAdministrator-oeAddHospital.msr.

C.8 File ./src-gen/messir-spec/operations/environment/environment-actAdministrator-oeDeleteCoordinator.msr

```
1 package icrash.operations.environment.actAdministrator.oeDeleteCoordinator {
2
3 import lu.uni.lassy.messir.libraries.primitives
4 import lu.uni.lassy.messir.libraries.math
5 import lu.uni.lassy.messir.libraries.calendar
6
7 import icrash.environment
8
9 import icrash.concepts.primarytypes.datatypes
10 import icrash.concepts.primarytypes.classes
11
12 Operation Model {
13
14 operation: actAdministrator.outactAdministrator.oeDeleteCoordinator(AdtCoordinatorID:dtCoordinatorID
      ):ptBoolean
15 {
16 preP{
17  let TheSystem: ctState in
18  let TheActor:actAdministrator in
19
20  self.rnActor.rnSystem = TheSystem
21  and self.rnActor = TheActor
22
23 /* PreP01 */
```

```
 and TheSystem.vpStarted = true
/* PreP02 */
 and TheActor.rnctAuthenticated.vpIsLogged = true
}
preF{
 let TheSystem: ctState in
 let TheActor:actAdministrator in

 self.rnActor.rnSystem = TheSystem
 and self.rnActor = TheActor
/* PreF01 */
 TheSystem.rnctCoordinator->select(id.eq(AdtCoordinatorID))
 = ColctCoordinators
 and ColctCoordinators->size().eq(1)
}
postF{
 let TheSystem: ctState in
 let TheActor:actAdministrator in
 let ThectCoordinator:ctCoordinator in
 self.rnActor.rnSystem = TheSystem
 and self.rnActor = TheActor
/* PostF01 */
 TheSystem.rnctCoordinator->select(id.eq(AdtCoordinatorID))
 = ThectCoordinator
 and ThectCoordinator.rnactCoordinator->forAll(msrIsKilled)
 and ThectCoordinator.msrIsKilled

 /* PostF02 */
 and TheActor.rnInterfaceIN^ieCoordinatorDeleted()

 /* Post Protocol:*/
/* PostP01 */
 and true
 }
 postP{true}

 prolog{"src/Operations/Environment/OUT/outactAdministrator-oeDeleteCoordinator.pl"}
 }
     }
    }
```

Listing C.8: Messir Spec. file environment-actAdministrator-oeDeleteCoordinator.msr.

C.9 File ./src-gen/messir-spec/operations/environment/environment-actAdministrator-oeDeleteHospital.msr

```
package icrash.operations.environment.actAdministrator.oeDeleteHospital {

import lu.uni.lassy.messir.libraries.primitives
import lu.uni.lassy.messir.libraries.math
import lu.uni.lassy.messir.libraries.calendar

import icrash.environment

import icrash.concepts.primarytypes.datatypes
import icrash.concepts.primarytypes.classes

Operation Model {

operation: actAdministrator.outactAdministrator.oeDeleteHospital(AdtHospitalID:dtHospitalID):
      ptBoolean
{
preP{
 let TheSystem: ctState in
 let TheActor:actAdministrator in

 self.rnActor.rnSystem = TheSystem
 and self.rnActor = TheActor
```

```
22
23 /* PreP01 */
24  and TheSystem.vpStarted = true
25 /* PreP02 */
26  and TheActor.rnctAuthenticated.vpIsLogged = true
27 }
28 preF{
29  let TheSystem: ctState in
30  let TheActor:actAdministrator in
31
32  self.rnActor.rnSystem = TheSystem
33  and self.rnActor = TheActor
34 /* PreF01 */
35  TheSystem.rnctHospital->select(id.eq(AdtHospitalID))
36  = ColctHospitals
37  and ColctHospitals->size().eq(1)
38 }
39 postF{
40  let TheSystem: ctState in
41  let TheActor:actAdministrator in
42  let ThectHospital:ctHospital in
43  self.rnActor.rnSystem = TheSystem
44  and self.rnActor = TheActor
45 /* PostF01 */
46  TheSystem.rnctHospital->select(id.eq(AdtHospitalID))
47  = ThectHospital
48  and ThectHospital.rnactHospital->forAll(msrIsKilled)
49  and ThectHospital.msrIsKilled
50
51  /* PostF02 */
52  and TheActor.rnInterfaceIN^ieHospitalDeleted()
53
54  /* Post Protocol:*/
55 /* PostP01 */
56  and true
57  }
58  postP{true}
59
60  prolog{"src/Operations/Environment/OUT/outactAdministrator-oeDeleteHospital.pl"}
61  }
62      }
63     }
```

Listing C.9: Messir Spec. file environment-actAdministrator-oeDeleteHospital.msr.

C.10 File ./src-gen/messir-spec/operations/environment/environment-actAuthenticated.msr

```
1 package icrash.operations.environment.actAuthenticated{
2
3 import lu.uni.lassy.messir.libraries.primitives
4
5 import icrash.concepts.primarytypes.datatypes
6 import icrash.concepts.primarytypes.classes
7 import icrash.concepts.secondarytypes.datatypes
8 import icrash.concepts.secondarytypes.classes
9 import icrash.environment
10
11 Operation Model {
12
13 operation: actAuthenticated.outactAuthenticated.oeLogin(AdtLogin:dtLogin, AdtPassword:dtPassword):
      ptBoolean
14 {
15 preP{
16  let TheSystem: ctState in
17  let TheActor:actAuthenticated in
18  self.rnActor.rnSystem = TheSystem
19  and self.rnActor = TheActor
```

```
20
21 /* PreP01 */
22  and TheSystem.vpStarted = true
23 /* PreP02 */
24  and TheActor.rnctAuthenticated.vpIsLogged = false
25 }
26 preF{
27 /* PreF01 */
28 true
29 }
30 postF{
31  let TheSystem: ctState in
32  let TheactAuthenticated:actAuthenticated in
33
34  let AptStringMessageForTheactAuthenticated: ptString in
35  let AptStringMessageForTheactAdministrator:ptString in
36
37  self.rnActor.rnSystem = TheSystem
38  and self.rnActor = TheactAuthenticated
39
40  and /* PostF01 */
41    if (TheactAuthenticated.rnctAuthenticated.pwd
42      = AdtPassword
43      and TheactAuthenticated.rnctAuthenticated.login
44        = AdtLogin
45      )
46    then (AptStringMessageForTheactAuthenticated.eq('You are logged ! Welcome ...')
47       and TheactAuthenticated.rnInterfaceIN^ieMessage(AptStringMessageForTheactAuthenticated)
48       )
49    else (AptStringMessageForTheactAuthenticated
50       .eq('Wrong identification information ! Please try again ...')
51       and TheactAuthenticated.rnInterfaceIN^ieMessage(AptStringMessageForTheactAuthenticated)
52       and AptStringMessageForTheactAdministrator.eq('Intrusion tentative !')
53       and TheSystem.rnactAdministrator
54         .rnInterfaceIN^ieMessage(AptStringMessageForTheactAdministrator)
55       )
56    endif
57 }
58 postP{
59  let TheSystem: ctState in
60  let TheactAuthenticated:actAuthenticated in
61
62  self.rnActor.rnSystem = TheSystem
63  and self.rnActor = TheactAuthenticated
64 /* PostP01 */
65  if (TheactAuthenticated.rnctAuthenticated.pwd = AdtPassword
66    and TheactAuthenticated.rnctAuthenticated.login = AdtLogin
67    )
68  then (TheactAuthenticated.rnctAuthenticated@post.vpIsLogged = true)
69  else true
70  endif
71 }
72 prolog{"src/Operations/Environment/OUT/outactAuthenticated-oeLogin.pl"}
73 }
74 /*-----------------------------------------------------------------------------*/
75
76 operation: actAuthenticated.outactAuthenticated.oeLogout():ptBoolean{
77
78 preP{
79  let TheSystem: ctState in
80  let TheActor:actAdministrator in
81  self.rnActor.rnSystem = TheSystem
82  and self.rnActor = TheActor
83
84 /* PreP01 */
85  and TheSystem.vpStarted = true
86 /* PreP02 */
87  and TheActor.rnctAuthenticated.vpIsLogged = true
88 }
89 preF{
```

```
90 /* PreF01 */
91 true
92 }
93 postF{
94  let TheSystem: ctState in
95  let TheactAuthenticated:actAuthenticated in
96  let AptStringMessageForTheactAuthenticated: ptString in
97
98  self.rnActor.rnSystem = TheSystem
99  and self.rnActor = TheactAuthenticated
100
101  /* PostF01 */
102  AptStringMessageForTheactAuthenticated.eq('You are logged out ! Good Bye ...')
103  and TheactAuthenticated.rnInterfaceIN^ieMessage(AptStringMessageForTheactAuthenticated)
104 }
105 postP{
106  let TheSystem: ctState in
107  let TheactAuthenticated:actAuthenticated in
108
109  self.rnActor.rnSystem = TheSystem
110  and self.rnActor = TheactAuthenticated.asSet
111 /* PostP01 */
112  TheactAuthenticated.rnctAuthenticated@post.vpIsLogged = false
113 }
114 prolog{"src/Operations/Environment/OUT/outactAuthenticated-oeLogout.pl"}
115 }
116 }
117 }
```

Listing C.10: Messir Spec. file environment-actAuthenticated.msr.

C.11 File ./src-gen/messir-spec/operations/environment/environment-actComCompany.msr

```
1 // Do not add/remove lines because code is inserted in slides
2
3 package icrash.operations.environment.actComCompany{
4
5 import lu.uni.lassy.messir.libraries.primitives
6 import lu.uni.lassy.messir.libraries.calendar
7 import lu.uni.lassy.messir.libraries.math
8
9 import icrash.concepts.primarytypes.datatypes
10 import icrash.concepts.primarytypes.classes
11 import icrash.concepts.secondarytypes.datatypes
12
13 import icrash.environment
14
15 Operation Model {
16
17 operation: actComCompany.outactComCompany.oeAlert(
18  AetKind:etHumanKind,
19  AdtMyDate:dtDate,
20  AdtTime:dtTime,
21  AdtPhoneNumber:dtPhoneNumber,
22  AdtGPSLocation:dtGPSLocation,
23  AdtComment:dtComment
24 ):ptBoolean{
25
26 preP{
27  let TheSystem: ctState in
28  self.rnActor.rnSystem = TheSystem
29
30 /* PreP01 */
31  and TheSystem.vpStarted = true
32 }
33 preF{
34  let TheSystem: ctState in
```

```
35  self.rnActor.rnSystem = TheSystem
36
37 /* PreF01 */
38  and (TheSystem.clock.date.gt(AdtDate)
39     or (TheSystem.clock.date.eq(AdtDate)
40       and TheSystem.clock.time.gt(AdtTime)
41      )
42    )
43 }
44 postF{
45  let TheSystem: ctState in
46
47  let ActHuman:ctHuman in
48  let TheactComCompany:actComCompany in
49  let ActAlert:ctAlert in
50  let AAlertInstant:dtDateAndTime in
51  let AetAlertStatus:etAlertStatus in
52  let ActAlertNearBy:ctAlert in
53  let ActCrisis:ctCrisis in
54  let AdtCrisisID:dtCrisisID in
55  let AetCrisisType:etCrisisType in
56  let AetCrisisStatus:etCrisisStatus in
57  let ACrisisInstant:dtDateAndTime in
58  let ACrisisdtComment:dtComment in
59  let AptStringMessage:ptString in
60  let AdtSMS:dtSMS in
61  let AdtAlertID:dtAlertID in
62
63  self.rnActor.rnSystem = TheSystem
64  and self.rnActor = TheactComCompany
65 /* PostF01 */
66  TheSystem.nextValueForAlertID=PrenextValueForAlertID
67  and PrenextValueForAlertID.add(1) = PostnextValueForAlertID // Inga
68  and TheSystem@post.nextValueForAlertID = PostnextValueForAlertID
69
70  /* PostF02 */
71 and AAlertInstant.date=AdtDate
72 and AAlertInstant.time=AdtTime
73
74 and AetAlertStatus=pending
75
76 and TheSystem.nextValueForAlertID.todtString().eq(AdtAlertID)
77
78 and ActAlert.init(AdtAlertID,
79          AetAlertStatus,
80          AdtGPSLocation,
81          AAlertInstant,
82          AdtComment)
83
84  /* PostF03 */
85
86 and TheSystem.rnctAlert -> select(location.isNearTo(AdtGPSLocation)) = ColctAlertsNearBy
87 //(TheSystem.rnCtCrisis -> select(c:ctCrisis | c.id.value = 1)).rnctAlerts
88
89 and if (ColctAlertsNearBy->size()=0)
90   then (TheSystem.nextValueForCrisisID = PrenextValueForCrisisID
91      and PrenextValueForCrisisID.add(1) = PostnextValueForCrisisID
92      and TheSystem@post.nextValueForCrisisID = PostnextValueForCrisisID
93      and TheSystem.nextValueForCrisisID.todtString().eq(AdtCrisisID)
94      and AdtCrisisType = small
95      and AetCrisisStatus = pending
96      and ACrisisInstant= AAlertInstant
97      and ACrisisdtComment = 'no reporting yet defined'
98      and ActCrisis.init( AdtCrisisID,
99                AdtCrisisType,
100               AetCrisisStatus,
101               AdtGPSLocation,
102               ACrisisInstant,
103               ACrisisdtComment)
104     )
```

```
105  else (ColctAlertsNearBy.rnTheCrisis->msrAny(true) = ActCrisis)
106  endif
107
108  /* PostF04 */
109 and ActAlert@post.rnTheCrisis = ActCrisis
110
111 /* PostF05 */
112 and TheSystem.rnctHuman->select(id.eq(AdtPhoneNumber)) = HumanCol1
113
114 and HumanCol1->select(kind.etEq(AetHumanKind)) = HumanCol2
115 and if (HumanCol2->msrIsEmpty)
116   then (ActHuman.init(AdtPhoneNumber,AetHumanKind)
117       and ActHuman@post.rnactComCompany = TheactComCompany
118       )
119   else (HumanCol2->any(true) = ActHuman)
120   endif
121
122  and ActHuman.rnSignaled->msrIncluding(ActAlert) = ColAlerts
123
124  and ActHuman@post.rnSignaled = ColAlerts
125
126 /* PostF06 */
127 AdtSMS.value = 'Your alert has been registered. We will handle it and keep you informed'
128 and TheactComCompany.rnInterfaceIN^ieSmsSend(AdtPhoneNumber,AdtSMS)
129 }
130  /* Post Protocol:*/
131 /* PostP01 */
132 postP{true}
133
134 prolog{"src/Operations/Environment/OUT/outactComCompany-oeAlert.pl"}
135 }
136 }
137 }
```

Listing C.11: Messir Spec. file environment-actComCompany.msr.

C.12 File ./src-gen/messir-spec/operations/environment/environment-actCoordinator-oeAddInjury.msr

```
1 /*
2 * @author subli
3 * @date Sun Dec 11 13:19:30 MSK 2016
4 */
5
6 package icrash.operations.environment.actCoordinator.oeAddInjury {
7
8 import lu.uni.lassy.messir.libraries.primitives
9
10  Operation Model {
11
12  }
13
14 }
```

Listing C.12: Messir Spec. file environment-actCoordinator-oeAddInjury.msr.

C.13 File ./src-gen/messir-spec/operations/environment/environment-actCoordinator-oeAddVictim.msr

```
1 package icrash.operations.environment.actCoordinator.oeAddVictim {
2
3 import lu.uni.lassy.messir.libraries.primitives
4
5 import icrash.concepts.primarytypes.datatypes
6 import icrash.concepts.primarytypes.classes
7 import icrash.environment
```

```

Operation Model {

operation: actCoordinator.outactCoordinator.oeAddVictim(AdtCrisisID:dtCrisisID):ptBoolean
{
preP{
 let TheSystem: ctState in
 let TheActor:actCoordinator in
 let AdtVictimID:dtVictimID in

 self.rnActor.rnSystem = TheSystem
 and self.rnActor = TheCoordinator

/* PreP01 */
 and TheSystem.vpStarted = true
/* PreP02 */
 and TheActor.rnctAuthenticated.vpIsLogged = true
  }
postF{
 let TheSystem: ctState in
 let TheActor:actCoordinator in
 let AdtVictimID:dtVictimID in

 self.rnActor.rnSystem = TheSystem
 and self.rnActor = TheActor

/* PostF01 */
 TheSystem.nextValueForVictimID=PrenextValueForVictimID
 and PrenextValueForVictimID.add(1) = PostnextValueForVictimID
 and TheSystem@post.nextValueForVictimID = PostnextValueForVictimID

/* PostF02 */
and TheSystem.nextValueForVictimID.todtString().eq(AdtVictimID)

and ActVictim.init(AdtVictimID)

and
TheSystem.rnCtCrisis@post.includes(ActVictim)

/* PostF03 */
and TheActor.rnInterfaceIN^ieVictimAdded()
}
postP{true}

}
}
}
```

Listing C.13: Messir Spec. file environment-actCoordinator-oeAddVictim.msr.

C.14 File ./src-gen/messir-spec/operations/environment/environment-actCoordinator-oeCloseCrisis.msr

```
package icrash.operations.environment.actCoordinator.oeCloseCrisis {

import lu.uni.lassy.messir.libraries.primitives
import lu.uni.lassy.messir.libraries.math
import lu.uni.lassy.messir.libraries.string
import lu.uni.lassy.messir.libraries.calendar
import icrash.concepts.primarytypes.datatypes
import icrash.environment

Operation Model {

operation: actCoordinator.outactCoordinator.oeCloseCrisis(AdtCrisisID:dtCrisisID):ptBoolean{
prolog{"src/Operations/Environment/OUT/outactCoordinator-oeCloseCrisis.pl"}
}
}
```

```
16 }
```

Listing C.14: Messir Spec. file environment-actCoordinator-oeCloseCrisis.msr.

C.15 File ./src-gen/messir-spec/operations/environment/environment-actCoordinator-oeDeleteInjury.msr

```
1 package icrash.operations.environment.actCoordinator.oeDeleteInjury {
2
3 import lu.uni.lassy.messir.libraries.primitives
4
5 import icrash.concepts.primarytypes.datatypes
6 import icrash.concepts.primarytypes.classes
7 import icrash.environment
8
9 Operation Model {
10
11 operation: actCoordinator.outactCoordinator.oeDeleteInjury(AdtInjuryID:dtInjuryID):ptBoolean
12 {
13 preP{
14  let TheSystem: ctState in
15  let TheActor:actCoordinator in
16  let AdtInjuryID:dtInjuryID in
17
18  self.rnActor.rnSystem = TheSystem
19  and self.rnActor = TheCoordinator
20
21 /* PreP01 */
22  and TheSystem.vpStarted = true
23 /* PreP02 */
24  and TheActor.rnctAuthenticated.vpIsLogged = true
25  }
26 preF{
27 /* PreF01 */
28  TheSystem.rnctInjury->select(id.eq(AdtInjuryID))
29  = ColctInjury
30  and ColctInjury->size().eq(1)
31 }
32
33 postF{
34  let TheSystem: ctState in
35  let TheActor:actCoordinator in
36
37  self.rnActor.rnSystem = TheSystem
38  and self.rnActor = TheActor
39
40 /* PostF01 */
41  TheSystem.rnctInjury->select(id.eq(AdtInjuryID))
42  = ColctInjury
43  and ColctInjury->size().eq(0)
44 /* PostF02 */
45  and TheActor.rnInterfaceIN^ieInjuryDeleted()
46 }
47 postP{true}
48
49 }
50 }
51 }
```

Listing C.15: Messir Spec. file environment-actCoordinator-oeDeleteInjury.msr.

C.16 File ./src-gen/messir-spec/operations/environment/environment-actCoordinator-oeDeleteVictim.msr

```
1 package icrash.operations.environment.actCoordinator.oeDeleteVictim {
2
```

```
3 import lu.uni.lassy.messir.libraries.primitives
4
5 import icrash.concepts.primarytypes.datatypes
6 import icrash.concepts.primarytypes.classes
7 import icrash.environment
8
9 Operation Model {
10
11 operation: actCoordinator.outactCoordinator.oeDeleteVictim(AdtVictimID:dtVictimID):ptBoolean
12 {
13 preP{
14  let TheSystem: ctState in
15  let TheActor:actCoordinator in
16  let AdtVictimID:dtVictimID in
17
18  self.rnActor.rnSystem = TheSystem
19  and self.rnActor = TheCoordinator
20
21 /* PreP01 */
22  and TheSystem.vpStarted = true
23 /* PreP02 */
24  and TheActor.rnctAuthenticated.vpIsLogged = true
25 /* PreF01 */
26  TheSystem.rnctVictim->select(id.eq(AdtVictimID))
27  = ColctVictims
28  and ColctVictims->size().eq(1)
29 }
30
31 postF{
32  let TheSystem: ctState in
33  let TheActor:actCoordinator in
34
35  self.rnActor.rnSystem = TheSystem
36  and self.rnActor = TheActor
37
38 /* PostF01 */
39  TheSystem.rnctVictim@post->select(id.eq(AdtVictimID))
40  = ColctVictims
41  and ColctVictims->size().eq(0)
42 /* PostF02 */
43  and TheActor.rnInterfaceIN^ieVictimDeleted()
44 }
45 postP{true}
46
47 }
48 }
49 }
```

Listing C.16: Messir Spec. file environment-actCoordinator-oeDeleteVictim.msr.

C.17 File ./src-gen/messir-spec/operations/environment/environment-actCoordinator-oeGetAlertsSet.msr

```
1 package icrash.operations.environment.actCoordinator.oeGetAlertsSet {
2
3 import lu.uni.lassy.messir.libraries.primitives
4 import lu.uni.lassy.messir.libraries.math
5 import lu.uni.lassy.messir.libraries.string
6 import lu.uni.lassy.messir.libraries.calendar
7
8 import icrash.concepts.primarytypes.datatypes
9 import icrash.environment
10
11 Operation Model {
12
13 operation: actCoordinator.outactCoordinator.oeGetAlertsSet(AetAlertStatus:etAlertStatus):ptBoolean{
14 prolog{"src/Operations/Environment/OUT/outactCoordinator-oeGetAlertsSet.pl"}
15 }
```

```
16 }
17 }
```

Listing C.17: Messir Spec. file environment-actCoordinator-oeGetAlertsSet.msr.

C.18 File ./src-gen/messir-spec/operations/environment/environment-actCoordinator-oeGetCrisisSet.msr

```
1 package icrash.operations.environment.actCoordinator.oeGetCrisisSet {
2
3 import lu.uni.lassy.messir.libraries.primitives
4 import lu.uni.lassy.messir.libraries.math
5 import lu.uni.lassy.messir.libraries.string
6 import lu.uni.lassy.messir.libraries.calendar
7 import icrash.concepts.primarytypes.datatypes
8 import icrash.environment
9
10 Operation Model {
11
12 operation: actCoordinator.outactCoordinator.oeGetCrisisSet(AetCrisisStatus:etCrisisStatus):ptBoolean
      {
13
14 prolog{"src/Operations/Environment/OUT/outactCoordinator-oeGetCrisisSet.pl"}
15 }
16 }
17 }
```

Listing C.18: Messir Spec. file environment-actCoordinator-oeGetCrisisSet.msr.

C.19 File ./src-gen/messir-spec/operations/environment/environment-actCoordinator-oeGetInjurySet.msr

```
1 package icrash.operations.environment.actCoordinator.oeGetInjurySet {
2
3 import lu.uni.lassy.messir.libraries.primitives
4
5 import icrash.concepts.primarytypes.datatypes
6 import icrash.concepts.primarytypes.classes
7 import icrash.environment
8
9 Operation Model {
10 operation: actCoordinator.outactCoordinator.oeGetInjurySet(AdtVictimID:dtVictimID):ptBoolean
11 {
12
13 preP{
14  let TheSystem: ctState in
15  let TheActor:actCoordinator in
16
17  self.rnActor.rnSystem = TheSystem
18  and self.rnActor = TheCoordinator
19
20 /* PreP01 */
21  and TheSystem.vpStarted = true
22 /* PreP02 */
23  and TheActor.rnctAuthenticated.vpIsLogged = true
24   }
25 postF{
26  let TheSystem: ctState in
27  let TheActor:actCoordinator in
28
29  self.rnActor.rnSystem = TheSystem
30  and self.rnActor = TheActor
31
32 /* PostF01 */
33 TheSystem.rnCtInjury -> select(ctVictim.id = dtVictimID) = colVictimsInjury
34 and colVictimsInjury->forAll(T:ctInjury | T.isSentToACoordinator(TheActor).eq(true))
```

```
35
36 }
37 postP{true}
38
39 }
40 }
41 }
```

Listing C.19: Messir Spec. file environment-actCoordinator-oeGetInjurySet.msr.

C.20 File ./src-gen/messir-spec/operations/environment/environment-actCoordinator-oeGetVictimSet.msr

```
1 package icrash.operations.environment.actCoordinator.oeGetVictimSet {
2
3 import lu.uni.lassy.messir.libraries.primitives
4
5 import icrash.concepts.primarytypes.datatypes
6 import icrash.concepts.primarytypes.classes
7 import icrash.environment
8
9 Operation Model {
10 operation: actCoordinator.outactCoordinator.oeGetVictimSet(AdtCrisisID:dtCrisisID):ptBoolean
11 {
12
13 preP{
14  let TheSystem: ctState in
15  let TheActor:actCoordinator in
16
17  self.rnActor.rnSystem = TheSystem
18  and self.rnActor = TheCoordinator
19
20 /* PreP01 */
21  and TheSystem.vpStarted = true
22 /* PreP02 */
23  and TheActor.rnctAuthenticated.vpIsLogged = true
24   }
25 postF{
26  let TheSystem: ctState in
27  let TheActor:actCoordinator in
28
29  self.rnActor.rnSystem = TheSystem
30  and self.rnActor = TheActor
31
32 /* PostF01 */
33 TheSystem.rnCtVictim -> select(ctCrisis.id = AdtCrisisID) = colCrisisVictims
34 and colCrisisVictims->forAll(T:ctVictim | T.isSentToACoordinator(TheActor).eq(true))
35
36 }
37 postP{true}
38
39 }
40 }
41 }
```

Listing C.20: Messir Spec. file environment-actCoordinator-oeGetVictimSet.msr.

C.21 File ./src-gen/messir-spec/operations/environment/environment-actCoordinator-oeInvalidateAlert.msr

```
1 package icrash.operations.environment.actCoordinator.oeInvalidateAlert {
2
3 import lu.uni.lassy.messir.libraries.primitives
4 import lu.uni.lassy.messir.libraries.math
5 import lu.uni.lassy.messir.libraries.string
6 import lu.uni.lassy.messir.libraries.calendar
```

```
7 import icrash.concepts.primarytypes.datatypes
8 import icrash.environment
9
10 Operation Model {
11
12 operation: actCoordinator.outactCoordinator.oeInvalidateAlert(AdtAlertID:dtAlertID):ptBoolean{
13 prolog{"src/Operations/Environment/OUT/outactCoordinator-oeInvalidateAlert.pl"}
14 }
15 }
16 }
```

Listing C.21: Messir Spec. file environment-actCoordinator-oeInvalidateAlert.msr.

C.22 File ./src-gen/messir-spec/operations/environment/environment-actCoordinator-oeReportOnCrisis.msr

```
1 package icrash.operations.environment.actCoordinator.oeReportOnCrisis {
2
3 import lu.uni.lassy.messir.libraries.primitives
4 import lu.uni.lassy.messir.libraries.math
5 import lu.uni.lassy.messir.libraries.string
6 import lu.uni.lassy.messir.libraries.calendar
7 import icrash.concepts.primarytypes.datatypes
8 import icrash.environment
9
10 Operation Model {
11
12 operation: actCoordinator.outactCoordinator.oeReportOnCrisis(AdtCrisisID:dtCrisisID, AdtComment:
      dtComment):ptBoolean{
13 prolog{"src/Operations/Environment/OUT/outactCoordinator-oeReportOnCrisis.pl"}
14 }
15
16 }
17 }
```

Listing C.22: Messir Spec. file environment-actCoordinator-oeReportOnCrisis.msr.

C.23 File ./src-gen/messir-spec/operations/environment/environment-actCoordinator-oeSetCrisisHandler.msr

```
1 package icrash.operations.environment.actCoordinator.oeSetCrisisHandler {
2
3 import lu.uni.lassy.messir.libraries.primitives
4 import lu.uni.lassy.messir.libraries.math
5 import lu.uni.lassy.messir.libraries.string
6 import lu.uni.lassy.messir.libraries.calendar
7
8 import icrash.concepts.primarytypes.datatypes
9 import icrash.concepts.primarytypes.classes
10 import icrash.concepts.secondarytypes.datatypes
11 import icrash.environment
12
13 Operation Model {
14
15 operation: actCoordinator.outactCoordinator.oeSetCrisisHandler(AdtCrisisID:dtCrisisID):ptBoolean{
16 prolog{"src/Operations/Environment/OUT/outactCoordinator-oeSetCrisisHandler.pl"}
17 }
18
19 }
20 }
```

Listing C.23: Messir Spec. file environment-actCoordinator-oeSetCrisisHandler.msr.

C.24 File ./src-gen/messir-spec/operations/environment/environment-actCoordinator-oeSetCrisisStatus.msr

```
package icrash.operations.environment.actCoordinator.oeSetCrisisStatus {

import lu.uni.lassy.messir.libraries.primitives
import lu.uni.lassy.messir.libraries.math
import lu.uni.lassy.messir.libraries.string
import lu.uni.lassy.messir.libraries.calendar
import icrash.concepts.primarytypes.datatypes
import icrash.environment

Operation Model {

operation: actCoordinator.outactCoordinator.oeSetCrisisStatus(AdtCrisisID:dtCrisisID,
    AetCrisisStatus:etCrisisStatus):ptBoolean{
prolog{"src/Operations/Environment/OUT/outactCoordinator-oeSetCrisisStatus.pl"}
}

}
}
```

Listing C.24: Messir Spec. file environment-actCoordinator-oeSetCrisisStatus.msr.

C.25 File ./src-gen/messir-spec/operations/environment/environment-actCoordinator-oeSetCrisisType.msr

```
package icrash.operations.environment.actCoordinator.oeSetCrisisType {

import lu.uni.lassy.messir.libraries.primitives
import lu.uni.lassy.messir.libraries.math
import lu.uni.lassy.messir.libraries.string
import lu.uni.lassy.messir.libraries.calendar
import icrash.concepts.primarytypes.datatypes
import icrash.environment

Operation Model {

operation: actCoordinator.outactCoordinator.oeSetCrisisType(AdtCrisisID:dtCrisisID, AetCrisisType:
    etCrisisType):ptBoolean{
prolog{"src/Operations/Environment/OUT/outactCoordinator-oeSetCrisisType.pl"}
}

}
}
```

Listing C.25: Messir Spec. file environment-actCoordinator-oeSetCrisisType.msr.

C.26 File ./src-gen/messir-spec/operations/environment/environment-actCoordinator-oeValidateAlert.msr

```
package icrash.operations.environment.actCoordinator.oeValidateAlert {

import lu.uni.lassy.messir.libraries.primitives

import icrash.concepts.primarytypes.datatypes
import icrash.concepts.primarytypes.classes
import icrash.environment

Operation Model {

operation: actCoordinator.outactCoordinator.oeValidateAlert(AdtAlertID:dtAlertID):ptBoolean{
preP{
 let TheSystem: ctState in
 let TheActor:actCoordinator in

 self.rnActor.rnSystem = TheSystem
 and self.rnActor = TheCoordinator

```

```
19 /* PreP01 */
20  and TheSystem.vpStarted = true
21 /* PreP02 */
22  and TheActor.rnctAuthenticated.vpIsLogged = true
23   }
24
25 preF{
26 /* PreF01 */
27  TheSystem.rnctAlert->select(id.eq(AdtAlertID))
28  = ColctAlert
29  and ColctAlert->size().eq(1)
30 }
31
32 postF{
33  let TheSystem: ctState in
34  let TheActor:actAuthenticated in
35
36  self.rnActor.rnSystem = TheSystem
37  and self.rnActor = TheActor
38
39  and TheSystem.rnctAlert->select(id.eq(AdtAlertID))
40  = ColctAlert
41
42  /* PostF01 */
43  and if (ColctAlert->forAll(T:ctAlert | T.corruption=regular or T.corruption=restored ))
44  then(
45  TheActor.rnInterfaceIN^ieMessage('The Alert is now declared as valid !')
46  true
47  )
48  else(
49  TheActor.rnInterfaceIN^ieMessage('Can\'t validate corrupted alert !')
50  )
51  endif
52
53 }
54 postP{
55  let TheSystem: ctState in
56
57  self.rnActor.rnSystem = TheSystem
58
59  TheSystem.rnctAlert->select(id.eq(AdtAlertID))
60  = ColctAlert
61
62  /* PostP01 */
63  and if (ColctAlert->forAll(T:ctAlert | T.corruption=regular or T.corruption=restored ))
64  then(
65  ColctAlert->forAll(T:ctAlert | T.status@post=valid)
66  )
67  else true
68  endif
69
70 }
71
72 prolog{"src/Operations/Environment/OUT/outactCoordinator-oeValidateAlert.pl"}
73 }
74
75 }
76 }
```

Listing C.26: Messir Spec. file environment-actCoordinator-oeValidateAlert.msr.

C.27 File ./src-gen/messir-spec/operations/environment/environment-actHospital-oeGetHospitalCrisisSet.msr

```
1 package icrash.operations.environment.actHospital.oeGetHospitalCrisisSet {
2
3 import lu.uni.lassy.messir.libraries.primitives
4
```

```
import icrash.concepts.primarytypes.datatypes
import icrash.concepts.primarytypes.classes
import icrash.environment

Operation Model {
operation: actHospital.outactHospital.oeGetHospitalCrisisSet(AetCrisisStatus:etCrisisStatus):
    ptBoolean
{

preP{
 let TheSystem: ctState in
 let TheActor:actHospital in

 self.rnActor.rnSystem = TheSystem
 and self.rnActor = actHospital

/* PreP01 */
 and TheSystem.vpStarted = true
/* PreP02 */
 and TheActor.rnctAuthenticated.vpIsLogged = true
  }
postF{
 let TheSystem: ctState in
 let TheActor:actHospital in
 let ThectHospital:ctHospital in

 self.rnActor.rnSystem = TheSystem
 and self.rnActor = TheActor
 and TheActor.rnctHospital = ThectHospital

/* PostF01 */
 TheSystem.rnCtCrisis -> select(ctCrisis.status = AetCrisisStatus) = colCrisis
 and colCrisis->forAll(T:ctCrisis | T.isSentToAHospital(TheActor).eq(true))
//and colCrisis->forAll(T:ctCrisis | TheActor.rnInterfaceIN^ieSendACrisisToHospital(T))

}
postP{true}

}
}
}
```

Listing C.27: Messir Spec. file environment-actHospital-oeGetHospitalCrisisSet.msr.

C.28 File ./src-gen/messir-spec/operations/environment/environment-actHospital-oeGetHospitalInjurySet.msr

```
package icrash.operations.environment.actHospital.oeGetHospitalInjurySet {

import lu.uni.lassy.messir.libraries.primitives

import icrash.concepts.primarytypes.datatypes
import icrash.concepts.primarytypes.classes
import icrash.environment

Operation Model {
operation: actHospital.outactHospital.oeGetHospitalInjurySet(AdtVictimID:dtVictimID):ptBoolean
{

preP{
 let TheSystem: ctState in
 let TheActor:actHospital in

 self.rnActor.rnSystem = TheSystem
 and self.rnActor = actHospital

/* PreP01 */
 and TheSystem.vpStarted = true
```

```
22 /* PreP02 */
23  and TheActor.rnctAuthenticated.vpIsLogged = true
24   }
25 postF{
26  let TheSystem: ctState in
27  let TheActor:actHospital in
28
29  self.rnActor.rnSystem = TheSystem
30  and self.rnActor = TheActor
31
32 /* PostF01 */
33 TheSystem.rnCtInjury -> select(ctVictim.id = dtVictimID) = colVictimInjuries
34 and colVictimInjuries->forAll(T:ctInjury | T.isSentToAHospital(TheActor).eq(true))
35
36 }
37 postP{true}
38
39 }
40 }
41 }
```

Listing C.28: Messir Spec. file environment-actHospital-oeGetHospitalInjurySet.msr.

C.29 File ./src-gen/messir-spec/operations/environment/environment-actHospital-oeGetHospitalVictimSet.msr

```
1 package icrash.operations.environment.actHospital.oeGetHospitalVictimSet {
2
3 import lu.uni.lassy.messir.libraries.primitives
4
5 import icrash.concepts.primarytypes.datatypes
6 import icrash.concepts.primarytypes.classes
7 import icrash.environment
8
9 Operation Model {
10 operation: actHospital.outactHospital.oeGetHospitalVictimSet(AdtCrisisID:dtCrisisID):ptBoolean
11 {
12
13 preP{
14  let TheSystem: ctState in
15  let TheActor:actHospital in
16
17  self.rnActor.rnSystem = TheSystem
18  and self.rnActor = actHospital
19
20 /* PreP01 */
21  and TheSystem.vpStarted = true
22 /* PreP02 */
23  and TheActor.rnctAuthenticated.vpIsLogged = true
24   }
25 postF{
26  let TheSystem: ctState in
27  let TheActor:actHospital in
28
29  self.rnActor.rnSystem = TheSystem
30  and self.rnActor = TheActor
31
32 /* PostF01 */
33 TheSystem.rnCtVictim -> select(ctCrisis.id = dtCrisisID) = colCrisisVictims
34 and colCrisisVictims->forAll(T:ctVictim | T.isSentToAHospital(TheActor).eq(true))
35
36 }
37 postP{true}
38
39 }
40 }
41 }
```

Listing C.29: Messir Spec. file environment-actHospital-oeGetHospitalVictimSet.msr.

C.30 File ./src-gen/messir-spec/operations/environment/environment-actMsrCreator-init.msr

```
package icrash.operations.icrash.environment.actMsrCreator.init {

import lu.uni.lassy.messir.libraries.primitives
import icrash.environment

Operation Model {

operation: actMsrCreator.init():ptBoolean{}
// generic operation provided by the simulator
}
}
```

Listing C.30: Messir Spec. file environment-actMsrCreator-init.msr.

C.31 File ./src-gen/messir-spec/operations/environment/environment-actMsrCreator-oeCreateSystemAndEnvironment.msr

```
package icrash.operations.environment.actMsrCreator.oeCreateSystemAndEnvironment{

import lu.uni.lassy.messir.libraries.primitives
import lu.uni.lassy.messir.libraries.math
import lu.uni.lassy.messir.libraries.calendar

import icrash.concepts.primarytypes.datatypes
import icrash.concepts.primarytypes.classes
import icrash.concepts.secondarytypes.datatypes
import icrash.concepts.secondarytypes.classes
import icrash.environment

Operation Model {

operation: actMsrCreator.outactMsrCreator.oeCreateSystemAndEnvironment(AqtyComCompanies:ptInteger):
    ptBoolean
{preP{true}
preF{true}
postF{
 let TheSystem: ctState in
 let AactMsrCreator: actMsrCreator in
 let AactAdministrator: actAdministrator in
 let AnextValueForAlertID: dtInteger in
 let AnextValueForCrisisID: dtInteger in
 let Aclock: dtDateAndTime in
 let AcrisisReminderPeriod: dtSecond in
 let AmaxCrisisReminderPeriod: dtSecond in
 let AvpStarted: ptBoolean in

 /* PostF01 -- MUST ALWAYS BE MADE FIRST -- */
 AnextValueForAlertID.value.eq(1)
 and AnextValueForCrisisID.value.eq(1)
 and Aclock.date.year.value = 1970
 and Aclock.date.month.value = 01
 and Aclock.date.day.value = 01
 and Aclock.time.hour.value = 00
 and Aclock.time.minute.value = 00
 and Aclock.time.second.value = 00

 and AcrisisReminderPeriod.value.eq(300)
 and AmaxCrisisReminderPeriod.value.eq(1200)
 and AvpStarted = true
 and TheSystem.init(AnextValueForAlertID,
          AnextValueForCrisisID,
          Aclock,
          AcrisisReminderPeriod,
          AmaxCrisisReminderPeriod,
```

```
47            Aclock,
48            AvpStarted
49           )
50  /* PostF02*/
51  and AactMsrCreator.init()
52  /* PostF03 */
53  and let AactComCompanyCol: Bag(actComCompany) in
54  AactComCompanyCol->size() = AqtyComCompanies
55  AactComCompanyCol-> forAll(init())
56  /* PostF04*/
57  and AactAdministrator.init()
58  /* PostF05*/
59  and let AactActivator:actActivator in
60  AactActivator.init()
61 /* PostF06 */
62  and let ActAdministrator:ctAdministrator in
63    let AdtLogin:dtLogin in
64    let AdtPassword:dtPassword in
65    AdtLogin.value.eq('icrashadmin')
66    and AdtPassword.value.eq('7WXC1359')
67    and ActAdministrator.init(AdtLogin,AdtPassword)
68  /* PostF07*/
69  and ActAdministrator@post.rnactAuthenticated = AactAdministrator}
70 postP{true}
71
72 prolog{ "src/Operations/Environment/OUT/outactMsrCreator-oeCreateSystemAndEnvironment.pl"}
73
74 }
75 }
76
77 }
```

Listing C.31: Messir Spec. file environment-actMsrCreator-oeCreateSystemAndEnvironment.msr.

C.32 File ./src-gen/messir-spec/environment/environment.msr

```
1 package icrash.environment{
2
3 import icrash.concepts.primarytypes.datatypes
4 import icrash.concepts.primarytypes.classes
5 import icrash.concepts.secondarytypes.datatypes
6 import lu.uni.lassy.messir.libraries.primitives
7 import lu.uni.lassy.messir.libraries.math
8 import lu.uni.lassy.messir.libraries.calendar
9
10  Environment Model {
11
12   actor actMsrCreator role rnactMsrCreator cardinality [1..1] {
13
14    operation init():ptBoolean
15
16    input interface inactMsrCreator {
17    }
18    output interface outactMsrCreator {
19     operation oeCreateSystemAndEnvironment(AqtyComCompanies:ptInteger ):ptBoolean
20    }
21   }
22
23   actor actAdministrator
24       role rnactAdministrator
25       cardinality [1..1]
26       extends actAuthenticated {
27
28    operation init():ptBoolean
29
30    output interface outactAdministrator{
31
32     operation oeAddCoordinator(
33            AdtCoordinatorID:dtCoordinatorID ,
```

```
34          AdtLogin:dtLogin ,
35          AdtPassword:dtPassword ):ptBoolean
36
37    operation oeDeleteCoordinator(
38          AdtCoordinatorID:dtCoordinatorID ):ptBoolean
39
40    operation oeAddHospital(
41          AdtCoordinatorID:dtHospitalID ,
42          AdtLogin:dtLogin ,
43          AdtPassword:dtPassword ):ptBoolean
44
45    operation oeDeleteHospital(
46          AdtHospitalID:dtHospitalID ):ptBoolean
47   }
48
49   input interface inactAdministrator{
50
51    operation ieCoordinatorAdded():ptBoolean
52    operation ieCoordinatorDeleted():ptBoolean
53    operation ieHospitalAdded():ptBoolean
54    operation ieHospitalDeleted():ptBoolean
55
56    }
57  }
58
59  actor actHospital
60   role rnactHospital
61   cardinality [0..*]
62   extends actAuthenticated{
63
64   operation init():ptBoolean
65
66   input interface inactHospital {
67    operation ieSendACrisisToHospital(ActCrisis:ctCrisis ):ptBoolean
68    operation ieSendAVictimToHospital(ActVictim:ctVictim ):ptBoolean
69    operation ieSendAInjuryToHospital(ActInjury:ctInjury ):ptBoolean
70   }
71   output interface outactHospital {
72    operation oeGetHospitalCrisisSet(AetCrisisStatus:etCrisisStatus):ptBoolean
73    operation oeGetHospitalVictimSet(AdtCrisisID:dtCrisisID):ptBoolean
74    operation oeGetHospitalInjurySet(AdtVictimID:dtVictimID):ptBoolean
75   }
76  }
77
78  actor actCoordinator
79      role rnactCoordinator
80      cardinality [0..*]
81      extends actAuthenticated{
82
83   operation init():ptBoolean
84
85   output interface outactCoordinator{
86    operation oeAddVictim(AdtCrisisID:dtCrisisID):ptBoolean
87    operation oeDeleteVictim(AdtVictimID:dtVictimID):ptBoolean
88    operation oeAddInjury(
89          AdtVictimID:dtVictimID,
90          AetInjuryKind:etInjuryKind
91          ):ptBoolean
92    operation oeDeleteInjury(AdtInjuryID:dtInjuryID):ptBoolean
93
94    operation oeInvalidateAlert(AdtAlertID:dtAlertID):ptBoolean
95    operation oeCloseCrisis(AdtCrisisID:dtCrisisID):ptBoolean
96    operation oeGetAlertsSet(AetAlertStatus:etAlertStatus):ptBoolean
97    operation oeGetCrisisSet(AetCrisisStatus:etCrisisStatus):ptBoolean
98    operation oeGetVictimSet(AdtCrisisID:dtCrisisID):ptBoolean
99    operation oeGetInjurySet(AdtVictimID:dtVictimID):ptBoolean
100   operation oeSetCrisisHandler(AdtCrisisID:dtCrisisID):ptBoolean
101   operation oeReportOnCrisis(
102         AdtCrisisID:dtCrisisID ,
103         AdtComment:dtComment
```

```
104          ):ptBoolean
105    operation oeSetCrisisStatus(
106          AdtCrisisID:dtCrisisID ,
107          AetCrisisStatus:etCrisisStatus
108          ):ptBoolean
109    operation oeSetCrisisType(
110                    AdtCrisisID:dtCrisisID ,
111                    AetCrisisType:etCrisisType
112                    ):ptBoolean
113    operation oeValidateAlert(AdtAlertID:dtAlertID ):ptBoolean
114   }
115
116   input interface inactCoordinator{
117    operation ieSendAnAlert(ActAlert:ctAlert ):ptBoolean
118    operation ieSendACrisis(ActCrisis:ctCrisis ):ptBoolean
119    operation ieSendAVictim(ActVictim:ctVictim ):ptBoolean
120    operation ieSendAInjury(ActInjury:ctInjury ):ptBoolean
121
122    operation ieVictimAdded():ptBoolean
123    operation ieVictimDeleted():ptBoolean
124    operation ieInjuryAdded():ptBoolean
125    operation ieInjuryDeleted():ptBoolean
126    }
127  }
128
129  actor actComCompany role rnactComCompany cardinality [0..*]{
130
131   operation init():ptBoolean
132
133   output interface outactComCompany{
134    operation oeAlert(
135          AetHumanKind:etHumanKind ,
136          AdtDate:dtDate ,
137          AdtTime:dtTime ,
138          AdtPhoneNumber:dtPhoneNumber ,
139          AdtGPSLocation:dtGPSLocation ,
140          AdtComment:dtComment
141          ):ptBoolean
142   }
143
144   input interface inactComCompany{
145    operation ieSmsSend(AdtPhoneNumber:dtPhoneNumber ,
146              AdtSMS:dtSMS
147              ):ptBoolean
148    }
149  }
150
151  actor actAuthenticated role rnactAuthenticated cardinality [0..*]{
152
153   operation init():ptBoolean
154
155   output interface outactAuthenticated{
156    operation oeLogin(AdtLogin:dtLogin , AdtPassword:dtPassword ):ptBoolean
157    operation oeLogout():ptBoolean
158   }
159
160   input interface inactAuthenticated{
161    operation ieMessage(AMessage:ptString):ptBoolean
162    }
163  }
164
165  actor actActivator[proactive] role rnactActivator cardinality [1..1]{
166
167   operation init():ptBoolean
168
169   output interface outactActivator{
170    proactive operation oeSollicitateCrisisHandling():ptBoolean
171    proactive operation oeSetClock(AcurrentClock:dtDateAndTime ):ptBoolean
172   }
173
```

```
174    input interface inactActivator{
175     }
176   }
177
178   actor actCorrupter[proactive] role rnactCorrupter cardinality [1..1]{
179
180    operation init():ptBoolean
181
182    output interface outactCorrupter{
183     proactive operation oeCorruptAlert(ActAlert:ctAlert):ptBoolean
184    }
185
186    input interface inactCorrupter{
187     }
188   }
189
190  }
191 }
```

Listing C.32: Messir Spec. file environment.msr.

C.33 File ./src-gen/messir-spec/operations/environment/evironment-actCoordinator-oeAddInjury.msr

```
 1 package icrash.operations.evironment.actCoordinator.oeAddInjury {
 2
 3 import lu.uni.lassy.messir.libraries.primitives
 4
 5 import icrash.concepts.primarytypes.datatypes
 6 import icrash.concepts.primarytypes.classes
 7 import icrash.environment
 8
 9 Operation Model {
10 operation: actCoordinator.outactCoordinator.oeAddInjury(AdtVictimID:dtVictimID, AetInjuryKind:
      etInjuryKind):ptBoolean
11 {
12
13 preP{
14  let TheSystem: ctState in
15  let TheActor:actCoordinator in
16  let AdtInjuryID:dtInjuryID in
17
18  self.rnActor.rnSystem = TheSystem
19  and self.rnActor = TheCoordinator
20
21 /* PreP01 */
22  and TheSystem.vpStarted = true
23 /* PreP02 */
24  and TheActor.rnctAuthenticated.vpIsLogged = true
25   }
26 postF{
27  let TheSystem: ctState in
28  let TheActor:actCoordinator in
29
30  self.rnActor.rnSystem = TheSystem
31  and self.rnActor = TheActor
32
33 /* PostF01 */
34  TheSystem.nextValueForInjuryID=PrenextValueForInjuryID
35  and PrenextValueForInjuryID.add(1) = PostnextValueForInjuryID
36  and TheSystem@post.nextValueForInjuryID = PostnextValueForInjuryID
37
38 /* PostF02 */
39 and TheSystem.nextValueForInjuryID.todtString().eq(AdtInjuryID)
40 and ActInjury.init(AdtInjuryID, AetInjuryKind)
41
42 /* PostF03 */
43 TheSystem.rnCtVictim -> select(ctVictim.id = dtVictimID) = TEMP
```

```
44 TEMP.rnTheInjury->msrAny(true) = ActInjury
45
46 /* PostF04 */
47 and TheActor.rnInterfaceIN^ieInjuryAdded()
48
49 }
50 postP{true}
51
52 }
53 }
54 }
```

Listing C.33: Messir Spec. file evironment-actCoordinator-oeAddInjury.msr.

C.34 File ./src-gen/messir-spec/concepts/primarytypes-associations.msr

```
1 package icrash.concepts.primarytypes.associations {
2
3 import icrash.concepts.primarytypes.datatypes
4 import icrash.concepts.primarytypes.classes
5 import icrash.environment
6 import lu.uni.lassy.messir.libraries.primitives
7
8  Concept Model {
9
10  Primary Types{
11
12 // Internal
13
14 association assctAlertctCrisis
15  ctAlert(rnAlerts)[1..*]
16  ctCrisis (rnTheCrisis)[1..1]
17
18 association assctAlertctHuman
19  ctAlert(rnSignaled)[1..*] // Why this role name?
20  ctHuman (rnSignaler)[1..1]
21
22 association assctCrisisctCoordinator
23  ctCrisis(rnHandled)[0..*]
24  ctCoordinator(rnHandler)[0..1]
25
26 association assctVictimCrisis
27  ctVictim(rnVictim)[1..*]
28  ctCrisis(rnTheCrisis)[1..1]
29
30 association assctInjuryVictim
31  ctInjury (rnInjury)[1..*]
32  ctVictim(rnVictim)[1..1]
33
34 // With Actors
35
36    association assctHumanactComCompany
37       ctHuman(rnctHuman)[0..*]
38       actComCompany(rnactComCompany)[1..1]
39
40    association assctCoordinatoractCoordinator
41       ctCoordinator(rnctCoordinator)[1..1]
42       actCoordinator(rnactCoordinator)[1..1]
43
44    association assctHospitalactHospital
45       ctHospital(rnctHospital)[1..1]
46       actHospital(rnactHospital)[1..1]
47
48    association assctAuthenticatedactAuthenticated
49       ctAuthenticated(rnctAuthenticated)[1..1]
50       actAuthenticated(rnactAuthenticated)[1..1]
51
```

```
  }
 }
}
```

Listing C.34: Messir Spec. file primarytypes-associations.msr.

C.35 File ./src-gen/messir-spec/operations/concepts/primarytypes-classes/primarytypes-classes-ctAdministrator.msr

```
package icrash.operations.concepts.primarytypes.classes.ctAdministrator{

import lu.uni.lassy.messir.libraries.primitives

import icrash.concepts.primarytypes.datatypes
import icrash.concepts.primarytypes.classes

Operation Model {

operation: icrash.concepts.primarytypes.classes.ctAdministrator.init(
 Alogin:dtLogin ,
 Apwd:dtPassword
 ):ptBoolean{
postF{
if
(
let Self:ctAdministrator in
/* Post F01 */
Self.login(Alogin)
and Self.pwd = Apwd
and Self.vpIsLogged = false

/* Post F02 */
and (Self.oclIsNew and self = Self)
)
then (result = true)
else (result = false)
endif
}
prolog{ "src/Operations/Concepts/PrimaryTypesClasses/PrimaryTypesClasses-ctAdministrator-init.pl"}
 }
}
}
```

Listing C.35: Messir Spec. file primarytypes-classes-ctAdministrator.msr.

C.36 File ./src-gen/messir-spec/operations/concepts/primarytypes-classes/primarytypes-classes-ctAlert.msr

```
package icrash.operations.concepts.primarytypes.classes.ctAlert{

import lu.uni.lassy.messir.libraries.primitives
import lu.uni.lassy.messir.libraries.calendar

import icrash.concepts.primarytypes.datatypes
import icrash.concepts.primarytypes.classes

import icrash.environment

Operation Model {

operation: icrash.concepts.primarytypes.classes.ctAlert.init(Aid:dtAlertID , Astatus:etAlertStatus ,
      Alocation:dtGPSLocation , Ainstant:dtDateAndTime , Acomment:dtComment
):ptBoolean{
postF{
let AetCorruptionKind:etCorruptionKind in
AetCorruptionKind = regular
```

```
18
19 if
20 (
21 /* Post F01 */
22 let Self:ctAlert in
23 Self.id = Aid
24 and Self.status = Astatus
25 and Self.location = Alocation
26 and Self.instant = Ainstant
27 and Self.comment = Acomment
28 and Self.corruption = AetCorruptionKind
29 /* Post F02 */
30 and (Self.oclIsNew and self = Self)
31 )
32 then (result = true)
33 else (result = false)
34 endif
35 }
36 prolog{"src/Operations/Concepts/PrimaryTypesClasses/PrimaryTypesClasses-ctAlert-init.pl"}
37 }
38
39 operation: icrash.concepts.primarytypes.classes.ctAlert.isSentToCoordinator(AactCoordinator:
       actCoordinator ):ptBoolean
40 {
41 postF{
42 if
43 (
44 /* Post F01 */
45 AactCoordinator.rnInterfaceIN.ieSendAnAlert(self)
46 )
47 then (result = true)
48 else (result = false)
49 endif
50 }
51 prolog{"src/Operations/Concepts/PrimaryTypesClasses/PrimaryTypesClasses-ctAlert-isSentToCoordinator.
       pl"}
52
53 }
54 }
55 }
```

Listing C.36: Messir Spec. file primarytypes-classes-ctAlert.msr.

C.37 File ./src-gen/messir-spec/operations/concepts/primarytypes-classes/primarytypes-classes-ctAuthenticated.msr

```
1 package icrash.operations.concepts.primarytypes.classes.ctAuthenticated {
2
3 import lu.uni.lassy.messir.libraries.primitives
4 import icrash.concepts.primarytypes.datatypes
5 import icrash.concepts.primarytypes.classes
6
7 Operation Model {
8
9 operation: icrash.concepts.primarytypes.classes.ctAuthenticated.init(Alogin:dtLogin, Apwd:dtPassword
       ):ptBoolean{
10 prolog{"src/Operations/Concepts/PrimaryTypesClasses/PrimaryTypesClasses-ctAuthenticated-init.pl"}
11 }
12 }
13
14 }
```

Listing C.37: Messir Spec. file primarytypes-classes-ctAuthenticated.msr.

C.38 File ./src-gen/messir-spec/operations/concepts/primarytypes-classes/primarytypes-classes-ctCoordinator.msr

```
package icrash.operations.concepts.primarytypes.classes.ctCoordinator.init {

import lu.uni.lassy.messir.libraries.primitives
import icrash.concepts.primarytypes.datatypes
import icrash.concepts.primarytypes.classes

Operation Model {

operation: icrash.concepts.primarytypes.classes.ctCoordinator.init(Aid:dtCoordinatorID, Alogin:
    dtLogin, Apwd:dtPassword):ptBoolean
{
postF{
if
(
/* Post F01 */
let Self:ctCoordinator in
Self.id = Aid
and Self.login = Alogin
and Self.pwd = Apwd
and Self.vpIsLogged = false
/* Post F02 */
and (Self.oclIsNew and self = Self)
)
then (result = true)
else (result = false)
endif}
prolog{"src/Operations/Concepts/PrimaryTypesClasses/PrimaryTypesClasses-ctCoordinator-init.pl"}
}
}
}
```

Listing C.38: Messir Spec. file primarytypes-classes-ctCoordinator.msr.

C.39 File ./src-gen/messir-spec/operations/concepts/primarytypes-classes/primarytypes-classes-ctCrisis.msr

```
package icrash.operations.concepts.primarytypes.classes.ctCrisis {

import lu.uni.lassy.messir.libraries.primitives
import lu.uni.lassy.messir.libraries.math
import lu.uni.lassy.messir.libraries.calendar

import icrash.concepts.primarytypes.datatypes
import icrash.concepts.primarytypes.classes
import icrash.concepts.secondarytypes.datatypes
import icrash.concepts.secondarytypes.classes
import lu.uni.lassy.messir.libraries.primitives

import icrash.environment

Operation Model {
//-------------------------------------------------------------
operation: icrash.concepts.primarytypes.classes.ctCrisis.init(
        Aid:dtCrisisID,
        Atype:etCrisisType,
        Astatus:etCrisisStatus,
        Alocation:dtGPSLocation,
        Ainstant:dtDateAndTime,
        Acomment:dtComment
       ):ptBoolean{
postF{
if
(
/* Post F01 */
let Self:ctCrisis in
Self.id = Aid
and Self.type = Atype
and Self.status = Astatus
```

```
33 and Self.location = Alocation
34 and Self.instant = Ainstant
35 and Self.comment = Acomment
36 /* Post F02 */
37 and (Self.oclIsNew and self = Self)
38 )
39 then (result = true)
40 else (result = false)
41 endif}
42 prolog{"src/Operations/Concepts/PrimaryTypesClasses/PrimaryTypesClasses-ctCrisis-init.pl"}}
43 //-----------------------------------------------------------
44 operation: icrash.concepts.primarytypes.classes.ctCrisis.handlingDelayPassed():ptBoolean
45 {
46 postF{
47 let TheSystem:ctState in
48 let CurrentClockSecondsQty:dtInteger in
49 let vpLastReminderSecondsQty:dtInteger in
50 let CrisisReminderPeriod:dtSecond in
51 if
52 ( /* Post F01 */
53  self.rnSystem = TheSystem
54  and self.status = pending
55  and TheSystem.clock.toSecondsQty() = CurrentClockSecondsQty
56  and TheSystem.vpLastReminder.toSecondsQty() = vpLastReminderSecondsQty
57  and TheSystem.crisisReminderPeriod = CrisisReminderPeriod
58  and CurrentClockSecondsQty.sub(vpLastReminderSecondsQty).gt(CrisisReminderPeriod) = true
59 )
60 then (result = true)
61 else (result = false)
62 endif
63 }
64 prolog{"src/Operations/Concepts/PrimaryTypesClasses/PrimaryTypesClasses-ctCrisis-handlingDelayPassed
       .pl"}}
65 //-----------------------------------------------------------
66 operation: icrash.concepts.primarytypes.classes.ctCrisis.maxHandlingDelayPassed():ptBoolean
67 {
68 postF{
69 let TheSystem:ctState in
70 let CurrentClockSecondsQty:dtInteger in
71 let CrisisInstantSecondsQty:dtInteger in
72 let MaxCrisisReminderPeriod:dtSecond in
73 if
74 ( /* Post F01 */
75  self.rnSystem = TheSystem
76  and self.status = pending
77  and TheSystem.clock.toSecondsQty() = CurrentClockSecondsQty
78  and Self.instant.toSecondsQty() = CrisisInstantSecondsQty
79  and TheSystem.maxCrisisReminderPeriod = MaxCrisisReminderPeriod
80  and CurrentClockSecondsQty.sub(CrisisInstantSecondsQty)
81                .gt(MaxCrisisReminderPeriod)
82 )
83 then (result = true)
84 else (result = false)
85 endif
86 }
87 prolog{"src/Operations/Concepts/PrimaryTypesClasses/PrimaryTypesClasses-ctCrisis-
       maxHandlingDelayPassed.pl"}}
88 //-----------------------------------------------------------
89 operation: icrash.concepts.primarytypes.classes.ctCrisis.isSentToCoordinator(AactCoordinator:
       actCoordinator):ptBoolean
90 {
91 postF{
92 if
93 (
94 /* Post F01 */
95  AactCoordinator.rnInterfaceIN.ieSendACrisis(self)
96 )
97 then (result = true)
98 else (result = false)
99 endif}
```

```
prolog{"src/Operations/Concepts/PrimaryTypesClasses/PrimaryTypesClasses-ctCrisis-isSentToCoordinator
    .pl" }}
//-----------------------------------------------------------
operation: icrash.concepts.primarytypes.classes.ctCrisis.isSentToHospital(AactHospital:actHospital):
    ptBoolean
{
postF{
if
(
/* Post F01 */
 AactHospital.rnInterfaceIN^ieSendACrisis(self)
)
then (result = true)
else (result = false)
endif}
prolog{"src/Operations/Concepts/PrimaryTypesClasses/PrimaryTypesClasses-ctCrisis-isSentToHospital.pl
    " }}
//-----------------------------------------------------------
operation: icrash.concepts.primarytypes.classes.ctCrisis.isAllocatedIfPossible():ptBoolean
{
postF{
if (
/* Post F01 */
self.maxHandlingDelayPassed()
and
 if (TheSystem.rnactCoordinator->msrIsEmpty = false)
 then (
   /* Post F02 */
   TheSystem.rnactCoordinator->msrAny(true) = TheCoordinatorActor
   and TheCoordinatorActor.rnctCoordinator = TheCoordinator
   and self@post.rnHandler = TheCoordinator
   and self@post.status = handled
   and self.id.value = TheCrisisIDptString
   and 'You are now considered as handling the crisis having ID: '
     .ptStringConcat(TheCrisisIDptString) = TheMessage
    and TheCoordinatorActor.rnInterfaceIN^ieMessage(TheMessage)
  )
 else ( /* Post F03 */
    TheSystem.rnactAdministrator
    ->forAll(rnInterfaceIN.ieMessage('Please add new coordinators to handle pending crisis !'))
  )
 endif
 )
then (result = true)
else (result = false)
endif
}
prolog{"src/Operations/Concepts/PrimaryTypesClasses/PrimaryTypesClasses-ctCrisis-
    isAllocatedIfPossible.pl"}
}
}
}
```

Listing C.39: Messir Spec. file primarytypes-classes-ctCrisis.msr.

C.40 File ./src-gen/messir-spec/operations/concepts/primarytypes-classes/primarytypes-classes-ctHuman.msr

```
package icrash.operations.concepts.primarytypes.classes.ctHuman.init {

import lu.uni.lassy.messir.libraries.primitives
import icrash.concepts.primarytypes.datatypes

import icrash.concepts.primarytypes.classes

Operation Model {

operation: icrash.concepts.primarytypes.classes.ctHuman.init(Aid:dtPhoneNumber, Akind:etHumanKind):
    ptBoolean
```

```
11 {
12 postF{
13 if
14 (
15 /* Post F01 */
16 let Self:ctHuman in
17
18 Self.id = Aid
19 and Self.kind = Akind
20
21 /* Post F02 */
22 and (Self.oclIsNew and self = Self)
23 )
24 then (result = true)
25 else (result = false)
26 endif
27 }
28 prolog{"src/Operations/Concepts/PrimaryTypesClasses/PrimaryTypesClasses-ctHuman-init.pl"}
29 }
30 operation: icrash.concepts.primarytypes.classes.ctHuman.isAcknowledged():ptBoolean{
31 prolog{"src/Operations/Concepts/PrimaryTypesClasses/PrimaryTypesClasses-ctHuman-isAcknowledged.pl"}
32 }
33 }
34 }
```

Listing C.40: Messir Spec. file primarytypes-classes-ctHuman.msr.

C.41 File ./src-gen/messir-spec/operations/concepts/primarytypes-classes/primarytypes-classes-ctInjury.msr

```
1 package icrash.operations.concepts.primarytypes.classes.ctInjury {
2
3 import lu.uni.lassy.messir.libraries.primitives
4 import lu.uni.lassy.messir.libraries.math
5 import lu.uni.lassy.messir.libraries.calendar
6
7 import icrash.concepts.primarytypes.datatypes
8 import icrash.concepts.primarytypes.classes
9 import icrash.concepts.secondarytypes.datatypes
10 import icrash.concepts.secondarytypes.classes
11 import lu.uni.lassy.messir.libraries.primitives
12
13 import icrash.environment
14
15 Operation Model {
16 //-----------------------------------------------------------
17 operation: icrash.concepts.primarytypes.classes.ctInjury.init(
18     Aid:dtInjuryID,
19         AetInjuryKind:etInjuryKind
20        ):ptBoolean{
21 postF{
22 if
23 (
24 /* Post F01 */
25 let Self:ctInjury in
26 Self.id = Aid
27 and Self.victimID = AvictimID
28 /* Post F02 */
29 and (Self.oclIsNew and self = Self)
30 )
31 then (result = true)
32 else (result = false)
33 endif}
34 prolog{"src/Operations/Concepts/PrimaryTypesClasses/PrimaryTypesClasses-ctInjury-init.pl"}}
35 //-----------------------------------------------------------
36 operation: icrash.concepts.primarytypes.classes.ctInjury.isSentToCoordinator(AactCoordinator:
      actCoordinator):ptBoolean
37 {
```

```
postF{
if
(
/* Post F01 */
 AactCoordinator.rnInterfaceIN.ieSendAInjury(self)
)
then (result = true)
else (result = false)
endif}
prolog{"src/Operations/Concepts/PrimaryTypesClasses/PrimaryTypesClasses-ctCrisis-isSentToCoordinator
      .pl" }}
//-----------------------------------------------------------
operation: icrash.concepts.primarytypes.classes.ctInjury.isSentToHospital(AactHospital:actHospital):
      ptBoolean
{
postF{
if
(
/* Post F01 */
 AactHospital.rnInterfaceIN.ieSendAInjury(self)
)
then (result = true)
else (result = false)
endif}
prolog{"src/Operations/Concepts/PrimaryTypesClasses/PrimaryTypesClasses-ctInjury-isSentToHospital.pl
      " }}
}
}
```

Listing C.41: Messir Spec. file primarytypes-classes-ctInjury.msr.

C.42 File ./src-gen/messir-spec/operations/concepts/primarytypes-classes/primarytypes-classes-ctState.msr

```
package icrash.operations.concepts.primarytypes.classes.ctState{

import lu.uni.lassy.messir.libraries.primitives
import lu.uni.lassy.messir.libraries.calendar
import lu.uni.lassy.messir.libraries.math

import icrash.concepts.primarytypes.classes

Operation Model {

 operation: icrash.concepts.primarytypes.classes.ctState.init(
  AnextValueForAlertID: dtInteger,
  AnextValueForCrisisID: dtInteger ,
  AnextValueForVictimID: dtInteger ,
  AnextValueForInjuryID: dtInteger ,
  dtAclock:dtDateAndTime,
  AcrisisReminderPeriod: dtSecond,
  AmaxCrisisReminderPeriod: dtSecond ,
  AvpLastReminder: dtDateAndTime ,
  AvpStarted:ptBoolean ):ptBoolean{
postF{
if
(
/* Post F01 */
let Self:ctState in

Self.nextValueForAlertID = AnextValueForAlertID
and Self.nextValueForCrisisID = AnextValueForCrisisID
and Self.nextValueForVictimID = AnextValueForVictimID
and Self.nextValueForInjuryID = AnextValueForInjuryID
and Self.clock = Aclock
and Self.crisisReminderPeriod = AcrisisReminderPeriod
and Self.maxCrisisReminderPeriod = AmaxCrisisReminderPeriod
and Self.vpLastReminder = AvpLastReminder
```

```
35 and Self.vpStarted = AvpStarted
36
37 and (Self.oclIsNew and self = Self)
38 )
39 then (result = true)
40 else (result = false)
41 endif
42 }
43 prolog{"src/Operations/Concepts/PrimaryTypesClasses/PrimaryTypesClasses-ctState-init.pl" }
44  }
45  }
46 }
```

Listing C.42: Messir Spec. file primarytypes-classes-ctState.msr.

C.43 File ./src-gen/messir-spec/operations/concepts/primarytypes-classes/primarytypes-classes-ctVictim.msr

```
1 package icrash.operations.concepts.primarytypes.classes.ctVictim {
2
3 import lu.uni.lassy.messir.libraries.primitives
4 import lu.uni.lassy.messir.libraries.math
5 import lu.uni.lassy.messir.libraries.calendar
6
7 import icrash.concepts.primarytypes.datatypes
8 import icrash.concepts.primarytypes.classes
9 import icrash.concepts.secondarytypes.datatypes
10 import icrash.concepts.secondarytypes.classes
11 import lu.uni.lassy.messir.libraries.primitives
12
13 import icrash.environment
14
15 Operation Model {
16 //-----------------------------------------------------------
17 operation: icrash.concepts.primarytypes.classes.ctVictim.init(
18         Aid:dtVictimID ):ptBoolean{
19 postF{
20 if
21 (
22 /* Post F01 */
23 let Self:ctVictim in
24 Self.id = Aid
25 and Self.crisisID = AcrisisID
26 /* Post F02 */
27 and (Self.oclIsNew and self = Self)
28 )
29 then (result = true)
30 else (result = false)
31 endif}
32 prolog{"src/Operations/Concepts/PrimaryTypesClasses/PrimaryTypesClasses-ctVictim-init.pl"}}
33 //-----------------------------------------------------------
34 operation: icrash.concepts.primarytypes.classes.ctVictim.isSentToCoordinator(AactCoordinator:
      actCoordinator):ptBoolean
35 {
36 postF{
37 if
38 (
39 /* Post F01 */
40  AactCoordinator.rnInterfaceIN.ieSendAVictim(self)
41 )
42 then (result = true)
43 else (result = false)
44 endif}
45 prolog{"src/Operations/Concepts/PrimaryTypesClasses/PrimaryTypesClasses-ctCrisis-isSentToCoordinator
      .pl" }}
46 //-----------------------------------------------------------
47 operation: icrash.concepts.primarytypes.classes.ctVictim.isSentToHospital(AactHospital:actHospital):
      ptBoolean
```

```
48 {
49 postF{
50 if
51 (
52 /* Post F01 */
53  AactHospital.rnInterfaceIN.ieSendAVictim(self)
54 )
55 then (result = true)
56 else (result = false)
57 endif}
58 prolog{"src/Operations/Concepts/PrimaryTypesClasses/PrimaryTypesClasses-ctCrisis-isSentToHospital.pl
       " }}
59 }
60 }
```

Listing C.43: Messir Spec. file primarytypes-classes-ctVictim.msr.

C.44 File ./src-gen/messir-spec/concepts/primarytypes-classes.msr

```
1 package icrash.concepts.primarytypes.classes {
2
3 import icrash.concepts.primarytypes.datatypes
4 import icrash.environment
5 import lu.uni.lassy.messir.libraries.primitives
6 import lu.uni.lassy.messir.libraries.math
7 import lu.uni.lassy.messir.libraries.calendar
8
9  Concept Model {
10
11   Primary Types{
12
13    state class ctState {
14     attribute nextValueForAlertID:dtInteger
15     attribute nextValueForCrisisID:dtInteger
16     attribute nextValueForVictimID:dtInteger
17     attribute nextValueForInjuryID:dtInteger
18     attribute clock:dtDateAndTime
19     attribute crisisReminderPeriod:dtSecond
20     attribute maxCrisisReminderPeriod:dtSecond
21     attribute vpLastReminder:dtDateAndTime
22     attribute vpStarted:ptBoolean
23
24     operation init( AnextValueForAlertID:dtInteger,
25             AnextValueForCrisisID:dtInteger,
26             AnextValueForVictimID:dtInteger,
27             AnextValueForInjuryID:dtInteger,
28              Aclock:dtDateAndTime,
29             AcrisisReminderPeriod:dtSecond ,
30             AmaxCrisisReminderPeriod:dtSecond ,
31             AvpLastReminder:dtDateAndTime ,
32             AvpStarted:ptBoolean ): ptBoolean
33    }
34
35    class ctAlert role rnctAlert cardinality [0..*]{
36     attribute id:dtAlertID
37     attribute status: etAlertStatus
38     attribute location:dtGPSLocation
39     attribute instant:dtDateAndTime
40     attribute comment:dtComment
41     attribute corruption:etCorruptionKind
42
43     operation init(    Aid:dtAlertID ,
44           Astatus:etAlertStatus ,
45           Alocation:dtGPSLocation ,
46           Ainstant:dtDateAndTime ,
47           Acomment:dtComment ):ptBoolean
48     operation isSentToCoordinator(AactCoordinator:actCoordinator ):ptBoolean
49
50    }
```

```
51
52   class ctCrisis role rnctCrisis cardinality [0..*]{
53    attribute id:dtCrisisID
54    attribute type:etCrisisType
55    attribute status: etCrisisStatus
56    attribute location:dtGPSLocation
57    attribute instant:dtDateAndTime
58    attribute comment:dtComment
59
60    operation init(
61          Aid:dtCrisisID ,
62          Atype:etCrisisType ,
63          Astatus:etCrisisStatus ,
64          Alocation:dtGPSLocation ,
65          Ainstant:dtDateAndTime ,
66          Acomment:dtComment ):ptBoolean
67
68    operation handlingDelayPassed():ptBoolean
69    operation maxHandlingDelayPassed():ptBoolean
70    operation isSentToCoordinator(AactCoordinator:actCoordinator ):ptBoolean
71    operation isSentToHospital(AactHospital:actHospital ):ptBoolean
72    operation isAllocatedIfPossible():ptBoolean
73   }
74
75   class ctVictim role rnctVictim cardinality [0..*]{
76    attribute id:dtVictimID
77
78    operation init(
79          Aid:dtVictimID ):ptBoolean
80
81    operation isSentToCoordinator(AactCoordinator:actCoordinator ):ptBoolean
82    operation isSentToHospital(AactHospital:actHospital ):ptBoolean
83   }
84
85   class ctInjury role rnctInjury cardinality [0..*]{
86    attribute id:dtInjuryID
87    attribute kind:etInjuryKind
88
89    operation init(
90          Aid:dtInjuryID,
91               AetInjuryKind:etInjuryKind
92           ):ptBoolean
93
94    operation isSentToCoordinator(AactCoordinator:actCoordinator ):ptBoolean
95    operation isSentToHospital(AactHospital:actHospital ):ptBoolean
96   }
97
98   class ctHuman role rnctHuman cardinality [0..*]{
99    attribute id:dtPhoneNumber
100   attribute kind:etHumanKind
101
102   operation init(
103         Aid:dtPhoneNumber ,
104         Akind:etHumanKind ):ptBoolean
105   operation isAcknowledged():ptBoolean
106  }
107
108  class ctAuthenticated
109    role rnctAuthenticated
110    cardinality [0..*]{
111
112   attribute login:dtLogin
113   attribute pwd: dtPassword
114   attribute vpIsLogged:ptBoolean
115
116   operation init(
117         Alogin:dtLogin ,
118         Apwd:dtPassword ):ptBoolean
119  }
120
```

```
  class ctCoordinator
    role rnctCoordinator
    cardinality [0..*]
    extends ctAuthenticated{

   attribute id:dtCoordinatorID

   operation init(
         Aid:dtCoordinatorID ,
         Alogin:dtLogin ,
         Apwd:dtPassword ):ptBoolean
  }

  class ctHospital
    role rnctHospital
    cardinality [0..*]
    extends ctAuthenticated{

   attribute id:dtHospitalID

   operation init(
         Aid:dtHospitalID ,
         Alogin:dtLogin ,
         Apwd:dtPassword ):ptBoolean
  }

  class ctAdministrator
    role rnctAdministrator
    cardinality [1..1]
    extends ctAuthenticated{

   operation init(
         Alogin:dtLogin ,
         Apwd:dtPassword ):ptBoolean
  }
 }
}
}
```

Listing C.44: Messir Spec. file primarytypes-classes.msr.

C.45 File ./src-gen/messir-spec/operations/concepts/primarytypes-datatypes/primarytypes-datatypes-dtAlertID.msr

```
package icrash.operations.concepts.primarytypes.datatypes.dtAlertID{

import lu.uni.lassy.messir.libraries.primitives

Operation Model {

 operation: icrash.concepts.primarytypes.datatypes.dtAlertID.is():ptBoolean{

 postF{
 let TheResult: ptBoolean in
 ( if
  ( AdtValue.value.length().gt(0)
   and AdtValue.value.length().leq(20)
  )
  then (TheResult = true)
  else (TheResult = false)
  endif
  result = TheResult
 )}
 prolog{"src/Operations/Concepts/PrimaryTypesDatatypes/PrimaryTypesDatatypes-dtAlertID-is.pl"}
}
}
```

```
}
```

Listing C.45: Messir Spec. file primarytypes-datatypes-dtAlertID.msr.

C.46 File ./src-gen/messir-spec/operations/concepts/primarytypes-datatypes/primarytypes-datatypes-dtComment.msr

```
package icrash.operations.concepts.primarytypes.datatypes.dtComment{

import lu.uni.lassy.messir.libraries.primitives

Operation Model {

  operation: icrash.concepts.primarytypes.datatypes.dtComment.is():ptBoolean{

    postF{
    let TheResult: ptBoolean in
    ( if
     ( MaxLength = 160
      and AdtValue.value.length().leq(MaxLength)
     )
     then (TheResult = true)
     else (TheResult = false)
     endif
     result = TheResult
    )
  }
  prolog{"src/Operations/Concepts/PrimaryTypesDatatypes/PrimaryTypesDatatypes-dtComment-is.pl"}
 }
}
}
```

Listing C.46: Messir Spec. file primarytypes-datatypes-dtComment.msr.

C.47 File ./src-gen/messir-spec/operations/concepts/primarytypes-datatypes/primarytypes-datatypes-dtCoordinatorID.msr

```
package icrash.operations.concepts.primarytypes.datatypes.dtCoordinatorID{

import lu.uni.lassy.messir.libraries.primitives

Operation Model {
  operation: icrash.concepts.primarytypes.datatypes.dtCoordinatorID.is():ptBoolean{

  postF{
    let TheResult: ptBoolean in
    ( if
     ( AdtValue.value.length().gt(0)
      and AdtValue.value.length().leq(5)
     )
     then (TheResult = true)
     else (TheResult = false)
     endif
     result = TheResult
    )
  }
  prolog{"src/Operations/Concepts/PrimaryTypesDatatypes/PrimaryTypesDatatypes-dtCoordinatorID-is.pl"
    }
 }
}
}
```

Listing C.47: Messir Spec. file primarytypes-datatypes-dtCoordinatorID.msr.

C.48 File ./src-gen/messir-spec/operations/concepts/primarytypes-datatypes/primarytypes-datatypes-dtCrisisID.msr

```
package icrash.operations.concepts.primarytypes.datatypes.dtCrisisID{

import lu.uni.lassy.messir.libraries.primitives

Operation Model {

  operation: icrash.concepts.primarytypes.datatypes.dtCrisisID.is():ptBoolean{

  postF{
    let TheResult: ptBoolean in
    ( if
     ( AdtValue.value.length().gt(0)
      and AdtValue.value.length().leq(10)
     )
     then (TheResult = true)
     else (TheResult = false)
     endif
     result = TheResult
    )
  }
  prolog{"src/Operations/Concepts/PrimaryTypesDatatypes/PrimaryTypesDatatypes-dtCrisisID-is.pl"}
 }
}
}
```

Listing C.48: Messir Spec. file primarytypes-datatypes-dtCrisisID.msr.

C.49 File ./src-gen/messir-spec/operations/concepts/primarytypes-datatypes/primarytypes-datatypes-dtGPSLocation.msr

```
package icrash.operations.concepts.primarytypes.datatypes.dtGPSLocation{

import lu.uni.lassy.messir.libraries.primitives
import lu.uni.lassy.messir.libraries.math

import icrash.concepts.primarytypes.datatypes
import icrash.concepts.primarytypes.classes
import icrash.concepts.secondarytypes.datatypes
import icrash.concepts.secondarytypes.classes

Operation Model {

  operation: icrash.concepts.primarytypes.datatypes.dtGPSLocation.is():ptBoolean{
  postF{
    let TheResult: ptBoolean in
    ( if
     ( AdtValue.latitude.is()
      and AdtValue.longitude.is
     )
     then (TheResult = true)
     else (TheResult = false)
     endif
     result = TheResult
    )
  }
  prolog{"src/Operations/Concepts/PrimaryTypesDatatypes/PrimaryTypesDatatypes-dtGPSLocation-is.pl"}
  }
  operation: icrash.concepts.primarytypes.datatypes.dtGPSLocation.isNearTo(aGPSLocation:
      dtGPSLocation):ptBoolean{
  postF{
    let TheResult: ptBoolean in true
    let EarthRadius: dtReal in
    let MaxDistance: dtReal in
```

```
33     let ComparedLatitude: dtLatitude in
34     let ComparedLongitude: dtLongitude in
35     let R1: dtReal in let R1a: dtReal in
36     let R2: dtReal in let R2a: dtReal in
37
38     ( if
39      ( EarthRadius.value = 6371
40       and MaxDistance.value = 100
41
42       and AdtValue.latitude = ComparedLatitude
43       and AdtValue.longitude = ComparedLongitude
44       and Self.latitude.sin() = R1a
45       and AdtValue.latitude.sin().mul(R1a) = R1
46       and Self.latitude.cos() = R2a
47       and AdtValue.latitude.cos().mul(R2a) = R2
48
49       and AdtValue.longitude = ComparedLongitude
50       and Self.longitude.sub(ComparedLongitude).cos().mul(R2)
51           .add(R1).acos().mul(EarthRadius).sub(MaxDistance)
52           .value.leq(0)
53      )
54      then (TheResult = true)
55      else (TheResult = false)
56      endif
57      result = TheResult
58     )
59   }
60   prolog{"src/Operations/Concepts/PrimaryTypesDatatypes/PrimaryTypesDatatypes-dtGPSLocation-isNearTo
       .pl"}
61   }
62   operation: icrash.concepts.primarytypes.datatypes.dtLatitude.is():ptBoolean{
63   postF{
64     let TheResult: ptBoolean in
65     ( if
66      ( AdtValue.value.geq(-90.0)
67       and AdtValue.value.leq(+90.0)
68      )
69      then (TheResult = true)
70      else (TheResult = false)
71      endif
72      result = TheResult
73     )}
74   prolog{ "src/Operations/Concepts/PrimaryTypesDatatypes/PrimaryTypesDatatypes-dtLatitude-is.pl"}
75   }
76   operation: icrash.concepts.primarytypes.datatypes.dtLongitude.is():ptBoolean{
77   postF{
78     let TheResult: ptBoolean in
79     ( if
80      ( AdtValue.value.geq(-180.0)
81       and AdtValue.value.leq(+180.0)
82      )
83      then (TheResult = true)
84      else (TheResult = false)
85      endif
86      result = TheResult
87     )}
88   prolog{ "src/Operations/Concepts/PrimaryTypesDatatypes/PrimaryTypesDatatypes-dtLongitude-is.pl"}
89  }
90 }
91 }
```

Listing C.49: Messir Spec. file primarytypes-datatypes-dtGPSLocation.msr.

C.50 File ./src-gen/messir-spec/operations/concepts/primarytypes-datatypes/primarytypes-datatypes-dtHospitalID.msr

```
1 package icrash.operations.concepts.primarytypes.datatypes.dtHospitalID {
2
```

```
import lu.uni.lassy.messir.libraries.primitives

Operation Model {
  operation: icrash.concepts.primarytypes.datatypes.dtHospitalID.is():ptBoolean{

  postF{
    let TheResult: ptBoolean in
    ( if
     ( AdtValue.value.length().gt(0)
      and AdtValue.value.length().leq(5)
     )
     then (TheResult = true)
     else (TheResult = false)
     endif
     result = TheResult
    )
  }
  prolog{"src/Operations/Concepts/PrimaryTypesDatatypes/PrimaryTypesDatatypes-dtHospitalID-is.pl"}
 }
}
}
```

Listing C.50: Messir Spec. file primarytypes-datatypes-dtHospitalID.msr.

C.51 File ./src-gen/messir-spec/operations/concepts/primarytypes-datatypes/primarytypes-datatypes-dtLogin.msr

```
package icrash.operations.concepts.primarytypes.datatypes.dtLogin{

import lu.uni.lassy.messir.libraries.primitives

Operation Model {

  operation: icrash.concepts.primarytypes.datatypes.dtLogin.is():ptBoolean{
  postF{
    let TheResult: ptBoolean in
    let MaxLength: ptInteger in
    ( if
     ( MaxLength = 20
      and AdtValue.value.length().leq(MaxLength)
     )
     then (TheResult = true)
     else (TheResult = false)
     endif
     result = TheResult
    )
    }
  prolog{"src/Operations/Concepts/PrimaryTypesDatatypes/PrimaryTypesDatatypes-dtLogin-is.pl"}
 }
}
}
```

Listing C.51: Messir Spec. file primarytypes-datatypes-dtLogin.msr.

C.52 File ./src-gen/messir-spec/operations/concepts/primarytypes-datatypes/primarytypes-datatypes-dtPassword.msr

```
package icrash.operations.concepts.primarytypes.datatypes.dtPassword{

import lu.uni.lassy.messir.libraries.primitives

Operation Model {

  operation: icrash.concepts.primarytypes.datatypes.dtPassword.is():ptBoolean{
  postF{
    let TheResult: ptBoolean in
```

```
10     let MinLength: ptInteger in
11     ( if
12      ( MinLength = 6
13       and AdtValue.value.length().geq(MinLength)
14      )
15      then (TheResult = true)
16      else (TheResult = false)
17      endif
18      result = TheResult
19     )
20 }
21   prolog{"src/Operations/Concepts/PrimaryTypesDatatypes/PrimaryTypesDatatypes-dtPassword-is.pl"}
22  }
23 }
24 }
```

Listing C.52: Messir Spec. file primarytypes-datatypes-dtPassword.msr.

C.53 File ./src-gen/messir-spec/operations/concepts/primarytypes-datatypes/primarytypes-datatypes-dtPhoneNumber.msr

```
1 package icrash.operations.concepts.primarytypes.datatypes.dtPhoneNumber{
2
3 import lu.uni.lassy.messir.libraries.primitives
4
5 Operation Model {
6
7   operation: icrash.concepts.primarytypes.datatypes.dtPhoneNumber.is():ptBoolean{
8
9   postF{
10   let TheResult: ptBoolean in
11   ( if
12    ( AdtValue.value.length().gt(4)
13     and AdtValue.value.length().leq(30)
14    )
15    then (TheResult = true)
16    else (TheResult = false)
17    endif
18    result = TheResult
19   )
20 }
21   prolog{"src/Operations/Concepts/PrimaryTypesDatatypes/PrimaryTypesDatatypes-dtPhoneNumber-is.pl"}
22  }
23 }
24 }
```

Listing C.53: Messir Spec. file primarytypes-datatypes-dtPhoneNumber.msr.

C.54 File ./src-gen/messir-spec/operations/concepts/primarytypes-datatypes/primarytypes-datatypes-etAlertStatus.msr

```
1 package icrash.operations.concepts.primarytypes.datatypes.etAlertStatus{
2
3 import lu.uni.lassy.messir.libraries.primitives
4
5 Operation Model {
6
7   operation: icrash.concepts.primarytypes.datatypes.etAlertStatus.is():ptBoolean{
8   postF{
9     let TheResult: ptBoolean in
10    ( if
11     ( self = pending
12      or self = valid
13      or self = invalid
14     )
15     then (TheResult = true)
```

```
16      else (TheResult = false)
17      endif
18      result = TheResult
19     )
20   }
21 prolog{"src/Operations/Concepts/PrimaryTypesClasses/PrimaryTypesDatatypes-etAlertStatus-is.pl"}
22 }
23 }
24 }
```

Listing C.54: Messir Spec. file primarytypes-datatypes-etAlertStatus.msr.

C.55 File ./src-gen/messir-spec/operations/concepts/primarytypes-datatypes/primarytypes-datatypes-etCorruptionKind.msr

```
1 package icrash.operations.concepts.primarytypes.datatypes.etCorruptionKind.msr {
2
3 import lu.uni.lassy.messir.libraries.primitives
4
5  Operation Model {
6   operation: icrash.concepts.primarytypes.datatypes.etCorruptionKind.is():ptBoolean{
7   postF{
8     let TheResult: ptBoolean in
9     ( if
10     ( self = regular
11      or self = corrupted
12      or self = restored
13     )
14     then (TheResult = true)
15     else (TheResult = false)
16     endif
17     result = TheResult
18     )
19   }
20 }
21  }
22
23 }
```

Listing C.55: Messir Spec. file primarytypes-datatypes-etCorruptionKind.msr.

C.56 File ./src-gen/messir-spec/operations/concepts/primarytypes-datatypes/primarytypes-datatypes-etCrisisStatus.msr

```
1 package icrash.operations.concepts.primarytypes.datatypes.etCrisisStatus{
2
3 import lu.uni.lassy.messir.libraries.primitives
4
5 Operation Model {
6
7   operation: icrash.concepts.primarytypes.datatypes.etCrisisStatus.is():ptBoolean{
8   postF{
9     let TheResult: ptBoolean in
10    ( if
11     ( self = pending
12      or self = handled
13      or self = solved
14      or self = closed
15     )
16     then (TheResult = true)
17     else (TheResult = false)
18     endif
19     result = TheResult
20     )
21   }
22   prolog{"src/Operations/Concepts/PrimaryTypesClasses/PrimaryTypesDatatypes-etCrisisStatus-is.pl"}
```

```
23 }
24 }
25 }
```

Listing C.56: Messir Spec. file primarytypes-datatypes-etCrisisStatus.msr.

C.57 File ./src-gen/messir-spec/operations/concepts/primarytypes-datatypes/primarytypes-datatypes-etCrisisType.msr

```
1 package icrash.operations.concepts.primarytypes.datatypes.etCrisisType{
2
3 import lu.uni.lassy.messir.libraries.primitives
4
5 Operation Model {
6
7   operation: icrash.concepts.primarytypes.datatypes.etCrisisType.is():ptBoolean{
8   postF{
9     let TheResult: ptBoolean in
10    ( if
11     ( self = small
12      or self = medium
13      or self = huge
14     )
15     then (TheResult = true)
16     else (TheResult = false)
17     endif
18     result = TheResult
19    )
20    }
21 prolog{"src/Operations/Concepts/PrimaryTypesClasses/PrimaryTypesDatatypes-etCrisisType-is.pl"}
22 }
23 }
24 }
```

Listing C.57: Messir Spec. file primarytypes-datatypes-etCrisisType.msr.

C.58 File ./src-gen/messir-spec/operations/concepts/primarytypes-datatypes/primarytypes-datatypes-etHumanKind.msr

```
1 package icrash.operations.concepts.primarytypes.datatypes.etHumanKind{
2
3 import lu.uni.lassy.messir.libraries.primitives
4
5 Operation Model {
6
7   operation: icrash.concepts.primarytypes.datatypes.etHumanKind.is():ptBoolean{
8   postF{
9     let TheResult: ptBoolean in
10    ( if
11     ( self = witness
12      or self = victim
13      or self = anonymous
14     )
15     then (TheResult = true)
16     else (TheResult = false)
17     endif
18     result = TheResult
19    )}
20 prolog{"src/Operations/Concepts/PrimaryTypesClasses/PrimaryTypesDatatypes-etHumanKind-is.pl"}
21 }
22 }
23 }
```

Listing C.58: Messir Spec. file primarytypes-datatypes-etHumanKind.msr.

C.59 File ./src-gen/messir-spec/operations/concepts/primarytypes-datatypes/primarytypes-datatypes-etInjuryKind.msr

```
package icrash.operations.concepts.primarytypes.datatypes.etInjuryKind {

import lu.uni.lassy.messir.libraries.primitives

Operation Model {

  operation: icrash.concepts.primarytypes.datatypes.etInjuryKind.is():ptBoolean{
  postF{
    let TheResult: ptBoolean in
    ( if
     ( self = arm
      or self = leg
      or self = chest
      or self = head
     )
     then (TheResult = true)
     else (TheResult = false)
     endif
     result = TheResult
    )
    }
prolog{"src/Operations/Concepts/PrimaryTypesClasses/PrimaryTypesDatatypes-etInjuryKind-is.pl"}
}
}
}
```

Listing C.59: Messir Spec. file primarytypes-datatypes-etInjuryKind.msr.

C.60 File ./src-gen/messir-spec/concepts/primarytypes-datatypes.msr

```
package icrash.concepts.primarytypes.datatypes {

import lu.uni.lassy.messir.libraries.primitives
import lu.uni.lassy.messir.libraries.string
import lu.uni.lassy.messir.libraries.math
import lu.uni.lassy.messir.libraries.calendar

 Concept Model {

  Primary Types {

    datatype dtAlertID extends dtString {
    operation is():ptBoolean
   }
   datatype dtVictimID extends dtString {
    operation is():ptBoolean
   }
   datatype dtInjuryID extends dtString {
    operation is():ptBoolean
   }
   datatype dtCrisisID extends dtString {
    operation is():ptBoolean
   }
   datatype dtLogin extends dtString {
    operation is():ptBoolean
   }
   datatype dtPassword extends dtString {
    operation is():ptBoolean
   }
   datatype dtCoordinatorID extends dtString {
    operation is():ptBoolean
   }
   datatype dtHospitalID extends dtString {
```

```
34     operation is():ptBoolean
35    }
36    datatype dtPhoneNumber extends dtString {
37     operation is():ptBoolean
38    }
39    datatype dtComment extends dtString {
40     operation is():ptBoolean
41    }
42    datatype dtLatitude extends dtReal {
43     operation is():ptBoolean
44    }
45    datatype dtLongitude extends dtReal {
46     operation is():ptBoolean
47    }
48    datatype dtGPSLocation {
49     attribute latitude: dtLatitude
50     attribute longitude: dtLongitude
51     operation is():ptBoolean
52     operation isNearTo(AGPSLocation:dtGPSLocation ):ptBoolean
53    }
54
55    enum etCrisisStatus {
56     constants["pending", "handled", "solved","closed"]
57     operation is():ptBoolean
58    }
59    enum etInjuryKind {
60     constants["head", "chest", "arm", "leg"]
61     operation is():ptBoolean
62    }
63    enum etAlertStatus {
64     constants["pending", "valid", "invalid"]
65     operation is():ptBoolean
66    }
67    enum etCrisisType {
68     constants["small", "medium", "huge"]
69     operation is():ptBoolean
70    }
71    enum etHumanKind {
72     constants["witness", "victim", "anonymous"]
73     operation is():ptBoolean
74    }
75    enum etCorruptionKind {
76     constants["corrupted", "restored", "regular"]
77     operation is():ptBoolean
78    }
79   }
80  }
81 }
```

Listing C.60: Messir Spec. file primarytypes-datatypes.msr.

C.61 File ./src-gen/messir-spec/operations/concepts/primarytypes-classes/ptimarytypes-classes-ctHospital.msr

```
1 package icrash.operations.concepts.primarytypes.classes.ctHospital.init {
2
3 import lu.uni.lassy.messir.libraries.primitives
4 import icrash.concepts.primarytypes.datatypes
5 import icrash.concepts.primarytypes.classes
6
7 Operation Model {
8
9 operation: icrash.concepts.primarytypes.classes.ctHospital.init(Aid:dtHospitalID, Alogin:dtLogin,
      Apwd:dtPassword):ptBoolean
10 {
11 postF{
12 if
13 (
```

```
/* Post F01 */
let Self:ctHospital in
Self.id = Aid
and Self.login = Alogin
and Self.pwd = Apwd
and Self.vpIsLogged = false
/* Post F02 */
and (Self.oclIsNew and self = Self)
)
then (result = true)
else (result = false)
endif}
prolog{"src/Operations/Concepts/PrimaryTypesClasses/PrimaryTypesClasses-ctHospital-init.pl"}
}
}
}
```

Listing C.61: Messir Spec. file ptimarytypes-classes-ctHospital.msr.

C.62 File ./src-gen/messir-spec/concepts/secondarytypes-associations.msr

```
package icrash.concepts.secondarytypes.associations {

 Concept Model {

  Secondary Types{

  }
 }
}
```

Listing C.62: Messir Spec. file secondarytypes-associations.msr.

C.63 File ./src-gen/messir-spec/concepts/secondarytypes-classes.msr

```
package icrash.concepts.secondarytypes.classes {

 Concept Model {

  Secondary Types{

  }
 }
}
```

Listing C.63: Messir Spec. file secondarytypes-classes.msr.

C.64 File ./src-gen/messir-spec/concepts/secondarytypes-datatypes.msr

```
package icrash.concepts.secondarytypes.datatypes {

import lu.uni.lassy.messir.libraries.primitives
import lu.uni.lassy.messir.libraries.string

import icrash.concepts.primarytypes.datatypes

 Concept Model {

  Secondary Types {

   datatype dtSMS {
```

```
13     attribute value: ptString
14     operation is():ptBoolean
15    }
16   }
17  }
18 }
```

Listing C.64: Messir Spec. file secondarytypes-datatypes.msr.

C.65 File ./src-gen/messir-spec/usecases/subfunctions-usecases.msr

```
1 package icrash.usecases.subfunctions {
2
3 import lu.uni.lassy.messir.libraries.primitives
4
5 import icrash.concepts.primarytypes.datatypes
6 import icrash.concepts.primarytypes.classes
7 import icrash.concepts.secondarytypes.datatypes
8 import lu.uni.lassy.messir.libraries.primitives
9 import lu.uni.lassy.messir.libraries.math
10 import lu.uni.lassy.messir.libraries.calendar
11
12 import icrash.environment
13
14  Use Case Model {
15 //-----------------------------------------------------
16  use case system subfunction oeAddVictim(AdtCrisisID:dtCrisisID){
17   actor actCoordinator[primary,active]
18   returned messages {
19    ieVictimAdded() returned to actCoordinator
20   }
21  }
22 //-----------------------------------------------------
23  use case system subfunction oeDeleteVictim(AdtVictimID:dtVictimID){
24   actor actCoordinator[primary,active]
25   returned messages {
26    ieVictimDeleted() returned to actCoordinator
27   }
28  }
29 //-----------------------------------------------------
30  use case system subfunction oeAddInjury(AdtVictimID:dtVictimID){
31   actor actCoordinator[primary,active]
32   returned messages {
33    ieInjuryAdded() returned to actCoordinator
34   }
35  }
36 //-----------------------------------------------------
37  use case system subfunction oeDeleteInjury(AdtInjuryID:dtInjuryID){
38   actor actCoordinator[primary,active]
39   returned messages {
40    ieInjuryDeleted() returned to actCoordinator
41   }
42  }
43 //-----------------------------------------------------
44  use case system subfunction oeAddCoordinator(AdtCoordinatorID:dtCoordinatorID, AdtLogin:dtLogin,
      AdtPassword:dtPassword){
45   actor actAdministrator[primary,active]
46   returned messages {
47    ieCoordinatorAdded() returned to actAdministrator
48   }
49  }
50 //-----------------------------------------------------
51 use case system subfunction oeAddHospital(AdtHospitalID:dtHospitalID, AdtLogin:dtLogin, AdtPassword:
      dtPassword){
52  actor actAdministrator[primary,active]
53  returned messages {
54   ieHospitalAdded() returned to actAdministrator
55  }
56 }
```

```
57 //-----------------------------------------------------
58 use case system subfunction oeAlert(
59                  AetKind:etHumanKind,
60                  AdtMyDate:dtDate,
61                  AdtTime:dtTime,
62                  AdtPhoneNumber:dtPhoneNumber,
63                  AdtGPSLocation:dtGPSLocation,
64                  AdtComment:dtComment) {
65  actor actComCompany[primary,active]
66  returned messages {
67   ieSmsSend(AdtPhoneNumber,AdtSMS) returned to actComCompany
68   }
69 }
70 //-----------------------------------------------------
71 use case system subfunction oeInvalidateAlert(AdtAlertID:dtAlertID) {
72  actor actCoordinator[primary,active]
73  actor actComCompany[secondary,passive]
74  returned messages {
75   ieMessage(AMessage) returned to actCoordinator
76  }
77 }
78 //-----------------------------------------------------
79 use case system subfunction oeCloseCrisis(AdtCrisisID:dtCrisisID) {
80  actor actCoordinator[primary,active]
81  returned messages {
82   ieMessage(AMessage) returned to actCoordinator
83  }  }
84 //-----------------------------------------------------
85 use case system subfunction oeCreateSystemAndEnvironment(AqtyComCompanies:ptInteger) {
86  actor actMsrCreator[primary,active]
87 }
88 //-----------------------------------------------------
89 use case system subfunction oeDeleteCoordinator(AdtCoordinatorID:dtCoordinatorID) {
90  actor actAdministrator[primary,active]
91  returned messages {
92   ieCoordinatorDeleted() returned to actAdministrator
93  }
94 }
95 //-----------------------------------------------------
96 use case system subfunction oeDeleteHospital(AdtHospitalID:dtHospitalID) {
97  actor actAdministrator[primary,active]
98  returned messages {
99   ieHospitalDeleted() returned to actAdministrator
100  }
101 }
102 //-----------------------------------------------------
103 use case system subfunction oeGetAlertsSet(AetAlertStatus:etAlertStatus) {
104  actor actCoordinator[primary,active]
105  returned messages {
106   ieSendAnAlert(ActAlert) returned to actCoordinator
107  }
108 }
109 //-----------------------------------------------------
110 use case system subfunction oeGetCrisisSet(AetCrisisStatus:etCrisisStatus) {
111 actor actCoordinator[primary,active]
112 returned messages {
113   ieSendACrisis(ActCrisis) returned to actCoordinator
114  }
115 }
116 //-----------------------------------------------------
117 use case system subfunction oeGetHospitalCrisisSet(AetCrisisStatus:etCrisisStatus){
118  actor actHospital[primary,active]
119  returned messages {
120   ieSendACrisisToHospital(ActCrisis) returned to actHospital
121  }
122 }
123 //-----------------------------------------------------
124 use case system subfunction oeGetHospitalVictimSet(AdtCrisisID:dtCrisisID){
125  actor actHospital[primary,active]
126  returned messages {
```

```
127    ieSendAVictimToHospital(ActVictim) returned to actHospital
128   }
129  }
130 //-----------------------------------------------------
131  use case system subfunction oeGetVictimSet(AdtCrisisID:dtCrisisID){
132   actor actCoordinator[primary,active]
133   returned messages {
134    ieSendAVictim(ActVictim) returned to actCoordinator
135   }
136  }
137 //-----------------------------------------------------
138  use case system subfunction oeGetHospitalInjurySet(AdtVictimID:dtVictimID){
139   actor actHospital[primary,active]
140   returned messages {
141    ieSendAInjuryToHospital(ActInjury) returned to actHospital
142   }
143  }
144 //-----------------------------------------------------
145  use case system subfunction oeGetInjurySet(AdtVictimID:dtVictimID){
146   actor actCoordinator[primary,active]
147   returned messages {
148    ieSendAInjury(ActInjury) returned to actCoordinator
149   }
150  }
151 //-----------------------------------------------------
152  use case system subfunction oeSetCrisisHandler(AdtCrisisID:dtCrisisID) {
153   actor actCoordinator[primary,active]
154   actor actCoordinator[secondary,passive]
155   actor actComCompany[secondary,passive,multiple]
156   returned messages {
157    ieMessage(AMessage)
158     returned to actCoordinator
159    ieSendAnAlert(ActAlert)
160     returned to actCoordinator
161    ieSmsSend(AdtPhoneNumber,AdtSMS)
162     returned to actComCompany
163   }
164  }
165 //-----------------------------------------------------
166  use case system subfunction oeLogin(AdtLogin:dtLogin , AdtPassword:dtPassword) {
167   actor actAuthenticated[primary,active]
168   returned messages {
169    ieMessage(AMessage) returned to actAuthenticated
170   }
171  }
172 //-----------------------------------------------------
173  use case system subfunction oeLogout() {
174   actor actAuthenticated[primary,active]
175   returned messages {
176    ieMessage(AMessage) returned to actAuthenticated
177   }
178  }
179 //-----------------------------------------------------
180  use case system subfunction oeReportOnCrisis(AdtCrisisID:dtCrisisID,AdtComment:dtComment) {
181   actor actCoordinator[primary,active]
182   returned messages {
183    ieMessage(AMessage) returned to actCoordinator
184   }
185  }
186 //-----------------------------------------------------
187  use case system subfunction oeSetClock(AcurrentClock:dtDateAndTime) {
188   actor actActivator[primary,proactive]
189  }
190 //-----------------------------------------------------
191  use case system subfunction oeSetCrisisStatus(AdtCrisisID:dtCrisisID ,AetCrisisStatus:
       etCrisisStatus) {
192   actor actCoordinator[primary,active]
193   returned messages {
194    ieMessage(AMessage) returned to actCoordinator
195   }
```

```
196 }
197 //-----------------------------------------------------
198 use case system subfunction oeSollicitateCrisisHandling() {
199  actor actActivator[primary,proactive]
200  actor actCoordinator[secondary,passive,multiple]
201  actor actAdministrator[secondary,passive]
202  returned messages {
203   ieMessage(AMessage) returned to actCoordinator
204  }
205 }
206 //-----------------------------------------------------
207 use case system subfunction oeValidateAlert(AdtAlertID:dtAlertID) {
208  actor actCoordinator[primary,active]
209  returned messages {
210   ieMessage(AMessage) returned to actCoordinator
211  }
212 }
213 //-----------------------------------------------------
214 use case system subfunction oeCorruptAlert(ActAlert:ctAlert) {
215  actor actCorrupter[primary,active]
216  returned messages {
217  }
218 }
219 }
220
221 }
```

Listing C.65: Messir Spec. file subfunctions-usecases.msr.

C.66 File ./src-gen/messir-spec/test/tc-testcase01.msr

```
1 package lu.uni.lassy.excalibur.examples.icrash.tests.testcase01 {
2
3 import lu.uni.lassy.messir.libraries.string
4 import lu.uni.lassy.messir.libraries.primitives
5 import lu.uni.lassy.messir.libraries.math
6 import lu.uni.lassy.messir.libraries.calendar
7
8 import icrash.concepts.primarytypes.associations
9 import icrash.concepts.primarytypes.classes
10 import icrash.concepts.primarytypes.datatypes
11 import icrash.concepts.secondarytypes.datatypes
12 import icrash.environment
13
14 Test Model{
15  test case testcase01 order 01 {
16 //-------------------------------------------------------------------
17   test step ts01oeCreateSystemAndEnvironment order 01 {
18    variables{
19     Creator:actMsrCreator
20     AqtyComCompanies: ptInteger
21     }
22    constraints{
23      AqtyComCompanies = 4
24     }
25    test message{
26     out:Creator sends to system actMsrCreator.outactMsrCreator.oeCreateSystemAndEnvironment(
      AqtyComCompanies)
27     }
28    oracle{
29     constraints{
30      true
31     }
32    }
33    prolog{"src/Tests/system/01/system-sim-01-01-oeCreateSystemAndEnvironment.pl"}
34   }
35 //-------------------------------------------------------------------
36 test step ts02oeSetClock order 02{
37     variables{
```

```
38      TheActor:actActivator
39      ACurrentClock:dtDateAndTime
40      }
41     constraints{
42      TheActor=TheSystem.rnactActivator->any2(true)
43
44      ACurrentClock.date.year.value = 2017
45      ACurrentClock.date.month.value = 11
46      ACurrentClock.date.day.value = 24
47      ACurrentClock.time.hour.value = 15
48      ACurrentClock.time.minute.value = 20
49      ACurrentClock.time.second.value = 00
50     }
51 test message{
52     out:TheActor sends to system actActivator.outactActivator.oeSetClock(ACurrentClock)
53    }
54  oracle{
55    constraints{
56     true
57    }
58   }
59   }
60 //-----------------------------------------------------------------------
61
62 test step ts03oeLogin order 03{
63    variables{
64      TheActor : actAdministrator
65      AdtLogin:dtLogin
66      AdtPassword:dtPassword
67     }
68    constraints{
69      TheActor=TheSystem.rnactAdministrator->any2(true)
70      AdtLogin.value.eq('icrashadmin')
71      AdtPassword.value.eq('7WXC1359')
72     }
73    test message{
74     out:TheActor sends to system actAdministrator.outactAdministrator.oeLogin(AdtLogin,AdtPassword)
75    }
76   oracle{
77    variables{
78     AMessage:ptString
79    }
80    constraints{
81     AMessage = 'You are logged ! Welcome ...'
82     TheActor.inactAdministrator.ieMessage(AMessage)
83    }
84   }
85   }
86 //-----------------------------------------------------------------------
87 test step ts04oeAddCoordinator order 04{
88    variables{
89      TheActor : actAdministrator
90      AdtCoordinatorID : dtCoordinatorID
91      AdtLogin:dtLogin
92      AdtPassword:dtPassword
93     }
94    constraints{
95      TheActor = TheSystem.rnactAdministrator->any2(true)
96      AdtCoordinatorID.value.eq('1')
97      AdtLogin.value.eq('steve')
98      AdtPassword.value.eq('pwdMessirExcalibur2017')
99     }
100    test message{
101     out:TheActor
102     sends to system actAdministrator.outactAdministrator.oeAddCoordinator
103                                (AdtCoordinatorID,
104                                 AdtLogin,
105                                 AdtPassword)
106    }
107    oracle{
```

```
108      constraints{
109        TheActor.inactAdministrator.ieCoordinatorAdded()
110      }
111    }
112    }
113 //-----------------------------------------------------------------
114  test step ts05oeLogout order 05{
115     variables{
116       TheActor : actAdministrator
117      }
118     constraints{
119       TheActor = TheSystem.rnactAdministrator->any2(true)
120      }
121     test message{
122      out:TheActor sends to system actAdministrator.outactAdministrator.oeLogout()
123     }
124     oracle{
125      variables{
126       AMessage:ptString
127      }
128      constraints{
129      AMessage = 'You are logged out ! Good Bye ...'
130      TheActor.inactAdministrator.ieMessage(AMessage)
131     }
132    }
133   }
134 //-----------------------------------------------------------------
135  test step ts06oeSetClock02 order 06{
136     variables{
137       TheActor:actActivator
138       ACurrentClock:dtDateAndTime
139      }
140     constraints{
141       TheActor=TheSystem.rnactActivator->any2(true)
142       ACurrentClock.date.year.value = 2017
143       ACurrentClock.date.month.value = 11
144       ACurrentClock.date.day.value = 26
145       ACurrentClock.time.hour.value = 10
146       ACurrentClock.time.minute.value = 15
147       ACurrentClock.time.second.value = 00
148      }
149     test message{
150      out:TheActor sends to system actActivator.outactActivator.oeSetClock(ACurrentClock)
151     }
152     oracle{
153      constraints{
154      true
155     }
156    }
157    }
158 //-----------------------------------------------------------------
159  test step ts07oeAlert1 order 07{
160     variables{
161       TheActor : actComCompany
162       AetHumanKind:etHumanKind
163       AdtDate:dtDate
164       AdtTime:dtTime
165       AdtPhoneNumber:dtPhoneNumber
166       AdtGPSLocation:dtGPSLocation
167       AdtComment:dtComment
168      }
169     constraints{
170       TheActor = TheSystem.rnactComCompany->any2(true)
171       AetHumanKind = witness
172       AdtDate.year.value = 2017
173       AdtDate.month.value = 11
174       AdtDate.day.value = 26
175       AdtTime.hour.value = 10
176       AdtTime.minute.value = 10
177       AdtTime.second.value = 16
```

```
178       AdtPhoneNumber.value = '+3524666445252'
179       AdtGPSLocation.latitude.value = 49.627675
180       AdtGPSLocation.longitude.value = 6.159590
181       AdtComment.value = '3 cars involved in an accident.'
182      }
183    test message{
184     out:TheActor
185     sends to system actComCompany.outactComCompany.oeAlert( AetHumanKind,
186                                 AdtDate,
187                                 AdtTime,
188                                 AdtPhoneNumber,
189                                 AdtGPSLocation,
190                                 AdtComment)
191    }
192    oracle{
193     variables{
194       AdtSMS:dtSMS
195     }
196     constraints{
197     AdtSMS.value = 'Your alert has been registered. We will handle it and keep you informed'
198     TheActor.inactComCompany.ieSmsSend(AdtPhoneNumber,AdtSMS)
199    }
200   }
201   }
202 //-----------------------------------------------------------------------
203 test step ts08oeSetClock03 order 08{
204    variables{
205      TheActor:actActivator
206      ACurrentClock:dtDateAndTime
207     }
208   constraints{
209      TheActor=TheSystem.rnactActivator->any2(true)
210      ACurrentClock.date.year.value = 2017
211      ACurrentClock.date.month.value = 11
212      ACurrentClock.date.day.value = 26
213      ACurrentClock.time.hour.value = 10
214      ACurrentClock.time.minute.value = 30
215      ACurrentClock.time.second.value = 00
216     }
217    test message{
218     out:TheActor sends to system actActivator.outactActivator.oeSetClock(ACurrentClock)
219    }
220    oracle{
221     constraints{
222      true
223     }
224   }
225   }
226 //-----------------------------------------------------------------------
227 test step ts09oeSollicitateCrisisHandling order 09{
228    variables{
229      TheActor : actActivator
230     }
231    constraints{
232      TheActor = TheSystem.rnactActivator->any2(true)
233     }
234    test message{
235     out:TheActor sends to system actActivator.outactActivator.oeSollicitateCrisisHandling()
236    }
237    oracle{
238    variables{
239      TheAdministrator:actAdministrator
240      TheCoordinator:actCoordinator
241      AMessageForCrisisHandlers:ptString
242     }
243     constraints{
244     TheAdministrator = TheSystem.rnactAdministrator->any2(true)
245     TheCoordinator = TheSystem.rnactCoordinator->any2(true)
246     AMessageForCrisisHandlers = 'There are alerts pending since more than the defined delay. Please
        REACT !'
```

```
247
248     TheAdministrator.inactAdministrator.ieMessage(AMessageForCrisisHandlers)
249     TheCoordinator.inactAdministrator.ieMessage(AMessageForCrisisHandlers)
250
251 /* this oracle should be written like this:
252
253    oracle{
254    variables{
255      TheAdministrator:actAdministrator
256      AMessageForCrisisHandlers:ptString
257     }
258     constraints{
259     AMessageForCrisisHandlers = 'There are alerts pending since more than the defined delay. Please
        REACT !'
260     TheAdministrator = TheSystem.rnactAdministrator->any2(true)
261
262     TheSystem.rnactCoordinator->forAll(TheCoordinator:actCoordinator | TheCoordinator.
       actAuthenticated.inactAuthenticated.ieMessage(AMessage))
263
264     // receives from system is for step instances
265
266  */
267    }
268   }
269   }
270 //-----------------------------------------------------------------
271  test step ts10oeLogin02 order 10{
272    variables{
273      TheActor : actCoordinator
274      AdtLogin:dtLogin
275      AdtPassword:dtPassword
276     }
277    constraints{
278      TheActor = TheSystem.rnactCoordinator->select(a | a.rnctCoordinator.login.value.eq('steve'))->
       any2(true)
279      AdtLogin.value.eq('steve')
280      AdtPassword.value.eq('pwdMessirExcalibur2017')
281     }
282    test message{
283     out:TheActor sends to system actAuthenticated.outactAuthenticated.oeLogin(AdtLogin,AdtPassword)
284    }
285    oracle{
286     variables{
287       AMessage:ptString
288     }
289     constraints{
290     AMessage = 'You are logged ! Welcome ...'
291     TheActor.inactAuthenticated.ieMessage(AMessage)
292    }
293   }
294   }
295 //-----------------------------------------------------------------
296  test step ts11oeGetCrisisSet order 11{
297    variables{
298      TheActor : actCoordinator
299      AetCrisisStatus : etCrisisStatus
300     }
301    constraints{
302      TheActor=TheSystem.rnactCoordinator
303      ->select(a | a.rnctCoordinator.login.value.eq('steve'))
304      ->any2(true)
305      AetCrisisStatus = pending
306     }
307    test message{
308     out:TheActor sends to system actCoordinator.outactCoordinator.oeGetCrisisSet(AetCrisisStatus)
309    }
310    oracle{
311 //TODO - make consistent with test step implementation by adding Prolog code for input messages
312    variables{
313      ActCrisis:ctCrisis
```

```
314     }
315     constraints{
316      TheActor.inactCoordinator.ieSendACrisis(ActCrisis)
317     }
318    }
319    }
320 //-----------------------------------------------------------------------
321  test step ts12oeSetCrisisHandler order 12{
322     variables{
323       TheActor : actCoordinator
324       AdtCrisisID : dtCrisisID
325      }
326     constraints{
327       TheActor=TheSystem.rnactCoordinator
328       ->select(a | a.rnctCoordinator.login.value.eq('steve'))
329       ->any2(true)
330       //and AdtCrisisID.value= '1'
331      }
332     test message{
333      out:TheActor sends to system actCoordinator.outactCoordinator.oeSetCrisisHandler(AdtCrisisID)
334     }
335     oracle{
336      variables{
337       AMessage:ptString
338       AdtPhoneNumber:dtPhoneNumber
339       AdtSMS:dtSMS
340       ActAlert:ctAlert
341
342       TheComCompany: actComCompany
343       TheCoordinator:actCoordinator
344      }
345      constraints{
346       AMessage = 'You are now considered as handling the crisis !'
347       AdtSMS.value = 'The handling of your alert by our services is in progress !'
348       TheComCompany.inactComCompany.ieSmsSend(AdtPhoneNumber,AdtSMS)
349       TheCoordinator.inactCoordinator.ieSendAnAlert(ActAlert)
350       TheActor.inactAuthenticated.ieMessage(AMessage)
351     }
352    }
353    }
354 //-----------------------------------------------------------------------
355  test step ts13oeSetClock04 order 13{
356     variables{
357       TheActor:actActivator
358       ACurrentClock:dtDateAndTime
359      }
360     constraints{
361       TheActor=TheSystem.rnactActivator->any2(true)
362       ACurrentClock.date.year.value = 2017
363       ACurrentClock.date.month.value = 11
364       ACurrentClock.date.day.value = 26
365       ACurrentClock.time.hour.value = 10
366       ACurrentClock.time.minute.value = 45
367       ACurrentClock.time.second.value = 00
368      }
369     test message{
370      out:TheActor sends to system actActivator.outactActivator.oeSetClock(ACurrentClock)
371     }
372     oracle{
373      constraints{
374      true
375     }
376    }
377    }
378 //-----------------------------------------------------------------------
379  test step ts14oeValidateAlert order 14{
380     variables{
381       TheActor : actCoordinator
382       AdtAlertID : dtAlertID
383      }
```

```
    constraints{
      TheActor=TheSystem.rnactCoordinator
      ->select(a | a.rnctCoordinator.login.value.eq('steve'))
      ->any2(true)
      //and AdtAlertID.value= '1'
     }
    test message{
     out:TheActor sends to system actCoordinator.outactCoordinator.oeValidateAlert(AdtAlertID)
    }
    oracle{
     variables{
       AMessage:ptString
     }
     constraints{
     AMessage = 'The Alert is now declared as valid !'
     TheActor.actAuthenticated.inactAuthenticated.ieMessage(AMessage)
    }
   }
   }
//-----------------------------------------------------------------------
 test step ts15oeAlert2 order 15{
    variables{
      TheActor : actComCompany
      AetHumanKind:etHumanKind
      AdtDate:dtDate
      AdtTime:dtTime
      AdtPhoneNumber:dtPhoneNumber
      AdtGPSLocation:dtGPSLocation
      AdtComment:dtComment
     }
    constraints{
      TheActor = TheSystem.rnactComCompany->any2(true)
      AetHumanKind = witness
      AdtDate.year.value = 2017
      AdtDate.month.value = 11
      AdtDate.day.value = 26
      AdtTime.hour.value = 10
      AdtTime.minute.value = 20
      AdtTime.second.value = 00
      AdtPhoneNumber.value = '+3524666445000'
      AdtGPSLocation.latitude.value = 49.627095
      AdtGPSLocation.longitude.value = 6.160251
      AdtComment.value = 'A car crash just happened.'
     }
    test message{
     out:TheActor
     sends to system actComCompany.outactComCompany.oeAlert( AetHumanKind,
                                AdtDate,
                                AdtTime,
                                AdtPhoneNumber,
                                AdtGPSLocation,
                                AdtComment)
    }
    oracle{
     variables{
      AdtSMS:dtSMS
     }
     constraints{
     AdtSMS.value = 'Your alert has been registered. We will handle it and keep you informed'
     TheActor.actComCompany.inactComCompany.ieSmsSend(AdtPhoneNumber,AdtSMS)
    }
    }
   }
//-----------------------------------------------------------------------
 test step ts16oeSetClock05 order 16{
    variables{
      TheActor:actActivator
      ACurrentClock:dtDateAndTime
     }
    constraints{
```

```
454       TheActor=TheSystem.rnactActivator->any2(true)
455       ACurrentClock.date.year.value = 2017
456       ACurrentClock.date.month.value = 11
457       ACurrentClock.date.day.value = 26
458       ACurrentClock.time.hour.value = 12
459       ACurrentClock.time.minute.value = 45
460       ACurrentClock.time.second.value = 00
461      }
462     test message{
463      out:TheActor sends to system actActivator.outactActivator.oeSetClock(ACurrentClock)
464     }
465     oracle{
466      constraints{
467      true
468     }
469    }
470    }
471 //-----------------------------------------------------------------------
472  test step ts17oeSetCrisisStatus order 17{
473     variables{
474       TheActor : actCoordinator
475       AdtCrisisID : dtCrisisID
476       AetCrisisStatus : etCrisisStatus
477      }
478     constraints{
479       TheActor=TheSystem.rnactCoordinator
480       ->select(a | a.rnctCoordinator.login.value.eq('steve'))
481       ->any2(true)
482       //and AdtCrisisID.value= '1'
483       //and AetCrisisStatus = solved
484      }
485     test message{
486      out:TheActor sends to system actCoordinator.outactCoordinator.oeSetCrisisStatus(AdtCrisisID,
     AetCrisisStatus)
487     }
488     oracle{
489      variables{
490       AMessage:ptString
491      }
492      constraints{
493      AMessage = 'The crisis status has been updated !'
494      TheActor.inactAuthenticated.ieMessage(AMessage)
495     }
496    }
497    }
498 //-----------------------------------------------------------------------
499  test step ts18oeReportOnCrisis order 18{
500     variables{
501       TheActor : actCoordinator
502       AdtCrisisID : dtCrisisID
503       AdtComment : dtComment
504      }
505     constraints{
506       TheActor=TheSystem.rnactCoordinator
507       ->select(a | a.rnctCoordinator.login.value.eq('steve'))
508       ->any2(true)
509       //and AdtCrisisID.value= '1'
510       //and AdtComment.value = '3 victims sent to hospital, 2 cars evacuated and 4 rescue unit
     mobilized'
511      }
512     test message{
513      out:TheActor sends to system actCoordinator.outactCoordinator.oeReportOnCrisis(AdtCrisisID,
     AdtComment)
514     }
515     oracle{
516      variables{
517       AMessage:ptString
518      }
519      constraints{
520      AMessage = 'The crisis comment has been updated !'
```

```
521      TheActor.inactAuthenticated.ieMessage(AMessage)
522     }
523    }
524    }
525 //-----------------------------------------------------------------
526  test step ts19oeCloseCrisis order 19{
527     variables{
528       TheActor : actCoordinator
529       AdtCrisisID : dtCrisisID
530      }
531     constraints{
532       TheActor=TheSystem.rnactCoordinator
533       ->select(a | a.rnctCoordinator.login.value.eq('steve'))
534       ->any2(true)
535       //and AdtCrisisID.value= '1'
536      }
537     test message{
538      out:TheActor sends to system actCoordinator.outactCoordinator.oeCloseCrisis(AdtCrisisID)
539     }
540     oracle{
541      variables {
542       AMessage:ptString
543      }
544      constraints{
545      AMessage = 'The crisis is now closed !'
546      TheActor.inactAuthenticated.ieMessage(AMessage)
547     }
548     }
549    }
550   }
551  }
552 }
```

Listing C.66: Messir Spec. file tc-testcase01.msr.

C.67 File ./src-gen/messir-spec/test/tci-testcase01-instance01.msr

```
 1 package lu.uni.lassy.excalibur.examples.icrash.tests.testcase01.instance01 {
 2
 3 import lu.uni.lassy.messir.libraries.string
 4 import lu.uni.lassy.messir.libraries.primitives
 5 import lu.uni.lassy.messir.libraries.math
 6 import lu.uni.lassy.messir.libraries.calendar
 7
 8 import icrash.concepts.primarytypes.associations
 9 import icrash.concepts.primarytypes.classes
10 import icrash.concepts.primarytypes.datatypes
11 import lu.uni.lassy.excalibur.examples.icrash.tests.testcase01
12 import icrash.environment
13
14 Test Model {
15  test case instance instance01:testcase01{
16 //-----------------------------------------------------
17   test step instance tsi01:testcase01.ts01oeCreateSystemAndEnvironment{
18    variables {
19     theCreator:testcase01.ts01oeCreateSystemAndEnvironment.Creator = "theCreator"
20     AqtyComCompanies : testcase01.ts01oeCreateSystemAndEnvironment.AqtyComCompanies="4"
21     }
22     oracle {
23      satisfaction = "true"
24     }
25     test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
26    }
27 //-----------------------------------------------------
28   test step instance tsi02: testcase01.ts02oeSetClock{
29    variables {
30     theClock:testcase01.ts02oeSetClock.TheActor = "theClock"
31     ACurrentClock : testcase01.ts02oeSetClock.ACurrentClock= "2017:11:24 - 03:20:00"
32     }
```

```
33     oracle {
34      satisfaction = "true"
35     }
36     test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
37    }
38 //-----------------------------------------------------
39   test step instance tsi03: testcase01.ts03oeLogin{
40    variables {
41     bill:testcase01.ts03oeLogin.TheActor="bill"
42     AdtLogin : testcase01.ts03oeLogin.AdtLogin= "icrashadmin"
43     AdtPassword : testcase01.ts03oeLogin.AdtPassword= "7WXC1359"
44     }
45     oracle {
46      satisfaction = "true"
47      received message {
48       AMessage : testcase01.ts03oeLogin.AMessage= 'You are logged ! Welcome ...'
49       tsi03.bill received from system actAuthenticated.inactAuthenticated.ieMessage(AMessage)
50      }
51     }
52     test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
53    }
54 //-----------------------------------------------------
55   test step instance tsi04: testcase01.ts04oeAddCoordinator{
56    variables {
57     reuse tsi03.bill as testcase01.ts04oeAddCoordinator.TheActor
58     AdtCoordinatorID : testcase01.ts04oeAddCoordinator.AdtCoordinatorID = "1"
59     AdtLogin :testcase01.ts04oeAddCoordinator.AdtLogin= "steve"
60     AdtPassword : testcase01.ts04oeAddCoordinator.AdtPassword = "pwdMessirExcalibur2017"
61    }
62    oracle {
63      satisfaction = "true"
64      received message {
65      tsi03.bill received from system actAdministrator.inactAdministrator.ieCoordinatorAdded()
66      }
67    }
68     test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
69    }
70 //-----------------------------------------------------
71   test step instance tsi05: testcase01.ts05oeLogout{
72    variables {
73     reuse tsi03.bill as testcase01.ts05oeLogout.TheActor
74    }
75    oracle {
76      satisfaction = "true"
77      received message {
78      AMessage : testcase01.ts05oeLogout.AMessage= 'You are logged out ! Good Bye ...'
79      tsi03.bill received from system actAuthenticated.inactAuthenticated.ieMessage(AMessage)
80      }
81    }
82     test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
83    }
84 //-----------------------------------------------------
85   test step instance tsi06: testcase01.ts06oeSetClock02{
86    variables {
87     reuse tsi02.theClock as testcase01.ts06oeSetClock02.TheActor
88     ACurrentClock : testcase01.ts06oeSetClock02.ACurrentClock= "2017:11:26 - 10:15:00"
89    }
90    oracle {
91      satisfaction = "true"
92    }
93     test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
94    }
95 //-----------------------------------------------------
96   test step instance tsi07: testcase01.ts07oeAlert1{
97    variables {
98     tango:testcase01.ts07oeAlert1.TheActor ="tango"
99     AetHumanKind : testcase01.ts07oeAlert1.AetHumanKind = "witness"
100    AdtDate : testcase01.ts07oeAlert1.AdtDate = "2017:11:26"
101    AdtTime : testcase01.ts07oeAlert1.AdtTime = "10:10:16"
102    AdtPhoneNumber : testcase01.ts07oeAlert1.AdtPhoneNumber = "+3524666445252"
```

```
103     AdtGPSLocation : testcase01.ts07oeAlert1.AdtGPSLocation = "49.627675:6.159590"
104     AdtComment : testcase01.ts07oeAlert1.AdtComment = "3 cars involved in an accident."
105     }
106    oracle {
107      satisfaction = "true"
108      received message {
109      AdtSMS : testcase01.ts07oeAlert1.AdtSMS= 'Your alert has been registered. We will handle it and
         keep you informed'
110      tsi07.tango received from system actComCompany.inactComCompany.ieSmsSend(AdtPhoneNumber,AdtSMS)
111
112      }
113    }
114     test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
115    }
116
117 //-----------------------------------------------------
118   test step instance tsi08: testcase01.ts08oeSetClock03{
119    variables {
120     reuse tsi02.theClock as testcase01.ts08oeSetClock03.ACurrentClock
121     ACurrentClock : testcase01.ts08oeSetClock03.ACurrentClock = "2017:11:26 - 10:30:00"
122    }
123    oracle {
124      satisfaction = "true"
125    }
126     test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
127    }
128 //-----------------------------------------------------
129   test step instance tsi09: testcase01.ts09oeSollicitateCrisisHandling{
130    variables {
131     reuse tsi02.theClock as testcase01.ts09oeSollicitateCrisisHandling.TheActor
132     steve:testcase01.ts09oeSollicitateCrisisHandling.TheCoordinator ="steve"
133     reuse tsi03.bill as testcase01.ts09oeSollicitateCrisisHandling.TheAdministrator
134    }
135    oracle {
136      satisfaction = "true"
137      received message {
138       AMessageForCrisisHandlers : testcase01.ts09oeSollicitateCrisisHandling.
         AMessageForCrisisHandlers= 'There are alerts pending since more than the defined delay. Please
         REACT !'
139
140      tsi03.bill received from system actAuthenticated.inactAuthenticated.ieMessage(
         AMessageForCrisisHandlers)
141      tsi09.steve received from system actAuthenticated.inactAuthenticated.ieMessage(
         AMessageForCrisisHandlers)
142            }
143    }
144     test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
145    }
146
147 //-----------------------------------------------------
148   test step instance tsi10: testcase01.ts10oeLogin02{
149    variables {
150     reuse tsi09.steve as testcase01.ts10oeLogin02.TheActor
151     AdtLogin : testcase01.ts10oeLogin02.AdtLogin = "steve"
152     AdtPassword : testcase01.ts10oeLogin02.AdtPassword= "pwdMessirExcalibur2017"
153    }
154    oracle {
155      satisfaction = "true"
156      received message {
157      AMessage : testcase01.ts10oeLogin02.AMessage= 'You are logged ! Welcome ...'
158      tsi09.steve received from system actAuthenticated.inactAuthenticated.ieMessage(AMessage)
159
160      }
161    }
162    test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
163    }
164 //-----------------------------------------------------
165   test step instance tsi11: testcase01.ts11oeGetCrisisSet{
166    variables {
167     reuse tsi09.steve as testcase01.ts11oeGetCrisisSet.TheActor
```

```
168     AetCrisisStatus : testcase01.ts11oeGetCrisisSet.AetCrisisStatus = "pending"
169    }
170    oracle {
171      satisfaction = "true"
172      received message {
173      ActCrisis : testcase01.ts11oeGetCrisisSet.ActCrisis= "crisis with ID 1 details"
174      tsi09.steve received from system actCoordinator.inactCoordinator.ieSendACrisis(ActCrisis)
175      }
176    }
177    test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
178    }
179 //-----------------------------------------------------
180   test step instance tsi12: testcase01.ts12oeSetCrisisHandler{
181    variables {
182     reuse tsi09.steve as testcase01.ts12oeSetCrisisHandler.TheActor
183     AdtCrisisID : testcase01.ts12oeSetCrisisHandler.AdtCrisisID = "1"
184
185      reuse tsi07.tango as testcase01.ts12oeSetCrisisHandler.TheComCompany
186
187    }
188    oracle {
189      satisfaction = "true"
190      received message {
191      AMessage : testcase01.ts12oeSetCrisisHandler.AMessage= 'You are now considered as handling the
       crisis !'
192      AdtSMS : testcase01.ts12oeSetCrisisHandler.AdtSMS= 'The handling of your alert by our services
           is in progress !'
193      AdtPhoneNumber : testcase01.ts12oeSetCrisisHandler.AdtPhoneNumber= "+3524666445252"
194
195      tsi07.tango received from system actComCompany.inactComCompany.ieSmsSend(AdtPhoneNumber,AdtSMS)
196      tsi09.steve received from system actAuthenticated.inactAuthenticated.ieMessage(AMessage)
197
198      }
199    }
200     test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
201    }
202 //-----------------------------------------------------
203   test step instance tsi13: testcase01.ts13oeSetClock04{
204    variables {
205     reuse tsi02.theClock as testcase01.ts13oeSetClock04.TheActor
206      ACurrentClock : testcase01.ts13oeSetClock04.ACurrentClock = "2017:11:26 - 10:45:00"
207    }
208    oracle {
209      satisfaction = "true"
210    }
211    test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
212    }
213 //-----------------------------------------------------
214   test step instance tsi14: testcase01.ts14oeValidateAlert{
215    variables {
216     reuse tsi09.steve as testcase01.ts14oeValidateAlert.TheActor
217     AdtAlertID : testcase01.ts14oeValidateAlert.AdtAlertID = "1"
218     }
219    oracle {
220      satisfaction = "true"
221      received message {
222      AMessage : testcase01.ts14oeValidateAlert.AMessage= 'The Alert is now declared as valid !'
223      tsi09.steve received from system actAuthenticated.inactAuthenticated.ieMessage(AMessage)
224
225      }
226    }
227     test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
228    }
229 //-----------------------------------------------------
230   test step instance tsi15: testcase01.ts15oeAlert2{
231    variables {
232     reuse tsi07.tango as testcase01.ts15oeAlert2.TheActor
233     AetHumanKind : testcase01.ts15oeAlert2.AetHumanKind ="witness"
234     AdtDate : testcase01.ts15oeAlert2.AdtDate= "2017:11:26"
235     AdtTime : testcase01.ts15oeAlert2.AdtTime= "10:20:00"
```

```
236     AdtPhoneNumber : testcase01.ts15oeAlert2.AdtPhoneNumber= "+3524666445000"
237     AdtGPSLocation : testcase01.ts15oeAlert2.AdtGPSLocation= "49.627095:6.160251"
238     AdtComment : testcase01.ts15oeAlert2.AdtComment= "A car crash just happened."
239    }
240    message {
241     tsi07.tango sent to system testcase01.ts15oeAlert2.out : actComCompany.outactComCompany.oeAlert(
            AetHumanKind,AdtDate,AdtTime,AdtPhoneNumber,AdtGPSLocation,AdtComment)
242
243    }
244    oracle {
245      satisfaction = "true"
246      received message {
247      AdtSMS : testcase01.ts15oeAlert2.AdtSMS= 'Your alert has been registered. We will handle it and
         keep you informed'
248      tsi07.tango received from system actComCompany.inactComCompany.ieSmsSend(AdtPhoneNumber,AdtSMS)
249
250      }
251    }
252     test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
253    }
254 //-----------------------------------------------------
255   test step instance tsi16: testcase01.ts16oeSetClock05{
256    variables {
257     reuse tsi02.theClock as testcase01.ts16oeSetClock05.TheActor
258     ACurrentClock : testcase01.ts16oeSetClock05.ACurrentClock = "2017:11:26 - 12:45:00"
259    }
260    oracle {
261      satisfaction = "true"
262      received message {
263
264      }
265    }
266     test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
267    }
268 //-----------------------------------------------------
269   test step instance tsi17: testcase01.ts17oeSetCrisisStatus{
270    variables {
271     reuse tsi09.steve as testcase01.ts17oeSetCrisisStatus.TheActor
272     AdtCrisisID : testcase01.ts17oeSetCrisisStatus.AdtCrisisID = "1"
273     AetCrisisStatus : testcase01.ts17oeSetCrisisStatus.AetCrisisStatus= "solved"
274    }
275    oracle {
276      satisfaction = "true"
277      received message {
278      AMessage : testcase01.ts17oeSetCrisisStatus.AMessage= "The crisis status has been updated !"
279      tsi09.steve received from system actAuthenticated.inactAuthenticated.ieMessage(AMessage)
280      }
281    }
282     test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
283    }
284 //-----------------------------------------------------
285   test step instance tsi18: testcase01.ts18oeReportOnCrisis{
286    variables {
287     reuse tsi09.steve as testcase01.ts18oeReportOnCrisis.TheActor
288     AdtCrisisID : testcase01.ts18oeReportOnCrisis.AdtCrisisID = "1"
289     AdtComment : testcase01.ts18oeReportOnCrisis.AdtComment= "3 victims sent to hospital, 2 cars
        evacuated and 4 rescue unit mobilized"
290    }
291    oracle {
292      satisfaction = "true"
293      received message {
294      AMessage : testcase01.ts18oeReportOnCrisis.AMessage= 'The crisis comment has been updated !'
295      tsi09.steve received from system actAuthenticated.inactAuthenticated.ieMessage(AMessage)
296
297      }
298    }
299     test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
300    }
301 //-----------------------------------------------------
302   test step instance tsi19: testcase01.ts19oeCloseCrisis{
```

```
303    variables {
304     reuse tsi09.steve as testcase01.ts19oeCloseCrisis.TheActor
305     AdtCrisisID : testcase01.ts19oeCloseCrisis.AdtCrisisID = "1"
306    }
307    oracle {
308      satisfaction = "true"
309      received message {
310      AMessage : testcase01.ts19oeCloseCrisis.AMessage= 'The crisis is now closed !'
311
312     tsi09.steve received from system actAuthenticated.inactAuthenticated.ieMessage(AMessage)
313
314      }
315    }
316     test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
317    }
318
319   }
320 //-----------------------------------------------------
321 //-----------------------------------------------------
322 //-----------------------------------------------------
323  test case instance instance01Part01:testcase01{
324 //-----------------------------------------------------
325   test step instance tsi01:testcase01.ts01oeCreateSystemAndEnvironment{
326    variables {
327     theCreator:testcase01.ts01oeCreateSystemAndEnvironment.Creator = "theCreator"
328     AqtyComCompanies : testcase01.ts01oeCreateSystemAndEnvironment.AqtyComCompanies="4"
329     }
330     oracle {
331      satisfaction = "true"
332     }
333     test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
334    }
335 //-----------------------------------------------------
336   test step instance tsi02: testcase01.ts02oeSetClock{
337    variables {
338     theClock:testcase01.ts02oeSetClock.TheActor = "theClock"
339     ACurrentClock : testcase01.ts02oeSetClock.ACurrentClock= "2017:11:24 - 03:20:00"
340     }
341     oracle {
342      satisfaction = "true"
343     }
344     test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
345    }
346 //-----------------------------------------------------
347   test step instance tsi03: testcase01.ts03oeLogin{
348    variables {
349     bill:testcase01.ts03oeLogin.TheActor="bill"
350     AdtLogin : testcase01.ts03oeLogin.AdtLogin= "icrashadmin"
351     AdtPassword : testcase01.ts03oeLogin.AdtPassword= "7WXC1359"
352     }
353     oracle {
354      satisfaction = "true"
355      received message {
356       AMessage : testcase01.ts03oeLogin.AMessage= 'You are logged ! Welcome ...'
357       tsi03.bill received from system actAuthenticated.inactAuthenticated.ieMessage(AMessage)
358      }
359     }
360     test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
361    }
362 //-----------------------------------------------------
363   test step instance tsi04: testcase01.ts04oeAddCoordinator{
364    variables {
365     reuse tsi03.bill as testcase01.ts04oeAddCoordinator.TheActor
366     AdtCoordinatorID : testcase01.ts04oeAddCoordinator.AdtCoordinatorID = "1"
367     AdtLogin :testcase01.ts04oeAddCoordinator.AdtLogin= "steve"
368     AdtPassword : testcase01.ts04oeAddCoordinator.AdtPassword = "pwdMessirExcalibur2017"
369    }
370    oracle {
371      satisfaction = "true"
372      received message {
```

```
373      tsi03.bill received from system actAdministrator.inactAdministrator.ieCoordinatorAdded()
374      }
375    }
376     test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
377    }
378 //-----------------------------------------------------
379   test step instance tsi05: testcase01.ts05oeLogout{
380    variables {
381     reuse tsi03.bill as testcase01.ts05oeLogout.TheActor
382    }
383    oracle {
384      satisfaction = "true"
385      received message {
386      AMessage : testcase01.ts05oeLogout.AMessage= 'You are logged out ! Good Bye ...'
387      tsi03.bill received from system actAuthenticated.inactAuthenticated.ieMessage(AMessage)
388      }
389    }
390     test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
391    }
392 //-----------------------------------------------------
393   test step instance tsi06: testcase01.ts06oeSetClock02{
394    variables {
395     reuse tsi02.theClock as testcase01.ts06oeSetClock02.TheActor
396     ACurrentClock : testcase01.ts06oeSetClock02.ACurrentClock= "2017:11:26 - 10:15:00"
397    }
398    oracle {
399      satisfaction = "true"
400    }
401     test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
402    }
403 //-----------------------------------------------------
404   test step instance tsi07: testcase01.ts07oeAlert1{
405    variables {
406     tango:testcase01.ts07oeAlert1.TheActor ="tango"
407     AetHumanKind : testcase01.ts07oeAlert1.AetHumanKind = "witness"
408     AdtDate : testcase01.ts07oeAlert1.AdtDate = "2017:11:26"
409     AdtTime : testcase01.ts07oeAlert1.AdtTime = "10:10:16"
410     AdtPhoneNumber : testcase01.ts07oeAlert1.AdtPhoneNumber = "+3524666445252"
411     AdtGPSLocation : testcase01.ts07oeAlert1.AdtGPSLocation = "49.627675:6.159590"
412     AdtComment : testcase01.ts07oeAlert1.AdtComment = "3 cars involved in an accident."
413     }
414    oracle {
415      satisfaction = "true"
416      received message {
417      AdtSMS : testcase01.ts07oeAlert1.AdtSMS= 'Your alert has been registered. We will handle it and
         keep you informed'
418      tsi07.tango received from system actComCompany.inactComCompany.ieSmsSend(AdtPhoneNumber,AdtSMS)
419
420      }
421    }
422     test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
423    }
424
425 //-----------------------------------------------------
426   test step instance tsi08: testcase01.ts08oeSetClock03{
427    variables {
428     reuse tsi02.theClock as testcase01.ts08oeSetClock03.ACurrentClock
429     ACurrentClock : testcase01.ts08oeSetClock03.ACurrentClock = "2017:11:26 - 10:30:00"
430    }
431    oracle {
432      satisfaction = "true"
433    }
434     test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
435    }
436 //-----------------------------------------------------
437   test step instance tsi09: testcase01.ts09oeSollicitateCrisisHandling{
438    variables {
439     reuse tsi02.theClock as testcase01.ts09oeSollicitateCrisisHandling.TheActor
440     steve:testcase01.ts09oeSollicitateCrisisHandling.TheCoordinator ="steve"
441     reuse tsi03.bill as testcase01.ts09oeSollicitateCrisisHandling.TheAdministrator
```

```
442    }
443    oracle {
444      satisfaction = "true"
445      received message {
446       AMessageForCrisisHandlers : testcase01.ts09oeSollicitateCrisisHandling.
       AMessageForCrisisHandlers= 'There are alerts pending since more than the defined delay. Please
       REACT !'
447
448      tsi03.bill received from system actAuthenticated.inactAuthenticated.ieMessage(
       AMessageForCrisisHandlers)
449      tsi09.steve received from system actAuthenticated.inactAuthenticated.ieMessage(
       AMessageForCrisisHandlers)
450           }
451    }
452     test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
453    }
454    }
455
456 //-----------------------------------------------------
457 //-----------------------------------------------------
458 //-----------------------------------------------------
459  test case instance instance01Part02:testcase01{
460
461   test step instance tsi10: testcase01.ts10oeLogin02{
462    variables {
463     steve : testcase01.ts10oeLogin02.TheActor
464     AdtLogin : testcase01.ts10oeLogin02.AdtLogin = "steve"
465     AdtPassword : testcase01.ts10oeLogin02.AdtPassword= "pwdMessirExcalibur2017"
466    }
467    oracle {
468      satisfaction = "true"
469      received message {
470      AMessage : testcase01.ts10oeLogin02.AMessage= 'You are logged ! Welcome ...'
471      steve received from system actAuthenticated.inactAuthenticated.ieMessage(AMessage)
472
473      }
474    }
475    test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
476    }
477 //-----------------------------------------------------
478   test step instance tsi11: testcase01.ts11oeGetCrisisSet{
479    variables {
480     reuse tsi10.steve as testcase01.ts11oeGetCrisisSet.TheActor
481     AetCrisisStatus : testcase01.ts11oeGetCrisisSet.AetCrisisStatus = "pending"
482    }
483    oracle {
484      satisfaction = "true"
485      received message {
486      ActCrisis : testcase01.ts11oeGetCrisisSet.ActCrisis= "crisis with ID 1 details"
487      tsi10.steve received from system actCoordinator.inactCoordinator.ieSendACrisis(ActCrisis)
488      }
489    }
490    test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
491    }
492 //-----------------------------------------------------
493   test step instance tsi12: testcase01.ts12oeSetCrisisHandler{
494    variables {
495     reuse tsi10.steve as testcase01.ts12oeSetCrisisHandler.TheActor
496     AdtCrisisID : testcase01.ts12oeSetCrisisHandler.AdtCrisisID = "1"
497     tango : testcase01.ts12oeSetCrisisHandler.TheComCompany
498    }
499    oracle {
500      satisfaction = "true"
501      received message {
502      AMessage : testcase01.ts12oeSetCrisisHandler.AMessage= 'You are now considered as handling the
       crisis !'
503      AdtSMS : testcase01.ts12oeSetCrisisHandler.AdtSMS= 'The handling of your alert by our services
       is in progress !'
504      AdtPhoneNumber : testcase01.ts12oeSetCrisisHandler.AdtPhoneNumber= "+3524666445252"
505
```

```
506     tango received from system actComCompany.inactComCompany.ieSmsSend(AdtPhoneNumber,AdtSMS)
507     tsi10.steve received from system actAuthenticated.inactAuthenticated.ieMessage(AMessage)
508
509     }
510   }
511    test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
512   }
513 //-----------------------------------------------------
514  test step instance tsi13: testcase01.ts13oeSetClock04{
515   variables {
516     theClock : testcase01.ts13oeSetClock04.TheActor
517     ACurrentClock : testcase01.ts13oeSetClock04.ACurrentClock = "2017:11:26 - 10:45:00"
518   }
519   oracle {
520     satisfaction = "true"
521   }
522   test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
523   }
524 //-----------------------------------------------------
525  test step instance tsi14: testcase01.ts14oeValidateAlert{
526   variables {
527    reuse tsi10.steve as testcase01.ts14oeValidateAlert.TheActor
528    AdtAlertID : testcase01.ts14oeValidateAlert.AdtAlertID = "1"
529    }
530   oracle {
531     satisfaction = "true"
532     received message {
533     AMessage : testcase01.ts14oeValidateAlert.AMessage= 'The Alert is now declared as valid !'
534     tsi10.steve received from system actAuthenticated.inactAuthenticated.ieMessage(AMessage)
535
536     }
537   }
538    test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
539   }
540 //-----------------------------------------------------
541  test step instance tsi15: testcase01.ts15oeAlert2{
542   variables {
543    reuse tsi12.tango as testcase01.ts15oeAlert2.TheActor
544    AetHumanKind : testcase01.ts15oeAlert2.AetHumanKind ="witness"
545    AdtDate : testcase01.ts15oeAlert2.AdtDate= "2017:11:26"
546    AdtTime : testcase01.ts15oeAlert2.AdtTime= "10:20:00"
547    AdtPhoneNumber : testcase01.ts15oeAlert2.AdtPhoneNumber= "+3524666445000"
548    AdtGPSLocation : testcase01.ts15oeAlert2.AdtGPSLocation= "49.627095:6.160251"
549    AdtComment : testcase01.ts15oeAlert2.AdtComment= "A car crash just happened."
550   }
551   message {
552    tsi12.tango sent to system testcase01.ts15oeAlert2.out : actComCompany.outactComCompany.oeAlert(
       AetHumanKind,AdtDate,AdtTime,AdtPhoneNumber,AdtGPSLocation,AdtComment)
553
554   }
555   oracle {
556     satisfaction = "true"
557     received message {
558     AdtSMS : testcase01.ts15oeAlert2.AdtSMS= 'Your alert has been registered. We will handle it and
        keep you informed'
559     tsi12.tango received from system actComCompany.inactComCompany.ieSmsSend(AdtPhoneNumber,AdtSMS)
560
561     }
562   }
563    test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
564   }
565 //-----------------------------------------------------
566  test step instance tsi16: testcase01.ts16oeSetClock05{
567   variables {
568    reuse tsi13.theClock as testcase01.ts16oeSetClock05.TheActor
569    ACurrentClock : testcase01.ts16oeSetClock05.ACurrentClock = "2017:11:26 - 12:45:00"
570   }
571   oracle {
572     satisfaction = "true"
573     received message {
```

```
574
575       }
576     }
577      test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
578     }
579 //-----------------------------------------------------
580   test step instance tsi17: testcase01.ts17oeSetCrisisStatus{
581    variables {
582     reuse tsi10.steve as testcase01.ts17oeSetCrisisStatus.TheActor
583     AdtCrisisID : testcase01.ts17oeSetCrisisStatus.AdtCrisisID = "1"
584     AetCrisisStatus : testcase01.ts17oeSetCrisisStatus.AetCrisisStatus= "solved"
585    }
586    oracle {
587      satisfaction = "true"
588      received message {
589      AMessage : testcase01.ts17oeSetCrisisStatus.AMessage= "The crisis status has been updated !"
590      tsi10.steve received from system actAuthenticated.inactAuthenticated.ieMessage(AMessage)
591      }
592    }
593     test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
594    }
595 //-----------------------------------------------------
596   test step instance tsi18: testcase01.ts18oeReportOnCrisis{
597    variables {
598     reuse tsi10.steve as testcase01.ts18oeReportOnCrisis.TheActor
599     AdtCrisisID : testcase01.ts18oeReportOnCrisis.AdtCrisisID = "1"
600     AdtComment : testcase01.ts18oeReportOnCrisis.AdtComment= "3 victims sent to hospital, 2 cars
        evacuated and 4 rescue unit mobilized"
601    }
602    oracle {
603      satisfaction = "true"
604      received message {
605      AMessage : testcase01.ts18oeReportOnCrisis.AMessage= 'The crisis comment has been updated !'
606      tsi10.steve received from system actAuthenticated.inactAuthenticated.ieMessage(AMessage)
607
608      }
609    }
610     test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
611    }
612 //-----------------------------------------------------
613   test step instance tsi19: testcase01.ts19oeCloseCrisis{
614    variables {
615     reuse tsi10.steve as testcase01.ts19oeCloseCrisis.TheActor
616     AdtCrisisID : testcase01.ts19oeCloseCrisis.AdtCrisisID = "1"
617    }
618    oracle {
619      satisfaction = "true"
620      received message {
621      AMessage : testcase01.ts19oeCloseCrisis.AMessage= 'The crisis is now closed !'
622
623     tsi10.steve received from system actAuthenticated.inactAuthenticated.ieMessage(AMessage)
624
625      }
626    }
627     test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
628    }
629
630   }
631  }
632
633 }
```

Listing C.67: Messir Spec. file tci-testcase01-instance01.msr.

C.68 File ./src-gen/messir-spec/usecases/ugHospitalMonitor.msr

```
1 package icrash.usecases.ugHospitalMonitor {
2
3  import icrash.environment
```

```
4 import icrash.usecases.subfunctions
5
6 Use Case Model {
7  use case system usergoal ugHospitalMonitor() {
8   actor icrash.environment.actHospital[primary,active]
9
10   reuse oeGetHospitalCrisisSet[0..*]
11   reuse oeGetHospitalVictimSet[0..*]
12   reuse oeGetHospitalInjurySet[0..*]
13
14   step a: icrash.environment.actHospital executes oeGetHospitalVictimSet
15   step b: icrash.environment.actHospital executes oeGetHospitalInjurySet
16   step c: icrash.environment.actHospital executes oeGetHospitalCrisisSet
17  }
18 }
19 }
```

Listing C.68: Messir Spec. file ugHospitalMonitor.msr.

C.69 File ./src-gen/messir-spec/usecases/usecase-suDeployAndRun.msr

```
1 package icrash.usecases.suDeployAndRun {
2 import icrash.concepts.primarytypes.datatypes
3 import icrash.environment
4 import icrash.usecases.suGlobalCrisisHandling
5 import icrash.usecases.suHospitalCrisisInteraction
6 import icrash.usecases.ugAdministrateTheSystem
7 import icrash.usecases.subfunctions
8
9 Use Case Model {
10  use case system summary suDeployAndRun() {
11   actor actAdministrator[primary,active]
12   actor actMsrCreator[secondary,active]
13   actor actCoordinator[secondary,active,multiple]
14   actor actHospital[secondary,active,multiple]
15   actor actActivator[secondary,proactive]
16   actor actComCompany[secondary,active]
17   actor actCorrupter[secondary,active]
18
19   reuse oeCreateSystemAndEnvironment[1..1]
20   reuse ugAdministrateTheSystem[1..*]
21   reuse suGlobalCrisisHandling[1..*]
22   reuse suHospitalCrisisInteraction[1..*]
23   reuse oeSetClock[1..*]
24   reuse oeSollicitateCrisisHandling[0..*]
25   reuse oeAlert[1..*]
26   reuse oeCorruptAlert[0..*]
27
28   step a: actMsrCreator executes oeCreateSystemAndEnvironment
29   step b: actAdministrator executes ugAdministrateTheSystem
30   step c: actComCompany executes oeAlert
31   step d: actActivator executes oeSetClock
32   step ^e: actActivator executes oeSollicitateCrisisHandling
33   step f: actCoordinator executes suGlobalCrisisHandling
34   step g: actHospital executes suHospitalCrisisInteraction
35   step h: actCorrupter executes oeCorruptAlert
36
37   ordering constraint
38    "step (a) must be always the first step."
39   ordering constraint
40    "step (f) can be executed by different actCoordinator actors."
41   ordering constraint
42    "step (g) can be executed by different actHospital actors."
43   ordering constraint
44    "if (e) then previously (d)."
45   ordering constraint
46    "if (h) then previously (c)."
```

```
47   }
48 //-----------------------------------------------------
49 //-----------------------------------------------------
50 //-----------------------------------------------------
51  use case instance uciSimpleAndComplete : suDeployAndRun {
52   actors {
53    theCreator : actMsrCreator
54    theClock : actActivator
55    bill : actAdministrator
56    tango : actComCompany
57    steve : actCoordinator
58    wozniak : actHospital
59    linus : actCorrupter
60   }
61   use case steps {
62 //-----------------------------------------------------
63    theCreator
64    executed instanceof subfunction
65       oeCreateSystemAndEnvironment("4"){}
66 //-----------------------------------------------------
67    theClock
68    executed instanceof subfunction
69       oeSetClock("2017:11:24 - 03:20:00"){}
70 //-----------------------------------------------------
71    bill
72    executed instanceof subfunction
73       oeLogin("icrashadmin","7WXC1359"){
74        ieMessage('You are logged ! Welcome ...') returned to bill
75       }
76 //-----------------------------------------------------
77    bill
78    executed instanceof subfunction
79       oeAddCoordinator("1","steve","pwdMessirExcalibur2017"){
80        ieCoordinatorAddedreturned returned to bill
81       }
82 //-----------------------------------------------------
83    bill
84    executed instanceof subfunction
85       oeAddHospital("2","wozniak","pwdMessirExcalibur2017bc"){
86        ieHospitalAddedreturned returned to bill
87       }
88 //-----------------------------------------------------
89    bill
90    executed instanceof subfunction
91       oeLogout{
92        ieMessage('You are logged out ! Good Bye ...') returned to bill
93       }
94 //-----------------------------------------------------
95    theClock
96    executed instanceof subfunction
97       oeSetClock("2017:11:26 - 10:15:00"){}
98 //-----------------------------------------------------
99    tango
100    executed instanceof subfunction
101       oeAlert("witness","2017:11:26","10:10:16","+3524666445252",
102           "49.627675:6.159590","3 cars involved in an accident."){
103         ieSmsSend("+3524666445252","Your alert has been registered. We will handle it and keep you
      informed") returned to tango
104       }
105 //-----------------------------------------------------
106    theClock
107    executed instanceof subfunction
108       oeSetClock("2017:11:26 - 10:30:00"){}
109 //-----------------------------------------------------
110    theClock
111    executed instanceof subfunction
112       oeSollicitateCrisisHandling{
113        ieMessage("There are alerts pending since more than the defined delay. Please REACT !")
114        returned to bill
115        ieMessage("There are alerts pending since more than the defined delay. Please REACT !")
```

```
116         returned to steve
117        }
118 //-----------------------------------------------------
119     steve
120     executed instanceof subfunction
121        oeLogin("steve","pwdMessirExcalibur2017"){
122         ieMessage('You are logged ! Welcome ...') returned to steve
123        }
124 //-----------------------------------------------------
125     steve
126     executed instanceof subfunction
127        oeGetCrisisSet("pending"){
128         ieSendACrisis("crisis with ID 1 details") returned to steve
129        }
130 //-----------------------------------------------------
131     steve
132     executed instanceof subfunction
133        oeSetCrisisHandler("1"){
134         ieSmsSend("+3524666445252","The handling of your alert by our services is in progress !")
135         returned to tango
136         ieMessage("You are now considered as handling the crisis !")
137         returned to steve
138        }
139 //-----------------------------------------------------
140     theClock
141     executed instanceof subfunction
142        oeSetClock("2017:11:26 - 10:45:00"){}
143 //-----------------------------------------------------
144     steve
145     executed instanceof subfunction
146        oeValidateAlert("1"){
147         ieMessage('The Alert is now declared as valid !')
148         returned to steve
149        }
150 //-----------------------------------------------------
151     tango
152     executed instanceof subfunction
153        oeAlert("witness","2017:11:26","10:20:00","+3524666445000",
154            "49.627095:6.160251","A car crash just happened."){
155          ieSmsSend("+3524666445000","Your alert has been registered. We will handle it and keep you
       informed") returned to tango
156        }
157 //-----------------------------------------------------
158     steve
159     executed instanceof subfunction
160        oeAddVictim("1"){
161         ieMessage('Victim has been added !')
162         returned to steve
163        }
164 //-----------------------------------------------------
165     steve
166     executed instanceof subfunction
167        oeAddInjury("1","head"){
168         ieMessage('Injury has been added !')
169         returned to steve
170        }
171 //-----------------------------------------------------
172     theClock
173     executed instanceof subfunction
174        oeSetClock("2017:11:26 - 12:45:00"){}
175 //-----------------------------------------------------
176     wozniak
177     executed instanceof subfunction
178        oeLogin("wozniak","pwdMessirExcalibur2017bc"){
179         ieMessage('You are logged ! Welcome ...') returned to wozniak
180        }
181 //-----------------------------------------------------
182     wozniak
183     executed instanceof subfunction
184        oeGetCrisisSet("pending"){
```

```
185         ieSendACrisis("crisis with ID 1 details") returned to wozniak
186        }
187 //-----------------------------------------------------
188    wozniak
189    executed instanceof subfunction
190       oeGetVictimSet("1"){
191        ieSendACrisis("Victim with ID 1 details") returned to wozniak
192       }
193 //-----------------------------------------------------
194    wozniak
195    executed instanceof subfunction
196       oeGetInjurySet("1"){
197        ieSendACrisis("Injury with ID 1 details") returned to wozniak
198       }
199 //-----------------------------------------------------
200    steve
201    executed instanceof subfunction
202       oeSetCrisisStatus("1","solved"){
203        ieMessage('The crisis status has been updated !')
204        returned to steve
205       }
206 //-----------------------------------------------------
207    steve
208    executed instanceof subfunction
209       oeReportOnCrisis("1","3 victims sent to hospital, 2 cars evacuated and 4 rescue unit
       mobilized"){
210        ieMessage('The crisis comment has been updated !')
211        returned to steve
212       }
213 //-----------------------------------------------------
214    steve
215    executed instanceof subfunction
216       oeCloseCrisis("1"){
217        ieMessage('The crisis is now closed !')
218        returned to steve
219       }
220 //-----------------------------------------------------
221    tango
222    executed instanceof subfunction
223       oeAlert("witness","2018:11:26","10:10:16","+3524666445252",
224           "49.627675:6.159590","3 cars involved in an accident."){
225         ieSmsSend("+3524666445252","Your alert has been registered. We will handle it and keep you
       informed") returned to tango
226       }
227 //-----------------------------------------------------
228    linus
229    executed instanceof subfunction
230       oeCorruptAlert("witness","2019:11:26","10:10:16","+3524666445252",
231           "49.627675:6.159590","3 cars involved in an accident."){
232       }
233 //-----------------------------------------------------
234    steve
235    executed instanceof subfunction
236       oeValidateAlert("2"){
237        ieMessage('The Alert is corrupted!')
238        returned to steve
239       }
240   }
241
242  }
243 //-----------------------------------------------------
244 //-----------------------------------------------------
245 //-----------------------------------------------------
246  use case instance uciSimpleAndCompletePart01 : suDeployAndRun{
247
248   actors {
249    theCreator : actMsrCreator
250    theClock : actActivator
251    bill : actAdministrator
252    tango : actComCompany
```

```
253     steve : actCoordinator
254    }
255    use case steps {
256 //-----------------------------------------------------
257     theCreator
258     executed instanceof subfunction
259        oeCreateSystemAndEnvironment("4"){}
260 //-----------------------------------------------------
261     theClock
262     executed instanceof subfunction
263        oeSetClock("2017:11:24 - 03:20:00"){}
264 //-----------------------------------------------------
265     bill
266     executed instanceof subfunction
267        oeLogin("icrashadmin","7WXC1359"){
268         ieMessage('You are logged ! Welcome ...') returned to bill
269        }
270 //-----------------------------------------------------
271     bill
272     executed instanceof subfunction
273        oeAddCoordinator("1","steve","pwdMessirExcalibur2017"){
274         ieCoordinatorAddedreturned returned to bill
275        }
276 //-----------------------------------------------------
277     bill
278     executed instanceof subfunction
279        oeAddHospital("2","wozniak","pwdMessirExcalibur2017bc"){
280         ieHospitalAddedreturned returned to bill
281        }
282 //-----------------------------------------------------
283     bill
284     executed instanceof subfunction
285        oeLogout{
286         ieMessage('You are logged out ! Good Bye ...') returned to bill
287        }
288 //-----------------------------------------------------
289     theClock
290     executed instanceof subfunction
291        oeSetClock("2017:11:26 - 10:15:00"){}
292 //-----------------------------------------------------
293     tango
294     executed instanceof subfunction
295        oeAlert("witness","2017:11:26","10:10:16","+3524666445252",
296            "49.627675:6.159590","3 cars involved in an accident."){
297          ieSmsSend("+3524666445252","Your alert has been registered. We will handle it and keep you
       informed") returned to tango
298        }
299 //-----------------------------------------------------
300     theClock
301     executed instanceof subfunction
302        oeSetClock("2017:11:26 - 10:30:00"){}
303 //-----------------------------------------------------
304     theClock
305     executed instanceof subfunction
306        oeSollicitateCrisisHandling{
307         ieMessage("There are alerts pending since more than the defined delay. Please REACT !")
308         returned to bill
309         ieMessage("There are alerts pending since more than the defined delay. Please REACT !")
310         returned to steve
311        }
312    }
313   }
314 //-----------------------------------------------------
315 //-----------------------------------------------------
316 //-----------------------------------------------------
317   use case instance uciSimpleAndCompletePart02 : suDeployAndRun{
318    actors {
319     theCreator : actMsrCreator
320     theClock : actActivator
321     bill : actAdministrator
```

```
322     tango : actComCompany
323     steve : actCoordinator
324    }
325    use case steps {
326
327 //-----------------------------------------------------
328     steve
329     executed instanceof subfunction
330        oeLogin("steve","pwdMessirExcalibur2017"){
331         ieMessage('You are logged ! Welcome ...') returned to steve
332        }
333 //-----------------------------------------------------
334     steve
335     executed instanceof subfunction
336        oeGetCrisisSet("pending"){
337         ieSendACrisis("crisis with ID 1 details") returned to steve
338        }
339 //-----------------------------------------------------
340     steve
341     executed instanceof subfunction
342        oeSetCrisisHandler("1"){
343         ieSmsSend("+3524666445252","The handling of your alert by our services is in progress !")
344         returned to tango
345         ieMessage("You are now considered as handling the crisis !")
346         returned to steve
347        }
348 //-----------------------------------------------------
349     theClock
350     executed instanceof subfunction
351        oeSetClock("2017:11:26 - 10:45:00"){}
352 //-----------------------------------------------------
353     steve
354     executed instanceof subfunction
355        oeValidateAlert("1"){
356         ieMessage('The Alert is now declared as valid !')
357         returned to steve
358        }
359 //-----------------------------------------------------
360     tango
361     executed instanceof subfunction
362        oeAlert("witness","2017:11:26","10:20:00","+3524666445000",
363            "49.627095:6.160251","A car crash just happened."){
364          ieSmsSend("+3524666445000","Your alert has been registered. We will handle it and keep you
       informed") returned to tango
365        }
366 //-----------------------------------------------------
367     theClock
368     executed instanceof subfunction
369        oeSetClock("2017:11:26 - 12:45:00"){}
370 //-----------------------------------------------------
371     steve
372     executed instanceof subfunction
373        oeSetCrisisStatus("1","solved"){
374         ieMessage('The crisis status has been updated !')
375         returned to steve
376        }
377 //-----------------------------------------------------
378     steve
379     executed instanceof subfunction
380        oeReportOnCrisis("1","3 victims sent to hospital, 2 cars evacuated and 4 rescue unit
       mobilized"){
381         ieMessage('The crisis comment has been updated !')
382         returned to steve
383        }
384 //-----------------------------------------------------
385     steve
386     executed instanceof subfunction
387        oeCloseCrisis("1"){
388         ieMessage('The crisis is now closed !')
389         returned to steve
```

```
        }

    }
   }
  }
}
```

Listing C.69: Messir Spec. file usecase-suDeployAndRun.msr.

C.70 File ./src-gen/messir-spec/usecases/usecase-suGlobalCrisisHandling.msr

```
package icrash.usecases.suGlobalCrisisHandling {
 import lu.uni.lassy.messir.libraries.primitives
 import icrash.environment
 import icrash.usecases.subfunctions
 import icrash.usecases.ugSecurelyUseSystem
 import icrash.usecases.ugManageCrisis
 import icrash.usecases.ugMonitor

 Use Case Model {
  use case system summary
   suGlobalCrisisHandling() {
   actor actCoordinator[primary,active]

   reuse ugSecurelyUseSystem[1..*]
   reuse ugMonitor[1..*]
   reuse ugManageCrisis[1..*]

   step a: actCoordinator
       executes ugSecurelyUseSystem
   step b: actCoordinator
       executes ugMonitor
   step c: actCoordinator
       executes ugManageCrisis

   ordering constraint
    "steps (a) (b) and (c) executions are interleaved
    (steps (b) and (c) have their protocol constrained by steps of (a))."
   ordering constraint
     "steps (a) (b) and (c) can be executed multiple times."
}
}}
```

Listing C.70: Messir Spec. file usecase-suGlobalCrisisHandling.msr.

C.71 File ./src-gen/messir-spec/usecases/usecase-suHospitalCrisisInteraction.msr

```
package icrash.usecases.suHospitalCrisisInteraction {

import lu.uni.lassy.messir.libraries.primitives

import icrash.environment
import icrash.usecases.subfunctions
import icrash.usecases.ugSecurelyUseSystem
import icrash.usecases.ugCrisisInteraction
import icrash.usecases.ugHospitalMonitor

Use Case Model {
 use case system summary
  suHospitalCrisisInteraction() {
  actor actHospital[primary,active]

  reuse ugSecurelyUseSystem[1..*]
  reuse ugHospitalMonitor[1..*]
```

```
18
19   step a: actHospital
20       executes ugSecurelyUseSystem
21   step b: actHospital
22       executes ugHospitalMonitor
23
24  ordering constraint
25   "steps (a) and (b) executions are interleaved
26   (step (b) have their protocol constrained by steps of (a))."
27  ordering constraint
28    "steps (a) and (b) can be executed multiple times."
29 }
30  }
31
32 }
```

Listing C.71: Messir Spec. file usecase-suHospitalCrisisInteraction.msr.

C.72 File ./src-gen/messir-spec/usecases/usecase-ugAdministrateTheSystem.msr

```
1 package icrash.usecases.ugAdministrateTheSystem {
2
3  import icrash.environment
4  import icrash.usecases.ugSecurelyUseSystem
5  import icrash.usecases.subfunctions
6
7  Use Case Model {
8
9   use case system usergoal
10   ugAdministrateTheSystem() {
11   actor actAdministrator[primary,active]
12
13   reuse ugSecurelyUseSystem[1..*]
14   reuse oeAddCoordinator[1..*]
15   reuse oeDeleteCoordinator[0..*]
16   reuse oeAddHospital[1..*]
17   reuse oeDeleteHospital[0..*]
18
19   step a: actAdministrator
20      executes ugSecurelyUseSystem
21   step b: actAdministrator
22      executes oeAddCoordinator
23   step c: actAdministrator
24      executes oeDeleteCoordinator
25   step d: actAdministrator
26      executes oeAddHospital
27   step e: actAdministrator
28      executes oeDeleteHospital
29
30   ordering constraint
31     "steps (a) (b) and (c) executions are interleaved
32     (steps (b) and (c) have their protocol constrained
33      by steps of (a) as well as steps (d) and (e))."
34   ordering constraint
35     "steps (a) (b) (c) (d) and (e) can be executed multiple times."
36  }
37 }
38 }
```

Listing C.72: Messir Spec. file usecase-ugAdministrateTheSystem.msr.

C.73 File ./src-gen/messir-spec/usecases/usecase-ugCrisisInteraction.msr

```
1 package icrash.usecases.ugCrisisInteraction { //_t7                    :
            ?
```

```
2
3 import lu.uni.lassy.messir.libraries.primitives
4 import icrash.environment
5 import icrash.usecases.subfunctions
6
7  Use Case Model {
8
9 //  use case system usergoal ugCrisisIntaraction() {
10 //   actor actHospital[primary, active]
11
12 //   reuse oeValidateAlert[0..*]
13 //   reuse oeSetCrisisStatus[0..*]
14 //   reuse oeSetCrisisHandler[0..*]
15 //   reuse oeReportOnCrisis[0..*]
16 //   reuse oeCloseCrisis[0..*]
17 //   reuse oeInvalidateAlert[0..*]
18 //
19 //   step a: actCoordinator executes oeValidateAlert
20 //   step b: actCoordinator executes oeSetCrisisStatus
21 //   step c: actCoordinator executes oeSetCrisisHandler
22 //   step d: actCoordinator executes oeReportOnCrisis
23 //   step f: actCoordinator executes oeCloseCrisis
24 //   step g: actCoordinator executes oeInvalidateAlert
25 //
26 //   ordering constraint "managing a crisis is doing one of the indicated use cases."
27
28 //  }
29
30  }
31
32 }
```

Listing C.73: Messir Spec. file usecase-ugCrisisInteraction.msr.

C.74 File ./src-gen/messir-spec/usecases/usecase-ugManageCrisis.msr

```
1 package icrash.usecases.ugManageCrisis {
2
3  import icrash.environment
4  import icrash.usecases.subfunctions
5
6  Use Case Model {
7
8   use case system usergoal ugManageCrisis() {
9    actor actCoordinator[primary, active]
10
11    reuse oeValidateAlert[0..*]
12    reuse oeSetCrisisStatus[0..*]
13    reuse oeSetCrisisHandler[0..*]
14    reuse oeReportOnCrisis[0..*]
15    reuse oeCloseCrisis[0..*]
16    reuse oeInvalidateAlert[0..*]
17    reuse oeAddVictim[0..*]
18    reuse oeDeleteVictim[0..*]
19    reuse oeAddInjury[0..*]
20    reuse oeDeleteInjury[0..*]
21
22    step a: actCoordinator executes oeValidateAlert
23    step b: actCoordinator executes oeSetCrisisStatus
24    step c: actCoordinator executes oeSetCrisisHandler
25    step d: actCoordinator executes oeReportOnCrisis
26    step f: actCoordinator executes oeCloseCrisis
27    step g: actCoordinator executes oeInvalidateAlert
28    step h: actCoordinator executes oeAddVictim
29    step i: actCoordinator executes oeDeleteVictim
30    step j: actCoordinator executes oeAddInjury
31    step k: actCoordinator executes oeDeleteInjury
```

```
32
33    ordering constraint "managing a crisis is doing one of the indicated use cases."
34
35   }
36
37  }
38 }
```

Listing C.74: Messir Spec. file usecase-ugManageCrisis.msr.

C.75 File ./src-gen/messir-spec/usecases/usecase-ugMonitor.msr

```
1 package icrash.usecases.ugMonitor {
2
3  import icrash.environment
4  import icrash.usecases.subfunctions
5
6  Use Case Model {
7   use case system usergoal ugMonitor() {
8    actor icrash.environment.actCoordinator[primary,active]
9
10    reuse oeGetCrisisSet[0..*]
11    reuse oeGetAlertsSet[0..*]
12    reuse oeGetVictimSet[0..*]
13    reuse oeGetInjurySet[0..*]
14
15    step a: icrash.environment.actCoordinator executes oeGetAlertsSet
16    step b: icrash.environment.actCoordinator executes oeGetCrisisSet
17    step c: icrash.environment.actCoordinator executes oeGetVictimSet
18    step d: icrash.environment.actCoordinator executes oeGetInjurySet
19   }
20  }
21 }
```

Listing C.75: Messir Spec. file usecase-ugMonitor.msr.

C.76 File ./src-gen/messir-spec/usecases/usecase-ugSecurelyUseSystem.msr

```
1 package icrash.usecases.ugSecurelyUseSystem {
2
3 import icrash.environment
4 import icrash.usecases.subfunctions
5
6 Use Case Model {
7
8 use case system usergoal
9  ugSecurelyUseSystem() {
10
11  actor actAuthenticated[primary,active]
12
13  reuse oeLogin[1..1]
14  reuse oeLogout[1..1]
15
16  step a: actAuthenticated
17     executes oeLogin
18  step b: actAuthenticated
19     executes oeLogout
20
21  ordering constraint
22   "step (a) must always precede step (b)."
23   }
24  }
25 }
```

Listing C.76: Messir Spec. file usecase-ugSecurelyUseSystem.msr.

C.77 File ./src-gen/messir-spec/usecases/usecaseinstance-ugSecurelyUseSystem-uciugSecurelyUseSystem.msr

```
package usecases.uciugSecurelyUseSystem {
 import icrash.usecases.ugSecurelyUseSystem
  import icrash.usecases.ugSecurelyUseSystem
  import icrash.concepts.primarytypes.datatypes
  import icrash.environment
  import icrash.usecases.suGlobalCrisisHandling
  import icrash.usecases.ugAdministrateTheSystem
  import icrash.usecases.subfunctions

 Use Case Model {

//-----------------------------
  use case instance uciugSecurelyUseSystem : ugSecurelyUseSystem {
   actors {
    bill:actAuthenticated
   }
   use case steps {
//-----------------------------
    bill
    executed instanceof subfunction
       oeLogin("icrashadmin","7WXC1359"){
        ieMessage('You are logged ! Welcome ...') returned to bill
       }
//-----------------------------
    bill
    executed instanceof subfunction
       oeLogout{
        ieMessage('You are logged out ! Good Bye ...') returned to bill
       }
   }
  }
 }
}
```

Listing C.77: Messir Spec. file usecaseinstance-ugSecurelyUseSystem-uciugSecurelyUseSystem.msr.

Appendix D

Listing of the Prolog Files Referenced in the Operation Model Specification

D.1 File ./src-gen/prolog-ref-spec/Operations/Environment/OUT/outactActi oeSetClock.pl

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%-------------------------------------------------------
msrop(outactActivator,
   oeSetClock,
   [preProtocol,Self,
    AcurrentClock
    ],
   []):-
/* Pre Protocol:*/
/* PreP01 */
 msrVar(ctState,TheSystem),
 msrVar(ptBoolean,AvpStarted),

 msrNav([Self],[rnActor,rnSystem],[TheSystem]),

 msrNav([Self],[rnActor,rnSystem,vpStarted],[AvpStarted]),
 AvpStarted = [ptBoolean,true],

 msrNav([TheSystem],
    [clock,lt,[AcurrentClock]],
    [[ptBoolean,true]])
 .

msrop(outactActivator,
   oeSetClock,
   [preFunctional,Self,
    AcurrentClock
    ],
   []):-
/* Pre Functional:*/
/* PreF01 */
true.

msrop(outactActivator,
   oeSetClock,
   [post,Self,
    AcurrentClock
    ],
    []):-

```

```
msrVar(ctState,TheSystem),

/* Post Functional:*/

msrNav([Self],[rnActor,rnSystem],[TheSystem]),

/* PostF01 */
msrNav([TheSystem],
    [msmAtPost,clock],
    [AcurrentClock]),

 /* Post Protocol:*/
/* PostP01 */
true
 .
```

Listing D.1: Prolog file outactActivator-oeSetClock.pl.

D.2 File ./src-gen/prolog-ref-spec/Operations/Environment/OUT/outact oeSollicitateCrisisHandling.pl

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%-------------------------------------------------------

msrop(outactActivator,
   oeSollicitateCrisisHandling,
   [preProtocol,Self
    ],
   []):-
/* Pre Protocol:*/
msrVar(ctState,TheSystem),
msrNav([Self],[rnActor,rnSystem],[TheSystem]),

msrVarCol(ctCrisis,_,ColctCrisisToHandle),

/* PreP01 */
msrNav([TheSystem],
    [vpStarted],
    [[ptBoolean,true]]),

/* PreP02 */
msrNav([TheSystem],
    [rnctCrisis,msrSelect,
     handlingDelayPassed,[]
    ],
    ColctCrisisToHandle),

msrNav(ColctCrisisToHandle,
    [msrSize,geq,[[ptInteger,1]]],
    [[ptBoolean,true]])
.

msrop(outactActivator,
   oeSollicitateCrisisHandling,
   [preFunctional,Self
    ],
   []):-
/* Pre Functional:*/
/* PreF01 */
true.

msrop(outactActivator,
   oeSollicitateCrisisHandling,
   [post,Self
    ],
```

```
   []):-

msrVar(ctState,TheSystem),
msrVar(dtComment,AMessageForCrisisHandlers),
msrVar(dtDateAndTime, TheClock),
msrVarCol(ctCrisis,_,ColctCrisisToAllocateIfPossible),

/* Post Functional:*/
msrNav([Self],[rnActor,rnSystem],[TheSystem]),

/* PostF01 */
msrNav([TheSystem],
   [rnctCrisis,msrSelect,
    maxHandlingDelayPassed,[]
   ],
   ColctCrisisToAllocateIfPossible),

msrNav(ColctCrisisToAllocateIfPossible,
   [msrForAll,isAllocatedIfPossible,[]],
   [[ptBoolean,true]]),

/* PostF02 */
msrNav([TheSystem],
   [rnctCrisis,msrSelect,
    handlingDelayPassed,[]
   ],
   ColctCrisisToHandle),

msrNav(ColctCrisisToHandle,
    [msrColSubtract,[ColctCrisisToAllocateIfPossible]
    ],
    ColctCrisisToRemind),

(msrNav(ColctCrisisToRemind,
    [msrSize,geq,[[ptInteger,1]]],
    [[ptBoolean,true]])
 -> (msrNav([AMessageForCrisisHandlers],
      [value],
      [[ptString,'There are alerts pending since more than the defined delay. Please REACT !']]),

   msrNav([TheSystem],
      [rnactAdministrator,rnInterfaceIN,
       ieMessage,[AMessageForCrisisHandlers]
      ],
      [[ptBoolean,true]]),

   msrNav([TheSystem],
      [rnactCoordinator,msrForAll,rnInterfaceIN,
       ieMessage,[AMessageForCrisisHandlers]
      ],
      [[ptBoolean,true]])
  )
; true
),

/* Post Protocol:*/
/* PostP01 */
msrNav([TheSystem],
   [clock],
   [TheClock]),

msrNav([TheSystem],
    [msmAtPost,vpLastReminder],
    [TheClock])
.
```

Listing D.2: Prolog file outactActivator-oeSollicitateCrisisHandling.pl.

D.3 File ./src-gen/prolog-ref-spec/Operations/Environment/OUT/outact oeAddCoordinator.pl

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%-----------------------------------------------------
msrop(outactAdministrator,
   oeAddCoordinator,
   [preProtocol,Self,
    AdtCoordinatorID,
    AdtLogin,
    AdtPassword
    ],
   []):-
/* Pre Protocol:*/
 msrVar(ctState,TheSystem),
 msrVar(actAdministrator,TheActor),
 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
 msrNav([Self],[rnActor],[TheActor]),

/* PreP01 */
 msrNav([TheSystem],
    [vpStarted],
    [[ptBoolean,true]]),

/* PreP02 */
 msrNav([TheActor],
    [rnctAuthenticated,vpIsLogged],
    [[ptBoolean,true]])

 .

msrop(outactAdministrator,
   oeAddCoordinator,
   [preFunctional,Self,
    AdtCoordinatorID,
    AdtLogin,
    AdtPassword
    ],
   []):-
/* Pre Functional:*/
 msrVar(ctState,TheSystem),
 msrVar(actAdministrator,TheActor),
 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
 msrNav([Self],[rnActor],[TheActor]),
/* PreF01 */
 msrNav([TheSystem],
     [rnctCoordinator,
     msrSelect,id,eq,[AdtCoordinatorID]],
     ColctCoordinators),
 msrNav(ColctCoordinators,
     [msrIsEmpty],
     [[ptBoolean,true]])
 .

msrop(outactAdministrator,
   oeAddCoordinator,
   [post,Self,
    AdtCoordinatorID,
    AdtLogin,
    AdtPassword
    ],
    []):-

/* Post Functional:*/
 msrVar(ctState,TheSystem),
 msrVar(actAdministrator,TheActor),
```

```
msrNav([Self],[rnActor,rnSystem],[TheSystem]),
msrNav([Self],[rnActor],[TheActor]),

msrVar(actCoordinator,TheactCoordinator),
msrVar(ctCoordinator,ThectCoordinator),

/* PostF01 */
msrNav([TheactCoordinator],
    [init,[]],
    [[ptBoolean,true]]),

/* PostF02 */
msrNav([ThectCoordinator],
    [init,[AdtCoordinatorID,AdtLogin,AdtPassword]],
    [[ptBoolean,true]]),

/* PostF03 */
msrNav([TheactCoordinator],
    [msmAtPost,rnctCoordinator],
    [ThectCoordinator]),

/* PostF04 */
 msrNav([ThectCoordinator],
    [msmAtPost,rnactAuthenticated],
    [TheactCoordinator]),

/* PostF05 */
 msrNav([TheActor],
    [rnInterfaceIN,
     ieCoordinatorAdded,[]],
    [[ptBoolean,true]]),

 /* Post Protocol:*/
/* PostP01 */
true
.
```

Listing D.3: Prolog file outactAdministrator-oeAddCoordinator.pl.

D.4 File ./src-gen/prolog-ref-spec/Operations/Environment/OUT/outactAdn oeDeleteCoordinator.pl

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%-------------------------------------------------------
msrop(outactAdministrator,
   oeDeleteCoordinator,
   [preProtocol,Self,
    AdtCoordinatorID
    ],
   []):-
/* Pre Protocol:*/
msrVar(ctState,TheSystem),
msrVar(actAdministrator,TheActor),
msrNav([Self],[rnActor,rnSystem],[TheSystem]),
msrNav([Self],[rnActor],[TheActor]),

/* PreP01 */
msrNav([TheSystem],
    [vpStarted],
    [[ptBoolean,true]]),

msrNav([TheActor],
    [rnctAuthenticated,vpIsLogged],
    [[ptBoolean,true]])
.
```

```
27
28msrop(outactAdministrator,
29   oeDeleteCoordinator,
30   [preFunctional,Self,
31    AdtCoordinatorID
32    ],
33   []):-
34/* Pre Functional:*/
35 msrVar(ctState,TheSystem),
36 msrVar(actAdministrator,TheActor),
37 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
38 msrNav([Self],[rnActor],[TheActor]),
39
40/* PreF01 */
41 msrNav([TheSystem],
42     [rnctCoordinator,
43     msrSelect,id,eq,[AdtCoordinatorID]],
44     ColctCoordinators),
45
46 msrNav(ColctCoordinators,
47     [msrSize,eq,[[ptInteger,1]]],
48     [[ptBoolean,true]]).
49
50msrop(outactAdministrator,
51   oeDeleteCoordinator,
52   [post,Self,
53    AdtCoordinatorID
54    ],
55    []):-
56
57/* Post Functional:*/
58 msrVar(ctState,TheSystem),
59 msrVar(actAdministrator,TheActor),
60 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
61 msrNav([Self],[rnActor],[TheActor]),
62
63/* PostF01 */
64 msrNav([TheSystem],
65    [rnctCoordinator,
66     msrSelect,id,eq,[AdtCoordinatorID]],
67    [ThectCoordinator]),
68
69 msrNav([ThectCoordinator],
70     [rnactCoordinator,msrForAll,msrIsKilled],
71     [[ptBoolean,true]]),
72
73 msrNav([ThectCoordinator],
74     [msrIsKilled],
75     [[ptBoolean,true]]),
76
77 /* PostF02 */
78 msrNav([TheActor],
79     [rnInterfaceIN,
80     ieCoordinatorDeleted,[]
81     ],
82     [[ptBoolean,true]]),
83
84 /* Post Protocol:*/
85/* PostP01 */
86 true
87 .
```

Listing D.4: Prolog file outactAdministrator-oeDeleteCoordinator.pl.

D.5 File ./src-gen/prolog-ref-spec/Operations/Environment/OUT/outact oeLogin.pl

```
1%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%-------------------------------------------------------
msrop(outactAuthenticated,
   oeLogin,
   [preProtocol,Self,
    AdtLogin,
    AdtPassword
    ],
   []):-
/* Pre Protocol:*/
msrVar(ctState,TheSystem),
msrVar(actAuthenticated,TheactAuthenticated),
msrNav([Self],[rnActor,rnSystem],[TheSystem]),
msrNav([Self],[rnActor],[TheactAuthenticated]),

/* PreP01 */
msrNav([TheSystem],
    [vpStarted],
    [[ptBoolean,true]]),

msrNav([TheactAuthenticated],
    [rnctAuthenticated,vpIsLogged],
    [[ptBoolean,false]])
.

msrop(outactAuthenticated,
   oeLogin,
   [preFunctional,Self,
    AdtLogin,
    AdtPassword
    ],
   []):-
/* Pre Functional:*/
/* PreF01 */
true
.

msrop(outactAuthenticated,
   oeLogin,
   [post,Self,
    AdtLogin,
    AdtPassword
    ],
    []):-

msrVar(ctState,TheSystem),
msrVar(actAuthenticated,TheactAuthenticated),

msrVar(ptString,AptStringMessageForTheactAuthenticated),
msrVar(ptString,AptStringMessageForTheactAdministrator),

/* Post Functional:*/

msrNav([Self],[rnActor],[TheactAuthenticated]),
msrNav([Self],[rnActor,rnSystem],[TheSystem]),

/* PostF01 */

 ( (msrNav([TheactAuthenticated],
      [rnctAuthenticated,pwd],
      [AdtPassword]),
   msrNav([TheactAuthenticated],
      [rnctAuthenticated,login],
      [AdtLogin])
  )
 -> ( msrNav([AptStringMessageForTheactAuthenticated],
       [eq,[[ptString,'You are logged ! Welcome ...']]],
       [[ptBoolean,true]]),
```

```
   msrNav([TheactAuthenticated],
      [rnInterfaceIN,
       ieMessage,[AptStringMessageForTheactAuthenticated]],
      [[ptBoolean,true]])
  )
; ( msrNav([AptStringMessageForTheactAuthenticated],
      [eq,[[ptString,'Wrong identification information ! Please try again ...']]],
      [[ptBoolean,true]]),
   msrNav([TheactAuthenticated],
      [rnInterfaceIN,
       ieMessage,[AptStringMessageForTheactAuthenticated]],
      [[ptBoolean,true]]),

   msrNav([AptStringMessageForTheactAdministrator],
      [eq,[[ptString,'Intrusion tentative !']]],
      [[ptBoolean,true]]),
   msrNav([TheSystem],
      [rnactAdministrator,rnInterfaceIN,
       ieMessage,[AptStringMessageForTheactAdministrator]],
      [[ptBoolean,true]])
   )
),

/* Post Protocol:*/
/* PostP01 */
( (msrNav([TheactAuthenticated],
     [rnctAuthenticated,pwd],
     [AdtPassword]),
  msrNav([TheactAuthenticated],
     [rnctAuthenticated,login],
     [AdtLogin])
 )
-> (msrNav([TheactAuthenticated],
      [rnctAuthenticated,msmAtPost,vpIsLogged],
      [[ptBoolean,true]])
   )
; true
)
.
```

Listing D.5: Prolog file outactAuthenticated-oeLogin.pl.

D.6 File ./src-gen/prolog-ref-spec/Operations/Environment/OUT/outact oeLogout.pl

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%-------------------------------------------------------
msrop(outactAuthenticated,
  oeLogout,
  [preProtocol,Self
   ],
  []):-
/* Pre Protocol:*/
msrVar(ctState,TheSystem),
msrVar(actAuthenticated,TheActor),
msrNav([Self],[rnActor,rnSystem],[TheSystem]),
msrNav([Self],[rnActor],[TheActor]),

/* PreP01 */
msrNav([TheSystem],
   [vpStarted],
   [[ptBoolean,true]]),

msrNav([TheActor],
   [rnctAuthenticated,vpIsLogged],
```

```
    [[ptBoolean,true]])
 .

msrop(outactAuthenticated,
   oeLogout,
   [preFunctional,Self
    ],
   []):-
/* Pre Functional:*/
/* PreF01 */
true
.

msrop(outactAuthenticated,
   oeLogout,
   [post,Self
    ],
    []):-

 msrVar(ctState,TheSystem),
 msrVar(actAuthenticated,TheactAuthenticated),

 msrVar(ptString,AptStringMessageForTheactAuthenticated),

/* Post Functional:*/
 msrNav([Self],[rnActor],[TheactAuthenticated]),
 msrNav([Self],[rnActor,rnSystem],[TheSystem]),

/* PostF01 */
 msrNav([AptStringMessageForTheactAuthenticated],
    [eq,[[ptString,'You are logged out ! Good Bye ...']]],
    [[ptBoolean,true]]),
 msrNav([TheactAuthenticated],
    [rnInterfaceIN,
     ieMessage,[AptStringMessageForTheactAuthenticated]],
    [[ptBoolean,true]]),

 /* Post Protocol:*/
/* PostP01 */
msrNav([TheactAuthenticated],
    [rnctAuthenticated,msmAtPost,vpIsLogged],
    [[ptBoolean,false]])
.
```

Listing D.6: Prolog file outactAuthenticated-oeLogout.pl.

D.7 File ./src-gen/prolog-ref-spec/Operations/Environment/OUT/outactCom oeAlert.pl

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%-------------------------------------------------------
nico(A):-
 trace,
 write('here'),
 write('\n').

msrop(outactComCompany,
   oeAlert,
   [preProtocol,Self,
    AetHumanKind,
    AdtDate,
    AdtTime,
    AdtPhoneNumber,
    AdtGPSLocation,
    AdtComment
```

```
20     ],
21    []):-
22/* Pre Protocol:*/
23 msrVar(ctState,TheSystem),
24 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
25/* PreP01 */
26 msrNav([TheSystem],
27     [vpStarted],
28     [[ptBoolean,true]])
29 .
30
31msrop(outactComCompany,
32    oeAlert,
33    [preFunctional,Self,
34     AetHumanKind,
35     AdtDate,
36     AdtTime,
37     AdtPhoneNumber,
38     AdtGPSLocation,
39     AdtComment
40     ],
41    []):-
42/* Pre Functional:*/
43/* PreF01 */
44 msrVar(ctState,TheSystem),
45 msrNav([Self],
46      [msmAtPre,rnActor,rnSystem],
47      [TheSystem]),
48
49  ( msrNav([TheSystem],[clock,date,gt,[AdtDate]],[[ptBoolean,true]])
50   ; (msrNav([TheSystem],[clock,date,eq,[AdtDate]],[[ptBoolean,true]])
51    , msrNav([TheSystem],[clock,time,gt,[AdtTime]],[[ptBoolean,true]])
52    )
53 )
54.
55
56msrop(outactComCompany,
57    oeAlert,
58    [post,Self,
59     AetHumanKind,
60     AdtDate,
61     AdtTime,
62     AdtPhoneNumber,
63     AdtGPSLocation,
64     AdtComment
65     ],
66     []):-
67
68 msrVar(ctState,TheSystem),
69 msrVar(ctHuman,ActHuman),
70 msrVar(actComCompany,TheactComCompany),
71 msrVar(ctAlert,ActAlert),
72 msrVar(dtDateAndTime,AAlertInstant),
73 msrVar(etAlertStatus,AetAlertStatus),
74% msrVar(ctAlert,ActAlertNearBy),
75 msrVar(ctCrisis,ActCrisis),
76 msrVar(dtCrisisID,AdtCrisisID),
77% msrVar(etCrisisType,AetCrisisType),
78 msrVar(etCrisisStatus,AetCrisisStatus),
79 msrVar(dtDateAndTime,ACrisisInstant),
80 msrVar(dtComment,ACrisisdtComment),
81% msrVar(ptString,AptStringMessage),
82 msrVar(dtSMS,AdtSMS),
83 msrVar(dtAlertID,AdtAlertID),
84
85% msrVar(ptInteger,TheNextptIntegerValue),
86% msrVar(ptInteger,UpdatedNextptIntegerValue),
87% msrVar(inactComCompany,TheComCompanyIN),
88% msrVar(dtComment,TheCommentStored),
89% msrVar(dtString,TheCommentStoreddtString),
```

```
/* Post Functional:*/

msrNav([Self],[rnActor],[TheactComCompany]),
msrNav([Self],[rnActor,rnSystem],[TheSystem]),

/* PostF01 */
msrNav([TheSystem],
    [nextValueForAlertID],
    [PrenextValueForAlertID]),
msrNav([PrenextValueForAlertID],
    [add,[[dtInteger,[[value,[ptInteger,1]]],[]]]],
    [PostnextValueForAlertID]),
msrNav([TheSystem],
    [msmAtPost,nextValueForAlertID],
    [PostnextValueForAlertID]),

/* PostF02 */
msrNav([AAlertInstant],[date],[AdtDate]),
msrNav([AAlertInstant],[time],[AdtTime]),

msrNav([AetAlertStatus],
    [],
    [[etAlertStatus,pending]]),

msrNav([TheSystem],
    [nextValueForAlertID,
    todtString,[],eq,[AdtAlertID]],
    [[ptBoolean,true]]),

msrNav([ActAlert],
    [init,[AdtAlertID,
       AetAlertStatus,
       AdtGPSLocation,
       AAlertInstant,
       AdtComment]],
    [[ptBoolean,true]]),

/* PostF03 */
msrNav([TheSystem],
    [rnctAlert,
     msrSelect,location,isNearTo,[AdtGPSLocation]],
     ColctAlertsNearBy),

( (msrNav(ColctAlertsNearBy,
     [msrIsEmpty],
     [[ptBoolean,true]])
  )
-> (
   msrNav([TheSystem],
      [nextValueForCrisisID],
      [PrenextValueForCrisisID]),
   msrNav([PrenextValueForCrisisID],
      [add,[[dtInteger,[[value,[ptInteger,1]]],[]]]],
      [PostnextValueForCrisisID]),
   msrNav([TheSystem],
      [msmAtPost,nextValueForCrisisID],
      [PostnextValueForCrisisID]),

  msrNav([TheSystem],
      [nextValueForCrisisID,
      todtString,[],eq,[AdtCrisisID]],
      [[ptBoolean,true]]),

  msrNav([AdtCrisisType],[],[[etCrisisType,small]]),
  msrNav([AetCrisisStatus],[],[[etCrisisStatus,pending]]),
  msrNav([ACrisisInstant],[],[AAlertInstant]),
  msrNav([ACrisisdtComment],
      [value],
      [[ptString,'no reporting yet defined']]),
```

```
160   msrNav([ActCrisis],[init,[AdtCrisisID,
161                 AdtCrisisType,
162                 AetCrisisStatus,
163                 AdtGPSLocation,
164                 ACrisisInstant,
165                 ACrisisdtComment]],
166       [[ptBoolean,true]])
167
168    )
169 ; (
170   msrNav(ColctAlertsNearBy,
171      [rnTheCrisis,msrAny,msrTrue],
172      [ActCrisis])
173  )
174 ),
175
176 /* PostF04 */
177
178 msrNav([ActAlert],
179     [msmAtPost,rnTheCrisis],
180     [ActCrisis]),
181
182/* PostF05 */
183
184 msrNav([TheSystem],
185    [rnctHuman,
186     msrSelect,id,eq,[AdtPhoneNumber]],
187    HumanCol1),
188
189 msrNav(HumanCol1,
190    [msrSelect,kind,etEq,[AetHumanKind]],
191    HumanCol2),
192
193 (msrNav(HumanCol2,[msrIsEmpty],[[ptBoolean,true]])
194 -> (msrNav([ActHuman],
195       [init,[AdtPhoneNumber,AetHumanKind]],
196       [[ptBoolean,true]]),
197   msrNav([ActHuman],
198       [msmAtPost,rnactComCompany],
199       [TheactComCompany])
200   )
201 ; msrNav(HumanCol2,
202     [msrAny],
203     [ActHuman])
204 ),
205
206msrNav([ActHuman],
207    [rnSignaled,msrIncluding,[ActAlert]],
208    ColAlerts),
209
210msrNav([ActHuman],
211    [msmAtPost,rnSignaled],
212    ColAlerts),
213
214/* PostF06 */
215msrNav([AdtSMS],
216    [value],
217    [[ptString,'Your alert has been registered. We will handle it and keep you informed']]),
218msrNav([TheactComCompany],
219    [rnInterfaceIN,
220    ieSmsSend,[AdtPhoneNumber,
221          AdtSMS]],[[ptBoolean,true]]),
222
223/*
224
225 */
226
227 /* Post Protocol:*/
228/* PostP01 */
229 true
```

```
.
```

Listing D.7: Prolog file outactComCompany-oeAlert.pl.

D.8 File ./src-gen/prolog-ref-spec/Operations/Environment/OUT/outactCoo oeCloseCrisis.pl

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%-------------------------------------------------------
msrop(outactCoordinator,
   oeCloseCrisis,
   [preProtocol,Self,
    AdtCrisisID
    ],
   []):-
/* Pre Protocol:*/
 msrVar(ctState,TheSystem),
 msrVar(actCoordinator,TheActor),
 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
 msrNav([Self],[rnActor],[TheActor]),

/* PreP01 */
 msrNav([TheSystem],
    [vpStarted],
    [[ptBoolean,true]]),

/* PreP02 */
 msrNav([TheActor],
    [rnctAuthenticated,vpIsLogged],
    [[ptBoolean,true]])
.

msrop(outactCoordinator,
   oeCloseCrisis,
   [preFunctional,Self,
    AdtCrisisID
    ],
   []):-
/* Pre Functional:*/
 msrVar(ctState,TheSystem),
 msrVar(actCoordinator,TheActor),

 msrVar(dtCrisisID,AdtCrisisID),

 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
 msrNav([Self],[rnActor],[TheActor]),

/* PreF01 */
 msrNav([TheSystem],
    [rnctCrisis,
     msrSelect,
     id,eq,[AdtCrisisID]
    ],
    ColCrisis),

 msrNav(ColCrisis,
     [msrSize,eq,[[ptInteger,1]]],
     [[ptBoolean,true]])
 .

msrop(outactCoordinator,
   oeCloseCrisis,
   [post,Self,
    AdtCrisisID
    ],
```

```
    []):-

/* Post Functional:*/
msrVar(ctState,TheSystem),
msrVar(actCoordinator,TheActor),

msrVar(ctCrisis,TheCrisis),
msrVar(dtCrisisID,AdtCrisisID),

msrNav([Self],[rnActor,rnSystem],[TheSystem]),
msrNav([Self],[rnActor],[TheActor]),

/* PostF01 */
msrNav([TheSystem],
   [rnctCrisis,
    msrSelect,
    id,eq,[AdtCrisisID]],
   [TheCrisis]),

msrNav([TheCrisis],
    [msmAtPost,status],
    [[etCrisisStatus,closed]]),

/* PostF02 */
msrNav([TheCrisis],
    [msmAtPost,rnHandler],
    []),

 /* PostF03 */
msrNav([TheCrisis],
    [rnAlerts,msrForAll,msrIsKilled],
    [[ptBoolean,true]]),

/* PostF04 */
msrNav([TheActor],
    [rnInterfaceIN,
    ieMessage,[[ptString,'The crisis is now closed !']]
    ],
    [[ptBoolean,true]]),

/* Post Protocol:*/
/* PostP01 */
true
.
```

Listing D.8: Prolog file outactCoordinator-oeCloseCrisis.pl.

D.9 File ./src-gen/prolog-ref-spec/Operations/Environment/OUT/outact oeGetAlertsSet.pl

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%-------------------------------------------------------
msrop(outactCoordinator,
   oeGetAlertsSet,
   [preProtocol,Self,
    AetAlertStatus
    ],
   []):-
/* Pre Protocol:*/
msrVar(ctState,TheSystem),
msrVar(actCoordinator,TheActor),
msrNav([Self],[rnActor,rnSystem],[TheSystem]),
msrNav([Self],[rnActor],[TheActor]),

/* PreP01 */
```

```
msrNav([TheSystem],
   [vpStarted],
   [[ptBoolean,true]]),

msrNav([TheActor],
   [rnctAuthenticated,vpIsLogged],
   [[ptBoolean,true]])
.

msrop(outactCoordinator,
  oeGetAlertsSet,
  [preFunctional,Self,
   AetAlertStatus
   ],
  []):-
/* Pre Functional:*/
/* PreF01 */
true
 .

msrop(outactCoordinator,
  oeGetAlertsSet,
  [post,Self,
   AetAlertStatus
   ],
   []):-

/* Post Functional:*/
msrVar(ctState,TheSystem),
msrVar(actCoordinator,TheActor),
msrNav([Self],[rnActor,rnSystem],[TheSystem]),
msrNav([Self],[rnActor],[TheActor]),

/* PostF01 */
msrNav([TheSystem],
   [rnctAlert,
    msrSelect,
    status,etEq,[AetAlertStatus]],
   ColAlertSet),

msrNav(ColAlertSet,
    [msrForAll,isSentToCoordinator,[TheActor]],
    [[ptBoolean,true]]),

/* Post Protocol:*/
/* PostP01 */
true
 .
```

Listing D.9: Prolog file outactCoordinator-oeGetAlertsSet.pl.

D.10 File ./src-gen/prolog-ref-spec/Operations/Environment/OUT/outactCo oeGetCrisisSet.pl

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%-------------------------------------------------------
msrop(outactCoordinator,
  oeGetCrisisSet,
  [preProtocol,Self,
   AetCrisisStatus
   ],
  []):-
/* Pre Protocol:*/
msrVar(ctState,TheSystem),
msrVar(actCoordinator,TheActor),
```

```
msrNav([Self],[rnActor,rnSystem],[TheSystem]),
msrNav([Self],[rnActor],[TheActor]),

/* PreP01 */
msrNav([TheSystem],
    [vpStarted],
    [[ptBoolean,true]]),

msrNav([TheActor],
    [rnctAuthenticated,vpIsLogged],
    [[ptBoolean,true]])
.

msrop(outactCoordinator,
   oeGetCrisisSet,
   [preFunctional,Self,
    AetCrisisStatus
    ],
   []):-
/* Pre Functional:*/
/* PreF01 */
true
 .

msrop(outactCoordinator,
   oeGetCrisisSet,
   [post,Self,
    AetCrisisStatus
    ],
    []):-

/* Post Functional:*/
msrVar(ctState,TheSystem),
msrVar(actCoordinator,TheActor),
msrNav([Self],[rnActor,rnSystem],[TheSystem]),
msrNav([Self],[rnActor],[TheActor]),

/* PostF01 */
msrNav([TheSystem],
    [rnctCrisis,
     msrSelect,
     status,etEq,[AetCrisisStatus]],
    ColCrisisSet),

msrNav(ColCrisisSet,
     [msrForAll,isSentToCoordinator,[TheActor]],
     [[ptBoolean,true]]),

 /* Post Protocol:*/
/* PostP01 */
 true
 .
```

Listing D.10: Prolog file outactCoordinator-oeGetCrisisSet.pl.

D.11 File ./src-gen/prolog-ref-spec/Operations/Environment/OUT/outac oeInvalidateAlert.pl

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%-------------------------------------------------------
msrop(outactCoordinator,
   oeInvalidateAlert,
   [preProtocol,Self,
    AdtAlertID
    ],
```

```
  []):-
/* Pre Protocol:*/
msrVar(ctState,TheSystem),
msrVar(actCoordinator,TheActor),
msrNav([Self],[rnActor,rnSystem],[TheSystem]),
msrNav([Self],[rnActor],[TheActor]),

/* PreP01 */
msrNav([TheSystem],
   [vpStarted],
   [[ptBoolean,true]]),

/* PreP02 */
msrNav([TheActor],
   [rnctAuthenticated,vpIsLogged],
   [[ptBoolean,true]])
.

msrop(outactCoordinator,
  oeInvalidateAlert,
  [preFunctional,Self,
   AdtAlertID
   ],
  []):-
/* Pre Functional:*/
msrVar(ctState,TheSystem),
msrVar(actCoordinator,TheActor),

msrVar(dtAlertID,AdtAlertID),

msrNav([Self],[rnActor,rnSystem],[TheSystem]),
msrNav([Self],[rnActor],[TheActor]),

/* PreF01 */
msrNav([TheSystem],
   [rnctAlert,
    msrSelect,
    id,eq,[AdtAlertID]
   ],
   ColAlert),

msrNav(ColAlert,
    [msrSize,eq,[[ptInteger,1]]],
    [[ptBoolean,true]])
.

msrop(outactCoordinator,
  oeInvalidateAlert,
  [post,Self,
   AdtAlertID
   ],
   []):-

/* Post Functional:*/
msrVar(ctState,TheSystem),
msrVar(actCoordinator,TheActor),

msrVar(ctAlert,TheAlert),
msrVar(dtAlertID,AdtAlertID),

msrNav([Self],[rnActor,rnSystem],[TheSystem]),
msrNav([Self],[rnActor],[TheActor]),

/* PostF01 */
msrNav([TheSystem],
   [rnctAlert,
    msrSelect,
    id,eq,[AdtAlertID]],
   [TheAlert]),

```

```
msrNav([TheAlert],
    [msmAtPost,status],
    [[etAlertStatus,invalid]]),

/* PostF02 */
msrNav([TheActor],
    [rnInterfaceIN,
    ieMessage,[[ptString,'The alert is now declared as invalid !']]
    ],
    [[ptBoolean,true]]),

/* Post Protocol:*/
/* PostP01 */
true
.
```

Listing D.11: Prolog file outactCoordinator-oeInvalidateAlert.pl.

D.12 File ./src-gen/prolog-ref-spec/Operations/Environment/OUT/outac oeReportOnCrisis.pl

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%-------------------------------------------------------
msrop(outactCoordinator,
  oeReportOnCrisis,
  [preProtocol,Self,
   AdtCrisisID,
   AdtComment
   ],
  []):-
/* Pre Protocol:*/
msrVar(ctState,TheSystem),
msrVar(actCoordinator,TheActor),
msrNav([Self],[rnActor,rnSystem],[TheSystem]),
msrNav([Self],[rnActor],[TheActor]),

/* PreP01 */
msrNav([TheSystem],
   [vpStarted],
   [[ptBoolean,true]]),

msrNav([TheActor],
   [rnctAuthenticated,vpIsLogged],
   [[ptBoolean,true]])
.

msrop(outactCoordinator,
  oeReportOnCrisis,
  [preFunctional,Self,
   AdtCrisisID,
   AdtComment
   ],
  []):-
/* Pre Functional:*/
msrVar(ctState,TheSystem),
msrVar(actCoordinator,TheActor),

msrVar(dtCrisisID,AdtCrisisID),

msrNav([Self],[rnActor,rnSystem],[TheSystem]),
msrNav([Self],[rnActor],[TheActor]),

/* PreF01 */
msrNav([TheSystem],
   [rnctCrisis,
```

```
    msrSelect,
    id,eq,[AdtCrisisID]
   ],
   ColCrisis),

msrNav(ColCrisis,
    [msrSize,eq,[[ptInteger,1]]],
    [[ptBoolean,true]])
.

msrop(outactCoordinator,
  oeReportOnCrisis,
  [post,Self,
   AdtCrisisID,
   AdtComment
   ],
   []):-

/* Post Functional:*/
msrVar(ctState,TheSystem),
msrVar(actCoordinator,TheActor),

msrVar(ctCrisis,TheCrisis),
msrVar(dtCrisisID,AdtCrisisID),
msrVar(dtComment,AdtComment),

msrNav([Self],[rnActor,rnSystem],[TheSystem]),
msrNav([Self],[rnActor],[TheActor]),

/* PostF01 */
msrNav([TheSystem],
   [rnctCrisis,
    msrSelect,
    id,eq,[AdtCrisisID]],
   [TheCrisis]),

msrNav([TheCrisis],
    [msmAtPost,comment],
    [AdtComment]),

msrNav([TheActor],
    [rnInterfaceIN,
    ieMessage,[[ptString,'The crisis comment has been updated !']]
    ],
    [[ptBoolean,true]]),

/* Post Protocol:*/
/* PostP01 */
true
.
```

Listing D.12: Prolog file outactCoordinator-oeReportOnCrisis.pl.

D.13 File ./src-gen/prolog-ref-spec/Operations/Environment/OUT/outactCo oeSetCrisisHandler.pl

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%-------------------------------------------------------
msrop(outactCoordinator,
  oeSetCrisisHandler,
  [preProtocol,Self,
   AdtCrisisID
   ],
  []):-
/* Pre Protocol:*/
```

```
13 msrVar(ctState,TheSystem),
14 msrVar(actCoordinator,TheActor),
15 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
16 msrNav([Self],[rnActor],[TheActor]),
17
18/* PreP01 */
19 msrNav([TheSystem],
20    [vpStarted],
21    [[ptBoolean,true]]),
22
23 msrNav([TheActor],
24    [rnctAuthenticated,vpIsLogged],
25    [[ptBoolean,true]])
26.
27
28msrop(outactCoordinator,
29   oeSetCrisisHandler,
30   [preFunctional,Self,
31    AdtCrisisID
32    ],
33   []):-
34/* Pre Functional:*/
35 msrVar(ctState,TheSystem),
36 msrVar(actCoordinator,TheActor),
37
38 msrVar(dtCrisisID,AdtCrisisID),
39
40 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
41 msrNav([Self],[rnActor],[TheActor]),
42
43/* PreF01 */
44 msrNav([TheSystem],
45    [rnctCrisis,
46     msrSelect,
47     id,eq,[AdtCrisisID]
48    ],
49    ColCrisis),
50
51 msrNav(ColCrisis,
52     [msrSize,eq,[[ptInteger,1]]],
53     [[ptBoolean,true]])
54 .
55
56msrop(outactCoordinator,
57   oeSetCrisisHandler,
58   [post,Self,
59    AdtCrisisID
60    ],
61    []):-
62
63/* Post Functional:*/
64 msrVar(ctState,TheSystem),
65 msrVar(actCoordinator,TheActor),
66 msrVar(ctCoordinator,TheCoordinator),
67 msrVar(ctCoordinator,TheCurrentHandler),
68
69 msrVar(ctCrisis,TheCrisis),
70 msrVar(dtCrisisID,AdtCrisisID),
71
72 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
73 msrNav([Self],[rnActor],[TheActor]),
74
75/* PostF01 */
76 msrNav([TheSystem],
77    [rnctCrisis,
78     msrSelect,
79     id,eq,[AdtCrisisID]],
80    [TheCrisis]),
81
82 msrNav([TheCrisis],
```

```
    [msmAtPost,status],
    [[etCrisisStatus,handled]]),

msrNav([TheActor],
    [rnctCoordinator],
    [TheCoordinator]),
msrNav([TheCrisis],
    [msmAtPost,rnHandler],
    [TheCoordinator]),

msrNav([TheActor],
    [rnInterfaceIN,
    ieMessage,[[ptString,'You are now considered as handling the crisis !']]
    ],
    [[ptBoolean,true]]),

/* PostF02 */
msrNav([TheCrisis],
    [rnAlerts,msrForAll,isSentToCoordinator,[TheActor]],
    [[ptBoolean,true]]),

/* PostF03 */
( msrNav([TheCrisis],
     [rnHandler,msrSize,eq,[[ptInteger,1]]],
     [[ptBoolean,true]])
 -> (msrNav([TheCrisis],
       [rnHandler],
       [TheCurrentHandler]),
   msrNav([TheCurrentHandler],
       [rnactCoordinator,rnInterfaceIN,
       ieMessage,[[ptString,'One of the crisis you were handling is now handled by one of your
    colleagues!']]
       ],
       [[ptBoolean,true]])
   )
 ; true
),

 /* PostF04 */
msrNav([TheCrisis],
    [rnAlerts,rnSignaler,msrForAll,isAcknowledged,[]],
    [[ptBoolean,true]]),

/* Post Protocol:*/
/* PostP01 */
true
.
```

Listing D.13: Prolog file outactCoordinator-oeSetCrisisHandler.pl.

D.14 File ./src-gen/prolog-ref-spec/Operations/Environment/OUT/outactCo oeSetCrisisStatus.pl

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%-------------------------------------------------------
msrop(outactCoordinator,
   oeSetCrisisStatus,
   [preProtocol,Self,
    AdtCrisisID,
    AetCrisisStatus
    ],
   []):-
/* Pre Protocol:*/
msrVar(ctState,TheSystem),
msrVar(actCoordinator,TheActor),
```

```
16 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
17 msrNav([Self],[rnActor],[TheActor]),
18
19/* PreP01 */
20 msrNav([TheSystem],
21    [vpStarted],
22    [[ptBoolean,true]]),
23
24 msrNav([TheActor],
25    [rnctAuthenticated,vpIsLogged],
26    [[ptBoolean,true]])
27.
28
29msrop(outactCoordinator,
30   oeSetCrisisStatus,
31   [preFunctional,Self,
32    AdtCrisisID,
33    AetCrisisStatus
34    ],
35   []):-
36/* Pre Functional:*/
37 msrVar(ctState,TheSystem),
38 msrVar(actCoordinator,TheActor),
39
40 msrVar(dtCrisisID,AdtCrisisID),
41
42 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
43 msrNav([Self],[rnActor],[TheActor]),
44
45/* PreF01 */
46 msrNav([TheSystem],
47    [rnctCrisis,
48     msrSelect,
49     id,eq,[AdtCrisisID]
50    ],
51    ColCrisis),
52
53 msrNav(ColCrisis,
54     [msrSize,eq,[[ptInteger,1]]],
55     [[ptBoolean,true]])
56 .
57
58msrop(outactCoordinator,
59   oeSetCrisisStatus,
60   [post,Self,
61    AdtCrisisID,
62    AetCrisisStatus
63    ],
64    []):-
65
66/* Post Functional:*/
67 msrVar(ctState,TheSystem),
68 msrVar(actCoordinator,TheActor),
69
70 msrVar(ctCrisis,TheCrisis),
71 msrVar(dtCrisisID,AdtCrisisID),
72 msrVar(etCrisisStatus,AetCrisisStatus),
73
74 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
75 msrNav([Self],[rnActor],[TheActor]),
76
77/* PostF01 */
78 msrNav([TheSystem],
79    [rnctCrisis,
80     msrSelect,
81     id,eq,[AdtCrisisID]],
82    [TheCrisis]),
83
84 msrNav([TheCrisis],
85     [msmAtPost,status],
```

```
    [AetCrisisStatus]),

msrNav([TheActor],
    [rnInterfaceIN,
    ieMessage,[[ptString,'The crisis status has been updated !']]
    ],
    [[ptBoolean,true]]),

/* Post Protocol:*/
/* PostP01 */
true
.
```

Listing D.14: Prolog file outactCoordinator-oeSetCrisisStatus.pl.

D.15 File ./src-gen/prolog-ref-spec/Operations/Environment/OUT/outactCo oeSetCrisisType.pl

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%-----------------------------------------------------
msrop(outactCoordinator,
  oeSetCrisisType,
  [preProtocol,Self,
   AdtCrisisID,
   AetCrisisType
   ],
  []):-
/* Pre Protocol:*/
msrVar(ctState,TheSystem),
msrVar(actCoordinator,TheActor),
msrNav([Self],[rnActor,rnSystem],[TheSystem]),
msrNav([Self],[rnActor],[TheActor]),

/* PreP01 */
msrNav([TheSystem],
   [vpStarted],
   [[ptBoolean,true]]),

msrNav([TheActor],
   [rnctAuthenticated,vpIsLogged],
   [[ptBoolean,true]])
.

msrop(outactCoordinator,
  oeSetCrisisType,
  [preFunctional,Self,
   AdtCrisisID,
   AetCrisisType
   ],
  []):-
/* Pre Functional:*/
msrVar(ctState,TheSystem),
msrVar(actCoordinator,TheActor),

msrVar(dtCrisisID,AdtCrisisID),

msrNav([Self],[rnActor,rnSystem],[TheSystem]),
msrNav([Self],[rnActor],[TheActor]),

/* PreF01 */
msrNav([TheSystem],
   [rnctCrisis,
    msrSelect,
    id,eq,[AdtCrisisID]
   ],
```

```
51    ColCrisis),
52
53 msrNav(ColCrisis,
54     [msrSize,eq,[[ptInteger,1]]],
55     [[ptBoolean,true]])
56 .
57
58msrop(outactCoordinator,
59   oeSetCrisisType,
60   [post,Self,
61    AdtCrisisID,
62    AetCrisisType
63    ],
64    []):-
65
66/* Post Functional:*/
67 msrVar(ctState,TheSystem),
68 msrVar(actCoordinator,TheActor),
69
70 msrVar(ctCrisis,TheCrisis),
71 msrVar(dtCrisisID,AdtCrisisID),
72 msrVar(etCrisisType,AetCrisisType),
73
74 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
75 msrNav([Self],[rnActor],[TheActor]),
76
77/* PostF01 */
78 msrNav([TheSystem],
79    [rnctCrisis,
80     msrSelect,
81     id,eq,[AdtCrisisID]],
82    [TheCrisis]),
83
84 msrNav([TheCrisis],
85     [msmAtPost,type],
86     [AetCrisisType]),
87
88 msrNav([TheActor],
89     [rnInterfaceIN,
90     ieMessage,[[ptString,'The crisis type has been updated !']]
91     ],
92     [[ptBoolean,true]]),
93
94/* Post Protocol:*/
95/* PostP01 */
96 true
97 .
```

Listing D.15: Prolog file outactCoordinator-oeSetCrisisType.pl.

D.16 File ./src-gen/prolog-ref-spec/Operations/Environment/OUT/outac oeValidateAlert.pl

```
1%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5%-----------------------------------------------------
6msrop(outactCoordinator,
7   oeValidateAlert,
8   [preProtocol,Self,
9    AdtAlertID
10    ],
11   []):-
12/* Pre Protocol:*/
13 msrVar(ctState,TheSystem),
14 msrVar(actCoordinator,TheActor),
15 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
```

```
16 msrNav([Self],[rnActor],[TheActor]),
17
18/* PreP01 */
19 msrNav([TheSystem],
20    [vpStarted],
21    [[ptBoolean,true]]),
22
23 msrNav([TheActor],
24    [rnctAuthenticated,vpIsLogged],
25    [[ptBoolean,true]])
26.
27
28msrop(outactCoordinator,
29   oeValidateAlert,
30   [preFunctional,Self,
31    AdtAlertID
32    ],
33   []):-
34/* Pre Functional:*/
35 msrVar(ctState,TheSystem),
36 msrVar(actCoordinator,TheActor),
37
38 msrVar(dtAlertID,AdtAlertID),
39
40 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
41 msrNav([Self],[rnActor],[TheActor]),
42
43/* PreF01 */
44 msrNav([TheSystem],
45    [rnctAlert,
46     msrSelect,
47     id,eq,[AdtAlertID]
48    ],
49    ColAlerts),
50
51 msrNav(ColAlerts,
52     [msrSize,eq,[[ptInteger,1]]],
53     [[ptBoolean,true]])
54 .
55
56msrop(outactCoordinator,
57   oeValidateAlert,
58   [post,Self,
59    AdtAlertID
60    ],
61    []):-
62
63/* Post Functional:*/
64 msrVar(ctState,TheSystem),
65 msrVar(actCoordinator,TheActor),
66
67 msrVar(ctAlert,TheAlert),
68 msrVar(dtAlertID,AdtAlertID),
69
70 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
71 msrNav([Self],[rnActor],[TheActor]),
72
73/* PostF01 */
74 msrNav([TheSystem],
75    [rnctAlert,
76     msrSelect,
77     id,eq,[AdtAlertID]],
78    [TheAlert]),
79
80 msrNav([TheAlert],
81     [msmAtPost,status],
82     [[etAlertStatus,valid]]),
83
84 msrNav([TheActor],
85     [rnInterfaceIN,
```

```
    ieMessage,[[ptString,'The Alert is now declared as valid !']]
    ],
    [[ptBoolean,true]]),

/* Post Protocol:*/
/* PostP01 */
true
.
```

Listing D.16: Prolog file outactCoordinator-oeValidateAlert.pl.

D.17 File ./src-gen/prolog-ref-spec/Operations/Environment/OUT/outac oeCreateSystemAndEnvironment.pl

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/*
/****************************************
MSRCreatorActor
****************************************/

/*** createSystemAndEnvironment ***/

msrop(outactMsrCreator,
  oeCreateSystemAndEnvironment,
  [preFunctional,_Self,_AqtyComCompanies],
  []):-
true.

msrop(outactMsrCreator,
  oeCreateSystemAndEnvironment,
  [preProtocol,_Self,_AqtyComCompanies],
  []):-
true.

msrop(outactMsrCreator,
  oeCreateSystemAndEnvironment,
  [post,_Self,AqtyComCompanies],
  []):-

msrVar(ctState,TheSystem),
msrVar(actMsrCreator,AactMsrCreator),
msrVar(actAdministrator,AactAdministrator),

msrVar(dtInteger, AnextValueForAlertID),
msrVar(dtInteger, AnextValueForCrisisID),
msrVar(dtDateAndTime, Aclock),
msrVar(dtSecond, AcrisisReminderPeriod),
msrVar(dtSecond, AmaxCrisisReminderPeriod),
msrVar(ptBoolean, AvpStarted),

/* PostF01 -- MUST ALWAYS BE MADE FIRST -- */
msrNav([AnextValueForAlertID],
    [value,eq,[[ptInteger,1]]],
    [[ptBoolean,true]]),

msrNav([AnextValueForCrisisID],
    [value,eq,[[ptInteger,1]]],
    [[ptBoolean,true]]),

msrNav([Aclock],
   [date,year,value],
   [[ptInteger,1970]]),
msrNav([Aclock],
   [date,month,value],
   [[ptInteger,01]]),
```

```
55msrNav([Aclock],
56    [date,day,value],
57    [[ptInteger,01]]),
58
59msrNav([Aclock],
60    [time,hour,value],
61    [[ptInteger,00]]),
62msrNav([Aclock],
63    [time,minute,value],
64    [[ptInteger,00]]),
65msrNav([Aclock],
66    [time,second,value],
67    [[ptInteger,00]]),
68
69 msrNav([AcrisisReminderPeriod],
70     [value,eq,[[ptInteger,300]]],
71     [[ptBoolean,true]]),
72
73 msrNav([AmaxCrisisReminderPeriod],
74     [value,eq,[[ptInteger,1200]]],
75     [[ptBoolean,true]]),
76
77 msrNav([AvpStarted],
78     [],
79     [[ptBoolean,true]]),
80
81 msrNav([TheSystem],
82     [init,[AnextValueForAlertID,
83        AnextValueForCrisisID,
84        Aclock,
85        AcrisisReminderPeriod,
86        AmaxCrisisReminderPeriod,
87        Aclock,
88        AvpStarted]
89        ],
90     [[ptBoolean,true]]),
91
92/* PostF02*/
93 msrNav([AactMsrCreator],
94     [init,[]],
95     [[ptBoolean,true]]),
96
97 /* PostF03 */
98 msrVarCol(actComCompany,AqtyComCompanies,AactComCompanyCol),
99
100 msrNav(AactComCompanyCol,
101     [msrForAll,init,[]],
102     [[ptBoolean,true]]),
103
104 /* PostF04*/
105 msrNav([AactAdministrator],
106     [init,[]],
107     [[ptBoolean,true]]),
108
109 /* PostF05*/
110 msrVar(actActivator,AactActivator),
111 msrNav([AactActivator],
112     [init,[]],
113     [[ptBoolean,true]]),
114
115/* PostF06 */
116 msrVar(ctAdministrator,ActAdministrator),
117 msrVar(dtLogin,AdtLogin),
118 msrVar(dtPassword,AdtPassword),
119
120 msrNav([AdtLogin],
121     [value,eq,[[ptString,'icrashadmin']]],
122     [[ptBoolean,true]]),
123
124 msrNav([AdtPassword],
```

```
    [value,eq,[[ptString,'7WXC1359']]],
    [[ptBoolean,true]]),

msrNav([ActAdministrator],
    [init,[AdtLogin,AdtPassword]],
    [[ptBoolean,true]]),

/* PostF07*/
msrNav([ActAdministrator],
    [msmAtPost,rnactAuthenticated],
    [AactAdministrator]),

/* Post Protocol:*/
/* PostP01 */
true
.
```

Listing D.17: Prolog file outactMsrCreator-oeCreateSystemAndEnvironment.pl.

D.18 File ./src-gen/prolog-ref-spec/Operations/Concepts/PrimaryTypesC ctAdministrator-init.pl

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

msrop(ctAdministrator,init,[Self,
          Alogin,
          Apwd],
   Result):-
(
msrVar(ctAdministrator,Self),

/* Post F01 */
msrNav([Self],[login],[Alogin]),
msrNav([Self],[pwd],[Apwd]),
msrNav([Self],[vpIsLogged],[[ptBoolean,false]]),

/* Post F02 */
 msrNav([Self],[msrIsNew],[Self])
)
-> Result = [ptBoolean,true]
; Result = [ptBoolean,false]
.
```

Listing D.18: Prolog file PrimaryTypesClasses-ctAdministrator-init.pl.

D.19 File ./src-gen/prolog-ref-spec/Operations/Concepts/PrimaryTypesC ctAlert-init.pl

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

msrop(ctAlert,init,[Self,
          Aid,
          Astatus,
          Alocation,
          Ainstant,
          Acomment],
   Result):-

/* Post F01 */
(
```

```
msrVar(ctAlert,Self),

msrNav([Self],[id],[Aid]),
msrNav([Self],[status],[Astatus]),
msrNav([Self],[location],[Alocation]),
msrNav([Self],[instant],[Ainstant]),
msrNav([Self],[comment],[Acomment]),

/* Post F02 */
 msrNav([Self],[msrIsNew],[Self])
)
-> Result = [ptBoolean,true]
; Result = [ptBoolean,false]
.
```

Listing D.19: Prolog file PrimaryTypesClasses-ctAlert-init.pl.

D.20 File ./src-gen/prolog-ref-spec/Operations/Concepts/PrimaryTypesClass ctAlert-isSentToCoordinator.pl

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

msrop(ctAlert,isSentToCoordinator,[Self,AactCoordinator],
   Result):-

/* Post F01 */
(
 msrNav([AactCoordinator],
     [rnInterfaceIN,ieSendAnAlert,[Self]],
     [[ptBoolean,true]])
)
-> Result = [ptBoolean,true]
; Result = [ptBoolean,false]
.
```

Listing D.20: Prolog file PrimaryTypesClasses-ctAlert-isSentToCoordinator.pl.

D.21 File ./src-gen/prolog-ref-spec/Operations/Concepts/PrimaryTypesClass ctAuthenticated-init.pl

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

msrop(ctAuthenticated,init,[Self,
              Alogin,
              Apwd],
   Result):-

/* Post F01 */
(
msrVar(ctAuthenticated,Self),

msrNav([Self],[login],[Alogin]),
msrNav([Self],[pwd],[Apwd]),
msrNav([Self],[vpIsLogged],[[ptBoolean,false]]),

/* Post F02 */
 msrNav([Self],[msrIsNew],[Self])
)
-> Result = [ptBoolean,true]
; Result = [ptBoolean,false]
```

```
.
```

Listing D.21: Prolog file PrimaryTypesClasses-ctAuthenticated-init.pl.

D.22 File ./src-gen/prolog-ref-spec/Operations/Concepts/PrimaryTypesC ctCoordinator-init.pl

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

msrop(ctCoordinator,init,[Self,
          Aid,
          Alogin,
          Apwd],
   Result):-

/* Post F01 */
(
msrVar(ctCoordinator,Self),

msrNav([Self],[id],[Aid]),
msrNav([Self],[login],[Alogin]),
msrNav([Self],[pwd],[Apwd]),
msrNav([Self],[vpIsLogged],[[ptBoolean,false]]),

/* Post F02 */
 msrNav([Self],[msrIsNew],[Self])
)
-> Result = [ptBoolean,true]
; Result = [ptBoolean,false]
.
```

Listing D.22: Prolog file PrimaryTypesClasses-ctCoordinator-init.pl.

D.23 File ./src-gen/prolog-ref-spec/Operations/Concepts/PrimaryTypesC ctCrisis-handlingDelayPassed.pl

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

msrop(ctCrisis,handlingDelayPassed,[Self],
   Result):-

/* Post F01 */
(
 msrVar(ctState,TheSystem),
 msrVar(dtInteger,CurrentClockSecondsQty),
 msrVar(dtInteger,LastReminderSecondsQty),
 msrVar(dtSecond,CrisisReminderPeriod),

 msrNav([Self],[rnSystem],[TheSystem]),

 msrNav([Self],
     [status],
     [[etCrisisStatus,pending]]),

 msrNav([TheSystem],
    [clock,toSecondsQty,[]],
    [CurrentClockSecondsQty]),

 msrNav([TheSystem],
    [vpLastReminder,toSecondsQty,[]],
```

```
    [LastReminderSecondsQty]),

msrNav([TheSystem],
    [crisisReminderPeriod],
    [CrisisReminderPeriod]),

msrNav([CurrentClockSecondsQty],
    [sub,[LastReminderSecondsQty],
     gt, [CrisisReminderPeriod]
     ],
    [[ptBoolean,true]])

)
-> Result = [ptBoolean,true]
; Result = [ptBoolean,false]
.
```

Listing D.23: Prolog file PrimaryTypesClasses-ctCrisis-handlingDelayPassed.pl.

D.24 File ./src-gen/prolog-ref-spec/Operations/Concepts/PrimaryTypesClass ctCrisis-init.pl

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

msrop(ctCrisis,init,[Self,
          Aid,
          Atype,
          Astatus,
          Alocation,
          Ainstant,
          Acomment],
   Result):-

/* Post F01 */
(
msrVar(ctCrisis,Self),

msrNav([Self],[id],[Aid]),
msrNav([Self],[type],[Atype]),
msrNav([Self],[status],[Astatus]),
msrNav([Self],[location],[Alocation]),
msrNav([Self],[instant],[Ainstant]),
msrNav([Self],[comment],[Acomment]),

/* Post F02 */
 msrNav([Self],[msrIsNew],[Self])
)
-> Result = [ptBoolean,true]
; Result = [ptBoolean,false]
.
```

Listing D.24: Prolog file PrimaryTypesClasses-ctCrisis-init.pl.

D.25 File ./src-gen/prolog-ref-spec/Operations/Concepts/PrimaryTypesClass ctCrisis-isAllocatedIfPossible.pl

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

msrop(ctCrisis,isAllocatedIfPossible,[Self],
   Result):-
```

```
(
msrVar(ctState,TheSystem),
msrNav([Self],[rnSystem],[TheSystem]),

msrVar(actCoordinator,TheCoordinatorActor),
msrVar(ctCoordinator,TheCoordinator),
msrVar(ptString,TheMessage),
msrVar(ptString,TheCrisisIDptString),

 (
 /* Post F01 */
msrNav([Self],
    [maxHandlingDelayPassed,[]],
    [[ptBoolean,true]]),

 ( msrNav([TheSystem],
     [rnactCoordinator,msrIsEmpty],
     [[ptBoolean,false]])
  -> (
    /* Post F02 */
    msrNav([TheSystem],
       [rnactCoordinator,msrAny,msrTrue],
       [TheCoordinatorActor]),

    msrNav([TheCoordinatorActor],
        [rnctCoordinator],
        [TheCoordinator]),

    msrNav([Self],
        [msmAtPost,rnHandler],
        [TheCoordinator]),

     msrNav([Self],
        [msmAtPost,status],
        [[etCrisisStatus,handled]]),

     msrNav([Self],
        [id,value],
        [TheCrisisIDptString]),

    msrNav([[ptString,'You are now considered as handling the crisis having ID: ']],
        [ptStringConcat,[TheCrisisIDptString]],
        [TheMessage]),

    msrNav([TheCoordinatorActor],
        [rnInterfaceIN,
        ieMessage,[TheMessage]
        ],
        [[ptBoolean,true]])
   )
 ; ( /* Post F03 */
   msrNav([TheSystem],
      [rnactAdministrator,msrForAll,rnInterfaceIN,
      ieMessage,[[ptString,'Please add new coordinators to handle pending crisis !']]],
      [[ptBoolean,true]])
  )
 )
 )
)
-> Result = [ptBoolean,true]
; Result = [ptBoolean,false]
.
```

Listing D.25: Prolog file PrimaryTypesClasses-ctCrisis-isAllocatedIfPossible.pl.

D.26 File ./src-gen/prolog-ref-spec/Operations/Concepts/PrimaryTypesC ctCrisis-isSentToCoordinator.pl

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

msrop(ctCrisis,isSentToCoordinator,[Self,AactCoordinator],
   Result):-

/* Post F01 */
(
 msrNav([AactCoordinator],
     [rnInterfaceIN,ieSendACrisis,[Self]],
     [[ptBoolean,true]])
)
-> Result = [ptBoolean,true]
; Result = [ptBoolean,false]
.
```

Listing D.26: Prolog file PrimaryTypesClasses-ctCrisis-isSentToCoordinator.pl.

D.27 File ./src-gen/prolog-ref-spec/Operations/Concepts/PrimaryTypesClass ctCrisis-maxHandlingDelayPassed.pl

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

msrop(ctCrisis,maxHandlingDelayPassed,[Self],
   Result):-

/* Post F01 */
(
 msrVar(ctState,TheSystem),
 msrVar(dtInteger,CurrentClockSecondsQty),
 msrVar(dtInteger,CrisisInstantSecondsQty),
 msrVar(dtSecond,MaxCrisisReminderPeriod),

 msrNav([Self],[rnSystem],[TheSystem]),

 msrNav([Self],
     [status],
     [[etCrisisStatus,pending]]),

 msrNav([TheSystem],
    [clock,toSecondsQty,[]],
    [CurrentClockSecondsQty]),

 msrNav([Self],
    [instant,toSecondsQty,[]],
    [CrisisInstantSecondsQty]),

 msrNav([TheSystem],
    [maxCrisisReminderPeriod],
    [MaxCrisisReminderPeriod]),

 msrNav([CurrentClockSecondsQty],
    [sub,[CrisisInstantSecondsQty],
     gt, [MaxCrisisReminderPeriod]
     ],
    [[ptBoolean,true]])

)
-> Result = [ptBoolean,true]
; Result = [ptBoolean,false]
.
```

Listing D.27: Prolog file PrimaryTypesClasses-ctCrisis-maxHandlingDelayPassed.pl.

D.28 File ./src-gen/prolog-ref-spec/Operations/Concepts/PrimaryTypesC ctHuman-init.pl

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

msrop(ctHuman,init,[Self,
          Aid,
          Akind],
   Result):-

/* Post F01 */
(
msrVar(ctHuman,Self),

msrNav([Self],[id],[Aid]),
msrNav([Self],[kind],[Akind]),

/* Post F02 */
 msrNav([Self],[msrIsNew],[Self])
)
-> Result = [ptBoolean,true]
; Result = [ptBoolean,false]
.
```

Listing D.28: Prolog file PrimaryTypesClasses-ctHuman-init.pl.

D.29 File ./src-gen/prolog-ref-spec/Operations/Concepts/PrimaryTypesC ctHuman-isAcknowledged.pl

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

msrop(ctHuman,isAcknowledged,[Self],Result):-

/* Post F01 */
(msrVar(dtPhoneNumber,AdtPhoneNumber),
 msrVar(dtSMS,AdtSMS),

 msrNav([Self],
     [id,eq,[AdtPhoneNumber]],
     [[ptBoolean,true]]),
 msrNav([AdtSMS],
     [value,eq,[[ptString,'The handling of your alert by our services is in progress !']]],
     [[ptBoolean,true]]),
 msrNav([Self],
     [rnactComCompany,rnInterfaceIN,ieSmsSend,[AdtPhoneNumber,AdtSMS]],
     [[ptBoolean,true]])
)
-> Result = [ptBoolean,true]
; Result = [ptBoolean,false]
.
```

Listing D.29: Prolog file PrimaryTypesClasses-ctHuman-isAcknowledged.pl.

D.30 File ./src-gen/prolog-ref-spec/Operations/Concepts/PrimaryTypesC ctState-init.pl

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

msrop(ctState,init,[Self,
          AnextValueForAlertID,
          AnextValueForCrisisID,
          Aclock,
          AcrisisReminderPeriod,
          AmaxCrisisReminderPeriod,
          AvpLastReminder,
          AvpStarted],
   Result):-

 /* Post F01 */
(
 msrVar(ctState,Self),

 msrNav([Self],[nextValueForAlertID],[AnextValueForAlertID]),
 msrNav([Self],[nextValueForCrisisID],[AnextValueForCrisisID]),
 msrNav([Self],[clock],[Aclock]),
 msrNav([Self],[crisisReminderPeriod],[AcrisisReminderPeriod]),
 msrNav([Self],[maxCrisisReminderPeriod],[AmaxCrisisReminderPeriod]),
 msrNav([Self],[vpLastReminder],[AvpLastReminder]),
 msrNav([Self],[vpStarted],[AvpStarted]),

 msrNav([Self],[msrIsNew],[Self])
)
-> Result = [ptBoolean,true]
; Result = [ptBoolean,false]
.
```

Listing D.30: Prolog file PrimaryTypesClasses-ctState-init.pl.

D.31 File ./src-gen/prolog-ref-spec/Operations/Concepts/PrimaryTypesData dtAlertID-is.pl

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

msrop(dtAlertID,is,[AdtValue],Result):-
% msd01
msrVar(ptBoolean,TheResult),
(
 ( msrNav([AdtValue],
    [value,length,[],gt,[[ptInteger,0]]],
    [[ptBoolean,true]]),
   msrNav([AdtValue],
    [value,length,[],leq,[[ptInteger,20]]],
    [[ptBoolean,true]])
 )
 -> (TheResult = [ptBoolean,true])
 ; (TheResult = [ptBoolean,false])
),
TheResult = Result
.

/*
| ?- X = [dtAlertID,[],[[dtString,[[value,[ptString,'0123456789']]],[]]]],
msrNav([X],[is,[]],[Result]).

X = [dtAlertID,[],[[dtString,[[value,[ptString,'0123456789']]],[]]]],
Result = [ptBoolean,true] ?

yes

| ?- X = [dtAlertID,[],[[dtString,[[value,[ptString,'01234567890123456789012345678 9']]],[]]]],
msrNav([X],[is,[]],[Result]).
```

```

X = [dtAlertID,[],[[dtString,[[value,[ptString,'0123456789012345678901234567 89']]],[]]]],
Result = [ptBoolean,false] ?

yes
*/
```

Listing D.31: Prolog file PrimaryTypesDatatypes-dtAlertID-is.pl.

D.32 File ./src-gen/prolog-ref-spec/Operations/Concepts/PrimaryTypesD dtComment-is.pl

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% dtComment

%msd01
msrop(dtComment,is,[AdtValue],Result):-
 msrVar(ptBoolean,TheResult),
 msrVar(ptInteger,MaxLength),
(
  (
    (
     MaxLength = [ptInteger,160],
     msrNav([AdtValue],
        [value,length,[],leq,[MaxLength]],
        [[ptBoolean,true]])
    )
     -> TheResult = [ptBoolean,true]
     ; TheResult = [ptBoolean,false]
  )
),
 Result = TheResult
.

/*
| ?- X = [dtComment,[],[[dtString,[[value,[ptString,'I broke my leg ! Please help ...']]],[]]]],
msrNav([X],[is,[]],[Result]).
X = [dtComment,[],[[dtString,[[value,[ptString,'I broke my leg ! Please help ...']]],[]]]],
Result = [ptBoolean,true] ?
yes

| ?- X = [dtComment,[],[[dtString,[[value,[ptString,'I broke my leg when I was running with my dog
      to go to the skate park because my friends called me on my mobile phone and told me that a skate
       star was doing triple back flips.']]],[]]]],
msrNav([X],[is,[]],[Result]).
X = [dtComment,[],[[dtString,[[value,[ptString,'I broke my leg when I was running with my dog to go
      to the skate park because my friends called me on my mobile phone and told me that a skate star
      was doing triple back flips.']]],[]]]],
Result = [ptBoolean,false] ?
yes
*/
```

Listing D.32: Prolog file PrimaryTypesDatatypes-dtComment-is.pl.

D.33 File ./src-gen/prolog-ref-spec/Operations/Concepts/PrimaryTypesD dtCoordinatorID-is.pl

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

msrop(dtCoordinatorID,is,[AdtValue],Result):-
```

```
% msd01
 msrVar(ptBoolean,TheResult),
(
 ( msrNav([AdtValue],
    [value,length,[],gt,[[ptInteger,0]]],
    [[ptBoolean,true]]),
   msrNav([AdtValue],
    [value,length,[],leq,[[ptInteger,5]]],
    [[ptBoolean,true]])
 )
 -> (TheResult = [ptBoolean,true])
 ; (TheResult = [ptBoolean,false])
),
TheResult = Result
.
```

Listing D.33: Prolog file PrimaryTypesDatatypes-dtCoordinatorID-is.pl.

D.34 File ./src-gen/prolog-ref-spec/Operations/Concepts/PrimaryTypesData dtCrisisID-is.pl

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

msrop(dtCrisisID,is,[AdtValue],Result):-
% msd01
 msrVar(ptBoolean,TheResult),
(
 ( msrNav([AdtValue],
    [value,length,[],gt,[[ptInteger,0]]],
    [[ptBoolean,true]]),
   msrNav([AdtValue],
    [value,length,[],leq,[[ptInteger,10]]],
    [[ptBoolean,true]])
 )
 -> (TheResult = [ptBoolean,true])
 ; (TheResult = [ptBoolean,false])
),
TheResult = Result
.
/*
| ?- X = [dtCrisisID,[],[[dtString,[[value,[ptString,'0123456789']]],[]]]],
msrNav([X],[is,[]],[Result]).
X = [dtCrisisID,[],[[dtString,[[value,[ptString,'0123456789']]],[]]]],
Result = [ptBoolean,true] ?
yes

| ?- X = [dtCrisisID,[],[[dtString,[[value,[ptString,'0123456789a']]],[]]]],
msrNav([X],[is,[]],[Result]).
X = [dtCrisisID,[],[[dtString,[[value,[ptString,'0123456789a']]],[]]]],
Result = [ptBoolean,false] ?
yes
*/
```

Listing D.34: Prolog file PrimaryTypesDatatypes-dtCrisisID-is.pl.

D.35 File ./src-gen/prolog-ref-spec/Operations/Concepts/PrimaryTypesData dtGPSLocation-is.pl

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```
%% dtPhoneNumber

% msd01
msrop(dtGPSLocation,is,[AdtValue],Result):-
msrVar(ptBoolean,TheResult),
(
  (
     msrNav([AdtValue],
        [latitude,is,[]],
        [[ptBoolean,true]]),
     msrNav([AdtValue],
        [longitude,is,[]],
        [[ptBoolean,true]])
  )
 -> TheResult = [ptBoolean,true]
 ; TheResult = [ptBoolean,false]
),

 Result = TheResult
.
```

Listing D.35: Prolog file PrimaryTypesDatatypes-dtGPSLocation-is.pl.

D.36 File ./src-gen/prolog-ref-spec/Operations/Concepts/PrimaryTypesD dtGPSLocation-isNearTo.pl

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% dtGPSLocation

msrop(dtGPSLocation,isNearTo,[Self,AdtValue],Result):-
msrVar(ptBoolean,TheResult),
msrVar(dtReal,EarthRadius),
msrVar(dtReal,MaxDistance),

msrVar(dtLatitude,ComparedLatitude),
msrVar(dtLongitude,ComparedLongitude),

msrVar(dtReal,R1),msrVar(dtReal,R1a),
msrVar(dtReal,R2),msrVar(dtReal,R2a),

(
  (
    (
     % msd01
     msrNav([EarthRadius],[value],[[ptReal,6371]]),
     msrNav([MaxDistance],[value],[[ptReal,100]]),

     msrNav([AdtValue],[latitude],[ComparedLatitude]),
     msrNav([AdtValue],[longitude],[ComparedLongitude]),

     msrNav([Self],[latitude,sin,[]],[R1a]),
     msrNav([AdtValue],[latitude,sin,[],mul,[R1a]],[R1]),

     msrNav([Self],[latitude,cos,[]],[R2a]),
     msrNav([AdtValue],[latitude,cos,[],mul,[R2a]],[R2]),

     msrNav([AdtValue],[longitude],[ComparedLongitude]),
     msrNav([Self],[longitude,sub,[ComparedLongitude],cos,[],mul,[R2],
            add,[R1],
            acos,[],
            mul,[EarthRadius],
            sub,[MaxDistance],
            value,leq,[[ptReal,0]]],
        [[ptBoolean,true]])
```

```
    )
     -> TheResult = [ptBoolean,true]
     ; TheResult = [ptBoolean,false]
  )
),
 Result = TheResult
.
```

Listing D.36: Prolog file PrimaryTypesDatatypes-dtGPSLocation-isNearTo.pl.

D.37 File ./src-gen/prolog-ref-spec/Operations/Concepts/PrimaryTypesData dtLatitude-is.pl

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% msd01
msrop(dtLatitude,is,[AdtValue],Result):-
msrVar(ptBoolean,TheResult),
(
 ( msrNav([AdtValue],
    [value,geq,[[ptReal,-90.0]]],
    [[ptBoolean,true]]),
   msrNav([AdtValue],
    [value,leq,[[ptReal,+90.0]]],
    [[ptBoolean,true]])
 )
 -> (TheResult = [ptBoolean,true])
 ; (TheResult = [ptBoolean,false])
),
Result = TheResult
.
```

Listing D.37: Prolog file PrimaryTypesDatatypes-dtLatitude-is.pl.

D.38 File ./src-gen/prolog-ref-spec/Operations/Concepts/PrimaryTypesData dtLogin-is.pl

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% dtComment

%msd01
msrop(dtLogin,is,[AdtValue],Result):-
 msrVar(ptBoolean,TheResult),
 msrVar(ptInteger,MaxLength),
(
  (
    (
     MaxLength = [ptInteger,20],
     msrNav([AdtValue],
        [value,length,[],leq,[MaxLength]],
        [[ptBoolean,true]])
    )
     -> TheResult = [ptBoolean,true]
     ; TheResult = [ptBoolean,false]
  )
),
 Result = TheResult
.
/*
| ?- X = [dtLogin,[],[[dtString,[[value,[ptString,'01234567']]],[]]]],
```

```
msrNav([X],[is,[]],[Result]).
X = [dtLogin,[],[[dtString,[[value,[ptString,'01234567']]],[]]]],
Result = [ptBoolean,true] ?
yes

| ?- X = [dtLogin,[],[[dtString,[[value,[ptString,'01234567a']]],[]]]],
msrNav([X],[is,[]],[Result]).
X = [dtLogin,[],[[dtString,[[value,[ptString,'01234567a']]],[]]]],
Result = [ptBoolean,false] ?
yes
*/
```

Listing D.38: Prolog file PrimaryTypesDatatypes-dtLogin-is.pl.

D.39 File ./src-gen/prolog-ref-spec/Operations/Concepts/PrimaryTypesD dtLongitude-is.pl

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% dtPhoneNumber

% msd01
msrop(dtLongitude,is,[AdtValue],Result):-
msrVar(ptBoolean,TheResult),
(
 ( msrNav([AdtValue],
    [value,geq,[[ptReal,-180.0]]],
    [[ptBoolean,true]]),
   msrNav([AdtValue],
    [value,leq,[[ptReal,+180.0]]],
    [[ptBoolean,true]])
 )
 -> (TheResult = [ptBoolean,true])
 ; (TheResult = [ptBoolean,false])
),

 Result = TheResult
.
```

Listing D.39: Prolog file PrimaryTypesDatatypes-dtLongitude-is.pl.

D.40 File ./src-gen/prolog-ref-spec/Operations/Concepts/PrimaryTypesD dtPassword-is.pl

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% dtComment

%msd01
msrop(dtPassword,is,[AdtValue],Result):-
 msrVar(ptBoolean,TheResult),
 msrVar(ptInteger,MinLength),
(
  (
    (
     MinLength = [ptInteger,6],
     msrNav([AdtValue],
        [value,length,[],geq,[MinLength]],
        [[ptBoolean,true]])
    )
     -> TheResult = [ptBoolean,true]
```

```
    ; TheResult = [ptBoolean,false]
  )
),
 Result = TheResult
.
/*
| ?- X = [dtPassword,[],[[dtString,[[value,[ptString,'012345']]],[]]]],
msrNav([X],[is,[]],[Result]).
X = [dtPassword,[],[[dtString,[[value,[ptString,'012345']]],[]]]],
Result = [ptBoolean,true] ?
yes

| ?- X = [dtPassword,[],[[dtString,[[value,[ptString,'01234']]],[]]]],
msrNav([X],[is,[]],[Result]).
X = [dtPassword,[],[[dtString,[[value,[ptString,'01234']]],[]]]],
Result = [ptBoolean,false] ?
yes
*/
```

Listing D.40: Prolog file PrimaryTypesDatatypes-dtPassword-is.pl.

D.41 File ./src-gen/prolog-ref-spec/Operations/Concepts/PrimaryTypesData dtPhoneNumber-is.pl

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% dtPhoneNumber

% msd01
msrop(dtPhoneNumber,is,[AdtValue],Result):-
msrVar(ptBoolean,TheResult),
(
 ( msrNav([AdtValue],
    [value,length,[],gt,[[ptInteger,4]]],
    [[ptBoolean,true]]),
   msrNav([AdtValue],
    [value,length,[],leq,[[ptInteger,30]]],
    [[ptBoolean,true]])
 )

 -> TheResult = [ptBoolean,true]
 ; TheResult = [ptBoolean,false]
),
 Result = TheResult
.
/*
| ?- X = [dtPhoneNumber,[],[[dtString,[[value,[ptString,'(+352) 46 66 44 60 00']]],[]]]],
msrNav([X],[is,[]],[Result]).

X = [dtPhoneNumber,[],[[dtString,[[value,[ptString,'(+352) 46 66 44 60 00']]],[]]]],

Result = [ptBoolean,true] ?

yes
*/
```

Listing D.41: Prolog file PrimaryTypesDatatypes-dtPhoneNumber-is.pl.

D.42 File ./src-gen/prolog-ref-spec/Operations/Concepts/PrimaryTypesClass etAlertStatus-is.pl

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
```

```
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% etAlertStatus

% msd01
msrop(etAlertStatus,is,[AdtValue],Result):-
msrVar(ptBoolean,TheResult),
(
  (
     member(AdtValue,[pending, valid, invalid])
  )
 -> TheResult = [ptBoolean,true]
 ; TheResult = [ptBoolean,false]
),
 Result = TheResult
.
```

Listing D.42: Prolog file PrimaryTypesDatatypes-etAlertStatus-is.pl.

D.43 File ./src-gen/prolog-ref-spec/Operations/Concepts/PrimaryTypesC etCrisisStatus-is.pl

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% etCrisisStatus

% msd01
msrop(etCrisisStatus,is,[AdtValue],Result):-
msrVar(ptBoolean,TheResult),
(
  (
     member(AdtValue,[pending, handled, solved, closed])
  )
 -> TheResult = [ptBoolean,true]
 ; TheResult = [ptBoolean,false]
),
 Result = TheResult
.
```

Listing D.43: Prolog file PrimaryTypesDatatypes-etCrisisStatus-is.pl.

D.44 File ./src-gen/prolog-ref-spec/Operations/Concepts/PrimaryTypesC etCrisisType-is.pl

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% etCrisisType

% msd01
msrop(etCrisisType,is,[AdtValue],Result):-
msrVar(ptBoolean,TheResult),
(
  (
     member(AdtValue,[small, medium, huge])
  )
 -> TheResult = [ptBoolean,true]
 ; TheResult = [ptBoolean,false]
),
 Result = TheResult
```

```
.
```

Listing D.44: Prolog file PrimaryTypesDatatypes-etCrisisType-is.pl.

D.45 File ./src-gen/prolog-ref-spec/Operations/Concepts/PrimaryTypesClass etHumanKind-is.pl

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% etHumanKind

% msd01
msrop(etHumanKind,is,[AdtValue],Result):-
msrVar(ptBoolean,TheResult),
(
  (
     member(AdtValue,[witness,victim,anonymous])
  )
 -> TheResult = [ptBoolean,true]
 ; TheResult = [ptBoolean,false]
),
 Result = TheResult
.
```

Listing D.45: Prolog file PrimaryTypesDatatypes-etHumanKind-is.pl.

D.46 File ./src-gen/prolog-ref-spec/Operations/Concepts/SecondaryTypesDa dtSMS-is.pl

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/* DISCONTIGUOUS PREDICATES */
:- multifile msrop/4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% dtComment

%msd01
msrop(dtSMS,is,[AdtValue],Result):-
 msrVar(ptBoolean,TheResult),
 msrVar(ptInteger,MaxLength),
(
  (
    (
     MaxLength = [ptInteger,160],
     msrNav([AdtValue],
        [value,length,[],leq,[MaxLength]],
        [[ptBoolean,true]])
    )
     -> TheResult = [ptBoolean,true]
     ; TheResult = [ptBoolean,false]
  )
),
 Result = TheResult
.
```

Listing D.46: Prolog file SecondaryTypesDatatypes-dtSMS-is.pl.

Glossary

abstract actor an actor that is not 22

actor An actor is a person, organization, or external system that plays a role in one or more interactions with the system 18

direct actor an actor that interacts directly with the system. It thus belongs to the environment. 22

indirect actor an actor that interacts indirectly with the system through a direct actor. It thus belongs the domain but not to the environment. 22

system operation a functionality of the system that can be triggered by a message sent by an actor belonging to the environment. 18

Bibliography

[1] Guelfi, N.: Messir: A Scientific Method for the Software Engineer. to be published (2017)

[2] Armour, F., Miller, G.: Advanced Use Case Modeling: Software Systems. Addison-Wesley (2001)

[3] ISO/IEC: ISO/IEC 25010 - Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models. (2011) ISO/IEC 13211-1.