# The Design and Implementation of a System for the Automatic Generation of Narrative Debriefs for AUV Missions

Nicholas Adam R. Johnson

Submitted for the degree of Doctor of Philosophy

Heriot-Watt University
School of Engineering and Physical Sciences

April 2011

# Abstract

Increased autonomy allows autonomous underwater vehicles to act without direct support or supervision. This requires increased complexity, however, and a deficit of trust may form between operators and these complex machines, though previous research has shown this can be reduced through repeated experience with the system in question. Regardless of whether a mission is performed with real vehicles or their simulated counterparts, effective debrief represents the most efficient method for performing an analysis of the mission.

A novel system is presented to maximise the effectiveness of a debrief by ordering the mission events using a narrative structure, which has been shown to be the quickest and most effective way of communicating information and building a situation model inside a person's mind. Mission logs are de-constructed and analysed, then optimisation algorithms used to generate a coherent discourse based on the events of the missions with any required exposition. This is then combined with a timed mission playback and additional visual information to form an automated mission debrief.

This approach was contrasted with two alternative techniques: a simpler chronological ordering; and a facsimile of the current state of the art. Results show that participant recall accuracy was higher and the need for redundant delivery of information was lower when compared to either of the baselines. Also apparent is a need for debriefs to be adapted to individual users and scenarios. Results are discussed in full, along with suggestions for future avenues of research.

## Acknowlegdements

As is the case in everything he does, the author must first thank his parents. Without their love, support and various other contributions the author would hope are obvious, he never would have made it even this far.

Furthermore, the author would like to thank David Lane, Yvan Petilot and Jonathan Evans for their supervision and guidance.

A large number of people helped out in smaller ways, such as offering suggestions and moral support, not to mention agreeing to be guinea pigs in a hazily defined experiment with nothing but a warm feeling (and some cake) as a reward. These include, but are almost certainly not limited to: Maryanne Heard, Dougal Stanton, Helen Hare, Rolf Baxter, Francesco Maurelli, Amy Baumann, Gareth Roberts, Dave Taddei, Ruth Benson, Sian Ringrose, Grainne O' Brien, Hassan Assalih, Maria Erla, Ronan Ryan, Derek Murphy, Hanna Miedema, Matthew Badger, Gordon Sim, Chris Sotzing, Joel Cartwright, Pedro Patron and Mark Carstairs. Additionally: Nigella Lawson, for providing excellent cake recipes.

Finally, the author would like to thank Robert Zemekis, Bob Vale, Dan Ackroyd, Harrold Ramis and Ivan Reitman for getting him interested in this whole science thing in the first place. Thanks, guys.

HERIOT WATT UNIVERSITY

# ACADEMIC REGISTRY
# Research Thesis Submission

| | | | |
|---|---|---|---|
| Name*:* | Nicholas Adam R. Johnson | | |
| School/PGI: | School of Engineering & Physical Sciences | | |
| Version: *(i.e. First, Resubmission, Final)* | Final | Degree Sought (Award **and** Subject area) | Doctor of Philosophy<br>Human Computer Interaction |

## Declaration

In accordance with the appropriate regulations I hereby submit my thesis and I declare that:

1) the thesis embodies the results of my own work and has been composed by myself
2) where appropriate, I have made acknowledgement of the work of others and have made reference to work carried out in collaboration with other persons
3) the thesis is the correct version of the thesis for submission and is the same version as any electronic versions submitted*.
4) my thesis for the award referred to, deposited in the Heriot-Watt University Library, should be made available for loan or photocopying and be available via the Institutional Repository, subject to such conditions as the Librarian may require
5) I understand that as a student of the University I am required to abide by the Regulations of the University and to conform to its discipline.

* *Please note that it is the responsibility of the candidate to ensure that the correct version of the thesis is submitted.*

| | | | |
|---|---|---|---|
| Signature of Candidate*:* | | Date: | |

## Submission

| | |
|---|---|
| Submitted By *(name in capitals):* | |
| Signature of Individual Submitting: | |
| Date Submitted: | |

## For Completion in Academic Registry

| | | | |
|---|---|---|---|
| Received in the Academic Registry by *(name in capitals):* | | | |
| *Method of Submission*<br>*(Handed in to Academic Registry; posted through internal/external mail):* | | | |
| *E-thesis Submitted (**mandatory for final theses from January 2009)*** | | | |
| Signature: | | Date: | |

Please note this form should bound into the submitted thesis.

Updated February 2008, November 2008, February 2009

# Contents

1 Introduction, or Prologue 1

I Establishment: Background and Supporting Work 5

2 Previous Work 7
2.1 What Are Autonomous Underwater Vehicles? . . . . . . . . . . . . . 8
2.2 What Are Multi-Agent Systems, and How Do They Relate to AUVs? 10
2.3 What is Planning, and Can It Be Used as a Basis for AUV Control? . 12
2.4 What Are the Issues of Trust? . . . . . . . . . . . . . . . . . . . . . . 15
2.5 What is Situational Awareness? . . . . . . . . . . . . . . . . . . . . . 17
2.6 What is Human Computer Interaction? . . . . . . . . . . . . . . . . . 21
2.7 Commercial Influences in Human Computer Interaction. . . . . . . . 26
2.8 How Can Visualisation Help? . . . . . . . . . . . . . . . . . . . . . . 29
2.9 How Can Natural Language Generation Help? . . . . . . . . . . . . . 33
2.10 How Can Narrative Help? . . . . . . . . . . . . . . . . . . . . . . . . 36
2.11 Summary . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 43

3 A Framework Designed to Increase Trust Between Operators and Autonomous Systems. 45
3.1 Components of the Framework . . . . . . . . . . . . . . . . . . . . . . 46
3.2 Migration of Data . . . . . . . . . . . . . . . . . . . . . . . . . . . . 48
3.3 Summary . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 50

4 Supporting Technologies 51
4.1 A Planner Based Control System for Multiple Real or Simulated AUVs 51
4.1.1 Planner Implementation . . . . . . . . . . . . . . . . . . . . . 51
4.1.2 Control System Integration . . . . . . . . . . . . . . . . . . . 56
4.1.3 Testing and Results . . . . . . . . . . . . . . . . . . . . . . . . 61
4.1.4 Future Work . . . . . . . . . . . . . . . . . . . . . . . . . . . . 63
4.2 High Speed Mission Simulation . . . . . . . . . . . . . . . . . . . . . 64
4.2.1 Accurate Faster than Real Time Simulation . . . . . . . . . . 65
4.2.2 Simulated Communication . . . . . . . . . . . . . . . . . . . . 66

4.2.3 Scene-graph Integration and Fast Sensor and Actuator Simulation . . . . . 68
4.2.4 Simulated System Failures . . . . . 72
4.2.5 Simulated Missions . . . . . 73
4.2.6 Future Work . . . . . 74
4.3 A Tailored Graphical User Interface for Rich Communication . . . . . 75
4.3.1 Vehicle and World Object Depiction . . . . . 76
4.3.2 Text and Graphics for Heads Up Display . . . . . 77
4.3.3 User Controlled Playback . . . . . 80
4.3.4 Scripted Playback . . . . . 81
4.3.5 Future Work . . . . . 82
4.4 Summary . . . . . 82

**II Confrontation: Glaykos, A Narrative Based System for Effective Mission Debriefing 84**

**5 The Bitmap Situation Model 88**
5.1 An Ontology for Representation of the Bitmap Situation Model . . . . . 88
5.2 Extracting Logs and Domain into the Bitmap Situation Model . . . . . 90
5.3 Further Processing of the Bitmap Situation Model . . . . . 93
5.3.1 Processing Existences, Beliefs, Desires, Intentions and Data Associations . . . . . 93
5.3.2 Processing Meta-Desires . . . . . 99
5.3.3 Processing Coherence and Expectations . . . . . 99
5.3.4 Processing Desire and Requirement Satisfaction . . . . . 102
5.3.5 Resolving Communications Events . . . . . 102
5.4 Summary . . . . . 104

**6 The Vector Situation Model 105**
6.1 Structure of the Vector Situation Model . . . . . 105
6.1.1 The Motivation Model . . . . . 106
6.1.2 The Causation Model . . . . . 108
6.1.3 Situational Axis Positions for Process Elements . . . . . 113
6.2 Extracting Data into the Vector Situation Model . . . . . 114
6.2.1 Finding the ground state . . . . . 114
6.2.2 Building the Motivation model . . . . . 114
6.2.3 Building the Causation Model . . . . . 117
6.2.4 Grouping and Simplification of the Causation Model . . . . . 125
6.3 Summary . . . . . 130

7 The Discourse 131
7.1 Building an Effective Discourse Ordering as an Optimisation Problem 131
7.1.1 Approaches to Optimisation . . . . . . . . . . . . . . . . . . . . 133
7.1.2 Fitness Function . . . . . . . . . . . . . . . . . . . . . . . . . 135
7.1.3 Genetic Algorithm Implementation . . . . . . . . . . . . . . . . 140
7.1.4 Bounded and Constrained Depth First Search . . . . . . . . . 142
7.1.5 Optimisation Results . . . . . . . . . . . . . . . . . . . . . . . 144
7.1.6 Potential Drawbacks of the TSP Model for Discourse . . . . . 145
7.2 Further Discourse Development . . . . . . . . . . . . . . . . . . . . . 148
7.2.1 Further Simplifications . . . . . . . . . . . . . . . . . . . . . . 148
7.2.2 Addition of Stasis Statements . . . . . . . . . . . . . . . . . . 149
7.3 Realisation of Natural Language . . . . . . . . . . . . . . . . . . . . . 151
7.3.1 Basic NLG Structures . . . . . . . . . . . . . . . . . . . . . . . 151
7.3.2 Pronominalization . . . . . . . . . . . . . . . . . . . . . . . . . 153
7.3.3 Fundamental Structures . . . . . . . . . . . . . . . . . . . . . . 154
7.3.4 Process and Stasis Statements . . . . . . . . . . . . . . . . . . 158
7.4 Generation of Additional Script Elements . . . . . . . . . . . . . . . . 162
7.4.1 The Audio, Time, Subtitle and Pause Scripts . . . . . . . . . . 163
7.4.2 The Camera Script . . . . . . . . . . . . . . . . . . . . . . . . 164
7.4.3 The Focus Script . . . . . . . . . . . . . . . . . . . . . . . . . 165
7.4.4 The Graphics Script . . . . . . . . . . . . . . . . . . . . . . . . 166
7.5 Summary . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 171

8 Scenarios and Generated Discourse 172
8.1 Offshore Maintenance . . . . . . . . . . . . . . . . . . . . . . . . . . 173
8.1.1 Initial State and Mission Goals . . . . . . . . . . . . . . . . . 173
8.1.2 Implementation . . . . . . . . . . . . . . . . . . . . . . . . . . 173
8.1.3 Discourse Excerpt . . . . . . . . . . . . . . . . . . . . . . . . . 175
8.1.4 Potentially Critical Incidents in Discourse . . . . . . . . . . . 176
8.2 Mine Counter Measures . . . . . . . . . . . . . . . . . . . . . . . . . 179
8.2.1 Initial State and Mission Goals . . . . . . . . . . . . . . . . . 179
8.2.2 Implementation . . . . . . . . . . . . . . . . . . . . . . . . . . 179
8.2.3 Discourse Excerpt . . . . . . . . . . . . . . . . . . . . . . . . . 181
8.2.4 Potentially Critical Incidents in Discourse . . . . . . . . . . . 182
8.3 Combination . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 185
8.3.1 Initial State and Mission Goals . . . . . . . . . . . . . . . . . 185
8.3.2 Implementation . . . . . . . . . . . . . . . . . . . . . . . . . . 185
8.3.3 Discourse Excerpt . . . . . . . . . . . . . . . . . . . . . . . . . 187
8.3.4 Potentially Critical Incidents in Discourse . . . . . . . . . . . 191
8.4 Summary . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 192

III Resolution: Experimentation and Discussion 194

9 Experimental Methodology 196

9.1 The Experiments . . . 197

9.1.1 Experiment 1 . . . 197

9.1.2 Experiment 2 . . . 199

9.1.3 Experiment 3 . . . 202

9.2 Expectations . . . 203

9.3 Summary . . . 204

10 Results 206

10.1 Were Pause Times Less for the *Experimental Case* than the *Base Case* in Experiment 1? . . . 206

10.2 Did Pause Times Increase with Distance Moved Along the Causal Dimension in Experiments 2 and 3? . . . 208

10.3 Did Users Prefer the *Experimental Case* to the *Base Case* in Experiment 1? . . . 209

10.4 Did Users Prefer the *Experimental Case* to the *User Case* in Experiments 2 and 3? . . . 212

10.5 Did Users Have Some Negative Reaction to the Lack of Control Afforded by the *Experimental Case*? . . . 215

10.6 Was the Participants' Accuracy Higher for the *Experimental Case* than Either the *Base* or *User Case*s? . . . 216

10.7 Did the User Need to Replay More Time in Order to Understand the Mission Fully in Experiments 2 and 3? . . . 220

10.8 Was There Considerable Variation in the Ways Participants Chose to Repeat the Events of the Mission? . . . 225

10.9 Additional Observations . . . 229

10.9.1 Effects of Potentially Critical Incidents . . . 229

10.9.2 Focus in the *User Case* . . . 232

10.9.3 Specific Opinions . . . 236

10.10Summary . . . 240

11 Discussion and Future Work 241

11.1 The Journey So Far . . . 241

11.2 The Present Location . . . 244

11.3 At the End of the Road . . . 248

11.4 First Steps . . . 249

11.4.1 Further Experiments With the Present Implementation . . . 249

11.4.2 Mid and Post Mission Analysis . . . 250

11.4.3 Dealing with Failure . . . 251

11.4.4 Generalisation . . . 251

11.4.5 Control of the "Camera" . . . . . . . . . . . . . . . . . . . . . . 252
11.4.6 Speed and Performance . . . . . . . . . . . . . . . . . . . . . . 252
11.4.7 The "Well Rested Traveling Salesman Problem" . . . . . . . . 253
11.5 Summary . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 254
11.6 Final Thought . . . . . . . . . . . . . . . . . . . . . . . . . . . 254

**IV Epilogue: Appendices 255**

**A Tools 257**
A.1 Languages . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 257
A.2 Applications . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 259
A.3 Released Source Code . . . . . . . . . . . . . . . . . . . . . . . . 260

**B Input Files 263**
B.1 Sensor and Actuator Descriptions . . . . . . . . . . . . . . . . . . 263
B.2 Module Descriptions . . . . . . . . . . . . . . . . . . . . . . . . . 265
B.3 World Descriptions . . . . . . . . . . . . . . . . . . . . . . . . . . 268
B.4 Simulation Descriptions . . . . . . . . . . . . . . . . . . . . . . . 270

**C Output Log Files 273**
C.1 Position Logs . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 273
C.2 Platform Logs . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 274
C.3 Planner Logs . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 275
C.3.1 Belief Change Events . . . . . . . . . . . . . . . . . . . . . 275
C.3.2 Desire Change Events . . . . . . . . . . . . . . . . . . . . . 279
C.3.3 Intention Change Events . . . . . . . . . . . . . . . . . . . 280
C.4 World Object Logs . . . . . . . . . . . . . . . . . . . . . . . . . . 281

**D Operational Parameters 283**
D.1 Base Genetic Algorithm Parameters . . . . . . . . . . . . . . . . . 283
D.2 Fitness Function Parameters . . . . . . . . . . . . . . . . . . . . . 283
D.3 Script Generation Parameters . . . . . . . . . . . . . . . . . . . . 285

**E SPARQL Queries 286**
E.1 Processing Existences, Beliefs, Desires, Intentions and Data Associations (Section 5.3.1) . . . . . . . . . . . . . . . . . . . . . . . . . 286
E.2 Resolving Communications Events (Section 5.3.5) . . . . . . . . . . . 288
E.3 Building the Motivation model (Section 6.2.2) . . . . . . . . . . . . 288

**F Complete List of Syntactic Structures 291**
F.1 Process Statements . . . . . . . . . . . . . . . . . . . . . . . . . . 291
F.2 Stasis Statements . . . . . . . . . . . . . . . . . . . . . . . . . . . 295

**G Example Narrative Script** **299**

**H Example Surveys and Instructions Given to Experiment participants** **322**

H.1 Background information survey . . . . . . . . . . . . . . . . . . . . . . . 323

H.2 Scripted Instructions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 325

H.3 User controlled Instructions . . . . . . . . . . . . . . . . . . . . . . . . . 326

H.4 Post experiment Surveys . . . . . . . . . . . . . . . . . . . . . . . . . . . 327

**Bibliography** **329**

# List of Tables

2.1 Example mail merge database. . . . . . . . . . . . . . . . . . . . . . . . 34

7.1 Example "random key" gene encoding. . . . . . . . . . . . . . . . . . . 140

7.2 Worked example of "parameterized uniform crossover". . . . . . . . . 141

8.1 Potentially critical incidents in the discourse for the offshore maintenance scenario. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 179

8.2 Potentially critical incidents in the discourse for the mine counter measures scenario. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 185

8.3 Potentially critical incidents in the discourse for the combination scenario. The equivalence value for incident 3 is marked in red in order to indicate that it shows differences to its equivalent in the mine counter measures scenario. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 192

10.1 Student's t-test results for comparisons between and within groups. Categories on the columns represent data from the *experimental case*, while categories on the columns represent the *base case*. All tests were homoscedastic with two tails. . . . . . . . . . . . . . . . . . . . . . . . 208

10.2 The Mann-Whitney-U $P$ value results for the between groups, within groups, and combined analysis for Experiment 1. . . . . . . . . . . . . 211

10.3 The Mann-Whitney-U $P$ value results for the combined analysis of Experiment 2. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 214

10.4 The Mann-Whitney-U $P$ value results for the combined analysis of Experiment 3. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 215

10.5 The t-test results for comparisons *experimental* vs. *base* and *experimental* vs. *user*. Groups are the same as in Figure 10.12. . . . . . . . 219

10.6 The t-test results for comparisons *experimental* vs. *base* and *experimental* vs. *user*, without the inclusion of the data from Participant 13. Groups are the same as in Figure 10.13. . . . . . . . . . . . . . . 219

10.7 The results of the t-test comparison between the generated and users' playback ratios with and without potential outliers. . . . . . . . . . . 223

10.8 The results of the t-test comparison between the generated and users' playback times with and without potential outliers. . . . . . . . . . . 225

10.9 Distribution of different combinations of orderings used by participants who viewed two scenarios. Rows represent the ordering used in the first survey, columns the second. . . . . . . . . . . . . . . . . . . 228

10.10 The correlation between the normalised histograms of the participants and the generated debrief in the offshore maintenance scenario (see Figure 10.23a). . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 234

10.11 The fraction of shared peaks between each combination of histograms for each scenario. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 235

10.12 T-Test results for comparisons of human versus human, and human versus generated debrief, as per Table 10.11. . . . . . . . . . . . . . . 236

# List of Figures

1.1 The structure of this thesis. . . . . . . . . . . . . . . . . . . . . . . . 4

2.1 Simplified Hybrid Architecture. . . . . . . . . . . . . . . . . . . . . . . 14
2.2 The SeeTrack user interface, with mission replay controls (top left). The vehicle's previous positions are shown in red, with the red arrow indicating its current trajectory. . . . . . . . . . . . . . . . . . . . . 19
2.3 Comparison of the paths generated by the current state of the art and those generated by the Triton system, to the vehicles real path, based on position updates received every 1080 seconds. . . . . . . . . 21
2.4 Annotated example of the Triton system's two dimensional "debug" interface. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 22
2.5 The three dimensional, Augmented Reality Framework based, "production" version of the Triton interface. . . . . . . . . . . . . . . . . 22
2.6 Examples of the graphical user interface from *Homeworld*[1]. . . . . . 28
2.7 Two Dimensional Hierarchical Task Network visualisation. . . . . . . 30
2.8 Three Dimensional Hierarchical Task Network visualisation. . . . . . 30
2.9 Example BIIMAPS visualisation. . . . . . . . . . . . . . . . . . . . . 31
2.10 An example Action Resource Formalism plan visualisation. Taken from [2], page 80. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 32
2.11 UML translation of the "Diagram of Narrative Structure" from page 267 of *Story and Discourse* [3]. . . . . . . . . . . . . . . . . . . . . 40

3.1 A framework designed to increase trust between operators and autonomous systems. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 46
3.2 Chronology of data migration within the framework. . . . . . . . . . 49
3.3 The components of the framework which have been developed as part of the work contained in this thesis. . . . . . . . . . . . . . . . . . . 49

4.1 The components described by this section and how they fit into the framework described in Chapter 3. . . . . . . . . . . . . . . . . . . . 52
4.2 A planner based control system for multiple real or simulated AUVs . 56
4.3 The Nessie III AUV passes through the validation gate at SAUC-E 2008. Image courtesy of Yves Gladu. . . . . . . . . . . . . . . . . . . 62
4.4 The simplified control system deployed on the Nessie III AUV. . . . . 62

4.5 The components described by this section and how they fit into the framework described in Chapter 3. . . . . . . . . . . . . . . . . . . . . . 65
4.6 Comparison of simulation approaches. . . . . . . . . . . . . . . . . . . . 66
4.7 Example sensor geometry. . . . . . . . . . . . . . . . . . . . . . . . . . . 69
4.8 Example actuator geometry. . . . . . . . . . . . . . . . . . . . . . . . . . 72
4.9 Example simulation geometry. . . . . . . . . . . . . . . . . . . . . . . . 73
4.10 The components described by this section and how they fit into the framework described in Chapter 3. . . . . . . . . . . . . . . . . . . . . . 76
4.11 Example vehicle models. From left: Nessie with yellow colouring, RAUVER with yellow colouring, REMUS with yellow colouring, and Nessie with Green colouring. . . . . . . . . . . . . . . . . . . . . . . . 76
4.12 Animated graphics in the Heads Up Display. . . . . . . . . . . . . . . 78
4.13 Example of features in the Heads Up Display. Including: 1) Mission time passed; 2) Mission time remaining; 3) Mission timeline; 4) Mission playback speed indicator; 5) Currently selected agent; and 6) Subtitle. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 79
4.14 Example of additional geometry used to help the user identify the relative positions of object in 3D space. The selected vehicle's position is projected onto the "sea bed" plain, while the position of other objects is projected onto the same plain as the selected vehicle. . . . . 80
4.15 The components described by the chapters in this part and how they fit into the framework described in Chapter 3. . . . . . . . . . . . . 85
4.16 Data Flow in the Narrative Debriefing System. The sections outlined with dashed lines refer back to the framework described in Chapter 3 and particularly Figure 3.1. . . . . . . . . . . . . . . . . . . . . . . . 86

5.1 The components described by this chapter and how they fit into the structure of the *Glaykos* system (as per Figure 4.15). . . . . . . . . . 89
5.2 Belief, Desire and Intention based ontology constructs. . . . . . . . . 89
5.3 Events used in the ontology. . . . . . . . . . . . . . . . . . . . . . . . . 90
5.4 Main structure of the ontology. . . . . . . . . . . . . . . . . . . . . . . 90
5.5 Example of base structure for a Meta-Desire. . . . . . . . . . . . . . . 91
5.6 Simple example knowledge base, in its initial state immediately after the assimilation of the mission logs. . . . . . . . . . . . . . . . . . . . 92
5.7 Additional links added to the knowledge based by the existents step shown by solid lines. Pre-existing links shown dashed. . . . . . . . . . 95
5.8 Additional links added to the knowledge base by the beliefs step. . . . 96
5.9 Additional links added to the knowledge base by the desires step. . . 97
5.10 Additional links added to the knowledge base by the intentions step. . 98
5.11 Additional links added to the knowledge base by the meta-desires step. 99

5.12 Additional properties added to the example knowledge base during the expectations step. . . . . . . . . . . . . . . . . . . . . . . . . . . . 101
5.13 Additional properties added to the example knowledge base during the satisfaction step. . . . . . . . . . . . . . . . . . . . . . . . . . . . 102

6.1 The components described by this chapter and how they fit into the structure of the *Glaykos* system (as per Figure 4.15). . . . . . . . . . 106
6.2 Example motivation model. . . . . . . . . . . . . . . . . . . . . . . 108
6.3 The direct relationships between the components of the motivation and causation models. . . . . . . . . . . . . . . . . . . . . . . . . . 110
6.4 Example causation model. Unbroken arrows indicate a "causes" relationship, dashed lines indicate a "contributes to" relationship. Colours indicate motivation strands from Figure 6.2. . . . . . . . . . . . . . . 112
6.5 The process elements of the example in chronological order. . . . . . . 128
6.6 Groupings and simplifications of the example causation model. . . . . 129

7.1 The components described by this chapter and how they fit into the structure of the *Glaykos* system (as per Figure 4.15). . . . . . . . . . 132
7.2 The "traveling salesman problem" edition of the XKCD webcomic, published online at http://xkcd.com/399/ . . . . . . . . . . . . . . . 133
7.3 Example of "inversion" applied to a Euclidean TSP. . . . . . . . . . . 141
7.4 Examples paths produced for simple asymmetric traveling salesman problem used for tuning of the genetic algorithm. Red lines indicate optimal solution produced via bounded depth first search, yellow lines indicated paths produced by the genetic algorithm. . . . . . . . . . . 143
7.5 Time taken to provide find a good solution with the bounded constrained depth first search and genetic algorithm based optimisation approaches for each of the mission described in Section 8. . . . . . . . 144
7.6 Chronological ordering of a subset of the example events from Figure 6.4. Ordering moves down the page. . . . . . . . . . . . . . . . . . . 145
7.7 Alternative causation based orderings of the example events from Figure 7.6. Ordering moves down the page. . . . . . . . . . . . . . . . 146
7.8 Causation based ordering of the example events from Figure 6.4 with repetition of pivotal event. Ordering moves down the page. . . . . . . 147
7.9 The legend for the syntax diagrams used in the following subsections. 151
7.10 Syntactic structure for intentions. . . . . . . . . . . . . . . . . . . . . 155
7.11 Syntactic structure for generalised intentions. . . . . . . . . . . . . . 155
7.12 Syntactic structure for beliefs. . . . . . . . . . . . . . . . . . . . . . . 156
7.13 Syntactic structure for desires. . . . . . . . . . . . . . . . . . . . . . . 156
7.14 Syntactic structure for complex meta-desires. . . . . . . . . . . . . . . 157
7.15 Syntactic structure for "Desire Satisfied" process statement. . . . . . . 159

7.16 Syntactic structure for “Conditions” stasis statement. . . . . . . . . . 159
7.17 Syntactic structure for the complex version of the “Capability” stasis statement. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 161
7.18 Syntactic structure for the simplified version of the “Capability” stasis statement. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 161
7.19 Top and side elevations, showing camera position calculated to make all elements visible. . . . . . . . . . . . . . . . . . . . . . . . . . . 165
7.20 Illustration of how focus of particular object is handled in a hypothetical script. The first two process statements refer to AUV 1, whereas the third refers to AUV 2. The second process statement is a *sensor detection* which refers to Object 1, where as the fifth stasis statement is an *object property* which refers to Object 2. . . . . . . . . . . . . 166
7.21 Example views for addition of meta-desires and desires. Note: text in the actual HUD appears white. . . . . . . . . . . . . . . . . . . . . 168
7.22 Example states for desire satisfaction, meta-desire satisfaction and mission succeeded views. . . . . . . . . . . . . . . . . . . . . . . . . 168
7.23 Addition of a new desire into a view which already contains satisfied desires. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 168
7.24 Example transition between intention views. . . . . . . . . . . . . . 170

8.1 The initial physical layout of the Offshore Maintenance Mission. . . . 174
8.2 The graph of activity during the Offshore Maintenance Mission. . . . 175
8.3 The initial physical layout of the Mine Counter Measures Mission. . . 180
8.4 The graph of activity during the Mine Counter Measures Mission. . . 181
8.5 The desire visualization for the point in the mission at which the search task is completed, showing a similar repetition to that which is found in the dialogue. . . . . . . . . . . . . . . . . . . . . . . . . . 184
8.6 The initial physical layout of the Combination Mission. . . . . . . . . 186
8.7 The graph of activity during the Combination Mission. . . . . . . . . 186
8.8 The graphics progression for the introduction of the desire that mine one should be destroyed in the combination scenario. . . . . . . . . . 187
8.9 The graphics displayed for the introduction of the plan to destroy mine one in the combination scenario. . . . . . . . . . . . . . . . . . . 188
8.10 The graphics progression for the completion of the Manipulator AUV’s approach to site alpha, which also moves between different motivations via a dual display. . . . . . . . . . . . . . . . . . . . . . . . . 188
8.11 The graphics progression for the satisfaction of the desire that detection two should be examined . . . . . . . . . . . . . . . . . . . . . . 190

9.1 The timing for each of the participants in Experiment 1. . . . . . . . 199
9.2 The timing for each of the participants in Experiment 2. . . . . . . . 201

9.3 The timing for each of the participants in Experiment 3. . . . . . . . 203

10.1 Base playback time (without users' pauses) and average playback time with users' pauses in experiment one. The error bars show the standard deviation. . . . . . . . . . . . . . . . . . . . . . . . . . . . 206
10.2 The average individual pauses between events for each scenario and case during experiment one. . . . . . . . . . . . . . . . . . . . . . . . . 207
10.3 Legend for scatter plots shown in Figures 10.4. . . . . . . . . . . . . . . 208
10.4 Scatter chart to show correlation between user pause times and causation gap between events in the *experimental case* for experiments two and three. The data set for each participant has been converted to the standard normal form. The legend for the different symbols can be found in Figure 10.3. . . . . . . . . . . . . . . . . . . . . . . . . 209
10.5 Participant preference for Experiment 1. . . . . . . . . . . . . . . . . 210
10.6 Subjective assessments for experiment 1. The error bars indicate the standard deviation. . . . . . . . . . . . . . . . . . . . . . . . . . . . 211
10.7 Participant preference for Experiment 2. . . . . . . . . . . . . . . . . 212
10.8 Subjective assessment for experiment 2. . . . . . . . . . . . . . . . . . 213
10.9 Subjective assessment for the Combination scenario during experiment 3. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 215
10.10Participant opinion of the lack of control in the *base* and *experimental cases*. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 216
10.11The hits, misses and false alarms for each of the participants in each of the scenarios. "M" indicates the mean for the preceding group. . . 218
10.12The average d-prime figures for each scenario and case. "All" averages all scenarios for each case, while "limited" excludes the combination scenario. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 219
10.13The average *d*-prime figures for each scenario and case, without the inclusion of the data from Participant 13. "All" averages all scenarios for each case, while "limited" excludes the combination scenario. . . . 220
10.14The complete temporal path taken by the participants and the generated playback for each scenario. . . . . . . . . . . . . . . . . . . . . . 222
10.15The playback ration for each of the participants in the *user case*, divided by scenario. "G" indicates the ratio for the debrief generated by the *Glaykos* system, "M" the mean for the participants in a scenario, and "-X" the mean with participant X removed. . . . . . . . . . . 223
10.16The playback time for each of the participants in the *user case*, divided by scenario. "B" indicates the actual time for the mission, "G" the time for the debrief generated by the *Glaykos* system, "M" the mean for the participants in a scenario, and "-X" the mean with participant X removed. . . . . . . . . . . . . . . . . . . . . . . . . . . 224

10.17 The different orderings used by participants for each case with the offshore maintenance and mine counter measures scenarios. . . . . . 227
10.18 The different orderings used by participants for each case with the combination scenario. . . . . . . . . . . . . . . . . . . . . . . . . 228
10.19 The normalised pauses for each of the participants who experienced the *experimental case* in Experiments 2 and 3. Symbols are as per Figure 10.3. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 229
10.20 The normalised pauses for each participant from Experiment 2 in the offshore maintenance scenario, with the first five pauses excluded. Shaded areas correspond to potentially critical incidents (see Section 8.3.4). Symbols are as per Figure 10.3. . . . . . . . . . . . . . . . . 230
10.21 The normalised pauses for each participant from Experiment 2 in the mine counter measures scenario, with the first five pauses excluded. Shaded areas correspond to potentially critical incidents (see Section 8.2.4). Symbols are as per Figure 10.3. . . . . . . . . . . . . . . . . 231
10.22 The normalised pauses for each participant from Experiment 3 in the combination scenario, with the first five pauses excluded. Shaded areas correspond to potentially critical incidents (see Section 8.3.4). Symbols are as per Figure 10.3. . . . . . . . . . . . . . . . . . . . 231
10.23 The normalised timespan each participant spent viewing each time in each scenario. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 233
10.24 Scatter chart showing the relationship between the normalised timespan spent at each time by the human participants using the *user case* for the offshore maintenance scenario. . . . . . . . . . . . . . . . . . . . . 234
10.25 Participant opinion of the text-to-speech voice synthesis. . . . . . . . 237
10.26 Participant opinion of non-linear ordering. . . . . . . . . . . . . . . 238
10.27 Participant opinion of intention graphics. . . . . . . . . . . . . . . 239

F.1 The legend for the syntax diagrams used in the following sections. . . 291
F.2 Syntactic structure for “Mission Succeeded” process statement. . . . . 291
F.3 Syntactic structure for “Desire Introduced” process statement. . . . . 292
F.4 Syntactic structure for “Intention Started” process statement. . . . . . 292
F.5 Syntactic structure for “Intention Successful” process statement. . . . 293
F.6 Syntactic structure for “Desire Satisfied” process statement. . . . . . . 293
F.7 Syntactic structure for “Meta-Desire Satisfied” process statement. . . 294
F.8 Syntactic structure for “Sensor Detection” process statement. . . . . . 294
F.9 Syntactic structure for the “Plan Introduced” process statement. . . . 295
F.10 Syntactic structure for the complex version of the “Capability” stasis statement. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 295
F.11 Syntactic structure for the simplified version of the “Capability” stasis statement. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 296

F.12 Syntactic structure for “Meta-Desire Explanation” stasis statement. . 296
F.13 Syntactic structure for “Object Location” stasis statement. . . . . . . 296
F.14 Syntactic structure for “AUV Explanation” stasis statement. . . . . . 297
F.15 Syntactic structure for “Effect” stasis statement. . . . . . . . . . . . . 297
F.16 Syntactic structure for “Conditions” stasis statement. . . . . . . . . . 298

G.1 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 299
G.2 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 300
G.3 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 300
G.4 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 300
G.5 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 301
G.6 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 301
G.7 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 301
G.8 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 302
G.9 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 302
G.10 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 302
G.11 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 303
G.12 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 304
G.13 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 304
G.14 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 305
G.15 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 305
G.16 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 306
G.17 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 306
G.18 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 307
G.19 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 308
G.20 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 308
G.21 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 309
G.22 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 310
G.23 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 310
G.24 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 311
G.25 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 311
G.26 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 312
G.27 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 312
G.28 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 312
G.29 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 313
G.30 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 313
G.31 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 314
G.32 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 314
G.33 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 315
G.34 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 315
G.35 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 316

G.36 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 316
G.37 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 317
G.38 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 317
G.39 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 318
G.40 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 318
G.41 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 319
G.42 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 319
G.43 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 320
G.44 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 320
G.45 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 320
G.46 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 321

H.1 Background information survey, page 1. . . . . . . . . . . . . . . . . . . 323
H.2 Background information survey, page 2. . . . . . . . . . . . . . . . . . . 324
H.3 Experimental case instructions for the installation repair scenario. . . 325
H.4 User case instructions for the installation repair scenario. . . . . . . . 326
H.5 Post experiment survey, page 1. . . . . . . . . . . . . . . . . . . . . . . . 327
H.6 Post experiment survey, page 2. . . . . . . . . . . . . . . . . . . . . . . . 328

# List of Algorithms

4.1 The algorithm used to perform replanning in response to a significant change in the world state. . . . . . . . . . . . . . . . . . . . . . . . . 55

4.2 The algorithm used by each agent to update their contact lists on receipt of a communication. . . . . . . . . . . . . . . . . . . . . . . . . . 68

4.3 The algorithm used by each agent to update the *existent* component of their world model on receipt of a communication. . . . . . . . . . . 69

4.4 The algorithm used by each agent to update the *statement* component of their world model on receipt of a communication. . . . . . . . . . . 70

5.1 The algorithm used to update the existents at each time in the bitmap situation model. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 94

5.2 The algorithm used to update the beliefs at each time in the bitmap situation model. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 96

5.3 The algorithm used to process the coherence between the beliefs of each agent. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 100

5.4 The algorithm used to process the expectations of each intention at each point in the timeline. . . . . . . . . . . . . . . . . . . . . . . . . 101

5.5 The algorithm used to process the satisfaction of the desires and requirements. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 103

6.1 The generation of desire and meta-desire wrappers for the motivation model. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 116

6.2 The algorithm used to calculate and track motivation frames. . . . . 118

6.3 Generation of “rectifying” motivation frames. . . . . . . . . . . . . . . 119

6.4 The generation of the nodes which make up the causation model. All process elements which are created are recorded. . . . . . . . . . . . . 120

6.5 The “update causation” function, which is used to update the causal connection between two process elements, taking into account the differing “caused by” and “contributed to by” relationships. . . . . . . 121

6.6 The “bind antecedents” function, which is used find the existences and beliefs which prompted the addition of a particular desire. . . . . 121

7.1 The algorithm used to check for validity when adding a new element to the discourse using bounded constrained depth first search. . . . . 143

7.2 The algorithm used to add stasis elements to the discourse, whilst avoiding giving redundant information to the user. . . . . . . . . . . . . 150

# Chapter 1

# Introduction, or Prologue

> “
>
> “Begin at the beginning,” the King said gravely, “and go on until you come to the end: then stop.”
>
> ”
>
> *“Alice’s Adventures in Wonderland” by Lewis Carrol*

In recent times, Autonomous Underwater Vehicles (or AUVs) have become a much sought after form of robotics technology. Much of this stems from applications which demand entry into areas which may be hazardous to human life.

Increased autonomy comes with pitfalls of its own, however. Autonomous systems require a plan in order to specify what tasks must be completed, and the operator must trust this plan completely. Even more so, they must trust that the autonomous system has the ability to complete it. This is made more difficult by the fact that an autonomous system must be able to adapt to circumstances, and therefore may not appear to be deterministic if the operator is not expecting this behaviour. There exist a large number of systems which allow a user to generate a plan based on their requirements, however many of these systems require the use of extremely complex notations in order to enable their complete expressivity and power. Additionally, the primary source of validation is often a direct test, either with the real system or a simulated equivalent, with any corrections to the plan being implemented through changes to the original specifications.

One potential solution to this problem is to provide operators with sufficient training to use these systems, both to create plans and to adapt them to deal with problems. However, since a strong foundation of logic and maths knowledge is often required to fully grasp the complete functionality of many planning systems, any training scheme runs the risk of being either extremely time consuming and expensive, or too superficial and highly inadequate.

This project is intended to explore the possibilities of the alternative solution. This approach attempts to bridge the gap between the complex notations used to generate plans and human expression itself, bringing the interface closer to the user, so only a very small amount of training (or ideally none) is required. This is a difficult approach, as it involves the translation of human expression, which is very simple to humans but very difficult for computers to understand, into planning notation, which may be very difficult for humans but is designed to be easy for computers to understand, and then the translation of the plan itself back into human expression. Further steps are required beyond this, though. The plan must be created, tested and verified, ideally in as short a space of time as possible.

This work concentrates on the process which translates the actions of the various AUVs during a mission into an effective debrief for the operator's information and verification. This being so, other components which are required to support this are also given time and attention as required. Thus, the aims of this research are as follows:

1. To perform an investigation into relevant previous work, in order to clarify the problem and help formulate a solution;

2. To propose a framework designed to guide the research for which this work is the first step;

3. To design and develop a system to create effective automated debriefs of autonomous missions;

4. To develop any supporting technologies within the framework proposed in point 2 which are required for the implementation and testing of point 3;

5. To test the effectiveness of the developed solution.

The remainder of this report is divided into four parts, themselves divided into nine further chapters and eight appendices. The structure of these parts is modeled roughly on the three act structure used by many narrative works, with the appendices forming an epilogue.

The first part is modeled upon the first act, or "establishment", in which the world and characters are traditionally introduced, together with problem the protagonist must face. As such, the three chapters in this part describe first the relevant previous work (as per aim 1), the framework which has been designed to guide the research carried out during this project (as per aim 2) and then the additional technology which has been developed to support the principle aims of this project (as per aim 4).

The second part is modeled after the second act, or "confrontation", in which the protagonist takes active steps to confront the problem. The three chapters in this part describe the principle stages used in the created system, as per aim 3. Respectively, these are: the bitmap situation model (Chapter 5), which is a complete representation of the mission, geared towards computational representation; the vector situation model (Chapter 6), a higher level representation of this, designed to mirror its representation within the human mind; and the discourse (Chapter 7), which is the concrete ordering of the data which is delivered to the user. Finally, Chapter 8 describes the scenarios which were used for testing, together with the discourses which were generated from them.

The third part is modeled after the third act, or “resolution”, in which the protagonist reaches the final confrontation, is tested, and then either succeeds or fails. As such, this part contains chapters which describe the methodology for the testing of the created system (Chapter 9), the results obtained (Chapter 10) and finally a chapter which a discusses these results (as per aim 5) and suggests possible areas of future work (Chapter 11).

The fourth and final part appears in the place of an epilogue, which typically acts as a dénouement, though in this case provides additional information not required in the main body. There are eight appendices in this part, which describe:

A The additional tools used within the project;

B The on disk data formats used to described the domain and problem information used for planning and simulation;

C the output logs of the control system;

D The specific parameters used to tune the various algorithms;

E The SPARQL queries which are used by the algorithms in Chapters 5 and 6.

F A complete list of the syntactic structures used by the natural language generation system described in Section 7.3;

G A complete example debrief script created by the system;

H Examples of the surveys and instructions which were given to the experimental participants.

This structure is represented diagrammatically in Figure 1.1.

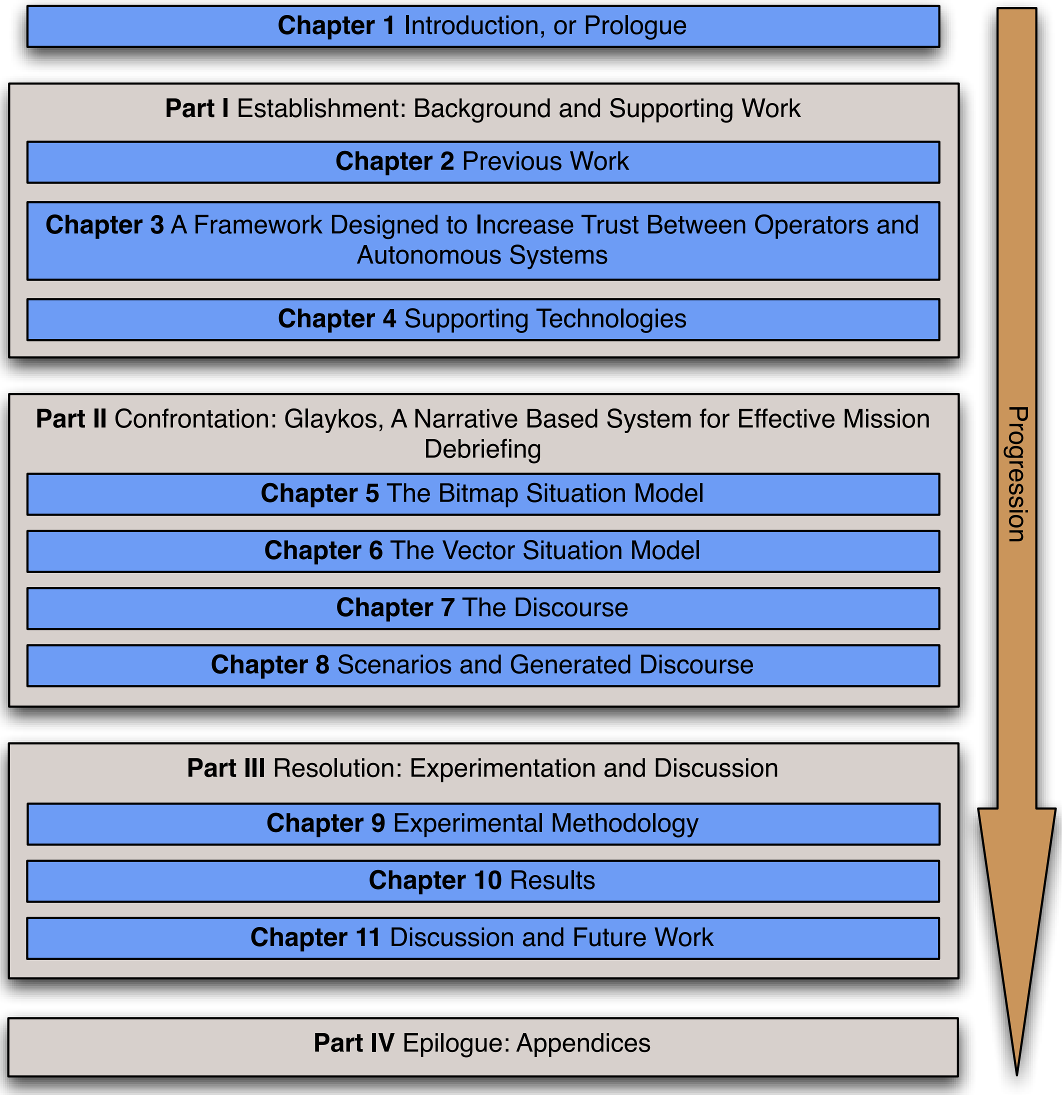

**Chapter 1** Introduction, or Prologue
**Part I** Establishment: Background and Supporting Work
**Chapter 2** Previous Work
**Chapter 3** A Framework Designed to Increase Trust Between Operators and Autonomous Systems
**Chapter 4** Supporting Technologies
**Part II** Confrontation: Glaykos, A Narrative Based System for Effective Mission Debriefing
**Chapter 5** The Bitmap Situation Model
**Chapter 6** The Vector Situation Model
**Chapter 7** The Discourse
**Chapter 8** Scenarios and Generated Discourse
**Part III** Resolution: Experimentation and Discussion
**Chapter 9** Experimental Methodology
**Chapter 10** Results
**Chapter 11** Discussion and Future Work
**Part IV** Epilogue: Appendices
Progression

Figure 1.1: The structure of this thesis.

# Part I

# Establishment: Background and Supporting Work

> “
>
> It would be a protean device that could turn into any tool you could ever need.
>
> ”
>
> *“Cryptonomicon” by Neal Stephanson*

The first part of this thesis mirrors the the first of the three acts used in many narratives. Sometimes referred to as the “establishment”, this introduces the protagonist of the story, establishes the world they inhabit and finally introduces the obstacle (often in the form of an embodied antagonist) against which they must battle.

This part contains three chapters which describe previous work which is relevant to this thesis, together with additional supporting work which has been carried out by the author. Chapter 2 describes the relevant previous work, as per research aim 1 described in Chapter 1. Next, Chapter 3 proposes the framework suggested by research aim 2. Chapter 4 describes the supporting technologies which have been developed as part of this research. Firstly, a prototype system which is able control multiple AUVs simultaneously, whilst using an autonomous planner in order to update the vehicles’ plans in the face of changing circumstances. Next, the methodology which has been developed for and used within this work in order to achieve high speed, accurate mission simulation. Lastly, the Graphical User Interface (or GUI) which has been developed to facilitate the delivery of the debriefs created by the system described in Part II to the user. Together, the work described in these three chapters fulfill research aim 4.

# Chapter 2

# Previous Work

This chapter will describe and discuss the previous work which has informed, influenced and, to some extent, necessitated this course of research.

The choice of chapter title, “Previous Work,” in preference to “Literature Review” is a deliberate one. Much of the work which will be discussed is indeed academic research, presented in papers and submitted for presentation at conferences or publication by journals. Some of it, however, is work undertaken in various industries. In cases such as this, there is often no publication of the theory and implementation behind the work, and so the final product must stand on its own and be examined as a black box. This should not diminish its potential for inspiration and exemplification, however. Observable work produced in industry is often in a final state, and is required to perform to a high enough standard that it is considered fit for purpose, and by extension a viable commercial proposition. A premium is therefore placed upon positive results in the form of a positive reception from the target audience. For this reason, the author gives these a consideration which, if not equal, is at least comparable.

The remainder of this chapter is divided into ten sections, which to some extent pivot around the fourth. The first three attempt to build a picture of Autonomous Underwater Vehicles, their advantages, and their complexities. In turn: AUVs themselves are covered, followed by systems which allow multiple cooperative AUVs to be deployed simultaneously, and then a description of how planning systems can be used to increase the autonomy of AUVs. By turns, each of these increase the complexity of the system and potential for catastrophic failure, which contributes to the significant issue of trust in AUVs, which is described in the pivotal fourth section.

The six sections which follow this attempt to build a picture of the various technologies which can be deployed to help develop a solution to the problem. Firstly, software designed to provide situational awareness is discussed, followed by the more general principles of human computer interaction, and the additional influences of human computer interaction which can be found in commercial products. Next two sections describe techniques for representing information visually and textually, via visualisation and natural language generation. The final section discusses narrative, how it can be used to help structure and order information and why this has the potential to provide an effective discourse to the user for debriefing purposes.

## 2.1 What Are Autonomous Underwater Vehicles?

In its *Unmanned Undersea Vehicle Master Plan*[4], the US Navy Office of Naval Research gives its road map for employing sub-sea robotics technologies. Through a breakdown of the areas in which the US Navy intends to employ UUVs (payload delivery, mine counter measures, etc.) a number of strengths of unmanned systems, as well as the difficulties of the domain are identified. These strengths include:

- **Autonomy**. AUVs have the ability to operate independently for extended periods of time.
- **Risk Reduction**. Their unmanned nature reduces the danger to human life.
- **Low Profile**. They can operate fully submerged, with potentially low acoustic and magnetic signatures.
- **Deployability**. They can be smaller than, and therefore easier to deploy than, manned systems.
- **Environmental Adaptability and Persistence**. They can operate regardless of harsh weather and conditions.

For these reasons, AUVs have found a place in the military toolkit and have been successfully deployed in various scenarios, particularly those related to Mine Counter Measures[5, 6].

These same strengths are now leading to the adoption of AUVs for complex tasks in the offshore oil industry, providing more efficient methods of carrying out tasks which would otherwise require ship borne Remotely Operated Vehicles (ROVs) or towed side scan sonar platforms (or “tow fishes”). At present, the most prevalent of these activities are geophysical seabed surveys and those of subsea pipelines[7]. New technology is developing the possibility to employ AUVs for more delicate inspection and intervention work[8].

AUVs also find use in other areas, such as marine biology [9] and geological and environmental surveys[10]. These tasks are typically more linear however, and have less requirements for decision making on the part of the vehicle. For this reason, the author concentrates on military and offshore oil applications when considering exemplar and testing scenarios.

As is the case with the majority of military operations, autonomous pipeline surveys are typically carried out using torpedo shaped, or “transit”, AUVs, such as the Hydroid REMUS[11] or Hafmynd Gavia[12] platforms. These are characterised by their torpedo like, highly hydrodynamic, shape and under-actuated control which leads to their being dynamically, but not statically, stable, and so they must maintain a reasonably constant speed. More recently, developments are being made on a second class of vehicle, the hover capable or "intervention" AUV, such as the PAIV vehicle described in [8], or the Bluefin HAUV[13]. These are fully actuated and so statically stable and holonomorphic. A valid analogy between the two types of AUV might compare the first to a jet, and the second to a helicopter.

The *master plan*[4] also identifies some challenges which vehicles in this subsea domain must face. These include:

- They will have a low communication bandwidth and range[1].
- They must face the possibility of enemy countermeasures.
- They can only can carry a limited amount of energy.
- They must be able to cope with variable water currents.
- There are limited sensors available for underwater environments[2].
- They have an increased need for reliability, as they cannot be directly supervised.

The first of these challenges raises difficulties for both inter-vehicle communication in missions carried out by multiple vehicles (which will be discussed in more detail in the next section), and also the communication of the vehicle's status to the operator (who is presumably on land or a ship from which the vehicle was launched). This issue is so pronounced, in fact, that even the most robust acoustic modem hardware requires a highly compressed data format[14] in order to achieve reliable communication.

The presence of ocean currents may require a significant amount of an AUV resources to be expended on simply maintaining a stable position. This will be at at the cost of energy, which as mentioned is in short supply when working with AUVs. These currents can also move other objects in the environment which the AUV may intend to track (such as mines in a mine counter measures mission, or a vertical oil pipe in a riser survey mission). As such, currents are the principle cause of the unstructured nature of underwater environments. Further trajectory planning work at Heriot Watt University's Ocean Systems Laboratory [15, 16] has

[1] Acoustic communication is the only currently available option, though some work is being done on underwater radio. Light based communication is unsuitable due to low visibility in most open water environments.

[2] The current technology essentially limits these to the various kinds of available sonar, descriptions of which are outside the scope of this thesis.

sought to overcome the difficulties of detecting and subsequently avoiding various in water obstacles.

The last of these challenges is of particular significance for this work, as it raises the need for a system which is able to provide the operator with situational awareness in order to overcome the issues of trust which arise. These will be discussed in more detail in Sections 2.5 and 2.4, respectively.

In order to help find new and novel solutions to many of these challenges, and further develop the AUV field in general, the Student Autonomous Underwater Competition-Europe[17, 18, 19] (SAUC-E) was started. SAUC-E is similar in nature to land based autonomous vehicle competitions designed to develop the state of the art, such as the DARPA urban challenge [20, 21] and the MOD challenge[22] (which also includes aerial vehicles), though significantly smaller in scope. For the past two years, SAUC-E has been convincingly won by Heriot-Watt University's Ocean Systems Laboratory[23, 24] and its "Nessie" AUV[25, 26].

## 2.2 What Are Multi-Agent Systems, and How Do They Relate to AUVs?

A multi-agent system is one which is made up of a number of discrete software modules which work together to solve a problem. They may do this with or without direct communication, and may operate on this same platform or on different ones. Thus, a robotic system may be considered multi-agent either because an individual robot is internally made up of agents, or because it is made up of many robots, each of which is considered to be an agent. This section is concerned largely with the second possibility, as it is this area which has most bearing on the subject matter of this report, and the internal workings of the individual robots is considered to be largely inconsequential. Some background in non-robotic multi-agent systems will first be given, both for context and for relevance in other areas of this project.

Agent and blackboard systems are intended to produce the same effect as multiple human experts, potentially with different skills, collaborating on the same task. In these systems a set of agents (which may or may not be identical) work on data in a single structure, to which they all have access (the blackboard). The classic agent/blackboard example is the *Hearsay II speech understanding system*[27], which uses an agent/blackboard system to understand human speech up to the semantic level.

V.R. Lesser, one of the authors of the *Hearsay* system, gives his own, explicitly personal, view on the field in [28]. Lesser identifies strong interaction as the key factor for effective cooperation in multi-agent systems, as without good communication the different agents cannot efficiently coordinate their actions. He also identifies moving multi-agent systems out of simulation and into the real world as a major goal of work in this area, a goal which multi-agent robotics actively pursues.

More recently, in [29], Daniel Corkill, one of the Pillars of the multi-agent and blackboard community, provides a brief explanation and history of multi-agent and blackboard systems and then looks to the future of what may be possible with such systems. In particular, traditional blackboard systems and more modern multi-agent systems are contrasted; the first favouring centralisation and the latter leaning more towards complete distribution. The paper concludes that the future may lie in combining these approaches, with each agent sharing the duties which would have previously fallen to the centralised element.

Several systems have specifically explored multi-agent robotic systems for underwater domains. The integration of tele-operation and multi-agent robotics is explored in [30]. In this system, user input is added as an additional behaviour, which gives each agent a predisposition to move in the direction specified by the user. The system performs well in the tasks to which it is applied, though the tests are performed in simulation with perfect communication assumed. Additionally, the removal, rather than the addition, of tele-operation is one of the aims of autonomy.

A different tactic altogether is taken in [31], which concentrates almost entirely on its biological influences, at the expense of a system capable of achieving viable goals beyond coordinated movement. Each robot in the system contains a simulated nucleus and cytoplasm, with lights and receivers linked to the "cytoplasm" used for communication. Though this system does present an interesting method for keeping robots in formation, it is not really viable for most AUV applications, as light based communication is unsuitable for communication in open water due to visibility issues. Additionally, the system's slavish adherence to its biological influences makes it non-deterministic, which has consequences in the area of operator trust, and renders it unable to handle complex user specified plans.

The DELPHIS system[32], a recent advancement on this state of the art, integrates an advanced plan representation[33], allowing it to decompose its tasks into the most efficient order. Additionally, it effectively copes with the communication problem by employing a prediction system, allowing each AUV agent to reason about the tasks each of its peers will attempt to complete. This combination of technologies allows the AUVs to maintain a high degree of efficiency, even in the face of vastly reduced communication reliability. The BIIMAPS[3] representation also requires a significant increase in complexity, however, and these plans must (at present) be created in their entirety by the user, who would have to be an expert in the BIIMAPS representation (and likely the DELPHIS system itself for context) in order to do so.

Another recent effort, the GREX project[34], is specifically aimed at missions in which multiple heterogeneous[35] vehicles swim in formation[36] in order to provide

[3] Blackboard Integrated Implicit Multi Agent Planning Strategy

increased coverage during the survey of a particular area, with the added possibility for one or more vehicles to break off to pursue or examine a significant target, and then rejoin the survey. Again, communication between the different vehicles becomes an important factor in these mission and so a reasonable proportion of the research is dedicated to this specific task[37, 38]. The system used by the GREX project is capable of only limited adaptation of its preprogrammed mission, which is essentially a sequential script designed to be carried out by multiple vehicles simultaneously. The mission plan must be generated in its entirety by the user as a sequential list of tasks, though a graphical planning environment has been created to help with this.

## 2.3 What is Planning, and Can It Be Used as a Basis for AUV Control?

Planners are able to generate a series of actions which achieve a particular goal. As such, a planner may be used in order to ease the burden of an AUV operator by allowing them to specify the high levels goals of the mission, and then have a detailed plan generated automatically. Likewise, a planner might also be deployed on the vehicle itself, allowing it to adapt its plan in the face of an unknown or dynamic environment. This represents a great potential strength for platforms which are unable to maintain reliable communication with their operator. This section will attempt to describe the background to planning in general, as well as a more contextual look at the benefits planners are able to provide and the difficulties associated with using them. What this section will not attempt to do is describe in depth the actual functioning of specific planners. This is outside the scope of this section, and indeed this thesis. A excellent reference is available in *Automated Planning: Theory and Practice*[39], however.

The *Stanford Research Institute Problem Solver* (or STRIPS), as first proposed in [40], is generally acknowledged as much of the basis for classical planning. Originally the name for the planner itself, STRIPS later became the name for the formal language used to express input to the planner. This input to a STRIPS system consists of:

- An initial state.
- A goal state (or set of goal states).
- A set of actions which the planner can perform. Each of these consists of:
    - A set of preconditions, which must be true for the action to be performed.
    - A set of postconditions, which will be true after the action has been completed.

The STRIPS planner than attempts to apply actions whose effects satisfy the goal, recursively applying other actions to meet the preconditions until a complete plan, which can be executed from the initial state and ends at the goal, is formed (if one is possible).

*Hierarchical Task Networks*, the basis for the plan representation[33] used by the previously mentioned DELPHIS system, and largely based upon [41] and [42], extend STRIPS and provide a more expressive planning framework. An HTN planning problem consist of:

- A set of primitive tasks, which are roughly analogous to the actions in the STRIPS system and can be executed directly.
- A set of compound tasks and rules for decomposing them into primitive tasks.
- A set of goal tasks, which are roughly analogous to the goals in the STRIPS system, but more general. These are specified in terms of conditions which must be made true.

The HTN planner attempts to satisfy the goal task by decomposing it into smaller and smaller sub-tasks, until the goal can be achieved entirely with primitive tasks. A complete HTN planning algorithm is presented in [43] (and expanded upon in [44] and [45]) which is used to prove that HTN planning is more expressive than the STRIPS planning system.

The Planning Domain Description Language, or PDDL[46, 47], is the state of the art planning language used each year as the basis of the International Planning Competition[4]. The language takes the same role as the input to STRIPS and HTN planning systems, but is vastly more expanded and flexible, with functionality for much more complex tasks, such as continual planning[48], and expressing user preference (as apposed to concrete rules). Despite its power and flexibility, PDDL is not an eminently suitable method of direct input for the task this project is intended to accomplish. Consider the following relatively simple PDDL clause meaning “We would like that the blocks forming the same tower always have the same colour”:

```
(:constraints
  and (preference
       (always (forall (?b1 ?b2 - block ?c1 ?c2 - colour)
                       (implies (and (on ?b1 ?b2)
                                     (colour ?b1 ?c1)
                                     (colour ?b2 ?c2))
                                (= ?c1 ?c2)))))
... )
```

[4] http://zeus.ing.unibs.it/ipc-5/

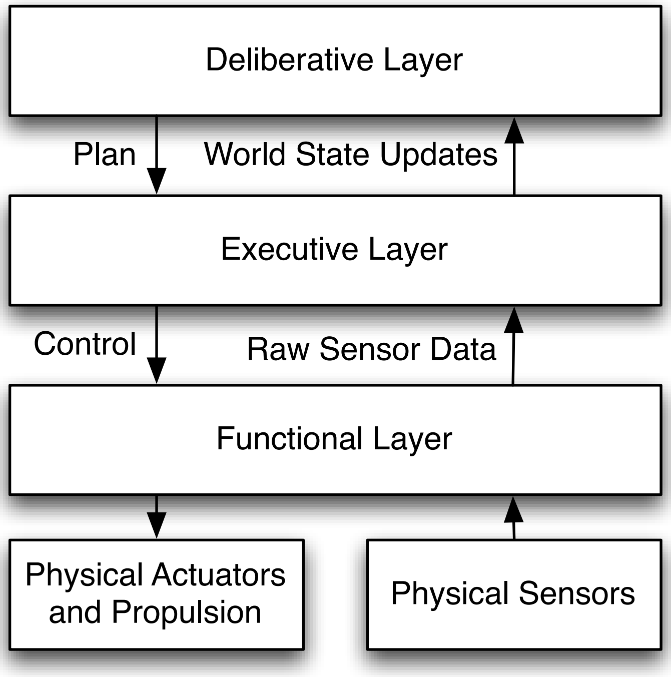


Figure 2.1: Simplified Hybrid Architecture.

Even this simple example might be nearly incomprehensible to someone with no planning, logic or programming background. The primary problem many people without this background would have stems from the fact that PDDL is based on the LISP programming language and therefore uses prefix operators instead of the more usual infix, so "3 + 5" would be written in PDDL (or LISP) as "+ 3 5". This means that the structure of the language lacks familiarity, and familiarity is a very important concept in human computer interaction, which will be dealt with in section 2.6. Though not suitable for direct input, PDDL's power, flexibility and expressivity make it ideal as an intermediate step to creating a plan to satisfy the user's requirements, especially since so much work has gone into creating algorithms which plan using it.

The underwater environment represents a particular set of challenges for planning, many of which are characterised in *The underwater environment: a challenge for planning*[49]. This review again highlights many of the difficulties which are discussed in Sections 2.1 and 2.2, but from the specific perspective of planning systems. These difficulties revolve largely around the variability of the underwater environment, the difficulty of communication, both between vehicles and to the operator, and the platforms' limitation of resources.

Hybrid, or three-layer, architectures have become one of the most popular and successful approaches to autonomous vehicle control. They are able to deploy the strengths of both reactive and deliberative architectures, whilst minimising the weaknesses found in both. Although there is some difference in the naming of the three layers, and the components which are placed in each, certain commonalities exist across most (if not all) implementations. Figure 2.1 shows a representation of a hybrid architecture, which has been somewhat simplified for illustrative purposes.

For the purposes of this example, the three layers are named *deliberative*, *executive* and *functional*. We restrict the functional layer to containing only components which control the physical systems of the vehicle. Reactive control operates within this layer, allowing the system to react quickly to sensor data when required. One example of this would be an emergency evasive manoeuvre in response to the sudden appearance of a large object in front of the vehicle (or at least the detection of one).

We also limit the deliberative layer to containing only the planning components. Thus the deliberative layer receives changes to the vehicle's perception of the world (represented at a symbolic level) and outputs a plan which is designed to achieve the mission goals.

As such, all other components required to make the system work are contained in the executive layer. These may include, but are not limited to: an autopilot, high level sensor processing and fusion, navigation, mission monitoring and action selection (such as the previously discussed DELPHIS system[32]). Also required is a system to convert high level actions into commands which control the vehicle's physical systems, an example of which can be found in the author's previous work *Abstracting The Planner Down Down: An Architecture for Planner Based Control of Autonomous Vehicles*[50], which is discussed in more detail in Chapter 4.1.

A more in depth review of hybrid architectures, together with a recent implementation, can be found in [51]. More recently still, work at Heriot-Watt University's Ocean Systems Laboratory has produced a new design proposal for a hybrid control implementation for autonomous underwater vehicle, which is integrated with an ontology based semantic world model[52, 53].

## 2.4 What Are the Issues of Trust?

Unmanned vehicles deployed in the ground and air domains are no less common than those deployed in the underwater domain, however there are some distinct differences. Communication to these platforms is significantly less problematic than with AUVs, and generally provides a higher bandwidth and lower latency. As a result, there is scope for direct inclusion of a human operator in their control processes. This could take the form of direct tele-operation or a higher level, waypoint based system. In tandem with this, the platform is able to provide the user with copious real time sensor data allowing fast and direct verification of the platform's actions. As such, there is a lower requirement for placing intelligence, and therefore autonomy, on the vehicle itself.

This is not the case with untethered unmanned vehicles which operate in the underwater domain, however. Communication in this case has a high latency and an extremely low bandwidth. As a result, the only opportunities for high bandwidth data exchange between the operator and the AUV are before and after the mission. Most major mission related decisions must be made before the vehicle is deployed, and the vast majority of the sensor data and other mission information cannot be reviewed until after the vehicle has been recovered. Additionally, the operator must make do with a bare minimum of status updates during the mission, which may contain little more than the position of the vehicle. As a result, a higher degree of intelligence must be placed on the vehicle and the operator must trust this to be able to correctly carry out the mission. As more complex requirements emerge for AUVs, so they develop a requirement for a larger degree of autonomy, and the issue of trust becomes even greater still. The issue is considered to be so crucial by some, in fact, that one study[54] was led to conclude:

> "Key challenges remain... Implementation is likely to be progressive in order to foster user trust and acceptance."

The implication here seems quite clear: development of AUV technology should be deliberately and artificially held back in order to prevent trust issues created by the advancement of autonomy from arising.

Sometime before this, however, when AUV technology was to some extent still in its infancy, an alternative possibility was noted in *Untethered AUV'S Can Reduce Costs For Offshore Inspection Jobs* [55]:

> "Once operational experience builds trust in AUVS, many tasks will be performed without the need for costly surface vessel support or personnel."

The trust issue is very clearly still in evidence here, though the solution suggested is less drastic: repeated use of the system should be employed to help reduce this deficit of trust. A near identical position was given much more recently from the perspective of the military domain in *Commander Trust in Autonomous Systems: the Role of Implicit Instructions and System Feedback* [56], which examines, among other things, the effect of feedback on commander trust and notes that the initial deficit is reduced through repeated use of a system, especially if adequate feedback can be provided. This last point is also carried by another paper presented at the same conference, *Human Centred Perspectives on Mission Planning for Autonomous Systems*[57], which states that there is a definite need for interfaces which provide just that:

> "A key aspect of the development of trust in autonomous systems will be the ability to share plans, a function for which interfaces will need to be developed."

Some of the author's previous work[58] described a framework for the creation of such an interface system, with the emphasis on three key factors:

- Using simulation in lieu of the real system to help build trust, whilst avoiding costly deployment of the real vehicle before this trust is built;
- Providing the best possible debrief for all missions, real or simulated, in order to gain as much knowledge as possible and most effectively build this trust;
- Moving the methods of communication as close to those which are native to humans as possible, in order to facilitate this.

This framework is described in more detail in Chapter 3, while later in this thesis the implementation of several key areas which compose it will be described.

## 2.5 What is Situational Awareness?

For the purposes of this thesis, software which provides situational unawareness is that which answers at least one of the following questions in the content of autonomous vehicle missions:

- What has happened in the **past**?
- What is happening in the **present**?
- What will happen in the **future**?

Concurrent to this, there are three phases of an AUV mission at which situational awareness can be provided. Chronologically, these are:

- **Before** the mission starts, when a plan is being created and verified.
- **During** the mission, for monitoring purposes.
- **After** the mission, during a debrief.

It should also be noted that the before phase has the potential to integrate the other two phases through simulation. As the author's previous work[58] shows, he considers this to be key, as it provides an avenue for the operator to gain the experience of multiple uses of the system required to build trust, without the need to deploy the real platform before this trust has been built. As such, most situational awareness technologies which relate to monitoring and debrief should be considered equally applicable to the verification component of the planning phase.

The remainder of this chapter will describe some of the previous approaches to situational awareness which have been applied to AUVs and other tangential technologies.

As most current generation AUVs are essentially deployed as sensor platforms, with missions consisting of a static list of waypoints, current generation operator interfaces[59, 60, 61] are designed with this application in mind. Missions are planned in their entirety by the operator on an entirely spatial basis. The system has no scope or integrated functionality for the verification of the mission after it has been planned. The Gavia Control Centre[61] also integrates a vehicle simulator which allows the planned mission to be tested in real time, allowing basic verification of the path plotted by the operator.

With the addition of a "plug-in", SeeTrack is used as the multi-vehicle planning environment for the GREX project (Figure 4 in [35] shows an example of this interface), which allows the operator to plan their mission as a static list of tasks, again largely based on locations to be visited by the AUVs.

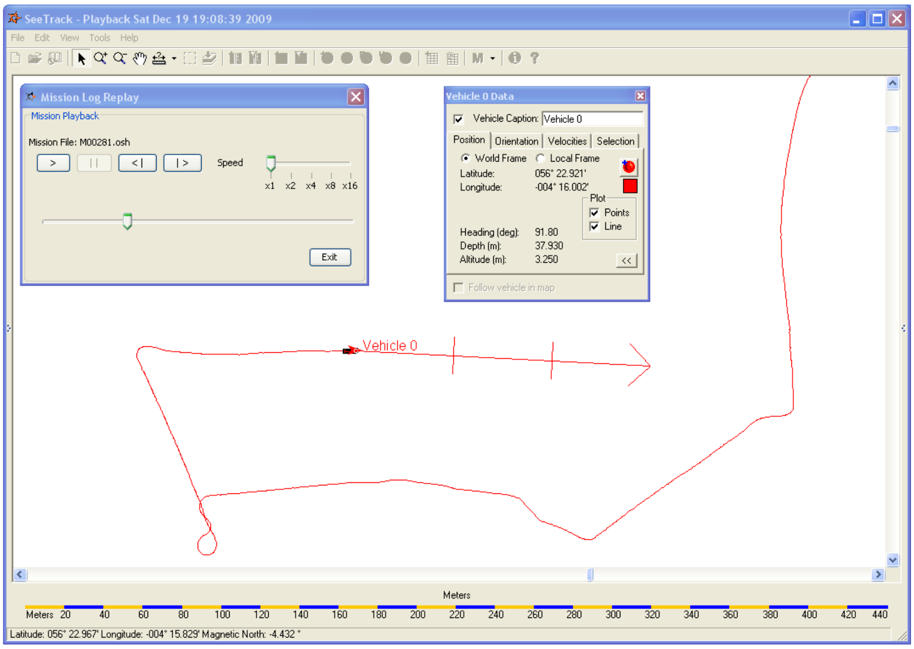
Mission monitoring with SeeTrack is limited to reporting the current position of the currently deployed vehicle(s), as and when this becomes available. When the previously discussed unreliability of acoustic communications is taken into account, the potential for this to produce an accurate representation of the path taken by the vehicle(s) is significantly reduced.

The REMUS VIP and Gavia Control Centre again provide similar functionality for their specific vehicles. The Gavia Control Centre also augments this functionality using its built in simulator, however. When the mission is started, the control centre also starts a simulated vehicle with the same mission, The position of this is then continually updated and shown in the interface, this provides the operator with a reasonable guide to the AUV's current position and likely path. Unfortunately, this is not reliably updated or recalculated based on new position updates received via an acoustic connection with the vehicle, which limits its potential accuracy.

The most important factor in missions carried out by the current generation of AUVs are the path taken by the vehicle and the sensor data collected while traveling along this path. As such, SeeTrack, VIP and Control Centre concentrate on delivering this information to the operator.

With the addition of the plug-in used to support the AutoTracker system[7], SeeTrack gains some additional debriefing functionality. The mission logs can be retrieved from the vehicle platform, and the inter-process messages replayed, allowing the operator to view a replay of the mission with a full "time slipping" functionality, such as variable playback speed, pause and rewind. This allows the user to view the events in the mission in any order which seems appropriate, and see the data being collected, rather than just viewing the completed data set. Figure 2.2 shows an example of this interface.

One example of a situational awareness solution from outside of the underwater domain is the Mobius Command and Control[62] by Autonomous Solutions, Inc. In contrast to the previously discussed solutions, Mobius is designed to provide real



SeeTrack - Playback Sat Dec 19 19:08:39 2009

Mission Log Replay

Mission Playback

Mission File: M00281.osh

Vehicle 0 Data

☑ Vehicle Caption: Vehicle 0

● World Frame ○ Local Frame

Latitude: 056° 22.921'

Longitude: -004° 16.002'

Heading (deg): 91.80

Depth (m): 37.930

Altitude (m): 3.250

Plot: ☑ Points ☑ Line

□ Follow vehicle in map

Latitude: 056° 22.967' Longitude: -004° 15.829' Magnetic North: -4.432 °

File Edit View Tools Help
>
| |
< |
| >
Speed
x1 x2 x4 x8 x16
Exit
Position
Orientation
Velocities
Selection
<<
Vehicle 0
Meters
Meters 20 40 60 80 100 120 140 160 180 200 220 240 260 280 300 320 340 360 380 400 420 440

Figure 2.2: The SeeTrack user interface, with mission replay controls (top left). The vehicle's previous positions are shown in red, with the red arrow indicating its current trajectory.

time feedback and control to the operator. As a result, it provides a visually rich user interface with numerous options and possibilities for direct data visualisation, as well the ability to issue new commands and plans to the autonomous assets during the mission. It requires a large amount of data throughput to the vehicle to do so however, making it more suited to autonomous ground and aerial vehicles, and less directly applicable to the underwater domain. It does however provide an example of the possibilities a rich user interface can provide to a situational awareness solution.

Work in the Ocean Systems Laboratory at Heriot-Watt university has produced the Augmented Reality Framework[63, 64] (ARF), which is designed to assist with mission simulation and visualisation. ARF integrates a 3D graphics and simulation environment with the facility for simulated sensors, which can be directly correlated to the real world if required. This allows tests to be performed entirely in simulation, or with a combination of simulated and real components. ARF is optimised for real time simulation, however, and so is less supportive of the faster than real time simulation which might be used for rapid mission assessment, or to provide a greater focus on the debriefing process.

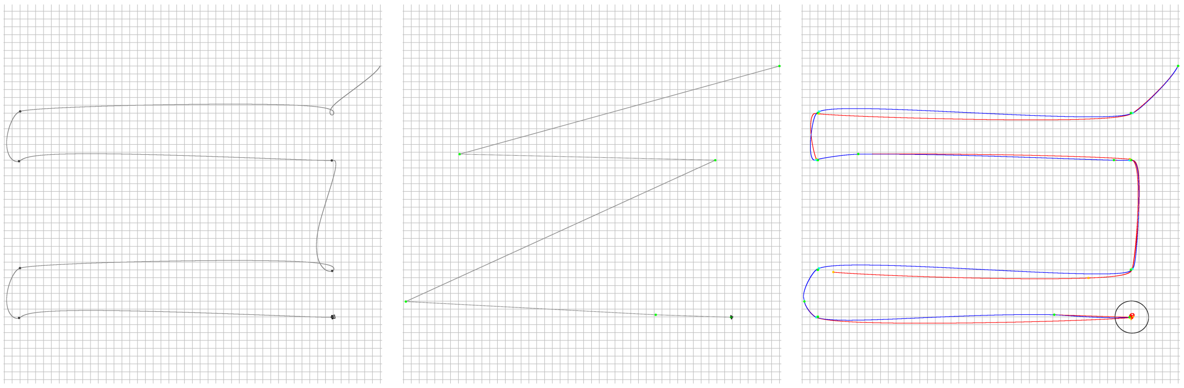
Moving closer to the question of *what is happening now*, the DARPA sponsored Pilot's Associate Program [65] was a project intended to create an intelligent situational awareness tool for military pilots. It analysed the current state of the pilot's mission, along with real-time data fed to them from both the aircraft's internal systems and external sensors, and then intelligently give the pilot whichever data was pertinent to the current mission objectives. So in a standard mission situation, the system might display the location of the next objective, together with a map of the local geography, whereas in the event that the plane is damaged, the displays will shift to show a path to the nearest friendly runway and an overview of the damage. More recently, the Rotary Pilot's Associate (RPA) Program extended this work, giving more focus to the Cockpit Information Manager [66], which dynamically decides which information should be given to the pilot on the various displays. This system performed well in simulated tests, when pilots were found to be notably willing to ignore some mistakes on the part of the system in light of the frequency at which it provided the right information, and the system's perceived predictability. Additionally, RPA is cited in [67] (albeit by the same author), which describes methods for finding the appropriateness of the information displayed by an interface, as an example system which successfully and accurately adapts itself to the needs of the user.

Military pilots find themselves in extremely dissimilar circumstances to AUV operators. They have instant and complete access to data, and are required to make split second decisions based on this which directly affect the kinematics of their vehicle. AUV operators, on the other hand, are supplied with delayed access to an extremely limited subset of their vehicle's data during a mission, and the decisions they are able to make are quite limited. In addition, as the state of the art progresses, they might be required to monitor multiple autonomous platforms simultaneously. Many of the principles which might be deployed to create a situation awareness interfaces are the same, though. All of the information should be available, but this should be streamlined according to its relevance in order to prevent cognitive overload on the part of the operator.

The Triton system[68], developed at the Ocean Systems Laboratory, is an attempt to maximise the amount of information which can be provided to an AUV operator despite the limited communications bandwidth provided by an acoustic link. The system utilises the prediction techniques used in the DELPHIS system (see Section 2.2), the faster than real time simulation techniques described in Chapter 4.2, and several other novel technologies.

On receipt of status update from a deployed vehicle, Triton employs a trajectory estimation system which computes the most likely path which would have taken the vehicle from its last known position to its new position.

Concurrent to these calculations, on receipt of the status update, an instance of an appropriate simulated vehicle is created. Data from this simulated instance is then used to give the user an estimate of the vehicle's current position. This simulation is performed faster than real-time, however, and this information cached, allowing the operator to also view the vehicle's likely actions in the future. Additional elements can also be added to the simulation, such as faults, using the fault simulation system described in section 4.2.4. To reflect the simulated nature of much of this data, a kinematic vehicle model is used to generate an ellipsoid which



(a) The vehicle's actual path. (b) The current state of the art. (c) The Triton system (blue line).

Figure 2.3: Comparison of the paths generated by the current state of the art and those generated by the Triton system, to the vehicles real path, based on position updates received every 1080 seconds.

represents the range of the vehicle since the last update. This is displayed whenever the operator views times subsequent to the most recent update, providing a measure of the potential error of the system.

As a result of these techniques, the Triton system is able to give the user reasonably accurate answers to each of three questions posed at the beginning of this section, together with more complex "what if?" questions regarding the vehicle's future actions. Additionally, it is able to produce predicted and calculated paths which much more closely resemble those of the actual vehicle (see Figure 2.3). Examples of its user interface can be found in Figure 2.4, which shows an annotated example of the system's two dimensional "debug" interface, and Figure 2.5, which shows an example of the system's three dimensional "production" interface.

## 2.6 What is Human Computer Interaction?

For our purposes, situational awareness is essentially a sub-set of the larger and more general field of Human Computer Interaction, or HCI. A maxim which was originally used in the world of semi-conductor physics[69], but has since been subsumed into the world of HCI is "the interface is the device" and this is particularly applicable to computational devices, such as AUVs, which do not have their own direct interface for human interaction. In these cases, information must be exchanged via a proxy interface, such as those described in the previous chapter, and so this interface must be as effective as possible. HCI is the study of all aspects of the interface between computers and their human operators, and so has a high degree of relevance for this work.

A much cited and seminal work in the area of human orientated design is Donald

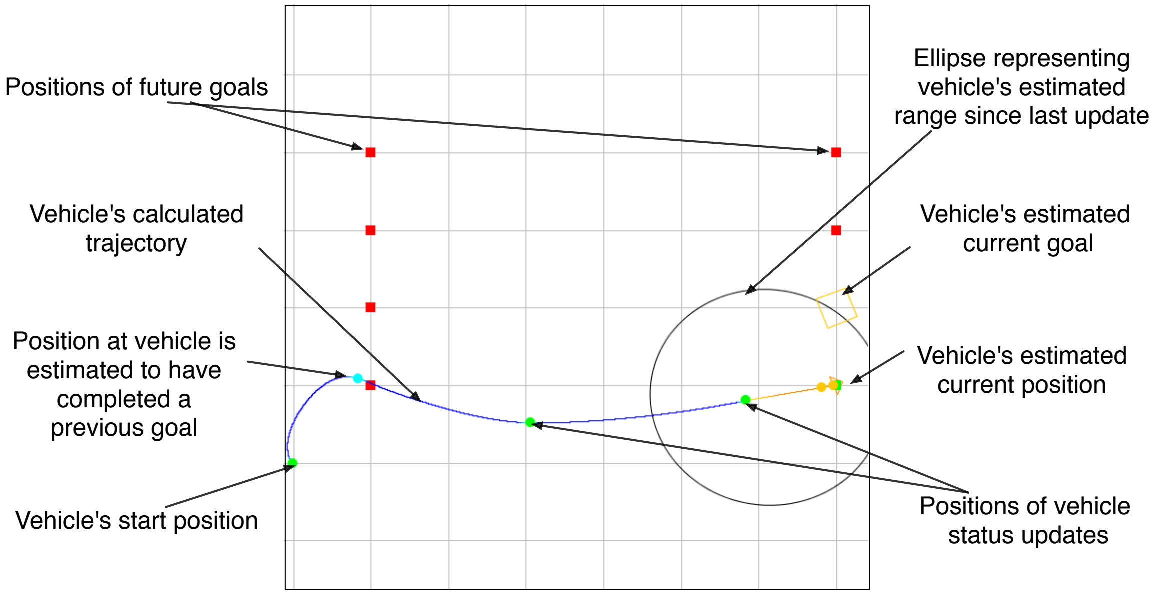

Positions of future goals
Ellipse representing vehicle's estimated range since last update
Vehicle's calculated trajectory
Vehicle's estimated current goal
Position at vehicle is estimated to have completed a previous goal
Vehicle's estimated current position
Vehicle's start position
Positions of vehicle status updates

Figure 2.4: Annotated example of the Triton system's two dimensional "debug" interface.

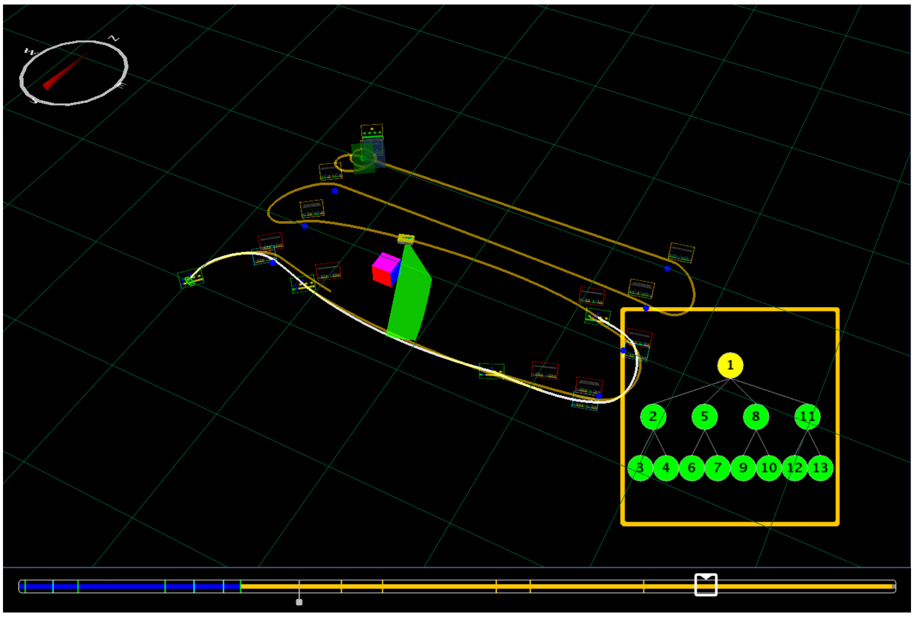


Figure 2.5: The three dimensional, Augmented Reality Framework based, "production" version of the Triton interface.

A. Norman's *The Design of Everyday Things*[70]. Although the specifically computer related content has become quite dated, the all pervasive use of metaphors (desktop, recycle bin, etc) in modern computer interfaces means that many of the principles, practices and cautions contained within are still very relevant. Taking the most relevant points and examples it is possible to give a very concise summery of [70] as a series of guidelines. These are:

1 Provide a good conceptual model - give the user enough clues about the way a system works that they can easily predict the outcome of their actions.

2 Make things visible - make it easy to see which actions the user can perform.

3 Provide strong mapping - actions should suggest their outcomes (e.g. turn the steering wheel of a car suggests turning in that direction).

4 Provide feedback - acknowledge user actions and show their effects.

5 Reduce the effect of errors - Errors should be easy to detect, they should have minimal consequences, and, if possible, their effects should be reversible.

6 Reduce memory load - put as much knowledge as you can in the world so that the user has to remember as little as possible.

7 Use all available means to convey information to the user.

8 Exploit constraints - where possible, don't give the user options which will certainly cause errors and prompt on actions which are likely to.

9 Avoid over automation - don't take control away from the user.

10 When all else fails, standardise.

This provides a good set of guidelines, which the design of any device should seek to follow. These are not computer specific, but this is part of their strength, as familiarity is an important factor when designing an interface to a computer system, as is suggested by the tenth guideline, and will be noted in the further, more computer specific guidelines listed later in this section.

A more recent work, which deals specifically with HCI, is *Human Computer Interaction[71]* by Alan Dix et Al. A standard work at the university level, it takes a lower level, more engineering based approach to [70], providing case studies and the actual implementation of systems, rather than their higher level design and functionality. Chapter 4.3 of this work provides a similar, but more focused, set of guidelines in the form of a list of properties a system should have:

1 **Predictability** - system behaviour is observably deterministic.

2 **Synthesisability** - the user can assess the effect of past actions.

3 **Familiarity** - match the interface to users' expectations.

4 **Dialogue Initiative** - give user control of the dialogue flow.

5 **Multi-Threading** - provide support for simultaneous tasks.

6 **Task-Migratability** - negotiability of function allocation between user and system.

7 **Substitutability** - equivalence for different forms of input expression.

8 **Customisability** - interface is capable of being adapted to suit different needs.

9 **Observability** - relationship between system state and its presentation.

10 **Recoverability** - support for undoing errors.

11 **Task Conformance** - interface functionality should match common tasks.

12 **Responsiveness** - feedback should be commensurate with action.

Clearly some of these guidelines echo those taken from [70], particularly those which relate to information visibility and providing feedback to the user. However these guidelines are aimed specifically at computers and thus are more specific. In particular, the Multi-Threading, Customisability, Observability add to our knowledge, introducing the idea that the system can do more than one task at a time and adjust its interface as appropriate to suit both the user and the task being carried out. In some ways this actually conflicts with the previous guidelines, which would tend to suggest that the interface to the system should not change, the key here clearly is that any changes must be called for by and appropriate to the task they are intended to suit. In this way a system can follow the new guidelines and stay true to the spirit of the previous ones. Additionally, the "Dialogue Initiative" guideline echoes the previous warning against "Over-automation", while further introducing the concept of an actual dialogue between the system and the user. The system should not be expected to just carry out the tasks it has been assigned, but also collaborate with the user in order to find out exactly what is required and the best way to achieve it, where possible.

Following on from his work on the Rotary Pilot's Associate, Christopher Miller produced the very interesting *Rules of Etiquette, or How a Mannerly AUI should Comport Itself to Gain Social Acceptance and be Perceived as Gracious and Well-Behaved in Polite Society* [72]. This very short paper does not present a human interface system or any results, but rather posits the idea that any advanced user interface is essentially having a conversation with the user and making decisions on their behalf, and in this case should follow rules, or etiquette, similar to those which would be followed by a human who was fulfilling the same function. He then goes on to suggest such a set of rules to accomplish this. These are:

1 Make many, many correct conversational moves for every error made.

2 Make it very, very easy to override and correct your errors.

3 Know when you are wrong, the easiest way to do this is to let the human tell you, and then get out of the way.

4 Don't make the same mistake twice.

5 Don't show off. Just because you can do something, doesn't always mean you should.

6 Be able to talk explicitly about what you're doing and why. Humans spend a lot of time in metacommunication activities facilitating coordination, especially in distributed work environments.

7 Make use of multiple modalities and information exchange channels redundantly; understand the implications of your communications on all the levels on which it operates.

8 Don't assume every user is the same, be sensitive to and adapt to individual, cultural, social, contextual differences.

9 Be aware of what the user knows, especially if s/he knows it because you recently conveyed it (i.e. don't repeat yourself).

10 Be cute only to the extent that it furthers your conversational goals.

Some of these guidelines clearly echo those we have seen seen previously. The second again emphasises the importance of recovery from errors, which both previous sets of guidelines have stressed. Guideline five echos the previous guidelines' warning against over-automation and specifying the need to give the user the dialogue initiative. The sixth is essentially a more specific version of the previous guidelines on visibility, as well supporting the need for a good conceptual model first given in the first set. The seventh guideline again calls for the interface to use every method of conveying and receiving information available, but also to take into account the appropriateness of the channel, and the different meanings which can be taken from the information. Finally, the eighth reflects the customisability guideline from the second set. However, whereas the previous guideline suggests that the user should be able to alter the interface to suit themselves, this guideline calls for the interface to adapt itself to suit the user. On the whole, these guidelines deal with much higher level issues and expect a lot more from the interface. Not only must it be well organised, it must be actively so. It should converse with the user to find their needs, instead of just presenting the possibilities. Then it must be able to explain what it is doing and, more importantly: why.

## 2.7 Commercial Influences in Human Computer Interaction.

Many developments in the HCI field have also been germinated in more commercial sectors, where methods are rarely published for public consumption. In these cases the relevant systems must often be analysed in a "black box" fashion. There are some exceptions to this, however. Apple Computers provide a set of extremely comprehensive guidelines[73] for anyone wishing to develop software for use with their OSX operating system and its highly acclaimed "Aqua" user interface. Though much of this document is concerned with ensuring that third party developers adhere to the established style of the Aqua interface, a large portion is also given over to maintaining the user friendliness of the system. Although much of the usability data contained within the document echoes [71], it is placed in context with what is essentially a huge case study.

Another software area which has created some advances in interface technology is computer games. Although quite easily dismissed due to what some may consider to be their frivolous nature, in this case there are two distinct reasons why they should be considered both valid and a source of highly relevant information.

The first is familiarity. The Entertainment Software Association, or ESA, indicates that the computer and video games industry in the United States of America generated $11.7 billion worth of revenue in 2008, with a total of 298.2 million software units being sold across consoles, computers and portable systems[74]. Furthermore, the same research indicates that 68% of American households play computer or video games, and within this the average game player is 35 years old, while 49% of gamers are between the ages of 18 and 49. Adult gamers having been playing computer and video games for an average of 12 years. The ESA's report *Video Games in the 21st Century: Economic Contributions of the US Entertainment Software Industry[75]* further indicates that while the growth of American economy as a whole (for the periods 2003-04 and 2005-06) was growing a rate of less than 4%, that of the US entertainment software industry exceeded 17.0% for the same periods. The market for the computer games in the United Kingdom is the third largest in the world, after the USA and Japan[76]. It is quite clear from these data that a reasonable proportion of the contact some people have with computers is through computer games. This being the case, the use of an interface which is reminiscent of one which may be found in a computer game will provide some familiarity, as suggested by the guidelines above.

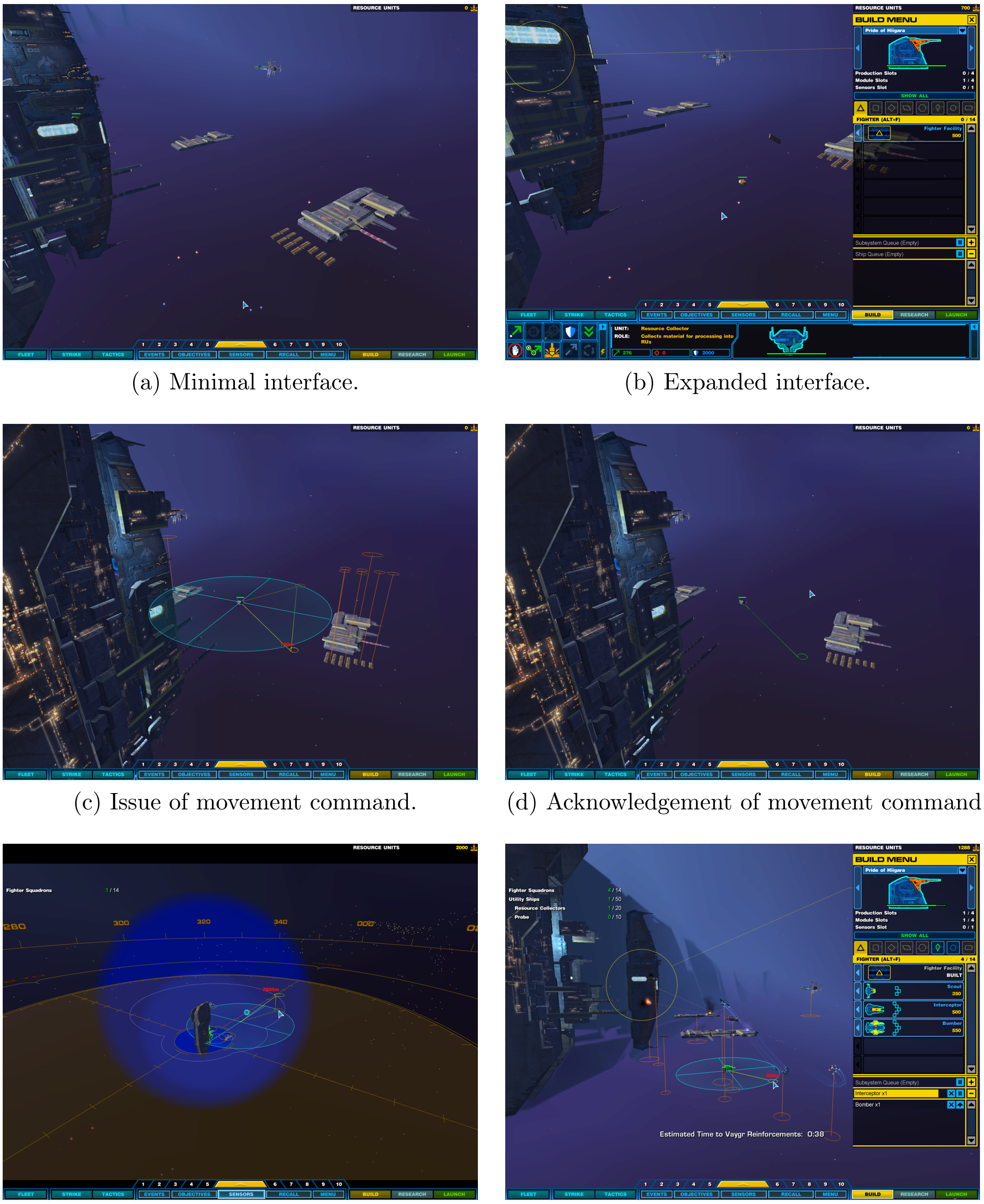
The second is the technology itself. The user interface is an extremely important element of any computer game. They aim to provide interactive entertainment and their interface must support this. If it does not then the game will quickly become frustrating, will not played and, more importantly to the companies which publish computer games, it will not sell.

The style of computer game whose interface is most appropriate for this work is almost certainly the “Real Time Strategy” (or RTS) genre. RTS refers to games in which the player takes the role of a general controlling an army, issuing instructions to single, or groups of, military units, in order to claim victory in the recreation of a battle. These games are typically viewed from a third person perspective, and characterised by the fact that the player has little or no control of the units beyond the high level instructions they can give (go to, attack, etc.) and the fact that play occurs in real time (as the name suggests), rather than the turn based structure of more traditional war games.

Homeworld[1] and its sequel[77] are two RTS games particularly known for their excellent interfaces, however this game is also notable due to the degrees of freedom involved in its gameplay. Where as most RTS games give the player control of ground units such as soldiers and tanks, in the Homeworld games the player controls spacecraft with the full six degrees of freedom. The increase in the complexity of the games requires a more complex interface, which the game provides without sacrificing elegance. The vast amount of additional data which the play requires (positions of friendly and enemy units, resources, ships which are being built etc.) is supplied to the player either intelligently by the game or on request, depending on the importance of the information. This means that the interface is never crowded and the player is always given the maximum amount of space to view the task they are currently working on.

Figure 2.6a shows the minimal version of the Homeworld interface, with as much of the screen as possible giving a world view, which is contrasted in Figure 2.6b by the more complete interface which gives more information. The switching between these two modes is controlled by the user, in line with the previously stated guidelines for Customisability. Figures 2.6c and 2.6d show a movement command being issued and the system acknowledging this, respectively. Notice how clear feedback is given, so that the user can asses what effect their instructions will have before they are issued and can see that they are being carried out, giving the system both Synthesisability and Observability. Finally, Figure 2.6e show a long range world overview and in Figure 2.6f the system provides relevant information to the user.

However, while RTS games do use a control method in which the player issues high level commands but has little to no control over the specific actions of the units these commands are issued to, these commands are issued in real time and can be directly countered by the player should the unit perform any undesirable action. This is not the case in AUV control, in which the “player” must make all their decisions beforehand and then allow the “units” to go about their tasks with little or no further control over them until the tasks have been completed. It is in this area where the analogy between RTS games and AUV control begins to break



(a) Minimal interface.

(b) Expanded interface.

(c) Issue of movement command.

(d) Acknowledgement of movement command.

(e) Long range view.

(f) Provision of contextually relevant information.

Figure 2.6: Examples of the graphical user interface from *Homeworld*[1].

down, because directly mimicking this entirely hands off approach is not possible in computer games, simply because it would isolate the player from the gameplay, and computer games are always intended to be immersive.

## 2.8 How Can Visualisation Help?

Visualisation could be considered one of the primary sets of techniques which the principles of human computer interaction are able to draw upon. Visualisation techniques allow broad spectra of data to be represented visually, and thus potentially be more easily understood by the human observer. In the literature, visualisation is defined, thus:

> “The standard argument to promote scientific visualization is that today’s researchers must consume ever higher volumes of numbers that gush, as if from a fire hose, out of supercomputer simulations or high-powered scientific instruments. If researchers try to read the data, usually presented as vast numeric matrices, they will take in the information at snail’s pace. If the information is rendered graphically, however, they can assimilate it at a much faster rate.”[78]

And thus:

> “[Visualisation] transforms the symbolic into the geometric, enabling researchers to observe their simulations and computations. [Visualisation] offers a method for seeing the unseen. It enriches the process of scientific discovery and fosters profound and unexpected insights. In many fields it is already [revolutionising] the way scientists do science.”[79]

Specifically, we are interested in those aspects of visualisation which deal with creation of a visual representations of symbolic data which describe the state of the world, goals which must be achieved and plans which are designed to accomplish these. These are, of course, the fundamental components which are required for planning, but might also be called beliefs, desires and intentions, in the parlance of multi-agent systems.

An example of a system specifically aimed at visualising a plan, specifically a hierarchical task network, can be found in [80]. The system remaps the HTN to a 3D representation, the tasks being displayed as boxes, which can be “opened” (the top two thirds made transparent) to reveal sub tasks, with dependencies displayed as arcs between them. The system is initially more easy to view then a standard 2D HTN representation, such as may be given by the SIPE-2 system described in [81] (see Figure 2.7 for an example), as the layout is less crowded, however, more

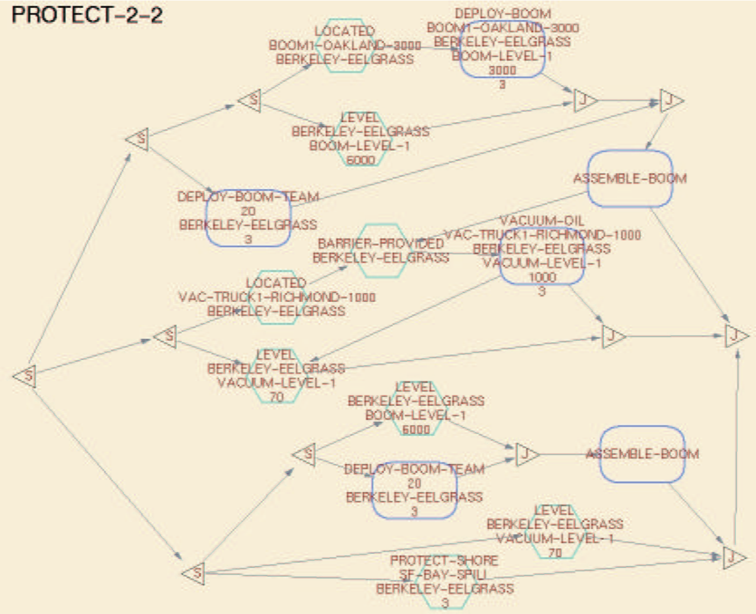
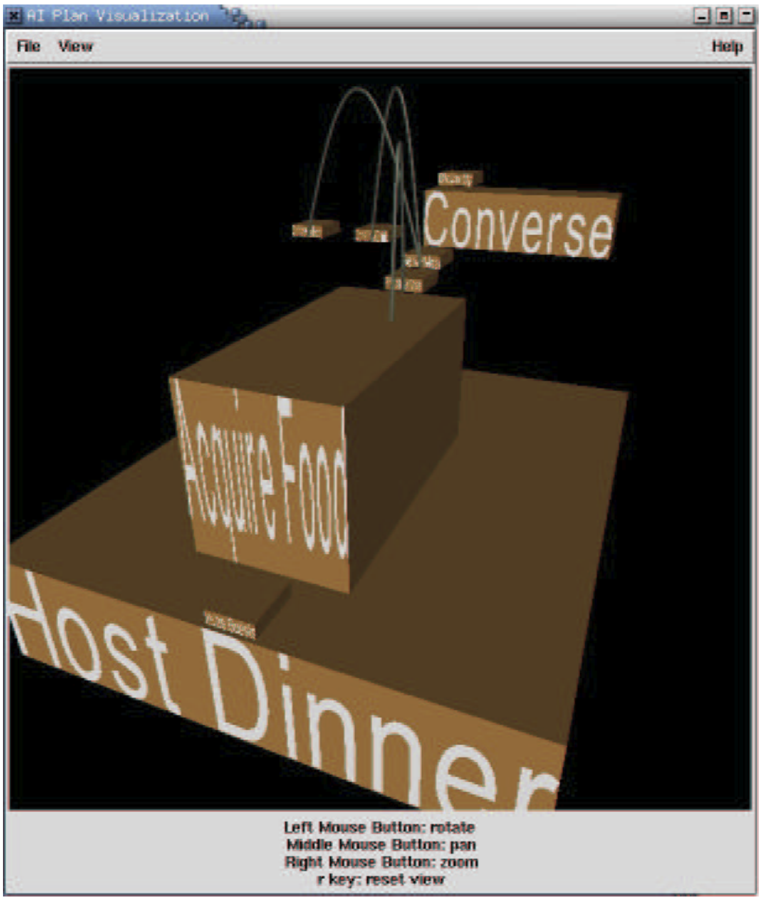


Figure 2.7: Two Dimensional Hierarchical Task Network visualisation.



Left Mouse Button: rotate
Middle Mouse Button: pan
Right Mouse Button: zoom
r key: reset view

Figure 2.8: Three Dimensional Hierarchical Task Network visualisation.

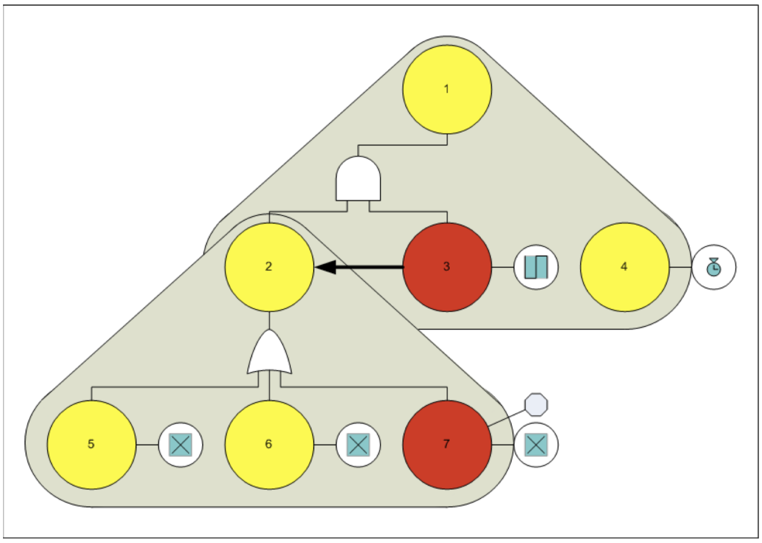


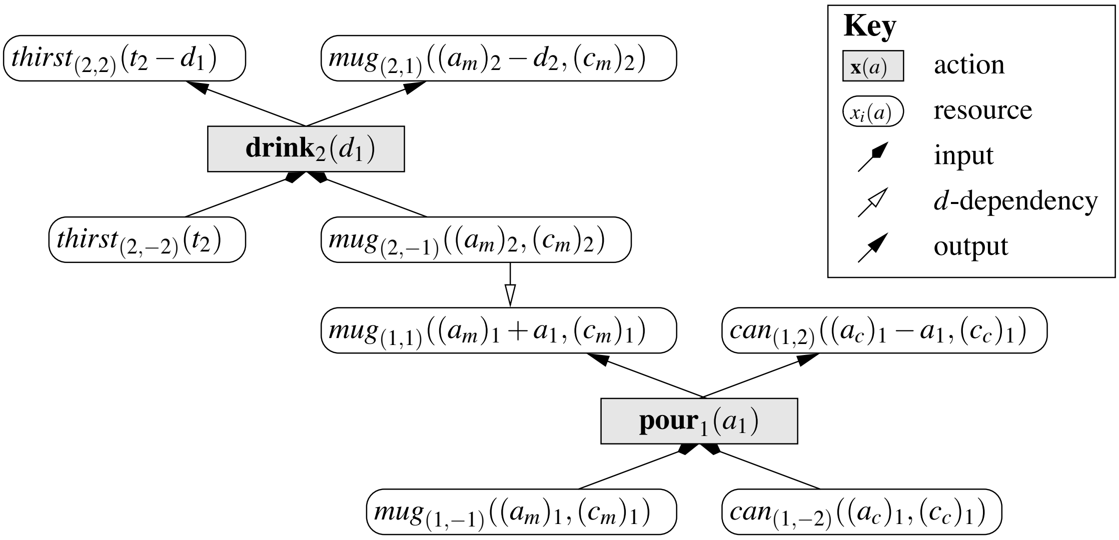
Figure 2.9: Example BIIMAPS visualisation.

effort is required for navigation, as for complete viewing the representation must be manipulated in 3D, thus only some information may be available from a particular angle and moving up and down the levels in the network can be quite fiddly (See Figure 2.8 for an example of this visualisation).

Some of the author's previous work produced the *Blackboard Integrated Implicit Multi-Agent Planning Strategy[33]*, or BIIMAPS, which is an example of a plan representation which features ease of visualisation among its design goals. This system is again based on hierarchical task networks, and its visualisation is intended to reflect this hierarchy by enforcing a pyramid shape, with each compound goal occupying the top corner of a rounded triangle which encloses its sub goals. BIIMAPS is designed to be an operational representation, rather than just a formalism, and so its visualisation is designed to be equally operational and able to represent the changing state of the plan. The goals are coloured red, amber, or green, to show that their status is respectively unavailable, available or completed. The logical combination of sub goals which is designed to satisfy a compound goal is represented using the same logic symbols employed for logic gates in electrical circuit diagrams. The actions required to complete the goals are represented visually by a glyph in a second circle to the right of the goal, connected to it by a horizontal line.

The BIIMAPS visualisation at present has a significant shortcoming, in that in order to make it as visually compact as possible it only displays a bare minimum of information about each goal and action. An example BIIMAPS visualisation, showing each of these elements, can be found in Figure 2.9.

While BIIMAPS has, to some extent, been proven as an operational plan representation [33, 32], its visualisation capabilities have yet to be significantly tested, however.



Key
x(a)
action
x_i(a)
resource
input
d-dependency
output
thirst_(2,2)(t_2 − d_1)
mug_(2,1)((a_m)_2 − d_2, (c_m)_2)
drink_2(d_1)
thirst_(2,−2)(t_2)
mug_(2,−1)((a_m)_2, (c_m)_2)
mug_(1,1)((a_m)_1 + a_1, (c_m)_1)
can_(1,2)((a_c)_1 − a_1, (c_c)_1)
pour_1(a_1)
mug_(1,−1)((a_m)_1, (c_m)_1)
can_(1,−2)((a_c)_1, (c_c)_1)

Figure 2.10: An example Action Resource Formalism plan visualisation. Taken from [2], page 80.

In his PhD thesis, *Plan Merging in Multi-Agent Systems*[82], Mathijs de Weerdt describes the *Action Resource Formalism.* This provides a complex formal algebra for the description of the actions which make up a plan, the resources these require and produce, and the dependencies between them. The formalism is intended to act as a *lingua franca* for planning systems, and has gained traction enough to be used in at least one other project [2, 83] (albeit one from the same laboratory). Demonstrated together with the description of the formalism is a strategy for the visualisation of the same, featuring a distinct symbology for each element.

The visual representation is expressive enough to represent plans in partial or total order, together with the changes to the world which result from these. It does, however, lack a specific representation for goals, or desires. These, instead, are represented in an identical manner to resources, or beliefs, with a lozenge shaped node, containing a symbolic representation of the resource. Actions, or intentions, are represented with a rectangular node, making them visually distinct. Connections between these are made by arrows, which have differing appearances depending on their function. Figure 2.10 shows an example of such a visualisation.

One potentially unfamiliar element of this visualisation style, however, is that it represents causation, and therefore time, as flowing *upwards.* In most traditional process representations, time is represented as moving down and to the right. This can also be found in many less formal circumstances, such as comic strips and texts in most western languages (such as this thesis, for example), together with media control systems, which uniformly represent a timeline as moving from left to right, and represent the "play" action with a right facing arrow.

A second potentially unfamiliar element is the labels placed in each node of the graph. These show the same symbols used in the textual representation of the Action Resource Framework. Thus they provide a convenient and familiar visual representation to an expert in this, but are likely to be highly unfamiliar to one who is not. As an example taken from Figure 2.10, the symbol $mug_{(2,1)}((a_m) - d_2, (c_m)_2)$ is used, which is likely to be less familiar to a system operator than an equivalent representation in natural language, such as "The mug is empty," or even a more complex phrase which maintains all of the information in the original symbol: "After the second action, the mug is empty."

The visualisation techniques discussed thus far have all produced essentially static visualisations. Some have represented a plan with little or no reference to time [80, 81, 33], while others have provided a representation of how a plan changes the world over time [82], but still via a static graph. A static graph for a complex plan has the potential to take up a large amount of space (particularly if natural language is used in place of symbolic references), which may be more than is available on a single computer screen. Thus, a methodology for displaying a portion of a plan, whilst maintaining the ability to connect this to the remainder of the plan, may be required. The first part of this may be relatively simple, as all that would be required is a schema for selecting which elements should be displayed at each time. The second has the potential to be more problematic, however. It is here that animation may be of some assistance.

One previous study[84] which demonstrated a file system view based on animated rotating "cone trees" remarked in its discussion:

> "It is easy to demonstrate that animation shifts cognitive load to the human perceptual system. While it is difficult to quantify the time it takes to cognitively re-assimilate structural relationships after a tree transformation without animation, it is clear from a simple demonstration that it is many seconds, and perhaps tens of seconds depending on the complexity of the [hierarchy]."[84]

The use of animation to show the relationships between differing partial views of the same data has also been found to be beneficial in other studies[85].

## 2.9 How Can Natural Language Generation Help?

Just as visualisation provides techniques for generating visual representations of symbolic data, so Natural Language Generation provides techniques for generating verbal representations. The complementary field to Natural Language Understanding, which seeks to convert natural language into symbolic representations, it has received significantly less attention. As a symptom of this, in the 934 page university level textbook *Speech and Language Processing*[86], only 35 pages are devoted to natural language generation.

Historically, there are three types of natural language generation systems. The first, and simplest is the "canned text" system. In these systems the individual

| name | item | data |
|---|---|---|
| Terry Pratchett | hat | 28/04/2009 |
| Douglas Adams | towel | 11/03/2001 |
| Neal Stephenson | computer | 31/10/2009 |

Table 2.1: Example mail merge database.

phrases are entirely static and unchanging, but may be reorganised into a different order.

The second most complex system is the "template based" natural language generation system. In this kind of system, template phrases with variable "slots" are used. These slots are then replaced with relevant words in order to build apparently unique phrases. An example of this type of system might be used in an automated "mail merge" type email system, such as might be used for sales receipts. Consider the following snippet of a template, with variables placed between "<%" and "%>" symbols:

> "*Dear* <%name%>*,*
>
> *Thanks you for your purchase of a* <%item%> *on* <%date%>*...*"

This could be combined with a data base, such as the example given in Table 2.1, to produce three apparently distinct and personalised email receipts. Another reasonably famous example of template based NLG is the ELIZA[87] system created by Joseph Weizenbaum, which held a dialogue with the user by simply rephrasing their answers as questions.

The third and most complex type of system is the "rule based" system, which uses actual grammatical rules to generate phrases based on symbolic information. As a result, these systems are potentially able to generate a larger number of phrases. They are also able to generate multiple variations of the same phrase, such as changing the tense of the expression, switching to a passive form of the verb, or changing number agreement ("this" as opposed to "these"). As a consequence of their need to perform computations beyond simple text replacement, they are considerably more complex than canned text and template based systems.

In more recent times, however, the line between template and rule based systems has begun to blur as several systems emerged which deployed elements of each. This phenomenon is discussed in detail in *Real vs. template-based natural language generation: a false opposition?*[88], which concludes that the line between the two types of system has been crossed by more modern template based systems which employ *syntax* based templates in place of the text based templates exemplified above, and rule based systems which employ template-like simplifications. Thus, hybrid systems have emerged which display the strengths of both approaches.

*Building Natural-Language Generation Systems*[89] identifies a generic pipeline to be used by a complete natural language generation system. It is made up of the three primary stages, each of which is subdivided, thus:

- **Document planner**.
  - **Content determination**, which decides what information will appear in the output text.
  - **Document structuring**, which decides how chunks of content should be grouped in a document, how to relate these groups to each other and in what order they should appear.
- **Microplanner**.
  - **Lexicalisation**, which decides what specific words should be used to express the content.
  - **Referring expressions**, which decides which expressions should be used to refer to entities (both concrete and abstract).
  - **Aggregation**, which decides how the structures created by document planning should be mapped onto linguistic structures such as sentences and paragraphs.
- **Surface realiser**.
  - **Linguistic realisation**, which uses rules of grammar (about morphology and syntax) to convert abstract representations of sentences into actual text.
  - **Structure realisation**, which converts abstract structures such as paragraphs and sentences into mark-up symbols which are used to display the text.

Supplemental to this, Reiter's own open source *SimpleNLG*[90] software package provides a near complete implementation of the surface realiser stage of this pipeline, with the intention to later also provide microplanning services. Reiter and his colleges' BabyTalk[91] project aims to create a complete natural language generation pipeline with a with a specific focus on creating textual summaries of temporal data from a Neonatal Intensive Care Units. These summaries are generated from the large volumes of data produced directly by the medical instruments attached to patients[92, 93], potentially simplifying the workflow of the doctors and nurses on the ward, whilst also providing summaries on demand to patients and their families. Among the issues brought up by this study is the importance of narrative when building natural language texts[94], a subject which will be covered in more detail in the next section.

Scott Nowson's facetiously titled MSc thesis, *Being John Motson*[95], provides a characterisation of one potential application of natural language generation: the dynamic generation of spoken commentary for football (soccer) computer games. Its conclusions, though, are likely to apply both to sports besides football and mediums outside of computer games. The current generation of football games (as of the thesis' writing) use prerecorded commentary (often made by well known real commentators) which is triggered by particular events during play. As result there are a limited number of unique phrases which the system is able to produce, and this leads to repetition. While the thesis leans towards the conclusion that computational commentary in general was of a low standard, repetition was found to became a more of a problem roughly in proportion to the uniqueness of the particular phrase:

> "Some utterances are even quite original and entertaining, but once they occur more than a couple of times, they actually become more annoying than the obviously predictable phrases."[95], page 79.

Also found to be exacerbating the problem of repetitions is the studied system's lack of discourse history:

> "The play-by-play commentator is agreeing with the colour commentator's agreement with something he just said. The system is failing to take into account discourse history... Again, like all errors, it sounds really bad."[95], page 72.

ILEX, as presented in [96] is an example of an natural language based interface implementation which attempts to avoid repetition, and thus follow the ideal of Miller's 9th rule (see Section 2.7). ILEX is a dynamic hypertext system, which displays information in a format that is similar to that found on many web pages. Where the system differs from standard hypertext is that it keeps track of which specifics the user knows, and then simplifies this data when it reoccurs. One potential failing of the system is connected to the human tendency to scan text rather than read it. Just because the system has presented the user with a page of text doesn't mean it has been read in its entirety, but the system is forced to make this assumption. Despite this potential weakness, the system seems successful.

## 2.10 How Can Narrative Help?

Whilst visualisation and natural language generation techniques provide methods for communicating the more atomic components which are required for situational awareness, such as how to display a plan and express its individual actions, they are less helpful for deciding what order information should be delivered to the user.

The authors of *Advanced Outlines, Familiarity and Text Genre on Retention of Prose*[97] attempt to study the effect of various variables on their subject's ability to recall prose passages. These variables were:

1 Whether the subject is provided with an outline in advance of reading the passage;

2 Whether the subject has a preexisting familiarity with the text;

3 Whether the text is narrative or expository in nature.

While the provision of an outline was found to have no effect on recall, and familiarity was found to have a marginally negative effect, this was not the case for the genre of the text. Retention scores for narrative texts were significantly higher than those for expository texts. There is a small flaw in the methodology used here, however, in that the expositive and narrative texts are not themselves necessarily directly comparable, in that they do not present the same information. The narrative texts used were:

- Noah's Ark;
- Story of Jonah;
- Snow White;
- Serpent Story;
- Story of Bodisat;
- The Princess and the Pea.

Whereas the expository texts were titled:

- Earthquakes;
- Emotions;
- The Earth's Orbit;
- Coal Energy;
- Harvester Ants;
- Armadillos.

While only "The Earth's Orbit" is given in full, it seems quite obvious that there is little, if any, crossover of subject matter between the narrative and expository texts. Regardless of this, however, this study provides a reasonably convincing suggestion that information which is presented narratively is more likely to be retained by the human mind than that which is present expositively.

Subsequent to this, the work of Pennington and Hastie[98, 99, 100, 101] discovered and demonstrated a different benefit of narrative, in an entirely different arena. In this work, they develop and test the "Story Model", an explanation based theory for juror decisions, and then test this by gauging the effects of various different factors on the decisions made by jurors. As a result they found, as they had posited, that the factor which had the largest effect on jurors' decisions, and their confidence in these decisions, was the coherence of the narrative which was presented to them. In particular, they found that evidence was more effective when presented in an order based on "story", rather than one based on witness or legal issue, and the provision of explicit inferences to the story had a greater effect still. They concluded:

> "These results... support the claim that stories are the mediating mental structures that cause decisions in the juror's judgement task."[100]

This represents an additional argument in favour of the use of narrative to structure information. Examples of narrative used to aid in the dissemination of information can also be found in popular literature. Two famous and critically acclaimed examples are *Zen and the Art of Motorcycle Maintenance*[102] and its sequel *Lila*[103], by Robert M. Pirsig. In each of these, Pirsig is able to fuse near text-book level philosophy with related narrative. The first follows the protagonist's motorcycle journey with his son following a mental breakdown, using almost all aspects of the narrative as hooks upon which to hang concepts of philosophy and logic. Not least of these is the motorcycle itself, which among other things becomes an exemplar for the difference between deductive and inductive logic.

Further example of this fusion of expository information and narrative can be found in Neal Stephenson's *Cryptonomicon*[104], which follows numerous narrative strands from differing periods of history, all having some relevance to the field of cryptography. Whereas Pirsig often breaks out of his narrative to deliver relevant philosophical lessons, or "Chautauquas" as they are referred to in the book, Stephenson often fuses his more directly, using a malfunctioning bicycle ridden by Alan Turing (a character in the story) as a metaphor with which to explain the mechanism of RSA encryption, for example.

What, precisely, is narrative, though? A more important question, perhaps, is: how can we define it with sufficient rigour and formality in order to be able to use it computationally in order to best structure information which is provided to an operator? In general, there are three fields which study narrative directly. These are literary criticism, cognitive psychology and interactive narrative, and each studies a different facet of narrative, from a different direction.

Literary Criticism is analytical in nature, specifically of the text itself, and to a lesser extent its author. Literary criticism is subjective by its very nature, and thus much work in this field does not appear in any sort of peer reviewed context. Some works which have been published in printed form do frequently get cited as inspiration in more academic papers, however. One example which has been quoted by works in the interactive narrative field is Seymour Chatman's *Story and Discourse*[3], which is written to be a guide and reference to "narrative structure in fiction and film." As well as taking an openly subjective critical look at many classic works and their use of many narrative devices, the book also described several important concepts in reasonably formal terms. Not least among these are the meaning of the two terms which make up its title:

- **Story**, which is analogous to content. It has internal structure, but not order. Story is data, and is subdivided, thus:
  - **Existents**, or stasis data, which describe the state of the world and are true for a period of time. They take the form "is."
  - **Events**, or process data, which occur at a particular time and change the state of the world. They take the form "does."
- **Discourse**, which is analogous to expression. It delivers the elements which make up the story to its audience, in a concrete order. Discourse is communication.

These are reasonably abstract concepts which wrap data, and at their base level they transfer directly to the AUV domain. Here an example of stasis data could be that an AUV is capable of performing a sidescan sonar survey, and an example of process data the same AUV's discovery of a mine like object.

Figure 2.11 presents the portion of the "Diagram of Narrative Structure" from page 267 of *Story and Discourse*[3] which describes the relationship between story and discourse, replotted in the form of UML. As shown in the diagram, there are additional properties of the events and existents. Events have a necessity, they are either a kernel or a satellite, specifying whether they contribute to the main plot or an additional side plot. They also have an agency, either action or happening, depending on whether they are triggered directly by a protagonist or by external forces, respectively. The first of these properties has little relevance in our domain, there are no side plots[5] in an AUV mission, there are only the goals which must be accomplished. The second is germane when operating in a dynamic world, however, where some events will be the results of the AUVs actions, whereas others might be the result of environmental factors or additional (potentially antagonistic) agents.

[5] In this context, a side plot is taken to be a secondary plot which exists for the purposes of characterisation, rather than to progress the main narrative.

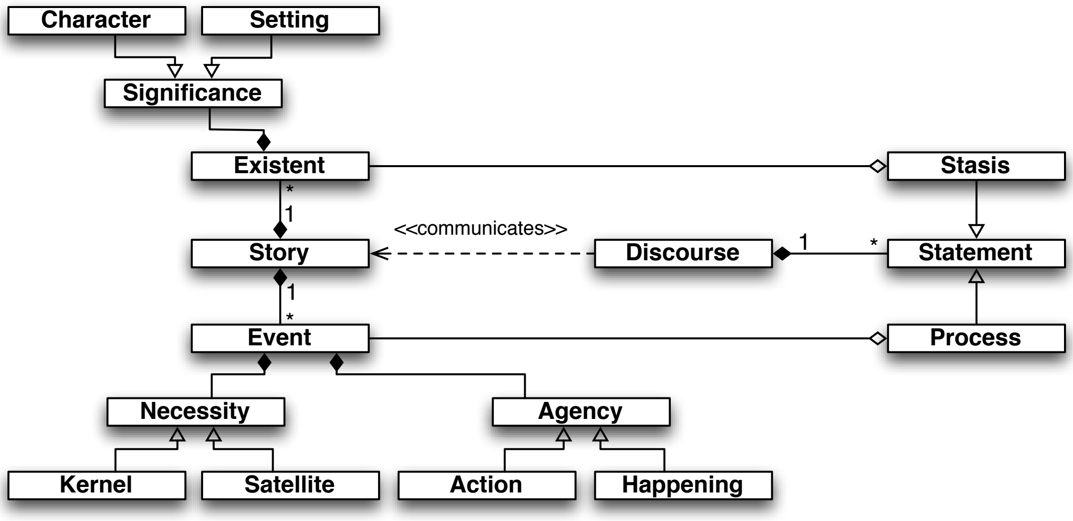


Figure 2.11: UML translation of the "Diagram of Narrative Structure" from page 267 of *Story and Discourse* [3].

Existents have the "significance" property, which states whether they refer to character or setting, which in the AUV domain might translate to details of an AUV versus details of the environment, such as the position and status of an offshore installation.

In *Morphology of the Folk Tale[105]*, Vladimir Propp took the concept of process statements to a less abstract level by breaking the plots of Russian folk tales down to their simplest irreducible narrative components. He found that a total of 31 plot functions was all that was required to represent each of the extensive list of stories he studied. Though specific to a particular genre of story telling, this provides an example of an abstraction of these narratives, and shows the potential for a similar deconstruction with autonomous underwater missions as the genre in question.

Cognitive psychology is, like literary criticism, analytical in nature. However, it seeks to analyse the audience rather than the author or the discourse itself. In the context of narrative, cognitive psychology is particularly concerned with the audience's understanding and mental representation of the story and discourse.

In *How does the mind construct and represent stories?*[106], the authors (who include Arthur Graesser, the co-author of [97]) discuss levels of representation for narrative in the human mind. Their model consists of six levels of representation, which are:

1 **Surface Code**. This code preserves the exact wording, syntax and intonation of the explicit text.

2 **Textbase**. This level captures the meaning of the explicit text in the form of a set of propositions and a small number of text-connecting inferences that link the propositions.

3 **Situation model**. A mental microworld of what the story is about, including the setting, the characters, and events in the plot.

4 **Thematic point**. The moral, adage, or main message of the story. A story may support multiple thematic points.

5 **Agent perspective**. The agent who tells the story creates a point of view (i.e. perspective) from which the story is told.

6 **Genre**. The category of narrative (e.g., mystery, folktale, romantic novel) has a typical context and structure.

Important corollaries can be drawn between the earlier three levels and other relevant work which has previously been discussed. Firstly, the first two levels correspond exactly to discourse, as described in *Story and Discourse*[3], initially in a fully realised form and latterly in a more abstract propositional form. The third level then corresponds to story as described in the same work. This correlation is to be expected, as the two works are essentially discussing the same subject, albeit from different points of view. This version adds an extra level of representation however, and a more formal description.

The situation model is potentially the most important of these representations for our purpose, at it represents both the sum total of the events which occurred during an AUV mission and all of the required supporting information. Described in [106], but discussed in considerably more detail in *Situation Models in Language Comprehension and Memory*[107], are the five situational dimensions along which each of the elements which make up the situation model can be plotted, giving it a defined internal structure. These dimensions are:

- **Space**. Where did the event happen?
- **Time**. When did the event happen?
- **Protagonist**. Who performed the event?
- **Motivation**. Why did the event happen? What goal was being worked towards?
- **Causality**. What caused this event to happen? What further events were caused as a result?

Previous research[108, 109, 110, 111] has found that a discontinuity between a newly introduced element and the audience's current situation model in one or more of these dimensions has a negative effect on the coherence of the discourse and leads to an increase in the comprehension time required for the reader (in the case of written text) to integrate the new knowledge into their situation model.

A second corollary which can be drawn to the previously mentioned levels of representation for narrative and discourse connects to the implicit levels of representation inherent in the natural language generation pipeline described in Section 2.9. The situation model would fit in between the "content determination" and "document structuring" stages of this pipeline, forming a symbolic representation of the information which is to be delivered to the audience with no specific ordering. The textbase then would be the still symbolic but now ordered discourse representation output by the "document structuring" stage for input to the "microplanner". Finally, the surface code would be the now fully realised discourse output by the "surface realiser" component.

It should be noted here that the levels of representations as stated by *Graeser et al.* are intended to be analytical in nature, taking the input and progressively deconstructing it into more abstract forms. The natural language generation pipeline, however, is obviously generative in nature. As a result this becomes a crucial mapping, as it translates the analytical framework onto a series of generative processes, as is required to generate a discourse from a story.

Interactive narrative, unlike literary criticism and cognitive psychology, is essentially purely generative in nature. It seeks to create narratives with which the user is able to interact with and affect. This combination of computational media and an intent to entertain invites comparison between interactive narrative and computer games. This is quickly refuted in *Chris Crawford on Interactive Storytelling*[112] by the author, who is himself one of the original progenitors of the computer games industry and founder of the *Game Developers' Conference*[113] and *The Journal of Computer Game Design*[114]. He argues:

> "Visualizing interactive storytelling in terms of games is rather like describing a whale by using a camel as a reference. Sure, they're both mammals, but they are so different that the effort is a waste of time and ultimately misleading." [112]

As interactive narrative seeks to generate valid narratives to entertain or otherwise involve a human audience, it draws quite heavily on the knowledge generated by the literary criticism and cognitive psychology fields[115]. Pure entertainment is not the only application of interactive narrative, however. James Niehaus and James Thomas, of the Liquid Narrative Group from North Carolina State University, describe the application of interactive narrative techniques to aid teaching and learning in their respective papers[116, 117]. Both attempt to employ the previously discussed benefits of narratively structured information in order to aid users in engaging with and recalling data. Both of these works essentially present proposals of work, rather than successful results, however.

The work of Yun-Gyung Cheong, also of the Liquid Narrative Group, focuses on taking user generated content and applying narrative techniques in order to generate a natural language[118] or visual summarisation[119]. The user generated content in this case is taken from log files generated by the user's play of a computer game, specifically a "long lived" game, such as an online role-playing game, which the user might play over a period of months or years, potentially with large gaps between periods of play. This work essentially seeks to develop a debriefing based situational awareness tool (as discussed in Section 2.5), designed to give the user an efficient way of viewing the events which occurred in the game world outside their own direct experience.

Based on the published description, this work seems successful, though very few results are published, and those which are published are somewhat anecdotal in nature. Particularly in the earlier work[118], the process used is also described in very general terms. The data analysis performed to generate the resultant discourse in the system also seems quite linear in nature, sticking to a purely chronological ordering and concentrating on filtering for the importance of events (described as "kernels" in line with the terminology used by Chatman[3]). This seems to be a valid filtration methodology for a system designed to summarise many months worth of play events, many of which may justifiably be considered inconsequential. When working for situational awareness of AUV missions, however, this assumption is likely to break. As noted, there are no "side plots" or "character moments" in AUV missions, thus all events must be considered to be kernels and very little can be considered to be inconsequential with absolute certainty.

In conclusion, narrative presents a potential methodology for structuring the information which is passed to the user. Structuring information in this way has been found to ease the audience's comprehension and retention of the information, whilst also lending credence to and helping to build trust in a particular version of events. Literary criticism and cognitive psychology present sources of information which can help to build a system for narrative structuring of information, whilst interactive narrative presents another valid use of some the same techniques.

## 2.11 Summary

This chapter has described previous work which is relevant to this thesis, either through clarification of the problem it is designed to solve or influence on the implemented solution. This work spans a wide variety of fields, and is not limited to academic work, but also encompasses products and methodologies developed in industry.

A total of ten areas of interest have been discussed, beginning with those which illuminate, and to some extent necessitate, the main body of the work described in this thesis. These sections described Autonomous Underwater Vehicles themselves, multi-agent systems and their relevance to AUVs, planning based control of AUVs, and culminated in the problem of trust which has emerged between autonomous systems and their human operators. Following on from this, the remainder of the chapter dealt with technologies which have potential to help bridge this gap, beginning with those, such as situational awareness and human computer interaction, which are directly related to the problem at hand. These were followed with discussions of technologies which have the potential to support advanced user interfaces, namely visualisation and natural language generation. Lastly, the field of narrative, which has the potential to provide structure to the flow of information between the autonomous system and its operator, was discussed.

Taken as a whole, this review of relevant previous work is intended to fulfil the research aim 1 given in Chapter 1.

# Chapter 3

# A Framework Designed to Increase Trust Between Operators and Autonomous Systems.

Autonomous Underwater Vehicles (AUVs) have several strong advantages over the more traditional remote presence systems. They have no constricting tether, are controlled by computers (which excel at tedious, repetitive tasks) and require very little surface support and are able run stealthily for long periods of time without needing to return to the surface. Their unmanned nature gives them the potential for a compact design, and the range of the mission is limited only by the amount of power carried by the AUV. Furthermore, whereas the loss of human life should be avoided at all costs, an AUV can have an exact fiscal cost attached to it.

Increased autonomy comes with pitfalls of its own, however. Autonomous systems require a plan in order to specify what tasks must be completed and what dependencies exist between them, and the operator must have complete trust in this plan. Furthermore, they must also trust that the autonomous system has the ability to complete it. This is made more difficult by the fact that an autonomous system must be able to adapt to circumstances, and therefore may appear to be not observably deterministic if the operator is not expecting this behaviour. This issue of trust is noted in both [56] and [57].

There exist a large number of systems which allow a user to generate a plan based on their requirements, however many of these systems require the use of extremely complex notations in order to enable their complete expressivity and power. Additionally, the primary source of validation is often a direct test, either with the real system or a simulated equivalent, with any corrections to the plan being implemented either through changes to the original specifications, or a direct edit of the plan itself.

One potential solution to this problem is to provide operators with sufficient training to use these systems, both to create plans and to adapt them to deal with

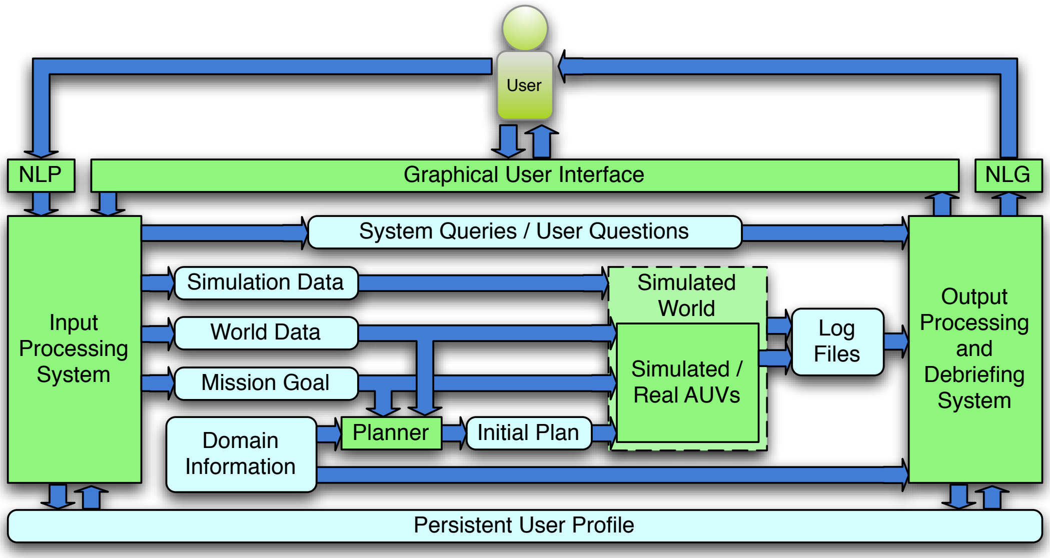


Figure 3.1: A framework designed to increase trust between operators and autonomous systems.

problems. However, since a strong foundation of logic and maths knowledge is often required to fully grasp the complete functionality of many planning systems, any training scheme runs the risk of being either extremely time consuming and expensive, or too superficial and highly inadequate.

This chapter presents a novel framework for the alternative approach: bringing the interface closer to the user, rather than taking the user closer to the interface. By leveraging and combining existing semantic technologies to bridge the gap between the complex notations used to generate plans and human expression itself, a system for which only a very small amount of training is required can be created. This ensures that the most appropriate communication vocabulary between the user and the system is selected, and furthermore that a coherent plan is generated. The plan can then be accurately simulated or performed by the real system. The framework aims to achieve a high level interface to the planning system and operator situation awareness at all stages of the mission plan creation and evolution process.

## 3.1 Components of the Framework

Figure 3.1 gives an overview of the framework as a whole, showing the most important processes and data representations together with the data flow between them. Figure 3.2 represents the chronology of this data flow as it forms a coherent system. The elements which make up these diagrams will now be described in more detail.

The **NLP** (Natural Language Processing) component processes human language input and passes it to the **input processing** component in a more computationally understandable form.

The **graphical user interface** serves the dual purpose of providing visual output to the user, as supplied by the **output processing** component; and accepting user input in the form of mouse clicks of on screen controls and elements, and passing them to the **input processing** component.

The **input processing** component takes the user input, passes it to the **output processing** component if it is a question about the current system state, or uses it to formulate the **unknown world**, **known world** and **mission goal** representations required for simulated and real missions. In the event that there is an ambiguity or contradiction in the current data, this component passes a query to the user (via **output processing**) to resolve this.

As previously mentioned, **system queries** and **user questions** are passed from the **input processing** component to the **output processing** component in order to answer the user's questions and prompt the user to clarify any ambiguities or contradictions the system has found, as suggested by Miller's third rule (see Section 2.6 and [72]).

The **unknown world** represents the parts of the simulated world the user does not wish the planning system to know about at the start of the mission. For example, in the case of a Mine Counter-Measures (MCM) mission, this would refer to the position of the simulated mines, as these are the things the user is testing the simulated system's reaction to and these data would not be available to the planner at the start of a real mission.

The **world data** refers to the initial state of the world used as the starting point for the planning system, including all the AUVs involved in a mission, their capabilities, any objects in the world which are known, and any specific properties they may have.

The **mission goal** is the goal used by the planning system, both as a world state to work towards and as a benchmark for the completion of the mission.

The **domain information** directly corresponds to the domain employed in the specific representation of a planning problem in PDDL. It contains the types which can exist in the world, the properties which can be assigned to these types and the actions which can be applied to the world by the planner. Also coupled with this information is any data required to convert the PDDL constructs and symbolic data to natural language for output to the user.

The **planner** component of this framework is a full planner which is used to create the **initial plan** for the AUVs to follow at the start of the mission, and may differ from the planning engine which might be used for re-planning on board the individual AUVs, which could be more stripped down and simple.

The **initial plan** is the plan which the AUVs are following at the beginning of the mission, but which may be changed by on-board re-planning systems. Again using the example of an MCM mission, the initial plan would most likely be that an AUV with a search capability should search the specified area. This plan would then be augmented should the search AUV discover any potential mines.

The **simulation data** is used to create a **simulated world** and the **known world**, **mission goal** and **domain information** used to create a set of **simulated AUVs** within it. These are then employed to perform a simulation of the mission. Alternatively, once a plan the operator is satisfied with has been found, the same box which represents the simulated AUVs again serves to represent the real AUVs which carry out the mission, as these should take the same input and produce the same output. This duality allows real missions to be debriefed in the exact same manner as those which have been simulated, maintaining familiarity for the user.

The **output processing** component of the framework attempts to answer any of the user's questions about the current plan and system state, and also prompts the user to resolve any ambiguities or contradiction detected by the **input processing** component. Its primary purpose, however, is to keep the user situated as to the progress of the mission in as much detail as is required, and moreover to provide these data in a structured manner, so the user is not bombarded by unnecessary and potentially confusing information, but also does not lack relevant data when it is required.

The **natural language generation** takes the output of the **output processing** component and converts in to (either spoken or written) natural language.

The purpose of the **persistent user profile** component is twofold. Firstly, it is used to maintain a store of all the information the system has given to the user, so the **output processing** component can avoid repeating redundant data, as suggested in Miller's ninth rule. Secondly, it stores data relating to the user's preferences and how any previous contradictions or ambiguities where resolved, in order to aid the **input processing** component's inference of the user's meaning, moving towards the ideas suggested by Miller's fourth rule. The **persistent user profile** as a whole supports Miller's eighth rule, allowing the system to adjust itself to different users.

## 3.2 Migration of Data

Figure 3.2 presents the sequence of the data migration within the system. At all stages in this pipeline it should be assumed, even when not explicitly stated, that the **output processing** module is passing all relevant information to the user regarding the current system state, and the **persistent user profile** is recording both the information being presented to the user and the user's responses to the system queries, as suggested by Miller's sixth rule.

As can be seen, the process is cyclical, enabling the operator to both adjust to the plan for correctness and test the system's reaction to different circumstances. It

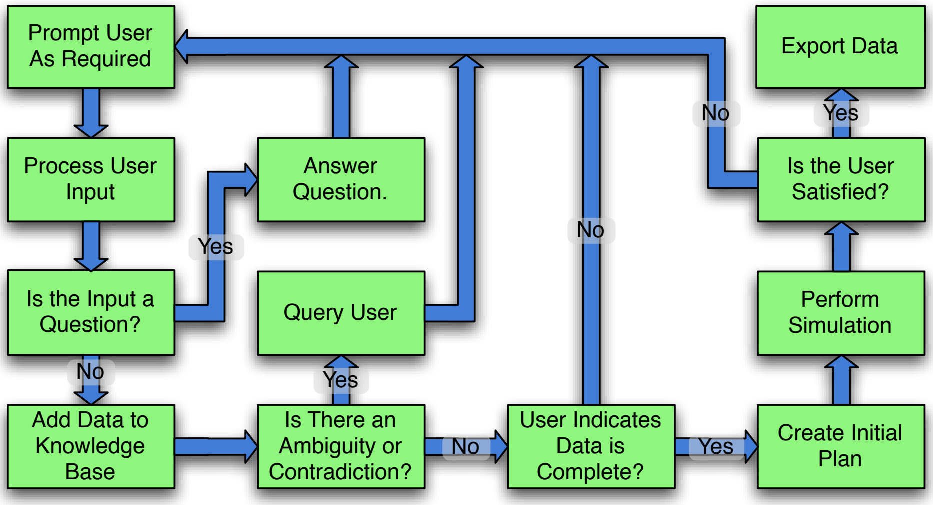


Figure 3.2: Chronology of data migration within the framework.

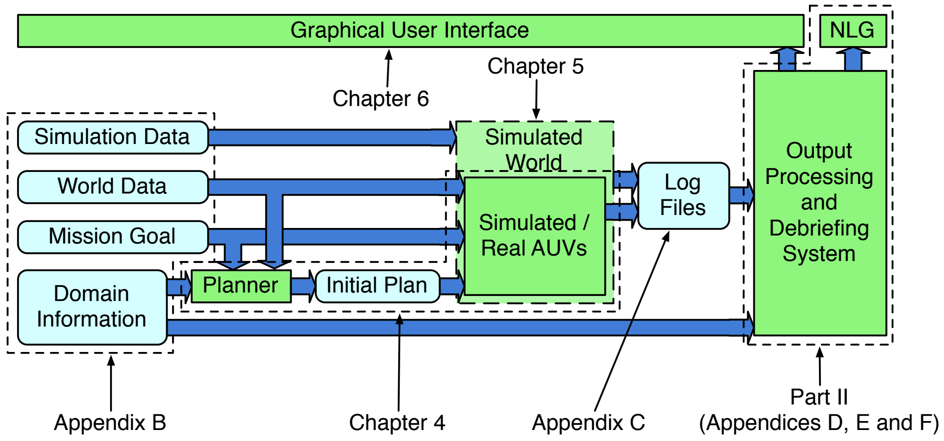


Figure 3.3: The components of the framework which have been developed as part of the work contained in this thesis.

is noted in [56] that the initial gulf in trust between the user and an autonomous asset can be reduced through repeated use of the system, and although these are not always tests performed with the real system, it is thought that these repeated simulated trials will act in lieu of the real system for the purpose of gaining this trust.

The system described here represents the larger, long term picture. Current work is focused around the output processing and user interface based components, with temporary systems in place to provide direct input to these. Presently, no development is taking place on the user input related components. Figure 3.3 shows a stripped down diagram of the framework, featuring only the components which have been developed as part of the work related to this thesis. Additionally, the diagram also illustrates the parts of the thesis which describe each of these components.

## 3.3 Summary

This chapter has described a novel framework designed to help bridge the gulf of trust between AUVs and their human operators. It is intended to bring the human computer interaction elements of an AUV system as close to the user as possible in order to mitigate the need for operators to possess deep knowledge of their internal function. Additionally, it employs simulation in order to provide the operator with verification of the mission parameters and enable iterative improvements to be made. Lastly, it is specifically designed to take the differences between different users into account and adjust its modes of communication and decisions as required.

This framework is intended to fulfil research aim 2 described in Chapter 1, and it is from it in turn that the specification for the supporting technologies described in the following chapter are taken.

# Chapter 4

# Supporting Technologies

This chapter describes three supporting technologies which have been developed in order to support the narrative debriefing system described in Part II of this thesis, as specified by research aim 4 in Chapter 1. The remainder of the chapter is divided into four sections, the first three of which each describe one of these supporting technologies, with the fourth providing a summary of the chapter as a whole.

## 4.1 A Planner Based Control System for Multiple Real or Simulated AUVs

This section describes a system which has been implemented to act as a prototype control system for autonomous vehicles. It uses the same planner to both create plans and replan them in real time during a mission. It is designed to act both as a testbed, by fulfilling the requirement for such a system in the framework described in chapter 3, and an incubator for more advanced systems. Figure 4.1 shows which components of the framework this chapter describes.

The remainder of this section is divided into four parts. The first describes the implementation of the planner itself, the second how the planner is used at the highest level of an autonomous vehicle control system, the third describes the results which have been obtained using this system, and the fourth describes some future work which might be done to improve the system.

### 4.1.1 Planner Implementation

The previously mentioned PDDL[46, 47] was chosen to be the the syntactic and semantic base of the planning system, as it is well established and much previous planning work has been done using it. To this end, a set of objects was created in Java, implementing the syntax and semantics of PDDL up to version 3.1. A parser was also created, allowing the correct object structures to be created from

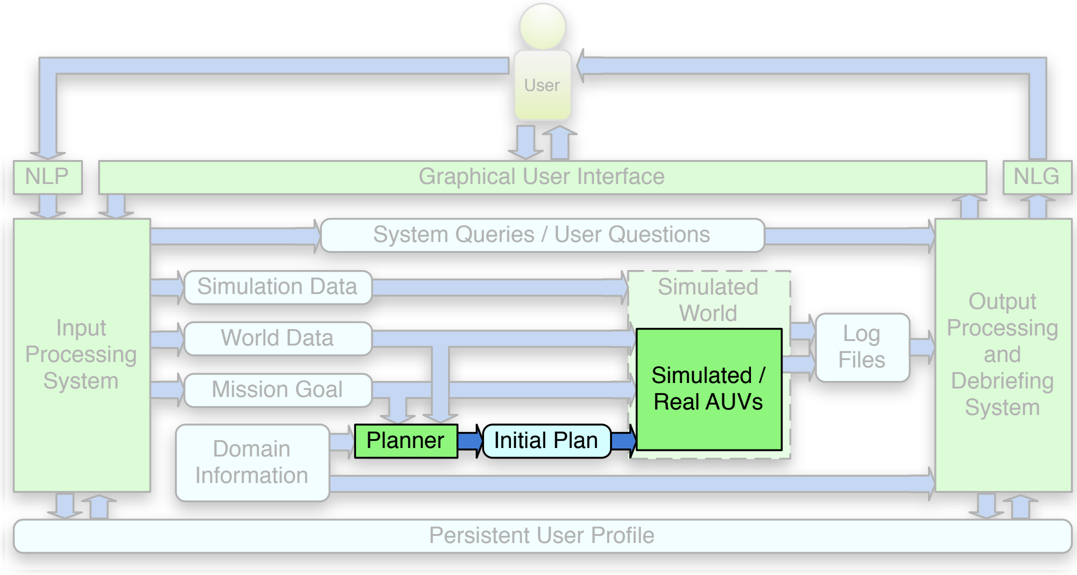


Figure 4.1: The components described by this section and how they fit into the framework described in Chapter 3.

PDDL in its textual form. Furthermore, a "World State" object was also created, further implementing the correct semantics and allowing "Actions" to be applied to a representation of the current world state (assuming the world state first satisfies their preconditions) and the internal state monitored. This object also keeps track of the status of the predefined goal, as well as the status of any preconditions and constraints.

A planner was required for the reasons previously noted, with the following preconditions:

1 The algorithm should be fast, reliable, and proven.

2 Time constraints mean it should require a minimum of time and effort for implementation.

3 In order to best develop the human interface systems, the functioning of the planner should be understood completely.

The second of these preconditions was considered to be best satisfied in one of two ways:

1 Its complete source code should be available and documented (ideally in Java); or

2 Its algorithm should be completely published and explained.

At the time this project was undertaken, the SGPlan[120] system was the champion of the international planning competition, giving it an excellent pedigree for the first of the preconditions. The SGPLan system works by taking each of the component terms of the goals individually and using a modified version of the MetricFF[121] planning system to create a plan to solve them. It then uses custom algorithms to recombine these into a single plan, performing any additional planning which is required to resolved any conflicts, but avoiding this where possible.

Unfortunately, it is not quite as exemplary in its relation to the second of the two preconditions. Though its partitioning and recombining algorithms are very well explained, it requires an extension to MetricFF to allow temporal planning, and this element is not well documented.

MetricFF in its standard form is extremely well documented, however, as well as being a well proven and extremely fast planning system. As no temporal planning was strictly required, MetricFF itself was an ideal candidate, as a Java implementation would be easy to create given the excellent reference materials available. Additionally, this would foster a complete understanding of how the system worked. As metric capabilities were not immediately required, it was decided to use the simpler "FF" version of the algorithm[122] in place of the more complex MetricFF.

The FF planning algorithm uses a heuristically guided search methodology, based on solving a relaxed version of the problem. The problem is relaxed by "ignoring delete lists", so an action is only allowed to add terms to the world and not remove them. A relaxed solution is then found by repeatedly applying all available actions to the world state until the goal is met (or the relaxed planning fails). This relaxation provides both a heuristic value of the current world state (based on the number of actions which are required to solve the relaxed problem), and a set of actions which are considered "helpful". In this context "helpful" is defined as the most likely candidates for advancing the world state towards satisfying the goals. Enforced hill climbing is used to ensure that the algorithm always advances the problem towards the goal and does not get stuck on false maximums.

As its simplification method involves ignoring delete lists, in its default form the FF planning algorithm is unable to solve any problem in which the initial state contains a proposition which the goal requires to be false (or not present). This also makes it impossible to apply any Action which has a similar precondition during the simplified planning stage. This problem was solved using a simple modification to the system.

Instead of having a single list containing all the propositions which are known to be true, we add a second list which maintains all of the propositions which are known to be false. Thus, during simplified planning the delete list of an action is not ignored, instead its contents are added to this negative list. During simplified planning, no term is ever removed from either of these lists, and the truth conditions become as follows:

1 A proposition is true if it is found in the positive list.

2 A proposition is false if it is found in the negative list *or* it is *not* found in the

positive list.

This technique mimics the method used in the full MetricFF algorithm to perform relaxed planning with metric terms. The negative list is also used during regular planning. Instead of simply being deleted from the positive list, propositions are moved into the negative list when they become false, and then back if they are reasserted. This means that the particular objects which form the propositions persist throughout the entire planning process. This being the case, data can then be attached to each and maintained, allowing the precise dependencies between the actions which make up a plan to be found. This allows the plan to be reduced to a partial order plan, and thus certain actions to actually be performed in parallel.

Replanning occurs at the end of the update cycle if it is requested. A set of simpler algorithms are applied to ensure that the actual planner is only ever engaged when absolutely necessary, and complete replans are seldom required. The algorithm used functions as represented by the pseudocode in Algorithm 4.1.

Following on from this, the agent creates a list of the actions which are assigned to it (the convention used is the the first parameter of an action is the name of the agent it is assigned to). Then it selects the first action in this list which has no dependencies on other actions in the current plan.

This methodology ensures that actions which are no longer required can be removed from the plan without engaging the planner itself. It also allows beneficial actions from the current plan to be carried over to the new plan, reducing the load on the planner when it does need to be run.

A system such as this, which is designed to operate in environments which are particularly hostile to both the agents themselves and the communication between them, must accept the possibility that the agents may be out of communication for long periods of time. Furthermore, the possibility must be accepted that this may be because one of the other agents is no longer functioning. The communication system (detailed in the next section) is designed to guard against instances where no direct contact is possible between two agents, but further modifications have been made to the planning system in order to deal with occurrences when an agent is completely out of contact for long periods of time, or indeed permanently.

In these circumstances the local agent must assume that the non-communicating agent is also non-functioning and therefore will be unable to complete the tasks which are assigned to it. However, it must also take the possibility that it is just temporarily (though protractedly) out of communication into account. To this end, a system has been implemented which allows the planning system to “freeze” any existent in the world state, and then “thaw” it again later. When an existent is “frozen”, it is removed from the world state and moved to another storage structure within the planning system. Also removed from the world state are any propositions which refer to this existent, which are also moved to the alternative storage and

**Algorithm 4.1** The algorithm used to perform replanning in response to a significant change in the world state.

```
Require: $currentWorld$ the current state of the world
Require: $currentPlan$ the current plan
Require: $goal$ the goal of the mission

$testWorld \leftarrow copy(currentWorld)$
for each $action$ in $currentPlan$ do
    attempt to apply $action$ to $testWorld$
end for
if $testWorld$ satisfies $goal$ then
    do nothing
else
    $planUpdate \leftarrow$ an empty plan
    if PLAN($testWorld$, $goal$, $planUpdate$) then
        $currentplan \leftarrow currentPlan + planUpdate$
    else if PLAN($currentWorld$, $goal$, $planUpdate$) then
        $currentplan \leftarrow planUpdate$
    else
        return $failure$
    end if
end if
$simplifyWorld \leftarrow copy(currentWorld)$
for each $action$ in $currentPlan$ do
    $newWorld \leftarrow$the product of applying $action$ to $simplifyWorld$
    if $newWorld$ is not an improvement over $simplifyWorld$ then
        remove $action$ from $currentPlan$
    end if
    $simplifyWorld \leftarrow newWorld$
end for
return $success$

function PLAN($stateState$, $goal$, $plan$)
    call the planner with $stateState$ as the initial state and $goal$ as the goal
    if planning succeeds then
        $plan \leftarrow$the output of the planner
        return $success$
    else
        return $failure$
    end if
end function
```

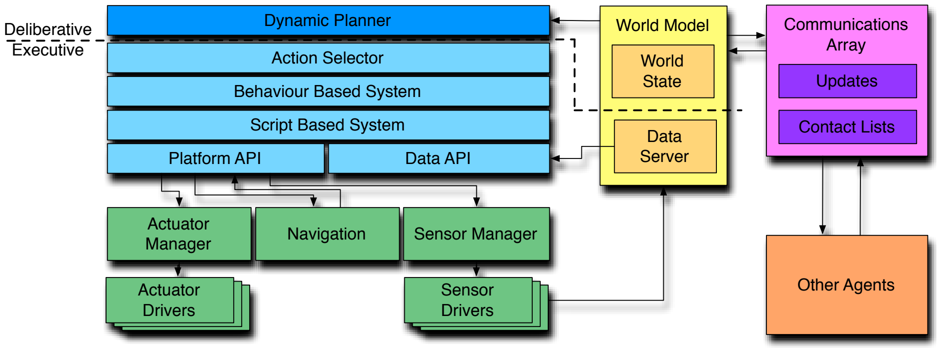


Figure 4.2: A planner based control system for multiple real or simulated AUVs

indexed against the frozen existent. When an existent is “thawed”, it it returned to the world state together with all of the propositions which are indexed against it. In either of these instances, a replan is requested.

This provides a mechanism for completely removing a malfunctioning agent from the plan, while allowing for it to be reintroduced at a later time if it regains its full capacity. As a result the completion of the plan may be slowed, but (provided that another agent with the required capabilities is available) it will not be stalled indefinitely. Either another agent will assume the duties of the malfunctioning agent and the plan will be completed, or it will fail quickly and cleanly if one is not available.

### 4.1.2 Control System Integration

Figure 4.2 shows how all of the components of the control system fit together. The structure of the system is heavily influenced by the three layer hybrid architectures described in Section 2.3, though at this point no reactive component is employed. The following subsections will describe each of the components shown in the diagram in more detail.

#### Augmented Planning Concepts

It is important that additional information should be attached to many of the PDDL constructs used in the planning system, a facility which PDDL does not allow for. To this end, an XML schema was created (see Appendix B.2), which uses XML tags to represent atomic constructs (such as predicate and action definitions), but raw PDDL (enclosed within XML tags) to represent compound clauses, such as goals and preconditions. The actual XML format is detailed in Appendix B, but the features it provides to the planning system will be discussed here. The most important function provided is the attachment of the script files (see Appendix B.2), which tell the lower level systems how to implement each action, to the action definition itself.

Secondly, this allows the consequences of each action to be separated into those which happen at the start, and those that happen at the end. Although it has been previously noted that temporal planning is not used, and therefore these are simply combined in the input to the planner, they are separated to allow the control system to more accurately reflect changes in the real world. Thus some of the consequences of the action are asserted in the control system's copy of the world state as soon as an action starts, and the remainder as soon as it finished. As an example, an autonomous vehicle is asserted to have left its current location as soon as it begins to move towards another, thus if a replan is triggered when the vehicle is halfway between two locations the world is accurately represented.

### The Action Selector

Below the dynamic planner itself in the architecture is the action selector. This selects which of the actions in the plan this agent should attempt. In the current implementation, it is very simple and merely selects the first action which is currently available (i.e. does not have any dependencies on any other actions in the plan) and is intended to be carried out by the current agent. A more complex system, such as the DELPHIS system[123, 33], might perform an analysis of the current state of the world and select an action intended to lead to the most efficient plan execution (by one metric or another). In either case, the result is essentially the same from the point of view of the next layer: an action, represented by a predicate term and a set of parameters, is passed down.

### The Behaviour Based Layer

The next level down in the architecture is a behaviour based system. This is based on instances of Java (or Java compatible) classes which must implement the methods specified in the following interface:

```java
public interface BehaviourScript {
 public void init(PlatformAPI platform, DataAPI data, String[]
     parameters);
 public String update(PlatformAPI platform, DataAPI data);
 public void clean(PlatformAPI platform, DataAPI data);
}
```

The `init` method initialises the script with the parameters which are passed to it. These parameters are the same as the parameters of the predicate term which represents the action the script is intended to carry out. The second method is the `update` method, which is run by the control system each times it cycles. This sends out the necessary instructions to the vehicle systems and has a return value which tells the control system whether the action still needs to run or has completed. In the current system this method will be called approximately five times a second. Finally, the `clean` method runs when an action has finished and releases any resources the script was using to implement the action, as well as leaving the vehicle in a stable state. Additionally, when a scripts indicates it has completed its intended action, a replan is requested.

### The Script Based System and Below

The next layer down in the architecture is the script based system. This layer provides some basic backbone, allowing scripts in a Java Virtual Machine compatible language to be executed. To facilitate this, it also provides access to the two API[1]s which are supplied to the methods specified by the behaviour based system. These APIs are detailed in the following sections.

### The Platform API

The first of the APIs mentioned above is the platform API. This gives the script the control it needs over the vehicle systems, such as requesting movement to specific locations and changing the mode of sensors and actuators. The platform API also provides feedback from the vehicle systems, such as the vehicle's current location and heading (in world frame) and the current mode of the sensors and actuators. All data sent to and received from sensors and actuators is routed through the sensor and actuator managers. These are simply small databases which index each of the sensors and actuators by name, allowing information to be passed back and forth as quickly as possible.

Vehicle navigation is controlled via waypoint requests. A waypoint represents a position in space, defined relative to either the vehicle or a fixed origin, a set of tolerances which decide when the vehicle has achieved the waypoint, a set of enables which define which axes the controller should take notice of, and also (depending on the navigation mode) a target attitude and a travel speed.

Control of sensors and actuators is achieved through very simple mode change requests. These modes are simply defined as text strings. The present implementation requires a driver for each sensor or actuator to be used in a real vehicle system to be implemented in Java. For simulated trials the sensors and actuators are specified via XML, and it is intended that this system should also be extended to allow the control of real sensors and actuators to be defined similarly. Methods are provided by the Platform API to allow the system to request a mode change and also to query the current mode of a sensor or actuator.

In simulation, the components of the architecture can be directly connected and information passed via simple method calls. On a real vehicle this is often not possible or even desirable, as different components below the level of this architecture

[1] Application Programming Interface

(such as navigation or SLAM[2] systems, and low level drivers) may be implemented in different languages, or distributed across multiple physical systems. For this reason, the system uses a UDP based communication system (called OceanSHELL) to communicate with everything below the level of the Platform API.

### The Data API and World Model

The second of the APIs is the Data API. This provides the script with access to the Data Server, allowing it to obtain the locations, areas, scalar values and times to which the existents in the world state relate. This API also allows the script to add additional key/value pairs to the Data Server, allowing some persistence of data between the different scripts.

Input from the vehicle sensors is assumed to have been fully processed before it reaches the control system. The receipt of this data will also most likely change the world state, either by adding new instances or predicates, or both of these. It may also add or update elements in the Data Server. As an example, the detection of a possible target might result in a `target` instance being added to the world, as well as a new `location` and an `at` predicate indicating that the target is at the location. These additions suffice for the planner to be able to adapt the mission accordingly. A new co-ordinate might also be added to Data Server, indexed to the name of the new location in order to provide a binding to the real world. The system ensures that duplicate objects are never added to the world state, but the locations in the data server are constantly updated. The second action that can occur after receiving new sensor data is for a new proposition to be added to the world state. If an entirely new existence or proposition is added to the world then a replan is requested.

### Communication

The communications array firstly keeps track of all changes which are made to the planning system's view of the world, and to a more limited extent the changes which are made to the Data Server. To do this it maintains three things specifically:

1 A list of updates to existents.

2 A list of updates to statements.

3 The (mission) time at which the last replan occurred.

Furthermore, it also maintains the last time any of these were updated and the last time it successfully communicated with the other agents in the system. These are maintained so that the communications system can judge whether any communication is required at a particular time. Also maintained are two "contact lists". The

[2] Simultaneous Localisation And Mapping

first of these, or “direct contact list”, maintains a list of the last time this agent directly communicated with the other agents. The second, or “proxy contact list” holds the times at which each of the other agents is last known to have communicated. The mechanism and use of these will be explained later.

Whenever the planning system indicates that a new existent is added to the world a new entry is added to the top of the (initially empty) list of updates to existents. Each of these entries contains a reference to the existent itself, a reference to a location which may optionally be attached to the existent, mirroring a subset of the functionality of the Data Server, the time at which the update occurred and the time at which the update was received. In the case of updates generated locally these last two will be identical, but this will not be the case if the update was received from another agent.

If an update to the location of an existent is received, a similar procedure is followed. If there is currently no reference to the existence in the list then one is created, otherwise the existing reference is removed from the current list. The location and time information is updated and then the entry is placed at the top of the list.

Again, a similar procedure is followed when a statement is updated. An entry is created if required, otherwise the existing entry is removed from the list. The entry for a statement contains the times at which it was received and updated, together with a reference to the statement itself. This reference contains the relevant predicate proposition, together with a Boolean status value declaring whether the statement is currently thought to be true or false.

### Example Behaviour Class

The class used for the implementation of each action is specified in its XML definition (see Appendix B.2). In the case of the examples shown here, the classes are described in a language called groovy[124], a dynamic scripting language which can be compiled to pure Java at run time. The back-end needed to ensure that this functionality is available at run time is provided by the scripting layer. The groovy class definition for the example used here is shown below:

```
class MoveControlScript implements BehaviourScript {
 LocalCoordinate3D waypoint
 int number
 void init(PlatformAPI platform, DataAPI data, String[] parameters) {
  waypoint = data.getVector(parameters[1])
  number = platform.getCurrentWaypointNo()++
  platform.setTolerences(1, 1, 1, 10)
  platform.setEnables(1, 1, 1, 1)
  platform.absoluteWayPointRequest(number, TRACK_MODE, waypoint.getNorth(), waypoint.getEast(), waypoint.getDepth(), 0, 1f)
 }

 String update(PlatformAPI platform, DataAPI data) {
  if (platform.inPosition()) return "succeed"
  platform.absoluteWayPointRequest(number, TRACK_MODE, waypoint.getNorth(), waypoint.getEast(), waypoint.getDepth(), 0, 1f)
  return "continue"
 }

 void clean(PlatformAPI platform, DataAPI data) {
  platform.stay()
 }
}
```

In this example, the class contains two member fields, one for the vehicle's destination and one for the index number of this waypoint. These are both initialised at the beginning of the `init` method, the first by obtaining from the Data Server the co-ordinate which is referred to by the second parameter[3] of the action predicate, and the second by obtaining the current waypoint number and incrementing it. Next, the tolerances and enables of the waypoint are set, indicating that the vehicle must be within one metre of the destination for the waypoint to be met and that all axes are to be used. Finally, the waypoint request itself is sent to the navigation system.

The `update` method first checks whether the waypoint has been achieved. If it has then ‘‘`succeed`’’ will be returned and the action will complete. Otherwise, the waypoint request will be repeated and ‘‘`continue`’’ will be returned.

When the `clean` method runs (on completion of the action) the vehicle is instructed to hold its current position.

### 4.1.3 Testing and Results

The system described here has been tested in two separate sets of circumstances. Firstly, the lower levels have been tested on a single real vehicle, with a simpler finite state machine based system taking the place of the replanning system. Secondly, the complete system has been tested in simulation.

The lower level systems were used as the basis of the control system used for the Nessie III AUV (see [25] and Figure 4.3) as part of The Ocean System Laboratory's entrance into the Student Autonomous Underwater Competition - Europe (or SAUC-E) 2008. The dynamic planning and action selection systems were replaced with a much simpler finite state machine based system, as illustrated in Figure 4.4. Otherwise, the system was unchanged and employed the same parameterised script

[3] Parameters are numbered from zero.

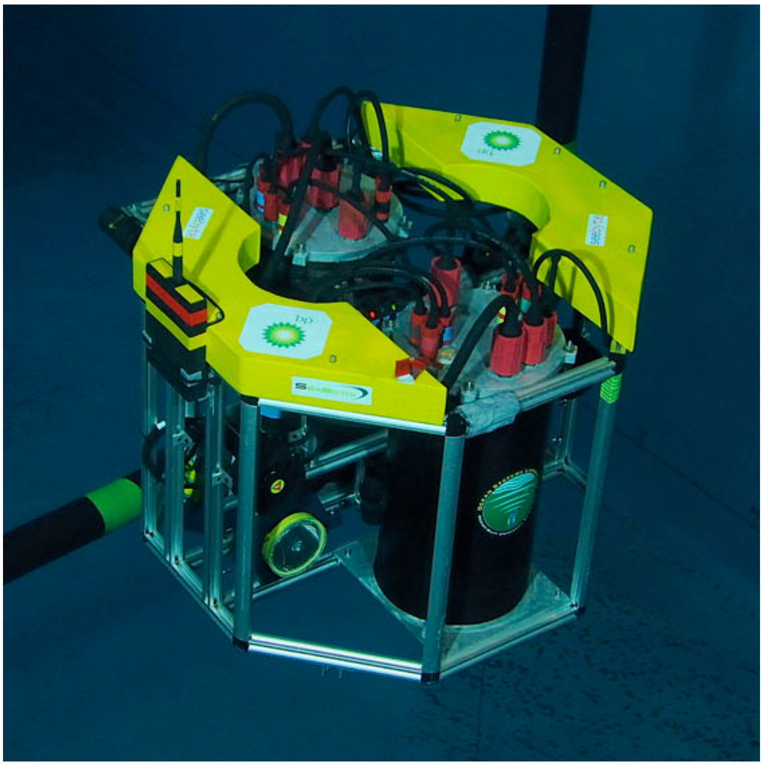


Figure 4.3: The Nessie III AUV passes through the validation gate at SAUC-E 2008. Image courtesy of Yves Gladu.

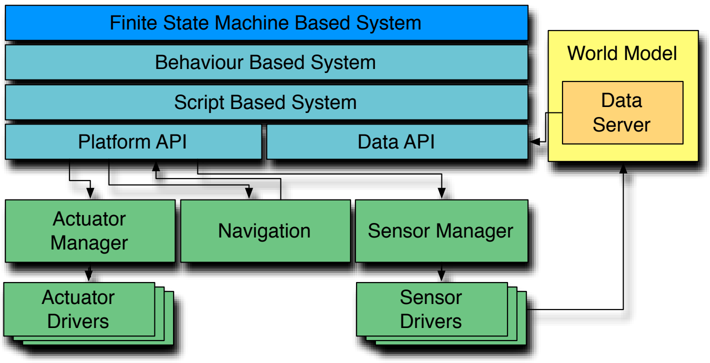


Figure 4.4: The simplified control system deployed on the Nessie III AUV.

based mechanism for control of the vehicle systems. This more simplified system was used as the task required of the vehicle in the competition was based upon an entirely static environment, and therefore a replanner was not required and would have led to unwarranted additional overhead. The employed system proved to be extremely robust and flexible, helping “Team Nessie” to take the first prize in the competition[23], as well as “The THALES Special Award for innovation in decision making autonomy”. Furthermore, the Nessie AUV completed all of the tasks laid, making it the first entry in the competition’s history to do so. The same system, virtually unchanged, was again fielded as part of Nessie IV[26], the Ocean Systems Laboratory’s subsequent entry to the 2009 competition. Nessie again took first prize in the competition[24], making it the first vehicle to successfully defend the title. Further information about SAUC-E can be found at the websites for the 2008[17] and 2009[18] competitions.

This system has also been given extensive testing in simulation, with the full dynamic planning system acting as its upper layer. As the planner which is currently employed is relatively simple, it was sometimes found to be necessary to make small alterations to the domain in order to ensure that an efficient plan was created. Other than this, the system was found to perform extremely well, with the control system integration functioning exactly as intended. In this configuration the system has be tested under various scenarios, some of which are explained in more detail in Section 8.

### 4.1.4 Future Work

Although the previously described system has led to some very positive results, there are various modifications which can be made in order increase efficiency (both in terms of the plans created and the resources required to do so) and power of the system. A selection of these improvements will now be described.

Currently, the planning system has no notion of the differing cost of each action, and thus makes no attempt to make the most efficient ordering or selection of actions. One example consequence of this is that the system is not aware that movement while carrying a large object has a higher cost than moving without such a burden. As a consequence of this, the action definitions in one scenario had to be specifically changed to prevent one of the agents carrying each of a set of large objects back to the base location for another to examine. Currently, as a by-product of the information storage used within the planning system, potential actions are considered in alphabetical order. Changing this to a system which considered them in reverse order of cost would most likely lead to a system which both produced more efficient plans, and also did not require such precise manipulation of the actions’ descriptions. This would of course require a reasonably accurate method for estimating the cost of actions, as well as a way of representing this within the action definitions.

Currently, the planning algorithm only makes use of the propositional capabilities of the MetricFF planning system, and does not support temporal planning. This is because the "fingerprints" algorithm used to deconstruct the produced total-order plans into more useful partial order plans becomes much more complicated given metric world states, and further, non trivial, modifications to the MetricFF algorithm are required to support temporal planning. So far, these modifications have not been carried out due to time constraints. The implementation of this functionality would lead to a large increase in the power and accuracy of the planner, allowing the system to attempt much more complex scenarios.

Currently, the communications system of each agent transmits the state of every proposition (and data association) which has changed since the start of the mission. Consequently, the amount of information which is sent will increase over the length of the mission. In the case of a simulated mission this is not a problem. In the case of a real mission in the underwater domain, however, this becomes a serious concern due to the limited bandwidth of acoustic communication systems. This problem can be alleviated to some extent if each agent keeps a record of the information which has been successfully communicated to other agents, and then only transmits that which has not.

The two major factors which lead to an increase in time taken for planning are the number of potential actions and the number of propositions present in the world. These both tend towards an increase over time, the first as more existents (and therefore potential parameters) are added to the world by sensor detections, the second also due to sensor detections, but also the effects of previous actions. This being the case, time taken for replanning the mission also increases over time. Metrics could be used which analyse the importance of each existent and proposition within the current mission and allow those with very low importance (i.e. those which do not contribute to the mission) to be "frozen", so they are no longer considered by the planner, but can be reactivated if their importance increases.

## 4.2 High Speed Mission Simulation

This section describes the simulation system which has been implemented in order to fulfil the requirement for one in the framework described in Chapter 3 (see Figure 4.5). This system allows for fast simulation of user specified missions, without the need to deploy real vehicles. This section is divided into six subsections. The first of these describes the methodology used for the faster than real time simulation. The second explains how the intra-vehicle communication is simulated. The third section explains how the simulated sensors and actuators function and integrate with the scene graph of the graphics system. The fourth section describes a system which

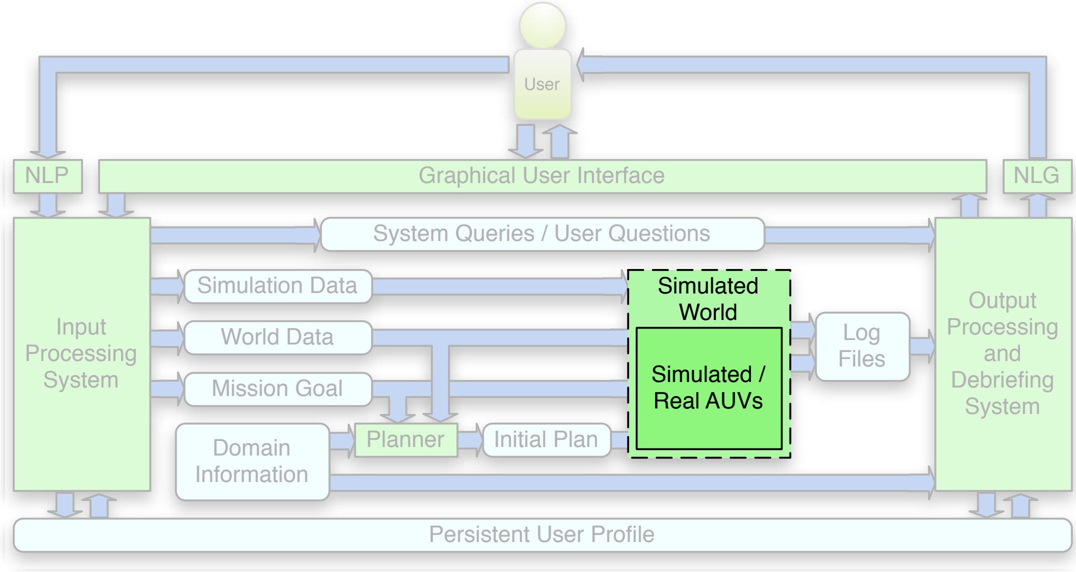


Figure 4.5: The components described by this section and how they fit into the framework described in Chapter 3.

has been implemented in order to provide a mechanism for the simulation of system failures. The fifth section describes how all these systems come together in order to run an actual simulation. Finally, section six details some improvements which could be implemented in future versions of this system.

### 4.2.1 Accurate Faster than Real Time Simulation

There are various approaches which can be taken to accomplish faster than real time simulation. Here, we will focus on three. Specifically, those which are most appropriate to the distributed architecture used by the systems employed in the Ocean Systems Laboratory. This systems are made up of discrete software modules which communicate using a UDP packet based mechanism (OceanSHELL). Figure 4.6 shows the three approaches which are considered, each of which will now be explained in more detail.

The first method exactly mimics the process employed on the real vehicles, with each module being deployed as a separate process, and communication being accomplished via OceanSHELL. The period of each module is then decreased in order to accomplish the faster than real time goal. Although this works quite well at slower speeds (up to ten times real time), the fixed costs required for the maintenance of each process can lead to a high resource cost when multiple vehicle simulations are run. More troublesome is the detrimental effect on synchronisation. As the speed of the simulation is increased, so the relative accuracy of the period each module runs at decreases. As no active synchronisation occurs, this can lead to less defined behaviours of the modules, which is not representative of their behaviour when run-

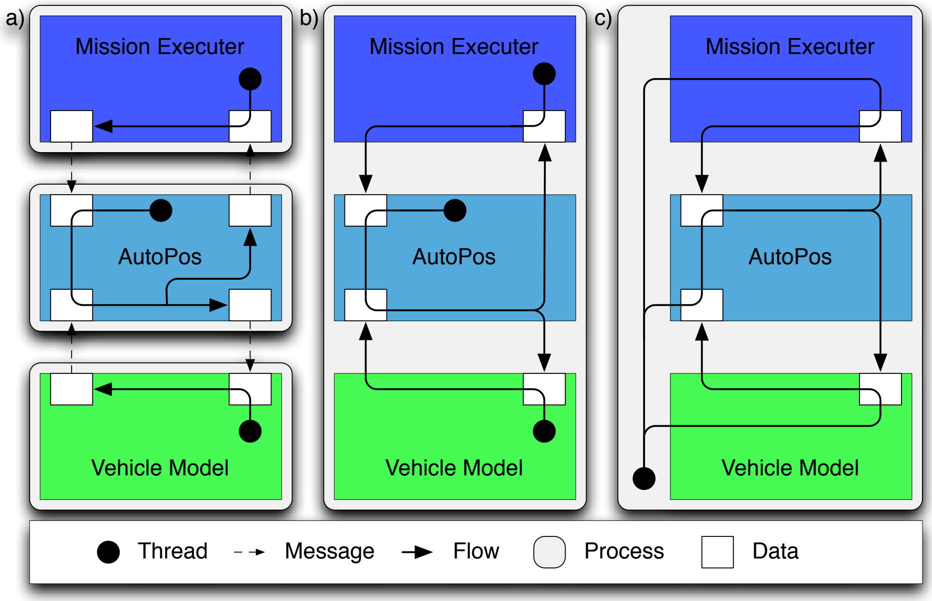


Figure 4.6: Comparison of simulation approaches.

ning in real time. Navigation and control, in particular, can visibly be seen to suffer because of this.

The second method eliminates the cost of additional processes by running each module as a thread in a single process. OceanSHELL is removed in favour of direct method calls for communication, which removes the cost associated with creating and distributing UDP packets. However, the synchronisation problems noted above are still found.

The third method uses a single thread to call each module in turn, ensuring perfect synchronisation, and additionally removing the cost associated with employing multiple threads. This method is therefore able to produce identical results, regardless of what speed the simulation is running at.

The synchronisation problems found in these first two methods make them unsuitable for simulation at very high speeds, whereas the third method has been shown to create accurate simulation with speeds reaching one hundred times real time (on the hardware used for testing; faster speeds would be possible with faster hardware). Additionally, the third method's fixed costs required for the simulation of additional vehicles grows more slowly, as no additional processes or threads are required. For these reasons, this third method is used for all simulation in this project.

### 4.2.2 Simulated Communication

The present simulated communication system employs a "token ring" style approach. The token is passed using the alphanumeric ordering of the ids of the agents. Each time the control system cycles the agent finds if it is currently its "turn" to communicate. If it is not then execution will continue, but if so then the communication system will make a decision as to whether to attempt to communicate or simply pass on the token, depending on whether any updates have been made since the last time this agent successfully communicated.

At this point it should be noted that the current system has only been deployed, tested and verified during simulated missions, and so is in many ways quite inefficient and un-robust.

If communication occurs then it does so in a broadcast fashion, using the following assumptions:

1 Communication either succeeds or fails.

2 If it succeeds then it does so completely, otherwise it has failed.

3 Success or failure of communication is considered independently for each receiving agent.

The agent broadcasts the list of changes to existents (planner objects), the list of changes to statements (planner predicates) and the last time it replanned, together with its proxy contact list (see Section 4.1.2). Each existent and statement maintains a record of its **update time** (the time at which the data was directly detected) and its **receive time** (the time at which this agent was informed of the data, either via direct observation or communication). The first is used to establish which of two conflicting observations is most recent, and therefore most likely to the true. The second is used to help remote agents calculate whether world model updates have prompted a replan in the local agent.

At present this broadcast is implemented through static method calls within the `SimulatedCommunicationsArray` class. A mechanism exists to apply a test to see if communication succeeds or fails (based on distance, occlusion or simply random chance), but currently this is not fully realised and as such communication always succeeds.

On receipt of this information, the receiving agent first updates its contact lists via Algorithm 4.2. The next step updates the local agent's record of all of the existents in the world. As part of the world model, each existent potentially has a location attached to it, along with its received and updated times. All of this data is additionally received from the remote agent, and then Algorithm 4.3 is used to perform the update. This algorithm also calculates whether a replan is warranted, based on the timings between the updates to the existents and the agent's last replan. If this indicates that the remote agent performed a replan after a new existent was added to its world model, then one will also be triggered locally. A similar procedure is followed for the received statement updates. Algorithm 4.4 is used to perform this

**Algorithm 4.2** The algorithm used by each agent to update their contact lists on receipt of a communication.

**Require:** *agent* a reference to the agent the communication was received from
**Require:** *directContact* a map containing this agent's direct contact list (*agent* → *time*)
**Require:** *proxyContact* a map containing this agent's proxy contact list (*agent* → *time*)
**Require:** *receivedContact* a map containing the received contact list (*agent* → *time*)

```
$time \leftarrow$ the current time
$directContact_{agent} \leftarrow time$
$proxyContact_{agent} \leftarrow time$
for each $remoteAgent$ in $receivedContact$ do
    if $remoteContant_{remoteAgent} > proxyContact_{remoteAgent}$ then
        $proxyContact_{remoteAgent} \leftarrow remoteContact_{remoteAgent}$
    end if
end for
```

update. This algorithm is slightly more complicated, as each statement also has a status attached to it which indicates whether the system considers it to be *true* or *false*. As with the previous algorithm, this has the potential to indicate that a local replan is required if received data indicates that one is required. Finally, the local system sends an acknowledgement back to the sending system, prompting it to update its own direct contact list (see Section 4.1.2).

Once the agent has finished communicating, it updates the status of the agents it currently considers to be frozen. For each of the currently frozen agents in the mission, if the entry for it in the proxy contact list is less than the contact time limit ago then it is unfrozen. Following on from this, for each of the currently unfrozen agents in the mission, if the entry for it in the proxy contact list is more than the contact limit ago then it is frozen. See Section 4.1.1 for an explanation of the concept of "freezing" in this context

### 4.2.3 Scene-graph Integration and Fast Sensor and Actuator Simulation

The primary requirement for the system discussed here is the simulation of high level control, and not sensor processing. Furthermore, it is highly desirable that this simulation should be completed as quickly as possible. To this end, a system has been implemented which provides a reasonably realistic facsimile of the detections produced by various sensors and the accompanying processing systems, while requiring a bare minimum of system resources (most notably processor time) in order to do so.

**Algorithm 4.3** The algorithm used by each agent to update the *existent* component of their world model on receipt of a communication.

**Require:** *localUpdate* a map containing this agent's existents and their update times ($existent \rightarrow time$)
**Require:** *localReceive* a map containing this agent's existents and their received times ($existent \rightarrow time$)
**Require:** *localLocations* a map containing this agent's existents and their real world locations ($existent \rightarrow location$)
**Require:** *remoteUpdate* a map containing the received existents and their update times ($existent \rightarrow time$)
**Require:** *remoteReceive* a map containing the received existents and their received times ($existent \rightarrow time$)
**Require:** *remoteLocations* a map containing the received existents and their real world locations ($existent \rightarrow location$)
**Require:** *remoteReplan* the time at which the remote agent last replanned
**Require:** *last* the time at which communication was last received from the remote agent

```
$replan \leftarrow false$
$time \leftarrow$ the current time
for each $existent : e$ in $remoteUpdate$ do
    if $e \notin localUpdate \vee remoteUpdate_e > localUpdate_e$ then
        if $e \notin localUpdate$ then
            $replan \leftarrow replan \vee (last < remoteReceive_e \leq remoteReplan)$
            add $e$ to the planner's world state
        end if
        $localUpdate_e \leftarrow remoteUpdate_e$
        $localReceive_e \leftarrow time$
        if $e \in remoteLocations$ then
            $localLocations_e \leftarrow remoteLocations_e$
        end if
    end if
end for
return $replan$
```

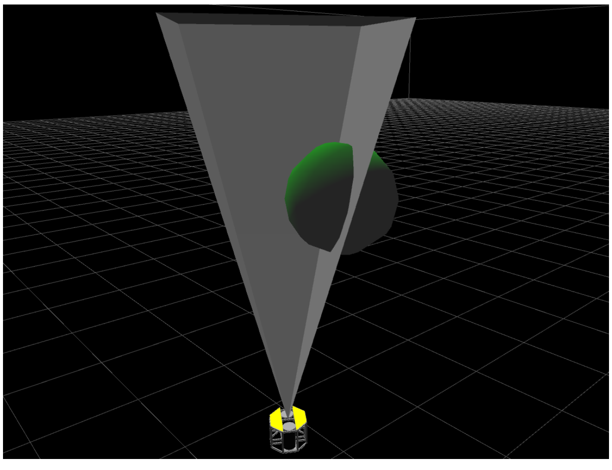


Figure 4.7: Example sensor geometry.

**Algorithm 4.4** The algorithm used by each agent to update the *statement* component of their world model on receipt of a communication.

**Require:** *localUpdate* a map containing this agent's statements and their update times ($existent \rightarrow time$)
**Require:** *localReceive* a map containing this agent's statements and their received times ($existent \rightarrow time$)
**Require:** *localStatus* a map containing this agent's statements and their status ($existent \rightarrow status$)
**Require:** *remoteUpdate* a map containing the received statements and their update times ($existent \rightarrow time$)
**Require:** *remoteReceive* a map containing the received statements and their received times ($existent \rightarrow time$)
**Require:** *remoteStatus* a map containing the received statements and their status ($existent \rightarrow status$)
**Require:** *remoteReplan* the time at which the remote agent last replanned
**Require:** *last* the time at which communication was last received from the remote agent

```
$replan \leftarrow false$
$time \leftarrow$ the current time
for each $statement : s$ in $remoteUpdate$ do
    if $s \in localUpdate$ then
        if $remoteUpdate_s > localUpdate_s$ then
            $localUpdate_s \leftarrow remoteUpdate_s$
            if $localStatus_s \neq remoteStatus_s$ then
                $replan \leftarrow replan \vee (last < remoteReceive_s \leq remoteReplan)$
                update the planners' world state for $s$
                $localReceive_s \leftarrow time$
                $localStatus_s \leftarrow remoteStatus_s$
            end if
        end if
    else
        $replan \leftarrow replan \vee (last < remoteReceive_s \leq remoteReplan)$
        update the planner's world state for $s$
        $localUpdate_s \leftarrow remoteUpdate_s$
        $localReceive_s \leftarrow time$
        $localStatus_s \leftarrow remoteStatus_s$
    end if
end for
return $replan$
```

The implemented system is designed to be as simple as possible. First of all, the system allows each object in the world to be labelled with zero or more existents, and zero or more propositions. Next, sensors carried by the various vehicles are defined as detecting zero or more types (relating to existences) and zero or more predicates (relating to propositions). Each of these has geometry in the simulation, a sphere in the case of objects and a sector in the case of sensors. The graphics engine detects any intersection of these geometries and a simple test is carried out. If any of the labels attached to the object geometry matches any label types the sensor is programmed to detect, the relevant existence or proposition is passed along to the control system of the agent carrying the sensor, along with additional location related information in the case of existences. An example of simulated sensor geometry is shown in Figure 4.7.

Additionally, a "reliability" value can be given to each instance of a sensor, leading to a random variable based probability measure which decides whether an individual detection should be passed on to the control system. This adds an extra degree of realism, allowing less reliable sensors to be simulated with a higher degree of accuracy. Also added to the simulated sensor definition is a "latency" parameter. This describes the amount of time (in milliseconds) the sensor must scan an object for before it detects any of the existences or propositions it contains. This reflects the fact that a sensor and its accompanying software might take some time to process the received data.

This methodology leads to a sensor simulation system which is extremely fast and reasonable realistic. As the intention of the simulation system as a whole is the testing and verification of high level control systems, the assumptions that all sensors have a quantifiable level of accuracy and their processing software has the ability to track individual detection is deemed to be a valid one.

As previously mentioned, world objects are represented as spheres in the 3D world. They also have named states, which are reflected by the colour of the sphere. A world object's initial state in set in the simulation definition input file, and then whenever it interacts with either a sensor or actuator its state is changed to the name of the sensor or actuator. These states are again defined in the simulation definition. As an example (with extraneous details removed):

```
<item name="o1" default="default">
 <state name="default" colour="black"/>
 <state name="sidescan" colour="grey"/>
 <state name="camera" colour="brown"/>
</item>
```

The object starts the simulation in the "default" state and is coloured black. After it is scanned by the "sidescan" sensor it switches to this mode and becomes grey, and then after it is examined by the "camera" it becomes brown.

Currently three types of simulated actuators have been implemented. The first

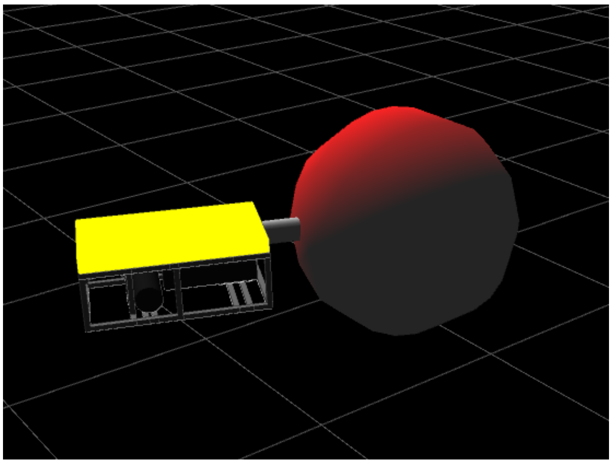


Figure 4.8: Example actuator geometry.

of these is the “destroy” actuator, which essentially simulates a bomb carried by the vehicle. This is parameterised by range and target type. The bomb is always simulated as a sphere centred on the vehicle, with the range being the radius of this sphere, which represents the bomb’s area of effectiveness. Within this area of effectiveness any world object which contains a label of the same type of the bomb’s target will have its status changed to “destroyed”. Additionally, and somewhat obviously, any vehicle which deploys one of these actuators will itself be destroyed (its propulsion and communication systems will be disabled).

The second type of implemented actuator is the “grab” actuator, which is intended to simulate a simplified manipulator or robotic arm. This is again defined as having a range and a target type. The actuator is represented as a cylinder projecting forwards from the front of the vehicle to the same distance as the defined range. When the actuator is activated the cylinder appears and any world object which intersects it and is labelled with the target type becomes attached to the end of it, until such a time as the actuator is deactivated, at which time the cylinder also disappears. An example of this kind of geometry is shown in figure 4.8, which shows the AUV using the actuator to lift an object.

The third implemented actuator is also the simplest. Named the “visible” actuator, Its geometry is identical to the “grab” actuator, but it has no effect on the simulated world. It exists simply to illustrate the fact that some action is being carried out, and is intended to act as a representation of close up intervention work (such as a repair).

### 4.2.4 Simulated System Failures

This section describes a method for the simulation of system failures. “System” in this case referring to the simulated physical systems. This allows to user to test how the software will react to the failure of, for example, the propulsion system, the

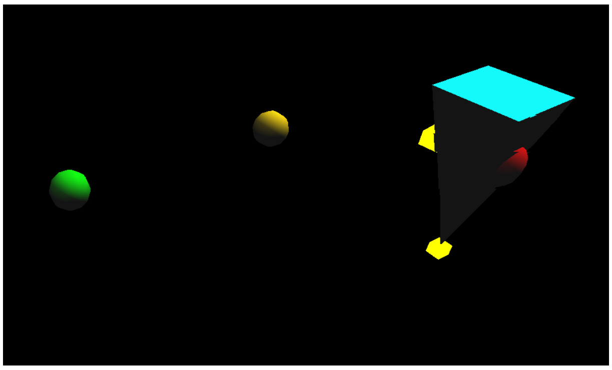


Figure 4.9: Example simulation geometry.

communication system or a sensor.

When each simulated vehicle is initialised, each simulated system which represents a possible point of failure registers itself (by name) with a “failure server”. This same system also stores the desired simulated failures as defined in the simulation input file (see Appendix B.4). These failures can be triggered in one of three ways. The first is simply at a specified time in the mission playback. Secondly, they might supply a probability that the failure might occur. Lastly, they can be triggered by the vehicle’s simulated battery falling below a particular level.

As well as the name of the failing system and a timing value, each failure definition also supplies a failure mode as a string. These strings are passed directly to the system in question when a failure is triggered and interpreted there. Currently these systems support boolean (“true” meaning a system is working, “false” meaning it is not) and scalar (a number between “0” and “1’ representing the reliability of the system) failure modes.

When the triggering of a failure is based upon a probability a random variable is used to implement this. A unique variable is used for each system, and each of these is initialised using a seed supplied in the simulation definition. This ensures that given the same seed, a particular sensor will always receive the same sequence of random values and therefore behave in the same way and will not be directly affected by the changes made to failure definitions for other systems (as would result from using shared random variables).

### 4.2.5 Simulated Missions

The section describes how all the previously described systems come together in order to perform an actual mission simulation.

The XML input files which define the mission and the simulation (described in Sections B.3 and B.4, respectively) are used to create two sets of definitions, the first for the vehicles and the second for the world objects. These are then used to initialise all of the objects which are required for the simulation.

For the purposes of simulation all geometry is made as simple as possible (see Figure 4.9 for an example). The number of polygons used in the spheres which represent world objects is reduced to a bare minimum and vehicle models are replaced with a simple cuboid derived from their bounding box. None of the geometry the user interface system uses to aid the user's perception of distance is used. This reduces the load on the graphics engine as much as possible, allowing the simulation to proceed at the fastest possible rate.

As previously stated, the third type of faster than real-time simulation discussed in Section 4.2.1 is utilised, again allowing the fastest possible simulation speed. A Java port of the the OSL "AutoPos" system (which is functionally identical to the original C++ version) is used for navigational control, with an accurate hydrodynamic model used to simulate the physics of the vehicle itself. High level control is achieved using the planner based control system discussed in Chapter 4.1.

An instance of the simulated communications array is given to the high level control system, presenting the exact same interface as the real module would. Based on the definition provided for the vehicle, instances of all the required simulated sensors and actuators are passed to the sensor and actuator managers (see Section 4.1.2), allowing them to be controlled in the exact same way the real sensors would be.

For each of the world object definitions, a relevant object is created, given the correct starting position and state, the conditions required for changes in states, and the correct geometry labels required for the fast sensor simulation system. The geometry for each of these is then added to the scene graph, as is required both for visualisation and geometry intersection testing.

As the intention of this system is for simulation to proceed as quickly as possible, the main loop is simply run over and over, without pauses, until the simulation completes. In each run of the main loop, all of the internal timers are incremented[4], next all the relevant processing required for the simulated sensors and actuators is carried out, then updates are made to the world objects and finally the vehicles themselves are updated. After this, the entire scene is rendered by the machine's graphics hardware.

When run, a complete mission simulation will produce a set of the logs described in Appendix C for each of the vehicles in the simulation, as well as a single log for all of the world objects.

### 4.2.6 Future Work

In the current single threaded version of the system, simulation simply pauses whenever replanning occurs. In the missions tested so far, replanning is relatively quick, taking no more than three seconds for each vehicle, so the detrimental ef-

[4] All of the systems used run at 5hz, so in each iteration the timers are increased by 200ms

fect on the viability of the simulation is minimal. With more complex missions this could become a problem, however. The proposed solution to this is that a new multi-threaded approach should be used, with the replanning run in a separate thread and the simulation running in real time during replanning. Of course multiple agents needing to replan simultaneously will result in replanning taking longer. This being the case, the actual speed of the simulation when any replanning is occurring should be one divided by the number of agents which are currently replanning. Thus for one agent the speed would be real-time, for two it would be one half real-time, for three one third real-time and so on.

Currently, the reliability of sensors can be adjusted by using a random variable to decide whether detection occurs, but some further control over this could lead to more accurate sensor simulation with minimal additional overhead. One possible implementation of this might vary the probability of a detection based on the angle of incidence of the sensor.

Currently a failure simply causes the affected system to cease functioning (or function at a decreased capacity). The inclusion of a simulated diagnostic system which was able to integrate with the planning system would allow for the latter to take these failures into account, alter the mission accordingly and pass this data on to the other agents. This would act as a simulated counterpart to systems such as RECOVERY[125], which act in a similar way for real vehicles.

Currently the simulated communications system always succeeds. A more realistic implementation might take into account the distance between the vehicles, as well as any objects which happen to be between them, and degrade the performance of the communication accordingly. This could be implemented in a similar way to the reliability measure used for the simulated sensors, or a more complex method, such as increasing the time taken for the communication.

## 4.3 A Tailored Graphical User Interface for Rich Communication

This section describes the graphical user interface which has been implemented in order to facilitate the playback of missions and provide effective audio visual debriefs. It implements the requirement for such a graphical user interface in the framework described in Chapter 3 (see Figure 4.10). The basic system provides a high fidelity graphics system to show the movements and actions of the vehicles and world objects during the mission, together with a highly flexible Heads Up Display (HUD) system. At present, the system operates in one of two ways. In the first the playback is entirely controlled by the user, whereas in the second the playback is entirely controlled by the system. The remainder of this section will consider all of

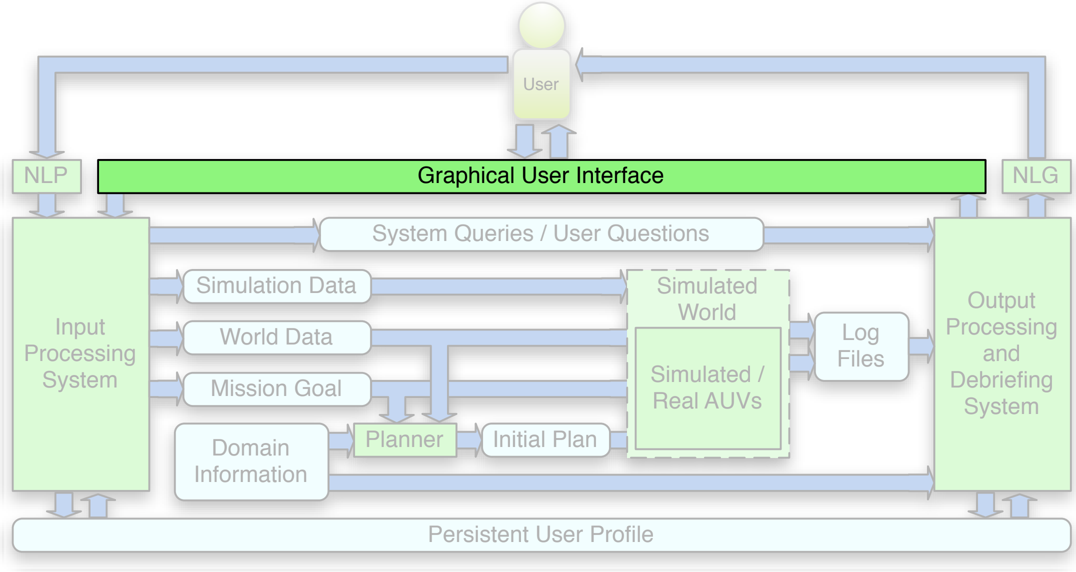


Figure 4.10: The components described by this section and how they fit into the framework described in Chapter 3.

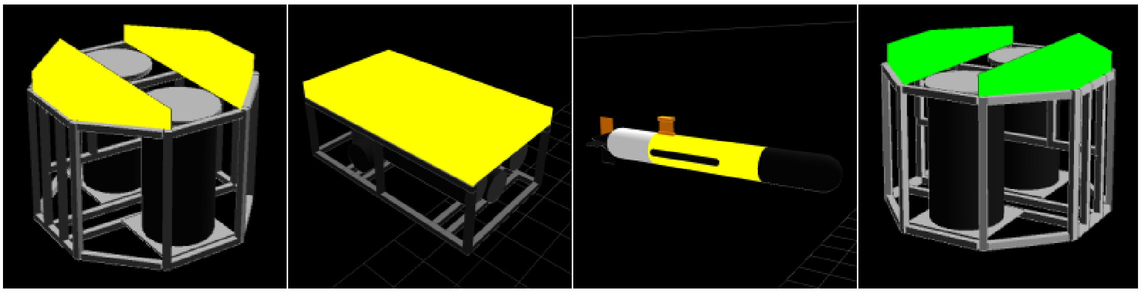


Figure 4.11: Example vehicle models. From left: Nessie with yellow colouring, RAUVER with yellow colouring, REMUS with yellow colouring, and Nessie with Green colouring.

this in more detail.

### 4.3.1 Vehicle and World Object Depiction

Unlike the vastly simplified vehicle representations used by the simulation system, the playback system attempts to leverage the graphical power made available by the JME graphics engine[126]. Vehicles are shown as highly detailed models, with an added white outline to ensure that they stand out from the background. The particular vehicle model is specified in the input file, with the currently implemented options being the Nessie[25], RAUVER and REMUS[11] AUVs. Examples of these models can be found in Figure 4.11.

The position and orientation of the vehicles are read in from the log files created during the mission (be it real or simulated) which is being played back. For each vehicle defined in the input files, the system searches for an appropriate log file.

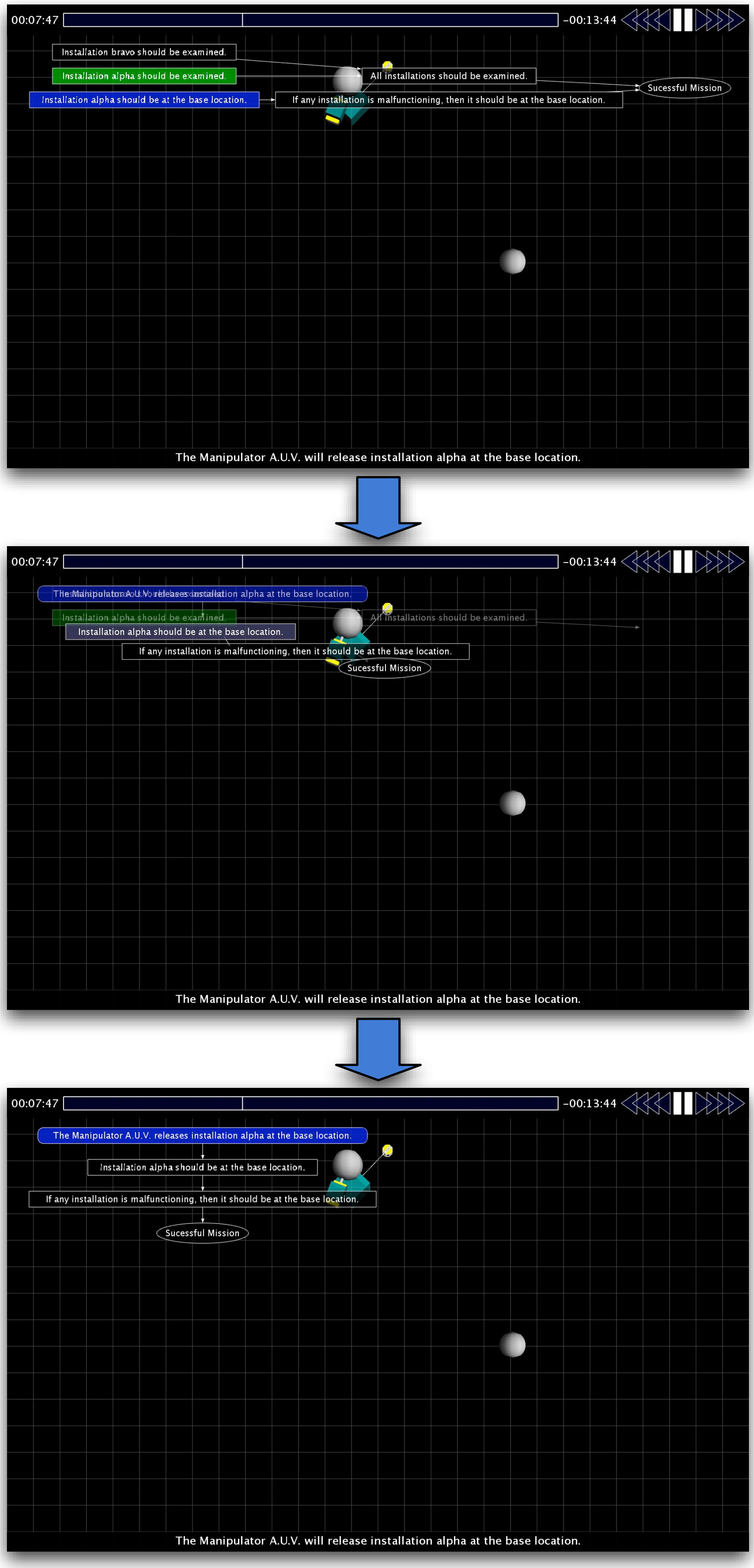
The various positions and orientations of each vehicle are then indexed by time, allowing the displayed position and orientation of the vehicle to match the same during the playback. A description of the input files can be found in Appendix B, while Appendix C describes the output log files.

Additional geometry is used to display the sensors and actuators of each vehicle, along with a representation of the vehicle's current waypoint. The status of each of these is read in from the platform log file, in a similar manner to the position log file. These changes in state are then reflected as the replay of the mission commences. Actuators are displayed using appropriate (if simplified) geometry, such as a silver cylinder projected forwards from the vehicle representing a manipulator arm. Sensors are displayed as a transparent sector, which represents their field of view. Finally, the vehicles' current waypoints are displayed as transparent blue cubes, whose centre represents the requested position, and whose dimensions represent the tolerance supplied as part of the waypoint request. Furthermore, each vehicle is linked to its current waypoint by a thin white line, aiding the identification of the particular waypoint a vehicle is trying to achieve.

Likewise, it is also possible for the world objects to change position and state during a mission. These are of course only recorded during a simulated mission, when each world object's accurate position and state are known. These are recorded in much the same ways as those of the vehicles, with the system again finding and using the appropriate logs for a particular named world object. Each world object is currently displayed as a sphere, which changes in state (such as "detected", or "destroyed") simply being reflected by changes in colour. As noted in Section 4.2.3, these state changes occur whenever a world object interacts with either a sensor or actuator, and so the changes in colour reflect the amount of information the system (and user) has about the particular object.

### 4.3.2 Text and Graphics for Heads Up Display

The JME graphics engine which creates the 3D environment used for the display of vehicles and objects during mission playback is a scene graph based system which sits on top of the Light Weight Java Game Library[127] (or LWJGL), a set of Java bindings to native OpenGL[128], which is used to control the graphics hardware. This being the case, it is also possible to sit other systems on top of these same bindings to display additional graphics. Using this method, the FengGUI[129] system is used to create 2D graphics in the manner of a Heads Up Display (or HUD). FengGUI provides an abstraction of the OpenGL commands, allowing lines, polygons and text to be drawn on the screen, in front of the 3D environment. Using this, a system has been implemented which allows these graphics to be defined as sets of named shapes, edges and text labels.



00:07:47

-00:13:44

Installation bravo should be examined.

Installation alpha should be examined.

Installation alpha should be at the base location.

All installations should be examined.

If any installation is malfunctioning, then it should be at the base location.

Sucessful Mission

The Manipulator A.U.V. will release installation alpha at the base location.

00:07:47

-00:13:44

The Manipulator A.U.V. releases installation alpha at the base location.

Installation alpha should be examined.

Installation alpha should be at the base location.

All installations should be examined.

If any installation is malfunctioning, then it should be at the base location.

Sucessful Mission

The Manipulator A.U.V. will release installation alpha at the base location.

00:07:47

-00:13:44

The Manipulator A.U.V. releases installation alpha at the base location.

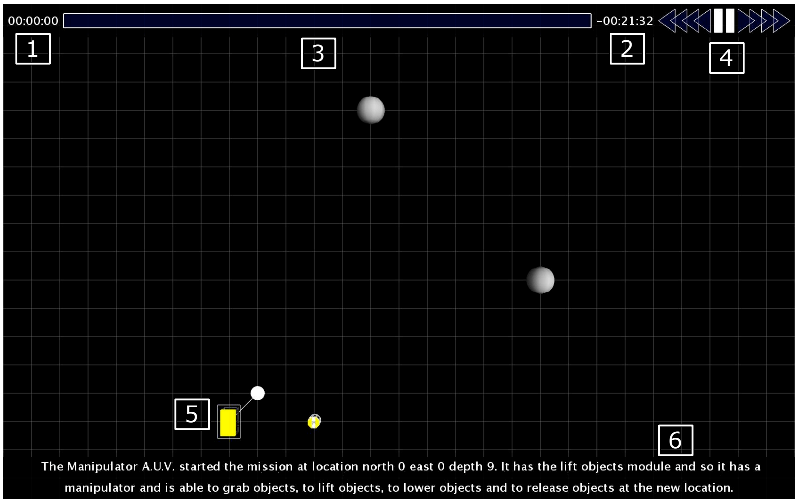
Installation alpha should be at the base location.

If any installation is malfunctioning, then it should be at the base location.

Sucessful Mission

The Manipulator A.U.V. will release installation alpha at the base location.

Figure 4.12: Animated graphics in the Heads Up Display.



The Manipulator A.U.V. started the mission at location north 0 east 0 depth 9. It has the lift objects module and so it has a manipulator and is able to grab objects, to lift objects, to lower objects and to release objects at the new location.

Figure 4.13: Example of features in the Heads Up Display. Including: 1) Mission time passed; 2) Mission time remaining; 3) Mission timeline; 4) Mission playback speed indicator; 5) Currently selected agent; and 6) Subtitle.

A further system has been implemented to order to provide interpolation between shapes, edges and labels which have been defined by the user. This a key-frame based animation system has been implemented to act as the primary source of graphics used for the HUD. The key-frames are defined using XML “scripts”, which supply multiple named shape, edge and label definitions, along with a time. The system then interpolates the additional frames needed for the display of each of the named objects, with the aid of the Java Timing Framework[130]. An example progression is shown in Figure 4.12.

The implemented HUD system allows for only one script to be displayed at a time, but multiple scripts to be added, and switched between based on user or system commands.

An additional set of classes allows for the GraphViz[131] “xdot” output language to be converted to the primitives required for this HUD system. This integration means that GraphViz can be use to provide formatting to the elements used in the display, and the interpolation system can create smooth transitions between multiple GraphViz charts. This allows for the automated construction of complex, well formatted graphics, with any progression that is required.

Two further elements have been added to the main HUD system. The first is a time display, which shows the temporal progression through the current playback, together with the speed the playback is currently moving at. Secondly, a subtitling systems has been implemented, allowing for any text which is required to be shown in large letters at the bottom of the screen. All of these features can be seen in Figure 4.13.

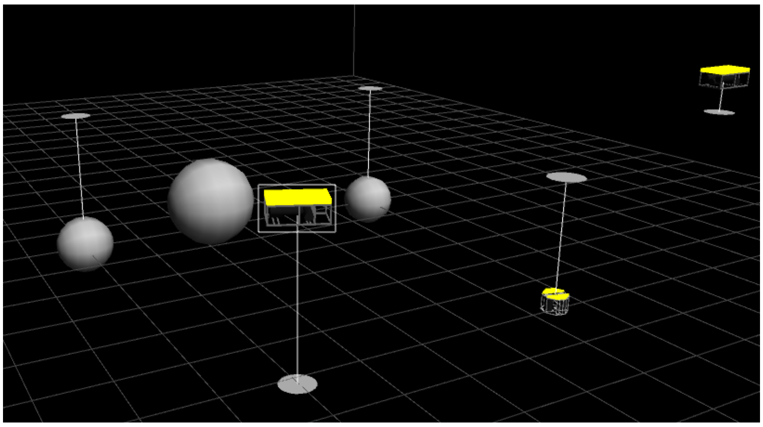


Figure 4.14: Example of additional geometry used to help the user identify the relative positions of object in 3D space. The selected vehicle's position is projected onto the "sea bed" plain, while the position of other objects is projected onto the same plain as the selected vehicle.

### 4.3.3 User Controlled Playback

The first of the two modes the playback systems function in is almost entirely user controlled. The state and position of each of the vehicles and world objects is governed by the logs of the mission which is being replayed, but essentially every other aspect of the playback is controlled by the user.

Positioning the curser over any vehicle or object and clicking with the left mouse button selects it. Selected elements are surrounded by a white box and additional geometry is displayed in order to give a better representation of the element's position in 3D space, particularly in reference to the other objects in the scene. The additional geometry is based on that used in the Homeworld series of computer games (see Section 2.7, and Figure 2.6 in particular), and consists of a disc capped vertical line leading down from the selected element to the grid at the bottom of the 3D environment, as well as similar constructs coming from each of the other elements. stopping at the same altitude as the selected element. The first of these provides a better estimate of the element's absolute position, by showing the point on the grid which the element is directly above, as well as its current altitude above the grid. The second provides similar information for the relative positions of the other elements in the scene. An example of this geometry can be found in Figure 4.14.

The middle mouse button allows the user to control the position of the camera. Clicking this button with any element selected centres the view upon it. Thus, if it moves the viewpoint will follow it. Clicking again with nothing selected will detach the camera, and the focus of the view will remain static at its last position. Moving the mouse wheel will zoom the viewpoint in and out of the current focus, up to a maximum and minimum. Holding down the middle mouse button and moving the mouse will rotate the viewpoint latitudinally and longitudinally around the current focus of the viewpoint.

In the initial state the playback is paused. Pushing the left and right arrow keys changes the default playback speed, moving it in and out of pause, making it faster and slower, and even making it run in reverse. Additionally, the user is able to directly jump to any point in the mission by clicking on the timeline at the top of the screen.

As previously mentioned, the HUD is able to accept multiple scripts, and the keyboard can be programmed to switch between these. The progression within this is keyed to the in mission time, so the HUD graphics will replay exactly like the representations of the vehicles and objects.

Every action the user makes is recorded by the system, particularly those which relate to the control of the camera and the time. These are stored in various log files, one set of which takes the same form as the input files used by the scripted playback system described in the next section. This allows the user's playback to replayed using this system, converted into a video, or analysed in some other way. In particular, the relationship between the time since the user started their playback (playback time) and the time in the mission they are currently viewing (mission time) is recorded in a Comma Separated Value (or CSV) file. This can easily be imported into many spreadsheet, statistics or graphics programs.

### 4.3.4 Scripted Playback

The second mode used by the graphical user interface for playback is controlled entirely by the system, with the user acting as a passive viewer. As implied by this, the entire system is controlled by a set of XML scripts[5], which are:

- The **audio** script, which specifies audio files which should be played at certain times during the debrief;
- The **subtitle** script, which specifies the subtitles which should be displayed along the bottom of the screen;
- The **time** script, which describes the relationship between playback time and mission time during the debrief;
- The **camera** script, which describes the movement of the virtual camera;
- The **focus** script, which specifies which vehicles or other objects in the world should be highlighted at which times during the playback;
- The **graphics** script, which specifies the additional graphics which should be shown to the user on the heads-up-display.

[5] "Scripts" in this case is used in the same context as it might be in film, television, or theatre, rather than its usual meaning in computing.

As indicated by this list, this version uses a *single* graphics script, rather than the multiple options made available by the version of the system described in the previous section. It is also possible to run this version of the system in an additional "analysis" mode, which also makes use of an additional script:

- The **pause** script, which will cause the system to pause at certain predefined times and wait for the user to push the space bar before continuing.

As the user has a minimum of interaction with this version of the system, significantly less data can be recorded. The exceptions to this are the total time taken to deliver the playback, and (since the beginning of Experiment 2, which is described in Section **??**) the amount of time taken by the user during each of the previously mentioned pauses.

### 4.3.5 Future Work

Various potential advancements are possible for this system, though many of these might be purely cosmetic in nature. These are ignored here, and those which advance the actual functionality of the system considered exclusively. Of these, the author considers two in particular to be of the most potential benefit.

The first of these is an advancement of the HUD system. Currently no user input is supported, however it would be possible with some extension to the existing implementation (and the scripting system) to allow the user to click on the objects displayed in the HUD and have the system respond by changing the graphics currently shown on the screen. This would essentially create a branching, menu like system, in which the user could click on a piece of information and have the system respond by providing more details. This would facilitate more of a dialogue between the system and the user.

Another possibility to aid in the creation of a dialogue with the user is to allow the scripting of discrete "scenes", as well the scripting of complete mission playback. This would allow for a branching approach to mission playback, wherein the user could choose which scenes they wished to watch next (controlling this, perhaps, with clicks on the HUD). This would also support the development of a system in which the user could submit a "query" (in a similar manner to SQL) and have the system assemble a set of "scenes" which satisfy this "query" and then play these back to the user.

## 4.4 Summary

The chapter has described the supporting technologies which were developed to facilitate the main aim of this thesis. The first section described a prototype control system able to make decisions at the deliberative level. This system was designed to be able to control multiple distributed autonomous platforms, and be equally suitable for both real and simulated missions. It was based on existing and well proven planning technologies, with some novel optimisation to facilitate distributed operation, communication between different layers of the system, operation in a continuous (non-discrete) physical environment, and also to avoid repeated replanning when possible.

This system was designed and implemented to act as a testbed for simulated missions to be used with the debriefing system described in Part II, but elements of it have also been successfully deployed on a real AUV.

The second section described a system and methodology to enable missions with multiple AUVs to be simulated at extremely high, faster than real time, speeds, whilst maintaining the accuracy of the simulation and allowing the simulation of sensors, actuators and communication with an acceptable degree of realism.

This allows the missions run with the control system described in the previous chapter to be run as quickly as possible, facilitating the verification and iterative improvement improvement required by the framework described in Chapter 3.

The third section described the implementation of a Graphical User Interface specifically designed to fulfil the need for one expressed in the framework described in Chapter 3. This GUI allows the visualisation of missions carried out with multiple real or simulated AUVs, whilst giving either the user or the system complete temporal and spatial control. In addition to this, a Head Up Display allows two dimensional animated graphics to be overlaid on top of this, allowing the provision of additional contextually relevant information.

This interface provides the rich graphics and interaction required for use with the debriefing system described in Part II of this thesis, as well as the additional test cases used in Part III.

The next part of this thesis will describe the primary technology which was developed as part of this project, a narrative based debriefing system which uses the first two of these supporting technologies to generate its input, and the last to deliver its output.

# Part II

# Confrontation: Glaykos, A Narrative Based System for Effective Mission Debriefing

> “
>
> “Of course it is happening in your head, Harry, but why on earth should that mean that it is not real?”
>
> ”
>
> *“Harry Potter and the Deathly Hallows” by J. K. Rowling*

This part is roughly modeled on the second act, or “confrontation”, of a traditional three act narrative. In this, the protagonist begins to take active steps towards defeating their obstacle or antagonist. In doing so, they gain a more complete understanding of the problem itself and the work which is required on their part.

The four chapters which make up this part describe a system intended to take all of the information produced during a simulated or real mission, process it, and then produce an automated debrief, as per the requirement for such a system in the framework described in Chapter 3 (see Figure 4.15). This debrief consists of a controlled replay of the mission, together with auditory narration in natural language (specifically English) and appropriate graphical overlays for illustrative purposes.

In order to create the debrief, the implemented system follows the levels of representation stated in *How does the mind construct and represent stories?*[106]. First it builds a complete situation model in the computer. It does so in two steps. The first is the **bitmap situation model**, which contains all of the data from the mission, and consists of a large and highly connected network of simple concepts. See Chapter 5 for a complete description of this.

From this is built the **vector situation model**, which is made up of a much smaller number of more complex concepts. These are contained within two acyclic

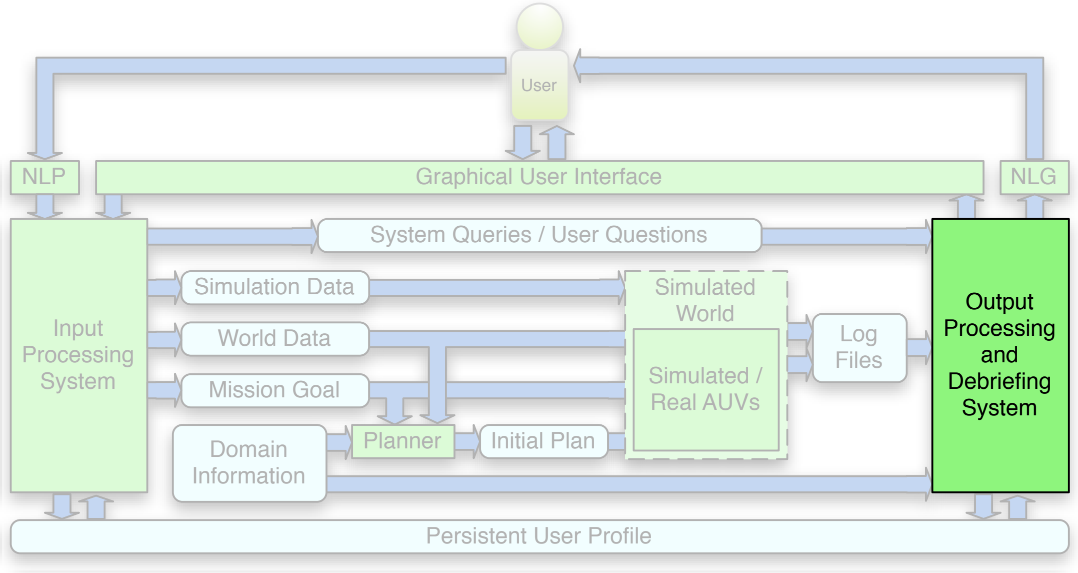


Figure 4.15: The components described by the chapters in this part and how they fit into the framework described in Chapter 3.

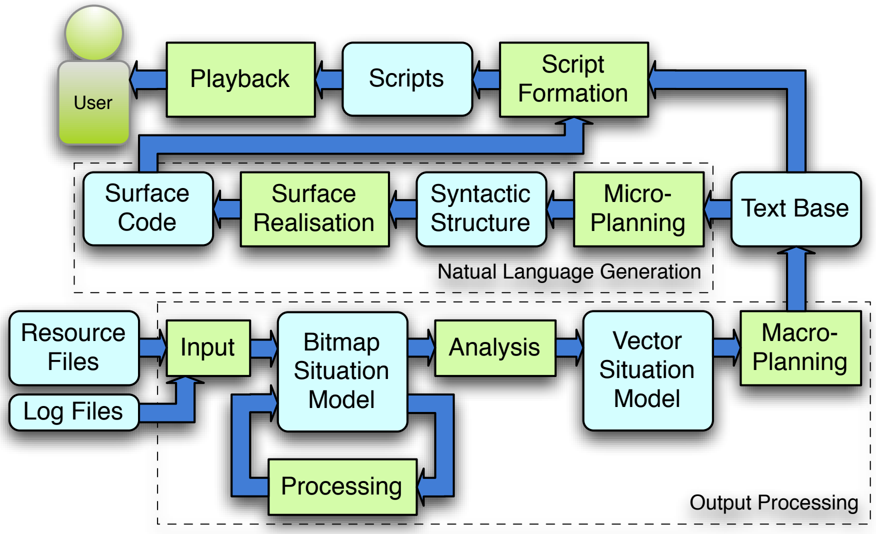


Figure 4.16: Data Flow in the Narrative Debriefing System. The sections outlined with dashed lines refer back to the framework described in Chapter 3 and particularly Figure 3.1.

directed graphs, which model the motivation and causation of the mission. The second of these is made up of **process elements**, which are modeled on "process statements", one of the two fundamental building blocks of narrative described in *Story and Discourse*[3]. Each of these are positioned on the five dimensions of narrative: space, time, protagonist, motivation and causation, as described in [106] and *Situation Models in Language Comprehension and Memory*[107]. The structure and generation of the vector situation model is described in Chapter 6.

These elements are then ordered in such a way as to reduce the total distance travelled across these axes to a minimum. This ordering forms the basis for the **text base**, which is then annotated with **stasis elements** as required. These are analogous to the second fundamental building block of narrative (the "stasis statement") from [3]. This gives us the complete text base.

Each of these elements is translated to natural language, which yields the **surface text**, in the form of an ordered set of sentences. Spoken language is generated using a speech synthesiser, and the time taken for each element recorded. This is used to build the the debrief's model of time during the replay. Also generated from the vector situation model are graphical representations of the motivation axis, which are used on the system's Heads Up Display (HUD) during the debrief. The process is described in detail in Chapter 7.

This set of processes together has been named the "Glaykos" system. This is to reflect the nomenclature convention established in the Ocean Systems Laboratory by the "Delphis" system[132]. Delphis is a system designed to simultaneously co-ordinate multiple AUVs, and so was named for the ancient Greek word for dolphin, an aquatic creature known for its sociability. *Glaykos* extends this convention by using the ancient Greek word for “owl,” a creature often associated with wisdom and knowledge, which the system is intended to impart to its user. The complete *Glaykos* system is summarized in Figure 4.16. Finally, Chapter 8 describes the AUV mission scenarios which were used to test the system, and the discourses which were generated as a result.

# Chapter 5

# The Bitmap Situation Model

This chapter describes the bitmap situation model, a more computer centric version of the "situation model" level of representation discussed in Section 2.10, as well as [106] and [107]. It is named "bitmap" in reference to bitmap images, which contain a large amount of simple constructs, representing the image in its entirety, without compression. As such, the bitmap situation model contains every known detail of the mission at the planning level.

The remainder of the chapter is divided into three sections. The first describes the ontology based structure of the bitmap situation model and the individual elements which constitute it. The second describes how the log files (see Appendix C) produced by the planner based control system (see Section 4.1) are extracted to create its basic structure. The third section describes the further processing which is performed on this original basic version in order to fully realise all of the available detail. Figure 5.1 illustrates how these components integrate into the *Glaykos* system as a whole. A final fourth section summarises the chapter.

## 5.1 An Ontology for Representation of the Bitmap Situation Model

The bitmap situation model is intended to act as a literal knowledge base for all of the data generated at the deliberative (or planning) level during a mission. It is a massive network of simple concepts which represent the progression of the mission, with a particular focus on the beliefs (world state), desires (goals) and intentions (planned actions) of each agent. This model has been extended further to specifically include:

- The **existents** (planner objects), which may be added to the world state during a mission.
- The **requirements** (preconditions) of the intentions.

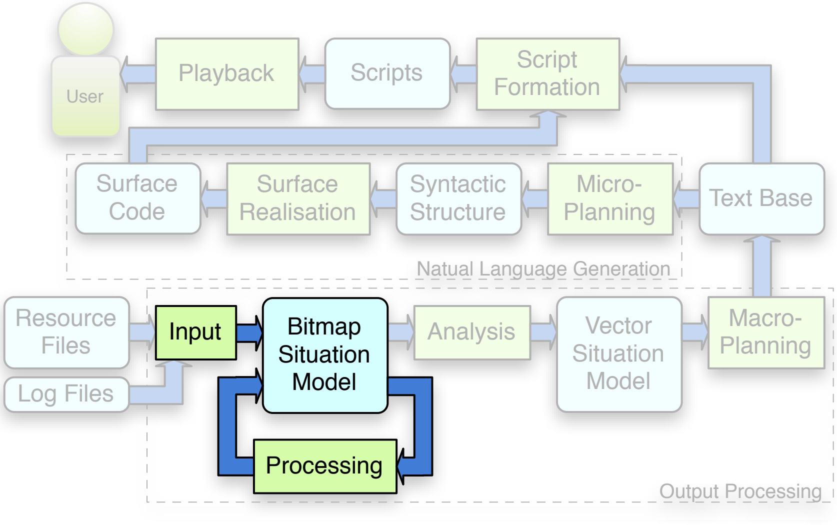


Figure 5.1: The components described by this chapter and how they fit into the structure of the *Glaykos* system (as per Figure 4.15).

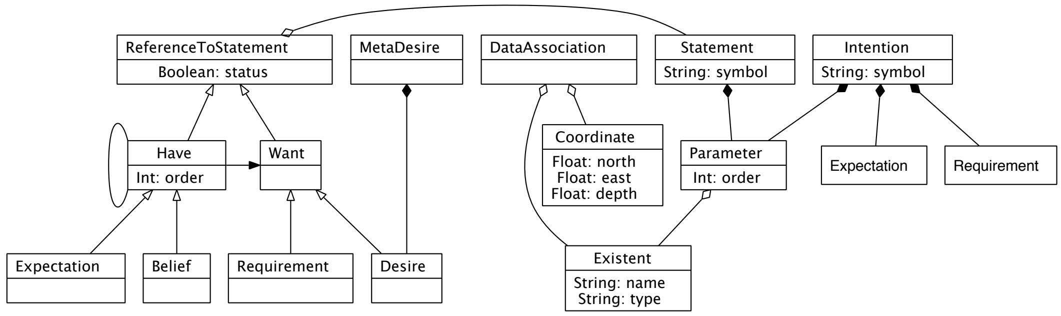


Figure 5.2: Belief, Desire and Intention based ontology constructs.

- The **expectations** (intended effects) of the intentions.
- The **data associations** between the existents and positions in the world.

Also calculated and stored are all of the interdependencies between the beliefs, desires, requirements and expectations. Together with this, a fine grained record of how each intention was expected to affect the world at each time during the mission.

These data are stored in a knowledge base over a custom ontology. This implementation was chosen for reasons of simplicity, flexibility and the ease of creating custom search methodologies. UML diagrams illustrating the basic structure of the ontology are shown in Figures 5.2, 5.3 and 5.4.

The structure of the knowledge base is built up in a series of steps, which are described in the following section. Owing to the complexity of the ontology, its

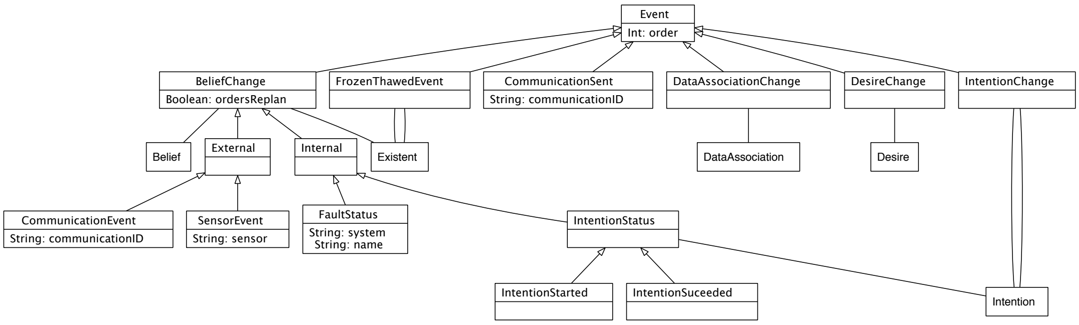


Figure 5.3: Events used in the ontology.

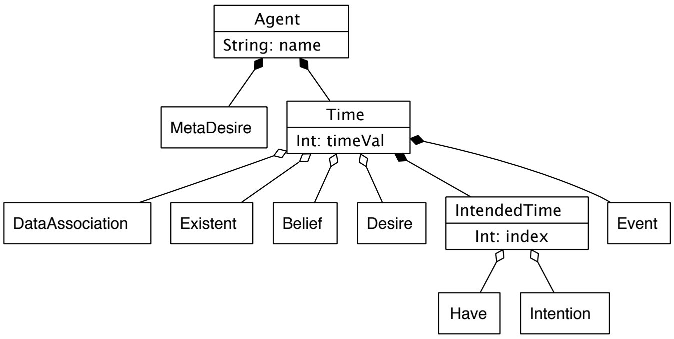


Figure 5.4: Main structure of the ontology.

structure will be described in more detail in parallel to the description of these steps.

## 5.2 Extracting Logs and Domain into the Bitmap Situation Model

As the high level events for each agent are stored in order of occurrence in a single file, the input of each event occurs agent by agent, and secondarily in chronological order. The input system takes advantage of this and caches references to all relevant structures in a local store. This reduces the time taken for input, as access to the local instances is much faster than access to the knowledge base. The reason for this is twofold. Firstly, the local access is inherently faster, as no search is required to find the instance. Secondly, an access to the knowledge base which is preceded by a change to the same requires the reasoner[1] to run, resulting in a large time penalty.

Additionally, this scheme means that searches of the knowledge base will be faster, as tests for equivalence will not be required, and the lack of duplication ensures that there will be less entities in the knowledge base as a whole.

[1] A reasoner is a system which, among things, ensures the consistency of a knowledge base and annotates it with implied knowledge.

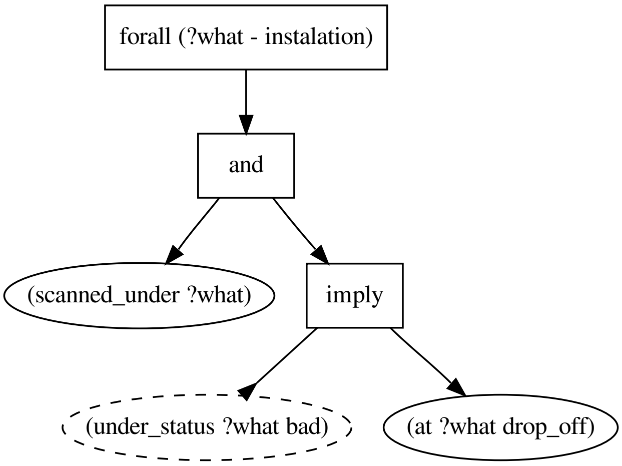


Figure 5.5: Example of base structure for a Meta-Desire.

The first set of references stored locally are those for the agents, which are indexed by name. Next is the set of time instances for each agent, which are indexed by the name of the agent, and then the time (in milliseconds from mission start) to which they refer. The events at each time are not cached, but a running total is maintained for reference.

Also stored are references to each existent. This ensures that the same reference is always used for the same existent, meaning search within the knowledge base can be vastly simplified. As existent names are required to be unique regardless of type, they are simply indexed by name.

References to each distinct statement are also stored. In much the same way as with the existents, it is ensured that each distinct statement is only created once, and then reused. Beliefs, desires, expectations and requirements are implemented as wrappers around the statement they refer to (see Figure 5.2). Again, this increases the speed of searching the ontology.

Updates to the plan are incremental, and for this reason each agent's last known plan is maintained. As updates are received in chronological order, this allows for the new plan to be created by making alterations to the previous plan.

The desires are unique to each agent and these are again maintained in a list indexed by the name of the agent they belong to.

Meta-desires are another globally unique structure which is cached. These are a more complex and high-level version of desires. Whereas the desires an agent has can change during the mission due to changes in circumstances, the meta-desires are fixed for the entire mission. They are generated from the tree structure which forms the basis for the goal of the planning system, and each desire in the mission is considered to be the child of exactly one meta-desire.

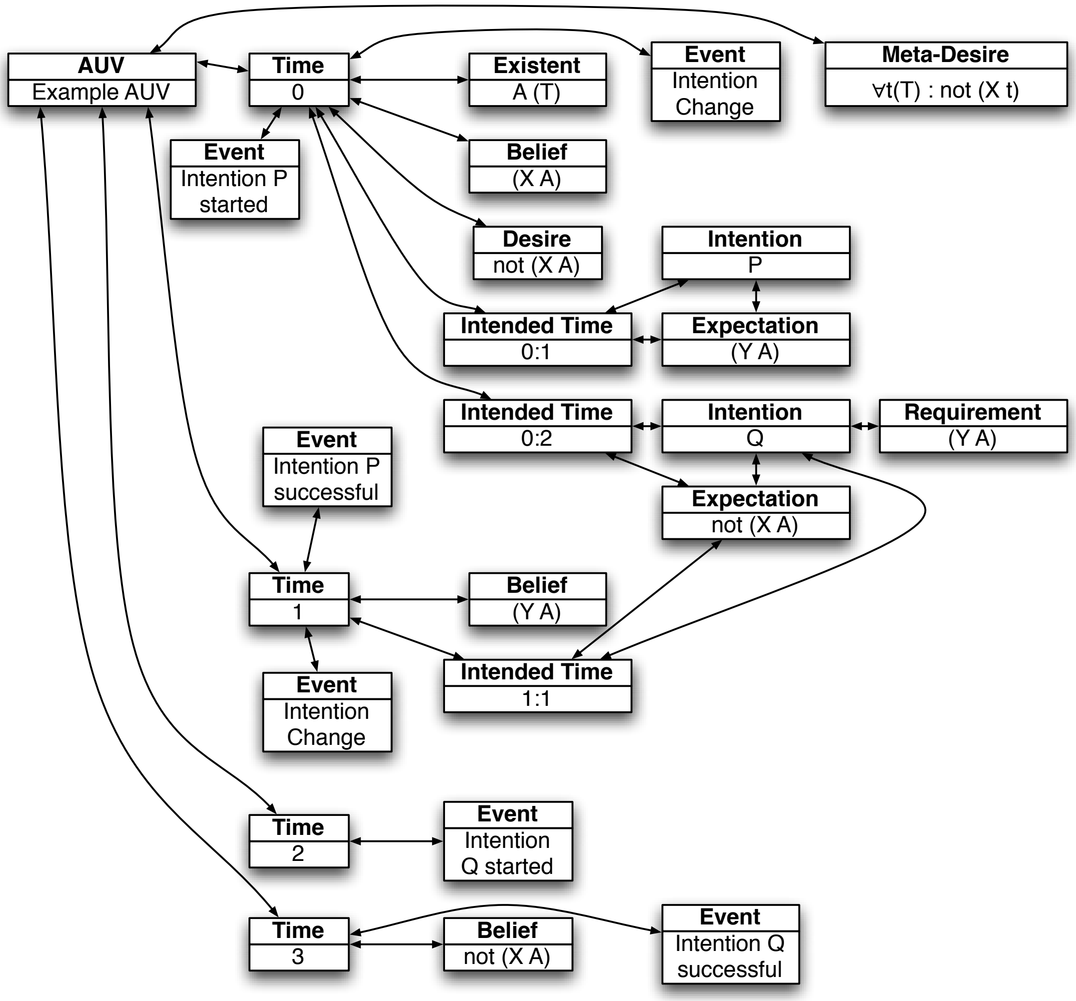


Figure 5.6: Simple example knowledge base, in its initial state immediately after the assimilation of the mission logs.

A meta-desire is a logical term, with exactly one consequent and zero or more antecedents. Consider the goal specification tree shown in Figure 5.5. This tree generates two meta desires. These are generated by taking each main leaf term (denoted by solid ellipses in the diagram) as the consequent, and then following the path to the root, adding the left hand side of any implication terms to the list of antecedents. The "for all" term is maintained by taking any variable starting with "?" as a wild-card. This methodology only supports the *implication*, *and*, *for all* and *not* logical terms, *or* and *exists* are not supported. The two meta-desires produced are "all installations should be examined," and "any installation which is malfunctioning should be at the base location," or more formally:

- → (scanned_under ?what)
- (under_status ?what bad) → (at ?what drop_off)

Finally, Cartesian co-ordinate *positions* are also cached, to insure that the same position is always referred to by the same instance.

With these methods in place, the log messages are processed in sequence, building up the basic structure of the knowledge base. At this early stage, each entity is only stated at the time it changes. So, existents are only stated to exist at the time they were first discovered, beliefs are only stated to be believed at the times that the belief changed, and so on. Figure 5.6 shows a simple example of the knowledge base in this early state. In this example there is a single meta-desire, which states that `forall t(T) : not (X t)`, for each existence `t` which is of type `T`, the proposition `not (X t)` should hold true. As `A` is the only existent with type `T`, this leads to a single desire, that the proposition `not (X A)` should hold true. Two intentions are required to achieve this. Firstly intention `P` makes true the belief that `(Y A)`, which is required by intention `Q`. Intention `P` starts at `Time 0` and succeeds at `Time 1`. Its requirement satisfied, intention `Q` starts at `Time 2` and succeeds at `Time 3`, making true the belief `not (X A)`.

## 5.3 Further Processing of the Bitmap Situation Model

After the mission logs have been input into the knowledge base, it exists in a simple form which contains only the information in the logs, without any of the additional relationships which are required to facilitate the later processing stages. The next stage adds these additional links in series of steps, each of which consists of a retrieval phase and an update phase. As the ontological reasoner is run each time data is retrieved following an update, this limits the number of costly reasoner runs which must occur. Each step makes additions to the knowledge base which are required by one or more of the following steps.

### 5.3.1 Processing Existences, Beliefs, Desires, Intentions and Data Associations

In its most basic form, the knowledge base only links relevant instances to the times at which they were created or changed. With the knowledge base in this form, not all `Time` instances in the ontology contain a complete record of the state of the world at the time they represent. This step seeks to rectify this by instantiating links from all existences, beliefs, desires, intentions and data associations to the times at which they hold true. This step has a number of sub-steps, one for each of the previously mentioned types. These are described in a serial fashion here, though it should be noted that in practice they are interleaved, with all search happening first and any changes made to the knowledge base last. This is done in order to minimise the number of calls made to the reasoner as previously mentioned.

The existents are the simplest of these types, and they are processed first. A SPARQL query (see Appendix E.1, listing 1) is run on the knowledge base, which returns a set of tuples. For each time in each agent's timeline at which no existent is added to the world, the tuple will contain just the name of the agent and a reference to the time. For each time at which an existent is added to the world state, the tuple will also contain a reference to this existent. In the event that multiple existents are added at the same time, one tuple will be returned for each. These are ordered, first by the name of the agent whose timeline they refer to, and then by the time. As existents are never removed from the world state once they are added, this allows the system to maintain a running list of what currently exists for each of the AUVs and append these to each subsequent time. This is accomplished by iterating through the results and applying the following rules:

1 If the AUV identifier has changed, assert that each existent in the list `existsAt` the previous time and clear the list of existents.

2 If the time has changed, assert that each existent in the list `existsAt` the previous time.

3 If the tuple contains an existent, add this to the list.

**Algorithm 5.1** The algorithm used to update the existents at each time in the bitmap situation model.

**Require:** *tuples* the set of tuples produced by the SPARQL query

```
currentAgent ← null
currentTime ← null
list ← empty list of existents
for each tuple ∈ tuples do
    if tuple[agent] ≠ currentAgent then
        if currentTime ≠ null then
            for each existent ∈ list do
                assert that existsAt(existent, currentTime)
            end for
        end if
        clear list
        currentAgent ← tuple[agent]
        currentTime ← tuple[time]
    else if tuple[time] ≠ currentTime then
        if currentTime ≠ null then
            for each existent ∈ list do
                assert that existsAt(existent, currentTime)
            end for
        end if
        currentTime ← tuple[time]
    end if
    if tuple[existent] ∉ list then
        list ← list + tuple[existent]
    end if
end for
```

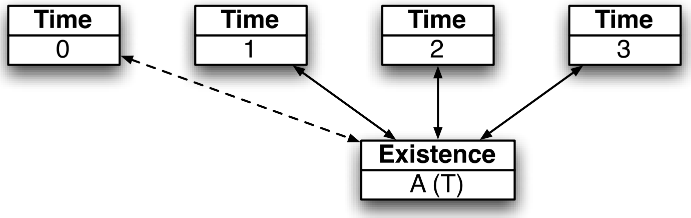


Figure 5.7: Additional links added to the knowledge based by the existents step shown by solid lines. Pre-existing links shown dashed.

This process is presented more formally in Algorithm 5.1, whilst Figure 5.7 summarises the additional links which are added to the example knowledge by this processing step. As existence `A` existed a `Time 0`, additional links are added to indicate that it also exists at every subsequent `Time`.

Next the beliefs are processed in a similar fashion, however as the beliefs change throughout the course of the mission the process is slightly more involved. A search of the knowledge base is performed using a SPARQL query (see Appendix E.1, listing 2), which returns data in a very similar format to the existents query, except looking for changes in belief, rather than existence. However, in this case the status of the belief is also returned, along with the actual statement it refers to (for identification purposes). The tuples are again ordered first by the name of the AUV they concern, and then by their chronological order, next by the order of the event within that time[2], and finally by their order within the belief change. This ensures that the beliefs are asserted in the world in the exact order they were written into the log. A list is again maintained of the state of the world at the current time. The tuples are iterated through and the knowledge base is updated using the following rules:

1 If the AUV identifier has changed, assert a `believedAt` property on the previous time for each of the belief updates in the list and clear the list of beliefs.

2 If the time has changed, assert a `believedAt` property on the previous time for each of the belief updates in the list.

3 If the tuple includes a belief update which makes an actual change to the status of the world, add this update to the list and assert an `updatedBy` property between the previous state of the belief and the new one.

Once again, this process is described more formally in Algorithm 5.2, whilst Figure 5.8 shows the additional properties which are added to the example knowledge base by this step. In similar fashion to the existence step, beliefs are asserted to be

[2] While in real life this is extremely unlikely to happen, in the current simulation methodology is both possible and common for several events to appear to occur at the same instant, whilst also having a distinct ordering.

**Algorithm 5.2** The algorithm used to update the beliefs at each time in the bitmap situation model.

```
Require: $tuples$ the set of tuples produced by the SPARQL query

$currentAgent \leftarrow null$
$currentTime \leftarrow null$
$beliefs \leftarrow$ empty map of statements indexing beliefs
$status \leftarrow$ empty map of statements indexing boolean status
for each $tuple \in tuples$ do
    if $tuple[agent] \neq currentAgent$ then
        if $currentTime \neq null$ then
            for each $belief \in beliefs$ do
                assert that $believedAt(belief, currentTime)$
            end for
        end if
        clear $beliefs$ and $status$
        $currentAgent \leftarrow tuple[agent]$
        $currentTime \leftarrow tuple[time]$
    else if $tuple.time \neq currentTime$ then
        if $currentTime \neq null$ then
            for each $belief \in beliefs$ do
                assert that $believedAt(belief, currentTime)$
            end for
        end if
        $currentTime \leftarrow tuple[time]$
    end if
    $statement \leftarrow tuple[statement]$
    if $statement \notin beliefs$ then
        $beliefs[statement] \leftarrow tuple[belief]$
        $status[statement] \leftarrow tuple[status]$
    else if $tuple[status] \neq status[statment]$ then
        assert that $updatedBy(beliefs[statement], tuple[belief])$
        $beliefs[statement] \leftarrow tuple[belief]$
        $status[statement] \leftarrow tuple[status]$
    end if
end for
```

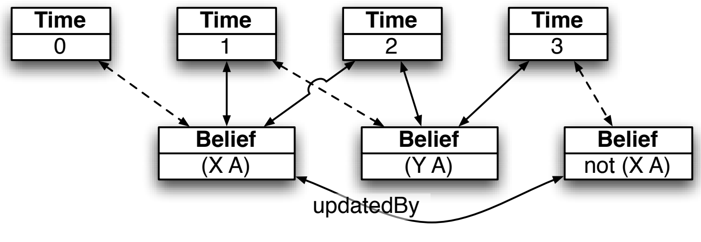


Figure 5.8: Additional links added to the knowledge base by the beliefs step.

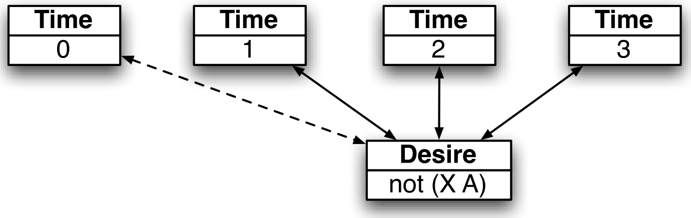


Figure 5.9: Additional links added to the knowledge base by the desires step.

believedAt times which occur subsequent to their creation. However, in this case this propagation stops at times when the status of the statement the belief refers to changes (as in Time 3 in this example). When this happens the two believes are additionally linked by an updatedBy property.

Next the desires are processed. For the purposes of this processing step these act in a very similar way to the existents, as in the present version of the system they are never changed or removed once added to the world state. Again a SPARQL query (see Appendix E.1, listing 3) is used to perform a search across the knowledge base which functions in a nearly identical fashion to the existents step, and is processed in much the same way. The sole difference being that it is a desiredAt property which is asserted between the time and the Desire instance.

Figure 5.9 shows the additional links which are added to the knowledge base by this step. As with the existents step, the Desire is now asserted to be desiredAt every Time subsequent to Time 0, when it was first added to the world state.

Penultimately, each agent's view of the current intentions is proliferated in a similar manner. Whereas the world state at each time is stored as unordered lists of existents, beliefs, desires and data associations (which will be covered next), the representation of intentions is somewhat more involved. Attached to each Time instance is an ordered list of IntendedTime objects, one for each of the intentions which make up the current plan. Each of these represents the projected time after the associated intention has been completed. As such, each has a reference to the Intention in question, as well as an IntendedWorld object, which represents the expected state of the world after the intention has succeeded. At this stage, these structures will only be in place for times at which an IntentionChange event has occurred, which will at present only happen either when an intention successfully completes and is removed from the plan, or when a sensor detection or communication causes additional intentions to be added to the plan. Additionally, for those which are in place, the only elements which will be present in each IntendedWorld are the Expectation instances associated with the Intention to which the relevant IntendedTime is linked. To begin the proliferation of the intentions across the different times, a SPARQL query (see Appendix E.1, listing 4) is performed over the knowledge base.

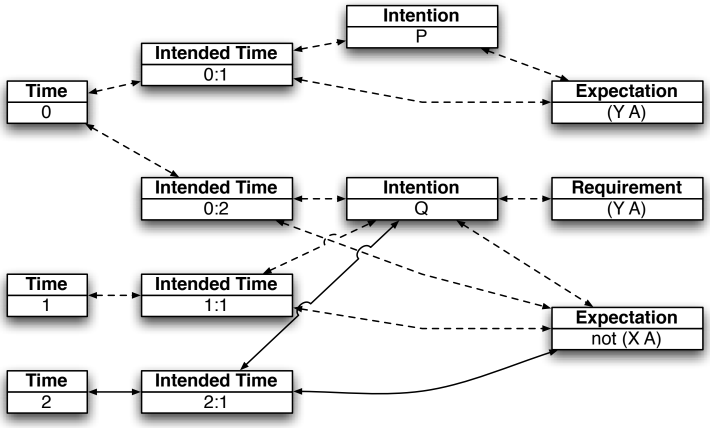


Figure 5.10: Additional links added to the knowledge base by the intentions step.

As with the query used previously for the desires step, this will return tuples for all `Time` instances, together with an `IntentionChange` if one occurred at this time. Again, these are ordered first by the agent whose timeline they refer to, and then chronologically. Though the data which is being processed is more complex, it is also more complete, as the entire plan is available at every time at which an `IntentionChange` occurred. This being the case, a record is maintained of the most recent plan, the data is iterated through, and the knowledge base and record updated via the following rules:

1 If the agent name contained in the tuple is not the same as that in the previous, clear the plan record.

2 If the tuple contains an `IntentionChange` update the plan record to reflect the plan at the associated time.

3 If the tuple does not contain an Intention change, create and add `IntendedTime` and `IntendedWorld` instances an accordance with the Intentions contained in the plan record.

Figure 5.10 shows the new properties which are added to the knowledge base by this step. A new `IntendedTime` instance with an associated `IntendedWorld` instance is created for `Intention Q` and its `Expectation`, then linked to `Time 3`.

Finally, the Data Associations are processed. These connect existents (typically those with a `location` type) in the world state with actual Cartesian coordinates in the real (or simulated) world. Once again, at this stage these are only recorded in the world state for the time at which they were updated. They are proliferated by a method identical to that used in the belief step, only with changes to the position attached to the associated `Existent`, rather than the status of the associated `Statement`, used as a trigger for an update. This begins by running a SPARQL query (see Appendix E.1, listing 5), which returns data in a similar format to that

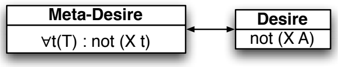


Figure 5.11: Additional links added to the knowledge base by the meta-desires step.

used in the belief step, which is then processed in a similar fashion. As there are no data associations in the example knowledge base, no changes would be made to it by this process.

### 5.3.2 Processing Meta-Desires

The next step links each of the desires to the meta-desire which generated it. Meta-desires are unique in the knowledge base. That is: only one instance of each exists for the mission and is referred to by all agents. Each agent has its own instance of each of the desires, however. Additionally, as the two structures are created by different processes, they are not linked together on creation and so this relationship is generated by implication and asserted into the knowledge base during the post processing stage.

This process makes use of the Jena API (See Appendix A.3) for direct access to the knowledge base. Each Desire is inspected in turn, and compared to each Meta-Desire. A valid match is considered to be found if the consequent (right hand side) of the meta desire matches the desire in the following respects:

1 They have the same status value.

2 They both refer to statements with the same predicate.

3 Each non-variable parameter in the consequent is an exact match for equivalent parameter in the desire.

4 Each variable parameter in the consequent has a type which is equal to the type of the equivalent parameter in the desire, or a super type of it.

If all of these conditions are met then a property is added indicating that the Desire is a sub desire of the Meta-Desire. In the example knowledge base, this results in a single property being added to the knowledge base, which is illustrated in Figure 5.11.

### 5.3.3 Processing Coherence and Expectations

The next stage in the post processing of the knowledge base covers two distinct and essentially unrelated matters: the coherence between agents and the expectations of the intentions. These are processed as part of the same stage as they have no interdependencies on each other and depend only on the previous stages. Processing them together again reduces the number of expensive calls which are required to the reasoner.

**Algorithm 5.3** The algorithm used to process the coherence between the beliefs of each agent.

```
for each Agent $a1$ do
    for each time $t1 \in a1$'s timeline do
        $b1 \leftarrow$ beliefs at $t1$
        $e1 \leftarrow$ existents at $t1$
        $d1 \leftarrow$ data associations at $t1$
        for each Agent $a2, a2 \neq a1$ do
            Find the time $t2 \in a2$'s timeline where $maximise(t2), t2 \leq t1$
            $b2 \leftarrow$ beliefs at $t2$
            $e2 \leftarrow$ existents at $t2$
            $d2 \leftarrow$ data associations at $t2$
            if $b1 = b2 \wedge e1 = e2 \wedge d1 = d2$ then
                assert that $t1$ is coherent with $a2$
            end if
        end for
    end for
end for
```

Firstly, the coherence is processed. This is a property which is potentially applied to each `Time` in the timeline of each agent, once for each other agent. If it is asserted for a particular time and particular agent, it indicates that the main agent's picture of the world state at this time is an exact match to the referred to agent's picture of the world state at the same time. Algorithm 5.3 illustrates how this is accomplished, using the Jena API. As our running example only contains a single agent, no diagrammatic representation is provided for this step, however.

Next, the `Expectation` elements are processed. These refer to the ways in the intentions are expected to change the world, as apposed to the beliefs, which describe the actual current state of the world (or at least the AUVs' perception of the same). The `ExpectedWorld` elements which are attached to each expected time contain both `Belief` and `Expectation` elements, illustrating the ways in which the world is expected to both change and stay the same as the intentions are carried out. As previously mentioned, at this stage the `ExpectedWorld` elements contain only the `Expectations` of the `Intentions` to which they are attached. The next process updates these so that the complete expected world is reflected in each element.

The Belief and Expectation elements are proliferated through the expected worlds using a process almost identical to that used for proliferating the beliefs across the `Time` elements described in section 5.3.1. In this instance the Jena API is employed for data collection in place of SPARQL queries, however. The process used to accomplish this is shown in Algorithm 5.4, while Figure 5.12 shows the additional links

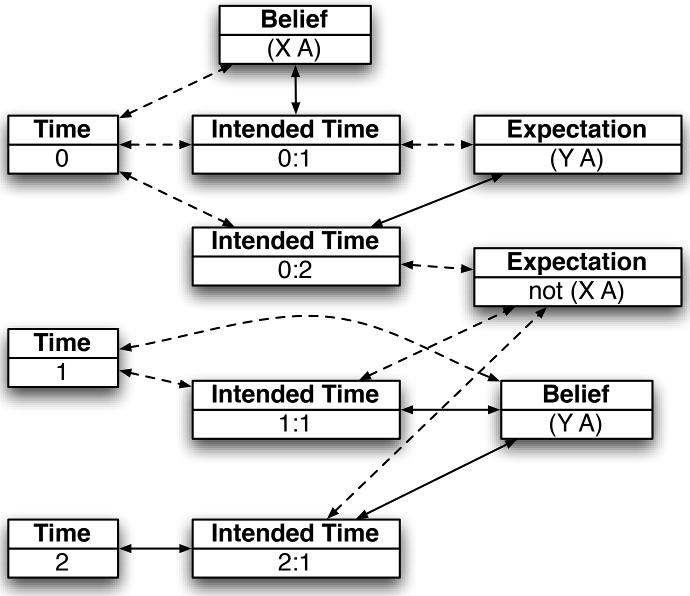


Figure 5.12: Additional properties added to the example knowledge base during the expectations step.

**Algorithm 5.4** The algorithm used to process the expectations of each intention at each point in the timeline.

```
for each Agent $a$ do
    for each time $t \in a$'s timeline do
        $w \leftarrow$ a copy of the beliefs at $t$
        for each ExpectedTime $eT \in t$, ordered by $eT.index$ do
            for each Expectation $e$ at $eT$ do
                for each entry $b \in w$ do
                    if $eT.statement = b.statement$ then
                        $w \leftarrow w - b$
                    end if
                end for
                $w \leftarrow w + e$
            end for
            Set the contents of $eT$ to be equal to $w$
        end for
    end for
end for
```

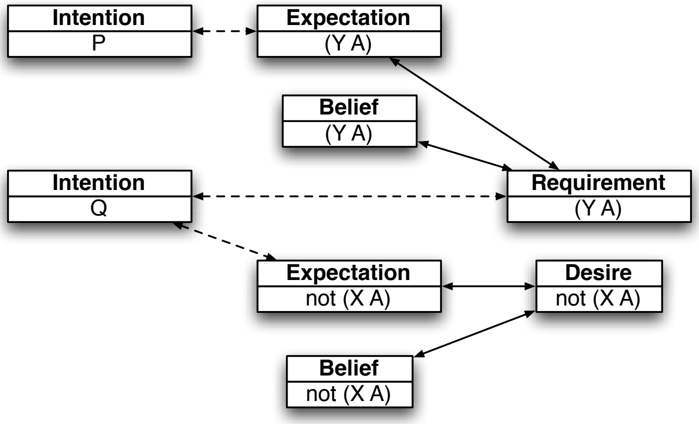


Figure 5.13: Additional properties added to the example knowledge base during the satisfaction step.

which are added to the example bitmap situation model after the expectations have been processed.

### 5.3.4 Processing Desire and Requirement Satisfaction

Desires are the conditions which must be met in order for the mission to be successful, whereas Requirements are the conditions which must be met in order for an Intention to be carried out. Each of these must be satisfied by a `Belief`, and may also be satisfied in potential by an `Expectation`. As the previous processing stage ensured that the beliefs and and expectations are properly represented, it is now possible for the system to match each of the desires and requirements with the beliefs and desires which satisfy them. Algorithm 5.5 describes the process which is used to accomplish this.

As before, all assertions are cached to ensure that no duplicates are created and additional calls to the reasoner are not made. Figure 5.13 shows the additional links which are added to the example data set by this step.

### 5.3.5 Resolving Communications Events

The final processing step performed on the bitmap situation model is to resolve the communication events. Many of the updates made to an AUV's set of beliefs about the world are made by communications events, which carry data which has been detected by another AUV. These events are opaque, however, and do not provide the original source of the data. This means that there is a break in the causal chain where communication events are concerned. This processing stage attempts to correct this, by tying the updates made by a communication event to their source on the originating AUV.

**Algorithm 5.5** The algorithm used to process the satisfaction of the desires and requirements.

```
for each Agent $a$ do
    for each time $t \in a$'s timeline do
        for each desire $d$ which is desired at $t$ do
            if there is a belief $b$ at $t$ which satisfies $d$ then
                assert that $b$ satisfies $d$
            else
                for each IntendedTime $iT \in t$, in reverse order do
                    $eW \leftarrow$ the ExpectedWorld attached to $iT$
                    if there is an Expectation $e \in eW$ which satisfies $d$ then
                        assert that $e$ satisfies $d$
                        break
                    end if
                end for
            end if
        end for
        for $n = 0$ to number of IntendedTimes in $t$ do
            $iT \leftarrow$ IntendedTime with index $n$
            $i \leftarrow$ the Intention attached to $iT$
            for each Requirement $r$ of $i$ do
                if $n = 0$ then
                    if there is a belief $b$ at $t$ which satisfies $r$ then
                        assert that $b$ satisfies $r$
                    end if
                else
                    $iTP \leftarrow$ IntendedTime with index $n - 1$
                    $eW \leftarrow$ the ExpectedWorld attached to $iTP$
                    if there is a Belief or Expectation $b$ at $t$ which satisfies $r$ then
                        assert that $b$ satisfies $r$
                    end if
                end if
            end for
        end for
    end for
end for
```

The first stage of this process uses a SPARQL query (see Appendix E.2, listing 1) to retrieve every belief which is changed by a communications event, the communication event which precipitated the change locally, and the remote event which caused the original change on the source vehicle.

The unique communication ID value which is attached to each communication event, along with the “Communication Sent” event which is recorded on the source AUV when the communication is made is the key piece of data which is used to reconstruct this. It allows the query to find the exact copy of the belief on the source vehicle which is being communicated.

A “data source” connection is then made between each of the communication events on the local AUV and the relevant belief change on the remote AUV, forging the missing link in the causal chain.

## 5.4 Summary

This chapter has described the bitmap situation model, a low level representation of every event which happened during the mission, and every piece of information which became available. The novel ontology used to represent this has been described, together with each of the algorithms used to generate it from the mission logs (see Appendix C), such as those produced by the prototype control system described in Section 4.1.

The bitmap situation model is used as codex for the mission, and from it is generated the more high level vector situation model, which is designed to more closely mimic the way information is thought to be represented in the human mind. This is described in the following chapter.

# Chapter 6

# The Vector Situation Model

This chapter describes the vector situation model, which is, as compared to the previous chapter's bitmap situation model, a more literal implementation of the "situation model" level of representation discussed in Section 2.10, as well as [106] and [107]. As with the bitmap situation model, it is named "vector" in reference to vector images. These contain a smaller number of more complex constructs, which allows the most important details of an image to be stored using less space for their representation, at the cost of some loss of fine detail. The vector situation model, then, attempts to represent only the elements and connections which are crucial to the understanding of the mission.

The remainder of the chapter is divided into three sections. The first of these describes the structure of the vector situation model and the two substructures within it. The second describes how the mass of information stored in the bitmap situation model is processed in order to generate the vector situation model. Finally, the third section summarises the chapter. Figure 6.1 illustrates how the components described in this chapter integrate into the *Glaykos* system as a whole.

## 6.1 Structure of the Vector Situation Model

The vector situation model consists of elements to represent process and stasis statements (see Section 2.10 for a description of these concepts), as well as wrapper objects used as bridges to information in the bitmap situation model. These form two distinct structures:

1 **The motivation model**, which is used to track the motivations of the AUVs at a high level; and

2 **The causation model**, which represents a lower level, and tracks the causal relationships between the individual events of the mission.

The following two subsections describe each of these structures in more detail. As will become obvious, the two structures are not separable, and have numerous inter-

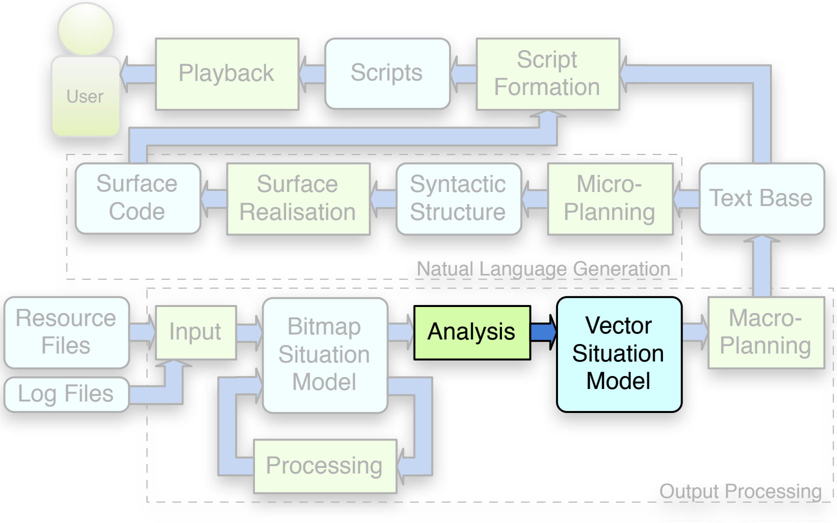


Figure 6.1: The components described by this chapter and how they fit into the structure of the *Glaykos* system (as per Figure 4.15).

connections. A third sub-section then describes how the elements of the causation model are positioned along the “five situational axes” (as described in [106] and [107]), which will be used in Section 7.1 to help construct an ideal discourse ordering for these elements.

### 6.1.1 The Motivation Model

The motivation model is an acyclic directed graph which describes the flow of motivation through the mission. This starts with the concept of a “successful mission”, though this to the meta-desires which specify the success conditions, though these to the atomic desires which these produce, and then from these the intentions carried out by the agents in order to satisfy the desires. As such, the motivation model is made up of structures which act as wrappers for three fundamental concepts.

“Meta desire wrappers” are at the highest level of the graph. Each of these contains:

- Its **consequent**;
- A list of its **antecedents** (which may be empty);
- The time at which it was **satisfied**;
- A list of the desire wrappers which represent its **sub-desires**.

The next level of the graph contains the “desire wrappers.” These contain:

- A predicate representation of the **desire** itself;
- The time at which it was **introduced**;

- The time at which it was **satisfied**;
- A reference to the Intention Wrapper which represents the intention which it was **satisfied by**.

Lastly, "intention wrappers" are used to represent the canonical instances of the intentions. That is, the references to the intentions from the agents which actually carried them out, as opposed to the reference to these intentions which were used by the other agents. Each of these contains:

- A predicate representation of the **intention** itself;
- The time at which it was **added** to the plan;
- The time at which its implementation **started**;
- The time at which its implementation **succeeded**;
- A list of **motivation frames**, representing the desires to which it contributes to in some way;
- The AUV **agent** which actually carried out the intention;
- A list of the desire wrappers which represent the desires which it **satisfies**;
- A list of intention wrappers which represent the intentions which it **relies upon**;
- A list of intention wrappers which represent the intentions which it is **relied upon by**;
- A list of intention wrappers which represent the intentions which it **invalidates**;
- A list of intention wrappers which represent the intentions which it is **invalidated by**.

These last four items refer to the ordering constraints between the different intentions. If intention A depends on intention B, then A must wait for B to succeed before it can be started, as B changes the world in a way which makes A possible. However, if intention A invalidates intention B, then A must wait for B to finish before it can start, as A makes a change to the world which means that B is no longer possible. This will be described in more detail in Section 6.2.2.

These wrappers also contain references to all of the "Process Elements" which concern them. These will be described in the next section.

Figure 6.2 shows an example motivation model, taken from a more complex mission than the one used for the knowledge base example in the previous chapter. This will be used as part of a running example for the remainder of this chapter.

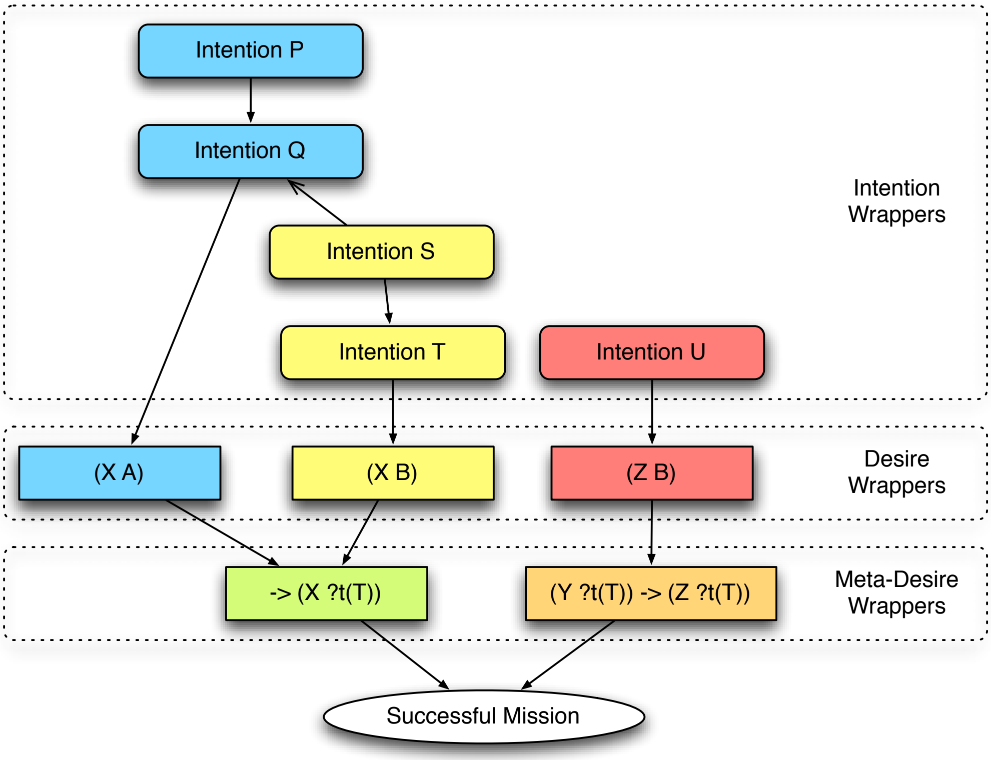


Figure 6.2: Example motivation model.

### 6.1.2 The Causation Model

The causation model is an acyclic directed graph which describes the causal relationships between the different events in the mission. Each node in this graph represents a "process element", which are analogous to the "process statements" discussed in Section 2.10 and [3]. Our "process elements" perform a near identical purpose within the causation model. The nomenclature change from "statement" to "element" is used to signify the fact that these structures are abstract and not fully realised natural language. They represent the active events which contribute to the development of the mission. At this stage, nine distinct types of process element are used to make up the causation model. We consider these to be analogous to the narratological units identified in Russian folk tales by Vladimir Propp (see Section 2.10 and [105]). They are:

- **Mission Started**;
- **Desire Introduced**;
- **Intention Introduced**, which indicates that an intention has been added to the plan;
- **Plan Introduced**, which indicates that a set of intentions with a particular purpose have have been added to the plan. This element is created by the simplification process described in Section 7.2.1.

- **Intention Started**, which also contains a list of the sensor and actuator mode changes which are triggered by this;
- **Intention Successful**, which also contains a list of the sensor and actuator mode changes which are triggered by this;
- **Sensor Event**, which contains a list of the *object location* and *object property* stasis elements which this sensor event added to the world state. Stasis elements will be discussed in more detail later in this section;
- **Desire Satisfied**;
- **Meta-Desire Satisfied**;
- **Mission Succeeded**.

As noted, each of the the "Intention Started" and "Intention Successful" process elements contains a list of the sensor and actuator mode changes they triggered. These structures are identical, and contain the follow information:

- The name of the AUV **agent** which the sensor or actuator actually belongs to.
- The **time** at which the mode change happened.
- The name of the **sensor**.
- The name of the new **mode** of the sensor.

According to the vocabulary used in *Story and Discourse*[3], the events represented by the process elements would all be categorised as "actions", rather than "happenings". This is as they related to the direct actions of our protagonists, rather than events resulting from factors beyond their control. At present we do not deal with dynamic environments or faults, but the events connected to these would likely be categorised as "happenings".

The links, or edges, which connect our nodes come in two forms:

1 **caused by**;

2 **contributed to by**.

The distinction between these is semantically slight and serves only to distinguish an event's most direct cause. With the exception of the *mission started* event, all events have exactly one *caused by* event, but can have any number of *contributed to by* events. The occurrence of all of these is required for the event take place, but the *caused by* event is that which happened chronologically last, and so is assumed to have acted as the direct trigger.

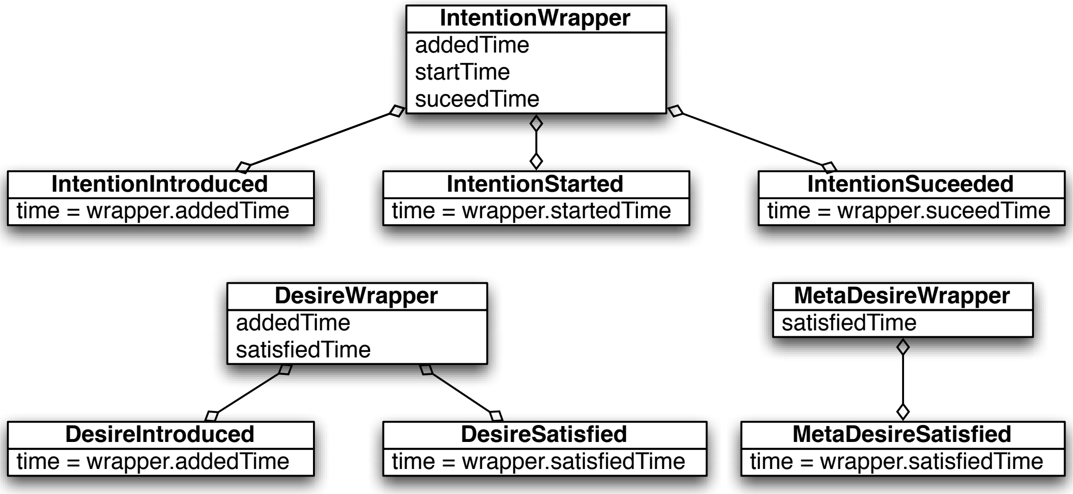


Figure 6.3: The direct relationships between the components of the motivation and causation models.

For example; Consider a meta-desire with three sub-desires. The satisfaction of all three of these sub-desires is required for the satisfaction of the meta-desire. However it is the sub-desire satisfaction which occurred temporally last which actually caused the meta-desire satisfaction to occur. Therefore, the meta-desire satisfaction is caused by the satisfaction of the third desire, and contributed to by the satisfaction of the first two.

In order to simplify the processing of the causation model, the inverse connections "causes" and "contributes to" are also made. As such, each process element has the following references to other process elements:

- A reference to the Process Element which it was **caused by**.
- A list of the Process Elements which it was **contributed to by**.
- A list of Process Elements which it **caused**.
- A list of Process Elements which it **contributed to**.

Many of the process elements directly correlate to the wrapper objects which make up the motivation model. Figure 6.3 shows these relationships in the form of a UML graph.

"Process statements" are the first fundamental building block of narrative employed in *Story and Discourse*[3]. The second is the "stasis statement," which provides information about the state of the world ("agent A was at location B," rather than "agent A moved to location B."). To fulfil this purpose we employ "stasis elements" to provide additional relevant information about the world. We define 10 stasis elements, which are the following:

- **Actuator mode explanation**, which describes the functioning of a particular mode of a particular type of actuator;
- **Sensor mode explanation**, as above, but for sensors;
- **AUV explanation**, which gives the name and starting location of a specific AUV;
- **Capability**, which explains the details of a capability which an AUV has (actions it enables, plus sensors and actuators it provides);
- **Conditions**, which states the requirements of an intention along with the intention(s) which satisfied or will satisfy them;
- **Effect**, which states the expectations of an intention, along with the intention requirements and desires which are satisfied by them;
- **Intention in progress**, which states that a particular intention is currently in progress;
- **Meta-desire explanation**, which provides details of a meta-desire;
- **Object location**, which gives the location of a (non AUV) object in the world;
- **Object property**, which describes a relevant property of a (non AUV) object in the world.

Each process element contains three lists of stasis elements which serve the following purposes:

- The stasis elements the process element **implies**. These are the stasis elements whose information is contained within that of the process element.
- The stasis elements the process element **relies upon**. These are the stasis elements which are considered to be relevant before the process element occurs.
- The stasis elements the process element **introduces**. These are the stasis elements which are considered to be relevant after the process element occurs.

In addition, each stasis element also has a list of the stasis elements which it implies. For functional purposes, each stasis element is defined to imply itself.

Figure 6.4 shows an example causation model, which relates directly to the example motivation model which is used shown in Figure 6.2.

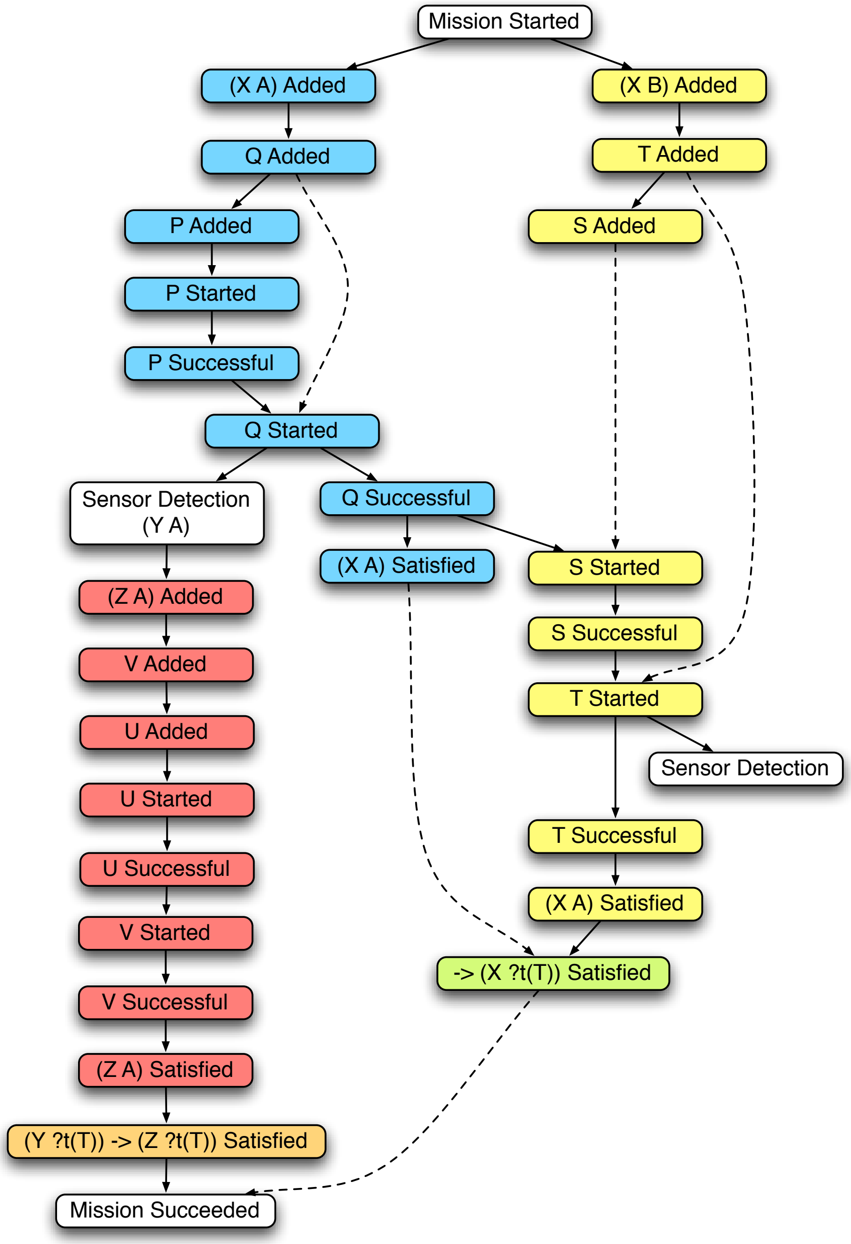


Figure 6.4: Example causation model. Unbroken arrows indicate a "causes" relationship, dashed lines indicate a "contributes to" relationship. Colours indicate motivation strands from Figure 6.2.

### 6.1.3 Situational Axis Positions for Process Elements

Each process element is considered to have a position on each of the five situational axes, as described in *Situation Models in Language Comprehension and Memory*[107] and *How does the mind construct and represent stories?*[106]. This section details how the events are placed on these axes.

#### Space

In traditional narratives space essentially consists of discrete locations. In the domain of underwater robotics, however, space is visibly continuous. As we are not considering any enclosed spaces, any one location can be seen from any other (with some exceptions due to occlusion by agents and objects in the world). Furthermore, the distance between any two locations can be simply and quickly bridged, facilitating the user's retention of the geographic model of the environment. This being the case, a continuous model of space is used, with the position of an event on the spatial axis simply being the Cartesian co-ordinate of the location at which it occurred, where applicable. More specifically, the location of the agent which either caused the event to occur, or the event occurred to, at the time the event occurred. The exceptions to this are the events, such as the start of the mission, which cannot be said to have occurred to one specific agent.

#### Time

Time is the most universal of all the axes, as all process elements by definition occur at a particular instant. It is also the simplest axis upon which to locate a process element. For our measure, we use time in milliseconds since the start of the mission.

#### Protagonist

Where applicable, the protagonist for a particular intention is simply the agent who carried out the event, or to whom the event occurred. As with the space axis, there are some events to which this axis is not applicable, such as the start and success of the mission.

#### Motivation

The motivation axis position for an event is directly connected to the motivation model described in Section 6.1.1. For the events which concern an intention, the motivation is considered to be the primary desire towards which the relevant intention is contributing towards, and secondarily the meta-desire of that desire. For events concerning desires, the primary motivation is considered to be the relevant desire, with the secondary motivation being its meta-desire. Finally, for events concerning meta-desires the motivation is considered to be the meta-desire itself. For other events the motivation axis is considered to be inapplicable.

#### Causation

The causal axis position of an event is essentially considered to be the event itself, as placed in the causation model described in Section 6.1.2.

## 6.2 Extracting Data into the Vector Situation Model

Whilst the previous section provided the relevant *stasis* information required for the understanding of the structure of the vector situation model, this section attempts to describe the *process* which is used to construct it from the existing bitmap situation model. The two substructures are assembled in the order they were previously described, beginning with the simpler motivation model and then proceeding to the more complex causation model. Before either of these is attempted, however, a record of the initial state of the world is made for comparative purposes. Each of these processes will be described in the following subsections.

### 6.2.1 Finding the ground state

The first step in the creation of the vector situation model is the creation of a record of the ground state. This is a list of all beliefs and existents which made up the world state at the beginning of the mission. For each `Agent` in the bitmap situation model's knowledge base, the `Time` object with a time value of zero is found, and then the beliefs and and existents which make up the world state at this time are recorded. This is performed for all agents to ensure that they are coherent at the start of the mission, which should always be the case. If this is not found to the the case, the system exists and indicates that an error has been found in the log files.

### 6.2.2 Building the Motivation model

The first step in the creation of the motivation model is to generate the intention wrappers. These are created from the canonical intention references in the bitmap situation model. While each AUV will carry a reference to each intention which has been part of the plan, the canonical references are those which are carried by the agent which actually carried out the intention in question. For this purpose, a SPARQL query (see Appendix E.3, listing 1) is used to retrieve a reference to each intention, the time at which it was added to the plan, the time it was started, the time it was completed and the name of the agent which carried it out.

Next the interdependencies between the intentions which affect their ordering are calculated. The first of these factors is reliance. Intention A is considered to be reliant on intention B if intention B changes the world in such as way as to make intention A possible. For example, consider the following two intentions:

1 Move to site A.

2 Examine object at site A.

Intention 2 is reliant on intention 1, as it is not possible for the object at site A to be examined without first moving to site A.

The first stage finds the requirements for each of the intention wrappers which are satisfied locally on the AUV which carried out the intention. For this a SPARQL query (see Appendix E.3, listing 2) is used to retrieve the intentions in question, and the intentions which satisfy their requirements. Also retrieved are the requirements which are satisfied, and the beliefs which satisfied them, but this information is mainly use for debugging purposes.

Next the requirements which are satisfied by intentions carried out by other agents are found. A SPARQL query (see Appendix E.3, listing 3) does this by tracing the belief which satisfied the requirement back to a Communication Event, and then via the "data source" property (see Section 5.3.5) back to the event which generated the original belief on the source AUV.

For each pair of intentions which is retrieved by the two previous queries, the first is added to the "relied upon by" list of the second, and the second is added to the "relies upon" list of the first.

The second factor which affects the ordering of the intentions is invalidation. One intention is considered to invalidate another if it changes the world in such a way as to make the second intention no longer possible. As an example, consider the following extension to the previous example:

1 Move to site A.

2 Examine object at site A.

3 Move to site B.

Intention 3 is considered to invalidate intention 2, as it is longer possible to examine the object at site A after moving to site B.

A similar process is used to find the invalidation connections as was used to find those for the reliance. Again, two SPARQL queries are used. The first (see Appendix E.3, listing 4), finds those invalidations which are generated by the actions of the local AUV. A second query (see Appendix E.3, listing 5) is then to find the invalidations generated on other AUVs and received via communication events.

Both queries work by finding the requirements of an intention, and then finding an event which occurred after the intention started which makes the intention no

**Algorithm 6.1** The generation of desire and meta-desire wrappers for the motivation model.

```
for each set of output values do
    $desire$ ← the reference to the desire
    $meta$ ← the reference to its meta-desire
    $intention$ ← the reference to the intention which satisfied the desire
    $added$ ← the time at which the desire was introduced
    if $meta$ does not match any known meta-desire then
        $newMetaDesire$ ← new Meta-Desire Wrapper
        retreive antecedants and consequant via the JENA API and add to $newMetaDesire$
        $newMetaDesire.satisfied$ ← 0
        add $newMetaDesire$ to list of meta-desires
    end if
    $intentionWrapper$ ← the Intention Wrapper which corresponds to $intention$
    $desireWrapper$ ← new Desire Wrapper
    $desireWrapper.introduced$ ← $added$
    $desireWrapper.satisfied$ ← $intentionWrapper.suceeded$
    $desireWrapper.satisfiedBy$ ← $intentionWrapper$
    add $desireWrapper$ to $intentionWrapper.satisfies$
    $metadesireWrapper$ ← the Meta-Desire Wrapper which corresponds to $meta$
    $desire.metaDesire$ ← $metadesireWrapper$
    $metadesireWrapper.satisfied$ ← MAX(metadesireWrapper.satisfied, desireWrapper.satisfied)
    add $desireWrapper$ to $metadesireWrapper.subDesires$
end for
```

longer possible. Some intentions essentially invalidate themselves after they start, so in the local case a `FILTER` is used to prevent these cases from being returned.

In a similar fashion to the reliance stage, for each pair of intentions which is retrieved by these two queries, the first is added to the "invalidated by" list of the second, and the second is added to the "invalidates" list of the first.

Next, the desires and meta-desires are retrieved from the bitmap situation model. As with the intentions, each AUV has its own reference to each desire, whereas meta-desires are globally unique (as they are present at the start of the mission and do not change throughout). This being the case, the canonical instance of a particular desire is taken to be the one used by the AUV which performed the intention which actually solved it. This is again done using a SPARQL query (see Appendix E.3, listing 6), which retrieves each desire, along with its meta-desire, the time which it was added to the mission and the intention which solved it.

The output of this query is then processed using Algorithm 6.1, which creates the desire and meta-desire wrappers as required, and generates the links between them and the intention wrappers.

Each intention can potentially contribute to more than one desire, or given a complicated motivation model this may appear to be the case when it in fact is not. Motivation frames are used to track this information and identify the desires an intention actually contributes to. A motivation frame contains the following information:

- The desire **track** to which it refers.
- Its **level**, which represents its distance from the desire in the motivation model.
- Whether it is considered to be **questionable** or not. A questionable motivation frame is one which potentially, but not definitely, contributes to its desires. These are used to generate correct ordering of the wrappers in the motivation model.

The creation of the motivation frames is handled by Algorithm 6.2, which first generates all of the motivation frames by "trickling" the information down from the desire wrappers to the intention wrappers, which then pass on the information to the other intention wrappers which they rely upon and invalidate. The questionable flag is used when passing the information on to invalidated intentions, as there is not a direct causal link here, but there is an ordering constraint. As Algorithm 6.2 illustrates, motivation frames which refer to the same desire are merged, maintaining the lowest ordering value and potentially converting a motivation frame from questionable to non-questionable. After all of the processing has been done, any surviving questionable frames are then removed, and the remaining frames sorted into ascending order by their level. The first motivation frame in the resultant list (the one with the lowest level, and therefore the highest proximity to the satisfaction of the desire) is considered to be the intention's "primary" motivation.

The final stage in the creation of the motivation model adds an additional type of motivation frame to the intention wrappers. These are the "rectifying" motivation frames, which indicate that an intention restores an element of the initial world state which has been changed during the satisfaction of another desire. In this way the intention can be thought of as contributing to the first desire, even though it was not required for its satisfaction. In order to distinguish them, rectifying frames are given a level of "-1". Algorithm 6.3 shows the process which is used to add these to the motivation model. Performing this action last means that rectifying motivation frames will always be at the bottom of the list of frames and so will not be counted as an intention's "primary" motivation.

### 6.2.3 Building the Causation Model

The process elements which make up the causation model are generated from the information contained in the motivation model, together which the additional events

**Algorithm 6.2** The algorithm used to calculate and track motivation frames.

```
for each DesireWrapper $d$ do
    $i \leftarrow$ the IntentionWrapper which satisfies $d$
    trackMotivation($i$, $d$, 1, $false$)
end for
for each IntentionWrapper $i$ do
    sort all of the Motivation Frames $i$ belongs to into acending order by $level$
    for each Motivation Frame $f$ which $i$ belongs to do
        if $f.questionable$ then
            remove $f$ from the list
        end if
    end for
end for

function trackMotivation(IntentionWrapper $i$, DesireWrapper $d$, int $level$, boolean $questionable$)
    if $d.addedTime > i.addedTime$ then
        return
    end if
    for each Motivation Frame $f$ which $i$ already belongs to do
        if $f.track = d$ then
            $f.level \leftarrow$ max($f.level$,$level$)
            $f.questionable \leftarrow f.questionable \wedge questionable$
        end if
    end for
    if No frame with $d$ as its track was found then
        $f \leftarrow$ new Motivation Frame
        $f.track \leftarrow d$
        $f.level \leftarrow n$
        $f.questionable \leftarrow questionable$
        add $f$ to the list of IntentionFrames $i$ belongs to
    end if
    for each IntentionWrapper $iW$ in $i.reliesUpon$ do
        $g \leftarrow true$ if all of the required changes made by $iW$ are present in the ground state
        trackMotivation($iW$, $d$, $n+1$, $q \vee g$)
    end for
    for each IntentionWrapper $iW$ in $i.invalidates$ do
        trackMotivation($iW$, $d$, $n+1$, $true$)
    end for
end function
```

**Algorithm 6.3** Generation of "rectifying" motivation frames.

```
for each IntentionWrapper $i$ do
    if any of the changes made by $i$ which are required by other intentions is not in the ground state then
        return
    end if
    $definitely \leftarrow$ an empty set of DesireWrappers
    for each IntentionWrapper $iW$ which $i$ depends on do
        if $iW$ does not belong to any of the same IntentionTracks as $i$ then
            for each IntentionFrame $f$ which $iW$ belongs to do
                $definitely \leftarrow definately + f.frame$
            end for
        end if
    end for
    for each DesireWrapper $d$ in $definitely$ do
        $f \leftarrow$ new IntentionTrack
        $f.track \leftarrow d$
        $f.level \leftarrow -1$
        $f.questionable \leftarrow false$
        $i.frames \leftarrow i.frames + f$
    end for
end for
```

which are known to have happened during the mission. After this, a set of rules are applied into order to build the causal links between these process elements. Finally, stasis elements are created and applied where appropriate.

The first stage of this is carried out by Algorithm 6.4. This creates the *mission started* element, then iterates through all of the intention, desire and meta-desire wrappers in the motivation model, creating all of the related process elements (see Figure 6.3) and giving each the correct time. The *intention started* and *successful* elements are given their *sensor* and *actuator mode change* components during this process. While iterating through the desire wrappers, the desire which was completed last is found and the time at which this happened recorded. The *mission succeeded* is then created and given the time at which the last desire was satisfied. Finally, the *sensor detection* elements are created, based on their analogous events from the bitmap situation model, and each of the appropriate existents and beliefs they added to the perceived world of the AUVs recorded.

After this has been completed, all of the process elements which make up the causation model have been created, but none of the all important connections between them have been found.

The "update causation" function (Algorithm 6.5), given a "cause" process element and an "effect" process element, will correctly update the *caused by*, *contributed to by*, *caused* and *contributed to* lists of each to correctly reflect the new cause/effect

**Algorithm 6.4** The generation of the nodes which make up the causation model. All process elements which are created are recorded.

```
Generate sensor and actuator mode changes from the Platform logs of each AUV
missionStarted ← new Mission Started
missionStarted.time ← 0
for each Intention Wrapper iW in the motivation model do
    added ← new Intention Introduced
    added.time ← iW.added
    started ← new Intention Started
    started.time ← iW.started
    suceeded ← new Intention Successful
    suceeded.time ← iW.suceeded
    for each sensor and actuator mode change mC do
        if mC.agent == iW.agent then
            if mC.time == started.time then
                add mC to added's list of mode changes
            else if mC.time == suceeded.time then
                add mC to suceeded's list of mode changes
            end if
        end if
    end for
end for
lastDesire ← 0
for each Desire Wrapper dW in the motivation model do
    introduced ← new Desire Introduced
    introduced.time ← dW.added
    satisfied ← new Desire Satisfied
    satisfied.time ← dW.satisfied
    lastDesire ← MAX(lastDesire, dW.suceeded)
end for
for each Meta-Desire Wrapper mW in the motivation model do
    satisfied ← new Meta-Desire Satisfied
    satisfied.time ← mW.suceeded
end for
missionSuceeded ← new Mission Suceeded
missionSuceeded.time ← lastDesire
for each Sensor Event in the bitmap situation model do
    sensorDetection ← new Sensor Detection
    sensorDetection.time ← the time of the sensor event
    for each existence added by the sensor Event do
        if existence has an attached location then
            objLoc ← new Object Location based on existent and location
            sensorDetection.implies ← objLoc
        end if
    end for
    for each belief added by the sensor event do
        objectProperty ← new Object Property based on belief
        sensorDetection.implies ← objectProperty
    end for
end for
```

**Algorithm 6.5** The "update causation" function, which is used to update the causal connection between two process elements, taking into account the differing "caused by" and "contributed to by" relationships.

```
Require: $cause$ the process element which is the cause
Require: $effect$ the process element which is the effect

if $effect.causedBy = null$ then
    $effect.causedBy \leftarrow cause$
    add $effect$ to $cause.causes$
else
    // If this potential cause is more recent than the previous cause
    if $cause.time > effect.causedBy.time$ then
        //Religate the old to cause to a contribution
        remove $effect$ from $effect.causedBy.causes$
        add $effect$ from $effect.causedBy.contibutesTo$
        add $effect.causesBy$ to $effect.contributedToBy$
        // Add the new cause
        $effect.causedBy \leftarrow cause$
        add $effect$ to $cause.causes$
    else
        // The candidate acts as a contribution
        add $cause$ to $effect.contributedToBy$
        add $effect$ to $cause.contributesTo$
    end if
end if
```

**Algorithm 6.6** The "bind antecedents" function, which is used find the existences and beliefs which prompted the addition of a particular desire.

```
$p_n$ denotes the $nth$ parameter of $p$

Require: $desire$ the desire for which the antecedents are to be bound

$existences \leftarrow$ empty list of existences
$beliefs \leftarrow$ empty list of beliefs
$metaDesire \leftarrow$ the meta-desire of $desire$
$consequent \leftarrow metaDesire$'s consequent
for $index = 0$ to number of $consequant$'s parameters do
    $parameter \leftarrow consequant_{index}$
    if $parameter$ is variable then
        $existences \leftarrow existences + desire_{index}$
        record that $parameter$ binds to $desire_{index}$
    end if
end for
for each of $metaDesire$'s antecedents $a$ do
    $bound \leftarrow a$, with each of it's variable parameters transformed to constants
        using the previously recorded bindings
    $beliefs \leftarrow beliefs + bound$
end for
    return $existences, beliefs$
```

relationship. In order to generate the causal connections, the "update causation" function is next called in the following ways:

1 As the successful completion of the mission depends on the satisfaction of the meta-desires, it is also called once with each of the **meta-desire satisfied** elements as cause and the **mission successful** element as effect.

2 As the satisfaction of a meta-desire depends on the satisfaction of its sub-desires, it is called once with each **desire satisfied** element as the cause and the relevant **meta-desire satisfied** as the effect.

3 As the satisfaction of a desire depends upon the successful completion of the intention which satisfies it, it is called once with each **desire satisfied** as effect and the **intention successful** elements for the intention which satisfied the desire as cause.

4 As an intention must start in order for it to be successful, it is called once for each intention, with the **intention started** element as the cause and the **intention successful** element as the effect.

5 As an intention must be added to the plan in order for it to start, it is called once for each intention, with the **intention introduced** element as the cause and the **intention started** element as the effect.

6 As it is also required that an intention's requirements should be met in order for it to start, for each intention it is called once for each other intention which it relies upon, with the **intention started** of the first as the effect, and the **intention successful** of the second as the cause. This rule is only applied if the two intentions have at least one *desire track* in common across their *motivation frames*.

7 As the intention which solves a desire is added to the plan in response to the creation of the desire, it is called once with each **desire introduced** element as the cause and the **intention introduced** for the intention which solved the desire as the effect.

8 As additional intentions are added to the plan in order that the requirements of others can be met, for each intention it is again called once for each other intention which it relies upon, but this time with the **intention introduced** of the first as the cause, and the **intention introduced** of the second as the effect. This rule is also only applied if the two intentions have at least one *desire track* in common across their *motivation frames*.

9 If a meta-desire is static (has no antecedents or variable parameters), then its sub-desire (as it would have only one) will not require any additional data to

trigger its creation. In these cases, the "update causation" function is called with the **mission started** element as the cause and the relevant **desire added** as the effect.

10 Alternatively, if a meta-desire is not static, then the creation of its sub-desires depends upon existents and beliefs which are added to the world at the start of the mission, by data gathered by sensors, or both. Thus, a more complex rule must be used. The existents and beliefs which triggered each are first calculated by the "bind antecedents" function (which is formalised in Algorithm 6.6). For each of these, if the information was present in the "ground state" (see section 6.2.1), the "update causation" function is run with the **mission started** element as the cause and **desire added** element as the effect. In this case, the existent or belief is recoded as being a "relevant" component of the ground state. Alternatively, the **sensor detection** which added the data to the world state is found and "update causation" is run with this as the cause and the **desire added** element as the effect.

11 Finally, as sensor events happen as a result of an intention being carried out (as the AUVs are otherwise inactive), for each sensor detection element, the **intention started** element with the highest time less than or equal to the time at which the sensor detection occurred, which was carried out by the same agent as had the detection is found. The "update causation" function is then called with this as the cause and the **sensor detection** as the effect.

Each of these connections can be seen in evidence in Figure 6.4, which shows a simple example causation model. The final stage in the construction of the causation model is the addition of the stasis elements, which add contextually relevant information and exposition to the skeleton created by the process elements. The stasis elements are created and added to the process elements in the following ways:

1 As the AUVs are crucial to the completion of the mission, an AUV explanation element is created for each and added to the *requires* list of the **mission started** element.

2 In tandem with the above, the capabilities which an AUV has *and* actually uses throughout the mission are found. A **capability** stasis element is created for each of these and added to the *requires* list of the **mission started** element. These are placed in the list in order to follow the explanation of the relevant AUV.

3 As the meta-desires are present at the start of the mission and remain constant throughout, a **meta-desire explanation** is created for each meta-desire and added to the *requires* list of the **mission started** process element.

4 As some existents and beliefs have been found to be relevant at the start of the mission (see rule 10, above), an **object location** or **object property** (as appropriate) is created for each and added to the *requires* list of the **mission started** process element.

5 As the first thing which is relevant to the creation of a desire is its meta-desire, for each **desire introduced** a **meta-desire explanation** is created for its meta-desire and added to its *requires* list.

6 As the second thing(s) which are relevant to the creation of a desire are the existents and beliefs which triggered its creation (see Algorithm 6.6), an **object location** or **object property** (as appropriate) is created for each and added to the *requires* list of the **desire introduced** element.

7 In order to give the important details about an intention, **conditions** and **effect** stasis elements are created for each and added to the *introduces* list of its **intention introduced** and **intention started** elements.

8 For each actuator or sensor mode change which is attached to each **intention started**, a **sensor mode explanation** or **actuator mode explanation** (as appropriate) is added to its *introduces* list.

9 An **effect** stasis element is created for each intention and added to the *introduces* list of its **intention succeeded** element. A conditions element is not used in this case, as the conditions are presumed to have already been met.

10 As capabilities play a role in deciding which AUV will carry out which intention, they are considered to be a contextually relevant requirement for an intention. For this reason, for any intention which requires an AUV to have a particular capability to carry it out, a **capability** element is added to the *requires* list of the **intention introduced** and **intention started** elements.

11 The fact that an intention has started implies that it is in progress, thus a relevant **intention in progress** stasis element is added to the *implies* list of each **intention started** element.

12 As a **sensor detection** element is connected to the intention which was in progress and caused it to happen, an **intention in progress** element is created for this intention and added to the *requires* list of the sensor detection.

13 The **object location** and **object property** elements which are linked to a **sensor detection** are added to its *implies* list, as they are stated directly when it is converted to discourse.

It should be noted that at this stage we are working with the content of the mission, rather than its expression. This is the story, rather than the discourse (as defined by Seymour Chatman in *Story and Discourse*[3] and discussed in Section 2.10). This being the case it is not important that this can result in a large amount of stasis elements being added to the causation model, many of which are potentially redundant. The stasis elements are added in every instance in which they might be relevant, and the decision as to whether they should actually be included in the discourse will be made later.

### 6.2.4 Grouping and Simplification of the Causation Model

Next, the causation model is simplified in order to ease the computational burden required to generate an ideal ordering of the process elements into a discourse for delivery to the user. This simplification groups some some process elements together, thereby reducing the number of distinct elements which must be ordered in order to form an effective discourse.

Each process element in the causation model is initially placed in a group of which it is the only member. Next, these groups are merged based on several sets of rules. The first set joins together elements with very strong causal connections based on the satisfaction of desires. In each case, cause → effect ordering is used within the groups created by these rules. They are:

1 The **mission succeeded** element, the **meta-desire satisfied** which caused it, the **desire satisfied** which caused this and finally the **intention successful** which caused this are all grouped.

2 For each **meta-desire satisfied** not covered by the previous rule, it is grouped with the **desire satisfied** which caused it and the **intention successful** which in turn caused this.

3 For each of the **desire satisfied** elements which is not covered by the previous two rules, it is grouped with the **intention successful** which caused it.

Next a set of grouping rules based on strong causal links related to the triggering of the creation of desires and the addition of intentions to the plan is applied:

4. Each element which is capable of causing a **desire introduced** element (the **mission started** element and any **sensor detection** elements) is grouped with any these elements which it does cause, as this is an extremely strong causal link.

5 Each of the **intention introduced** elements is added to the group which contains the **desire introduced** element for its primary motivation. While it is possible for an intention to support the satisfaction of multiple desires, the

constraints which are used to process the motivation of the intentions mean that this will only apply to desires which are created at the same time.

6 **Sensor detection** elements are still generated when the detection has no effect on the mission. As these are guaranteed not to cause any other process elements they represent a dead end in the causation model, but also have a strong causal relationship to the intention which triggered them. For this reason, all **sensor detection** elements of this type are grouped with the **intention started** element which causes them, but only if the intention started element triggers a single sensor detection.

All the simplification rules used so far have applied to very specific elements. The final two, however, are much more general and attempt to provide as much additional simplification as possible. In order to facilitate these rules, the current groups of process elements are first sorted into chronological order, based on the time of the first element in the group. The rules are as follows:

7. It is quite likely that the events which make up the end of the mission are all the result of the satisfaction of a similar desire. As these will be a single causal strand in chronological order, they are grouped. Taking the first element in the group which contains the *mission succeeded* element (which will be an *intention succeeded* element to begin with), if its cause is the last element of the group which immediately precedes it chronologically, this second group is prepended to the first group. This process is then repeated until the condition no longer holds.

8 Finally, if any run of groups represents a minimum distance along all of the situational axes (see section 6.1.3), they are joined. For this to be the case, the following conditions must be met:

- They are chronologically adjacent;
- They all have the same protagonist;
- They do not have differing primary motivations; and
- The last element of each group is the cause of the first element of the next, *and no others.*

This rule is applied in a similar way to rule 7, excepting that it starts at the beginning and works forwards. Starting with the first group, the conditions are checked against the following group (meaning that the first conditions will always be true). If the conditions hold the groups are merged and the test is made again, otherwise the algorithm advances to the next group.

This final rule does not consider the space axis, as if two elements are chronologically adjacent and carried out by the same AUV then it is considered to be highly likely that they will also happen in a similar spatial region.

Rules 4 and 5 have the potential to lead to groups which consist of a data adding element such as the *mission started* element or a *sensor detection* element, one or more *desire introduced* elements and a potentially large number of *intention introduced* elements. It seems clear that the data adding element should come first, but an ordering strategy is required for the other elements. Two core strategies were devised for this purpose:

- **Ungrouped**. In this strategy, all of the *desire introduced* elements are ordered first, followed by the *intention introduced* elements in either chronological or reverse chronological order of the time they were satisfied.
- **Grouped**. In this case the *desire introduced* and *intention introduced* elements are interleaved. Each *desire introduced* is followed by the *intention introduced* elements which have it as their primary motivation, in either chronological or reverse chronological ordering of the time the intention succeeded. The ordering within the *desire introduced* elements is the chronological order of the time they were satisfied. This ordering leads to a cleaner partition of the information. It is, however, less suitable in instances where an intention makes a strong contribution to more than one desire, or there is an interdependency between the plans for two different desires.

Both of these strategies allow the possibility of both chronological or reverse chronological ordering between the intention introduced elements. This leads to four effective ordering strategies. While the reverse ordering more clearly expresses the causal links between the intention introduced elements, it is a less normal ordering to use for communicating to humans, as plans are normally expressed chronologically. As such, both of these options have advantages and disadvantages. For the easiest understanding, the grouped chronological strategy is preferred, however if any interdependency is detected then the strategy is changed to the ungrouped chronological.

For illustrative purposes, Figure 6.5 shows the process elements which make up the example causation model shown in Figure 6.4 in strict chronological ordering. Figure 6.6 then shows the groupings created in the example causation model by applying of the simplification rules rules given here. This results in the number of distinct components which make up the model being reduced from thirty to seven. As a brute force depth first search with no optimisation has an complexity of:

$$O(n!) \tag{6.1}$$

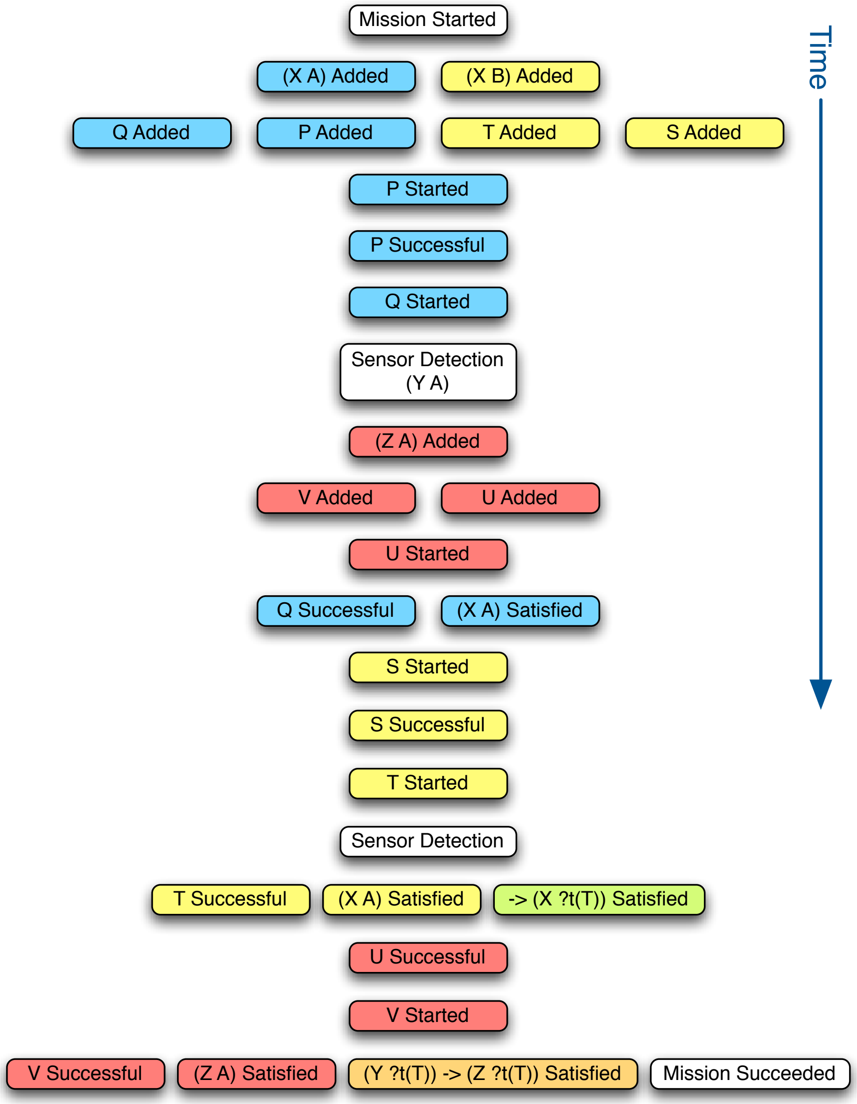


Figure 6.5: The process elements of the example in chronological order.

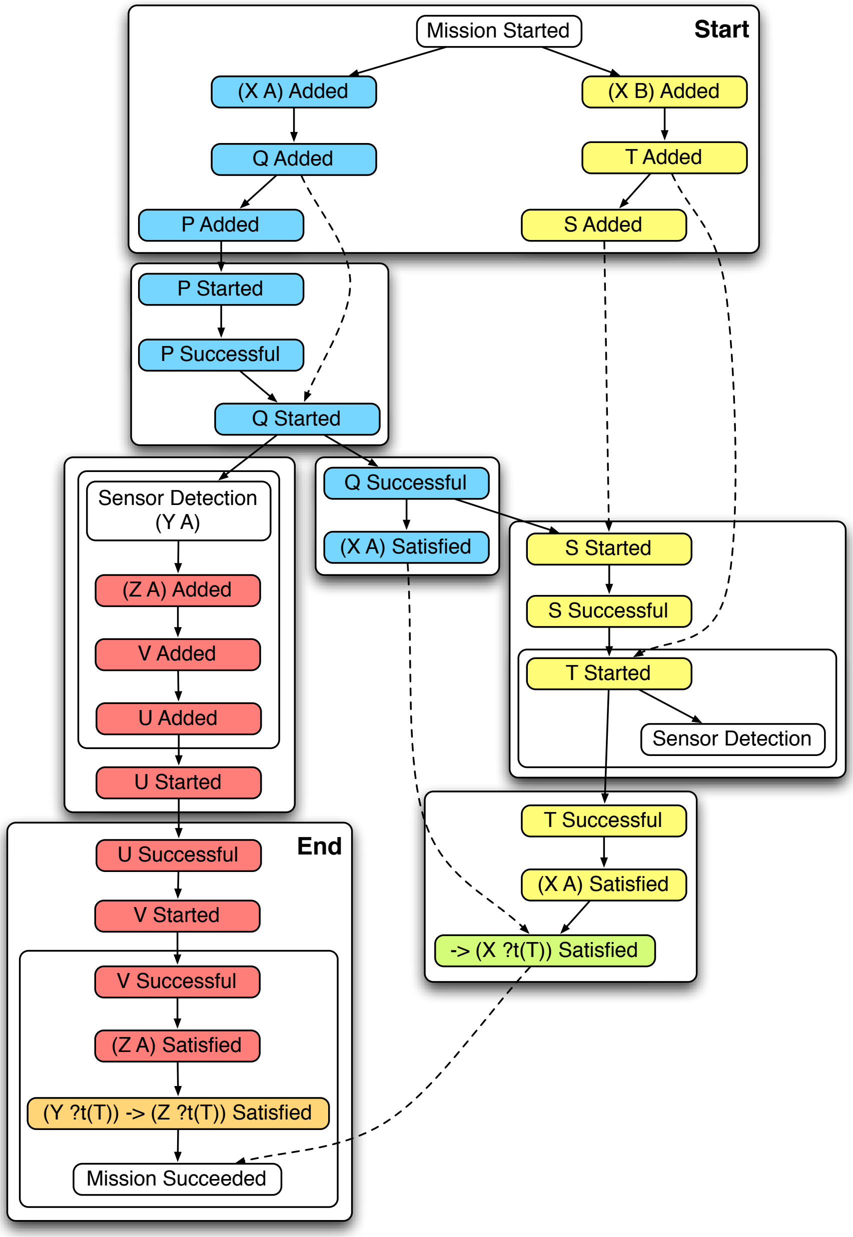


Figure 6.6: Groupings and simplifications of the example causation model.

this has the potential to reduce the complexity from:

$$30! = 2.6525286 \times 10^{32} \tag{6.2}$$

to:

$$7! = 5040 \tag{6.3}$$

Which, for this very small example, give a reduction in complexity of:

$$\frac{30!}{7!} = 5.26295357 \times 10^{28} \tag{6.4}$$

Even though depth first search represents a worse case scenario, this simplification represents a worthwhile reduction in complexity for any search based algorithm.

## 6.3 Summary

This chapter has described the Vector Situation Model (VSM), which represents the mission at a higher level than the Bitmap Situation Model (BSM) described in the previous chapter. Where as the BSM was highly computational in nature, containing every detail of the mission in close to atomic form, the VSM is designed to more closely mimic the levels of representation thought to be found in the human mind. In particular, it represents the flow of motivation and causation within the mission.

As well as describing the structure of the vector situation model, this chapter has also described the algorithms used to generate it from the BSM and infer the connections between its elements, which are smaller in number than those in the BSM, but greater in significance. Although the VSM does contain a smaller number of elements than the BSM, it is still potentially large enough to cause problems of scale when attempting to generate a discourse (which is discussed in the next chapter). This being case, techniques for grouping and simplification were also discussed.

# Chapter 7

# The Discourse

The chapter describes the discourse which is generated from the vector situation model described in the previous chapter and delivered to the user. This covers, in the parlance of the levels of representation discussed in Section 2.10, the formation of initial symbolic representation found in the "text base" and the realisation of this into the "surface code." Of course the form of the discourse which is produced by the *Glaykos* system is not simply textual in nature, but audio visual as well, and these additional components must also be created.

The remainder of this chapter is divided into four sections. The first discusses the problem of, and the methodology used for, ordering the "process elements" which make up the vector situation model's causation model into the most coherent discourse available. The second covers the further development of this skeleton discourse, both through additional simplifications which are applied and the addition of expository information in the form of "stasis elements." This combination results in the creation of the completed text base. The third section describes the methodology used to realise the symbolic text base into the natural language which makes up the surface code. Finally, the fourth section describes the processes which generate the complete audio visual debrief, based on both the surface code and the other information attached to the vector situation model. Figure 7.1 illustrates how these components integrate into the *Glaykos* system as a whole.

## 7.1 Building an Effective Discourse Ordering as an Optimisation Problem

Up until this point we have been entirely focused on creating a situation model which represents the content (or story) of the mission which is to be reviewed. The next stage of the process is to generate an expression (or discourse) of this content, which transforms each of the elements of the causation model into *statements* and delivers these to the user in a particular order.

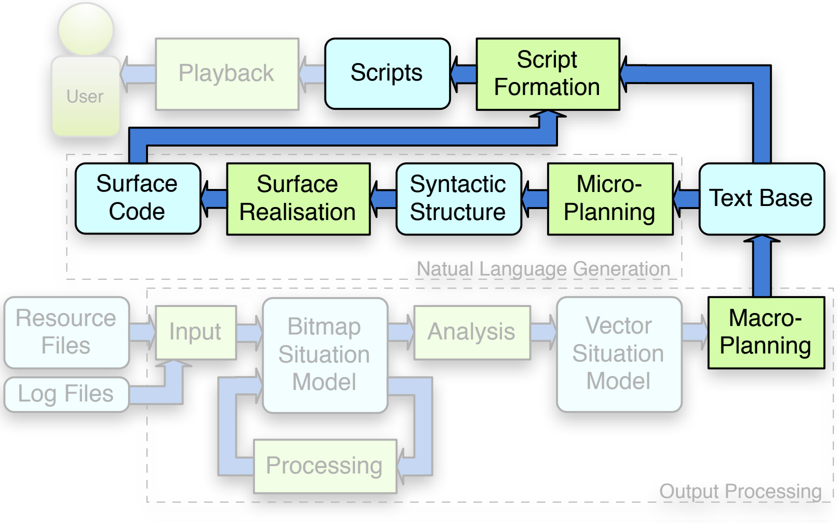


Figure 7.1: The components described by this chapter and how they fit into the structure of the *Glaykos* system (as per Figure 4.15).

Previous literature has shown the benefit of delivering information in the form of a narrative [97, 100, 101]. In order to utilise these benefits we have based the structure of our causation model on process and stasis elements, the building blocks which make up a narrative[3], and positioned these on the five situational axes[106, 107]. To most effectively organise these statements into a cogent discourse they should be ordered in such a way as to reduce the amount of displacement along these axes from statement to statement as far as possible. This task closely resembles a particular class of optimisation problem known as the "traveling salesman problem."

The "traveling salesman problem" is the generalised form of a class of mathematical problems first studied in the 1800s by Sir William Rowan Hamilton and Thomas Kirkman (see [133] for a summarisation of their work). The generalised form which became known as the Traveling Salesman Problem (or TSP), however, was first studied by various mathematicians in the 1930s, most notably Karl Menger. It can be simply stated thus:

> "Given a set of cities along with the cost of travel between each pair of them, the traveling salesman problem, or TSP for short, is to find the cheapest way of visiting all the cities and returning to the starting point." [134]

There exist two classes of TSP, the more general one which has been implicitly specified, known as the Symmetric Traveling Salesman Problem, and a more constrained version, known as the Asymmetric Traveling Salesman Problem. The difference between the two is that in the former the distance between two cities is the same in both directions, where as this is not necessarily the case in the later. TSP is one of the most notorious and studied computational problems of the last

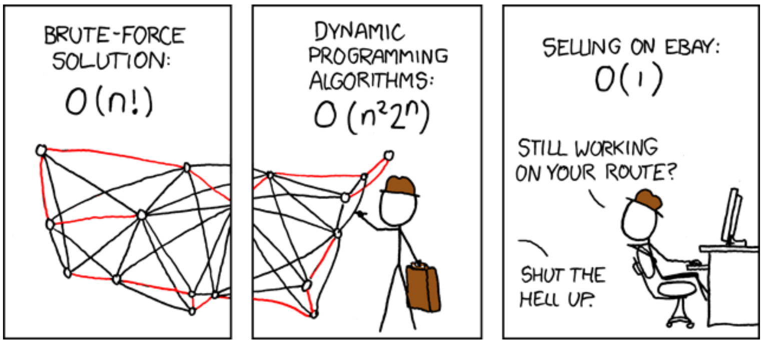


Figure 7.2: The "traveling salesman problem" edition of the XKCD webcomic, published online at http://xkcd.com/399/

century, to the extent that it has even penetrated popular culture (see Figure 7.2 for an example of this). As such a deep investigation of it is outside the scope of this thesis. The following subsection, however, will give some further background to some of the current solutions, both optimal and sub-optimal.

### 7.1.1 Approaches to Optimisation

The simplest method of finding an optimal solution to the problem (in terms of algorithmic complexity) is "brute-force" or exhaustive search. This method consists of generating each potential solution to a problem and testing each, finally outputting the one with the best score as the solution (hence the alternative name "generate and test"). As is often the case with this methodology, the worst case computation complexity for exhaustive search with the TSP is $O(n!)$, or the factorial of the number of cities. This being the case, the time taken to compute the best solution to the problem grows rapidly as the number of cities increases and becomes intractable for even relatively small problems.

The branch and bound algorithm can potentially provide an optimal solution to the TSP with reduced computational complexity. First proposed by Land and Doig in 1960[135] as a solution to linear programming problems, this method simplifies the problem through a "divide and conquer" style method. The complete set is recursively divided into smaller subsets (branching), and then the upper and lower bounds across each of these calculated (bounding). If any set of candidates then has a lower bound which is greater than the upper bound of any other, it can be discarded. Thus, the size of the search space is reduced. In order to be successful, this method requires efficient techniques for both the branching and bounding phases. Further example of the branch and bound technique can be found in [136, 137, 138, 139].

Proposed slightly earlier than this in 1954, the Dantzig-Fulkerson-Johnson, or Cutting Plane, algorithm[140] was originally put forward as a method of solving the TSP for one city in each of the (then) 48 states in the USA. This method forms the basis for the CONCORDE algorithm[141, 134], which is currently the most successful algorithm for solving the TSP, having been successfully applied to a problem consisting of all 85,900 "cities" in Sweden.

Various heuristic based, and sub-optimal, solutions also exist for the TSP. The Nearest Neighbour algorithm (sometimes referred to as a "Greedy" algorithm [142]) was one of the first to yield a solution to the TSP. It consists of the following steps:

1 Stand on an arbitrary vertex.

2 Find out the lightest edge connecting current vertex and an unvisited vertex V.

3 Set current vertex to V.

4 Mark V as visited.

5 If all the vertices in domain are visited, then terminate.

6 Go to step 2.

This can rapidly yield a good solution to the problem, with the computational complexity of $O(n)$ for a single run, and a worst case of $O(n^2)$ for a more exhaustive analysis which tries using each of the cities as the starting point. This method is highly unlikely to produce optimal results, however, and has been shown under some circumstances to actually produce the longest possible route [143, 144, 145].

Several biomimetic approaches to the TSP have also been tried. These include solutions based on simulated ant colonies[146], and genetic algorithms [147, 148, 149, 150]. The latter of these is a very common and effective tool, utilised across a wide variety of optimisation problems.

Genetic Algorithms [151, 152] are a class of general problem solving algorithms which encode potential solutions to a problem as a "genome" and then use the principles of genetics to attempt to "evolve" a solution. An initial population of genomes is generated, often randomly, but potentially seeded with sub-optimal but reasonable solutions produced by other algorithms. Each of these is then evaluated for fitness, and placed in order. A number of the highest scoring genes may then be directly copied into the next "generation" ("cloning"). A further proportion of the next generation is generated by randomly selected pairs of genomes from those with the highest scores and performing a "crossover" operation to form a new genome. The remainder of the next generation is then created by introducing new randomly generated genomes in a similar method to the one used to the create the initial population ("immigration"). Finally, the members of the new generation may be altered by the processes of "mutation" or "improvement". Mutation makes random changes to genomes based on a predefined statistical likelihood. Together with immigration, mutation encourages variation in the gene pool and helps ensure that it does not "stagnate." Improvement[1], on the other hand uses heuristic techniques to

[1] The analogously correct term here is probably "eugenics," however this is a terminology which is probably best avoided, and so "improvement" is used in its place.

attempt to make definite improvement to each genome in the new generation. This process is then repeated, either until a set number of generations has been reached, or the fitness of the population has plateaued or reached a pre-set target.

The particular form of the traveling salesman problem which we are using as an analogy for the creation of our discourse is both asymmetric and "ordered." That is, the distance to a new "city" does not depend just on the previous city, but also every other in the route which has been taken so far. The reason for this will be explained in the Section 7.1.2. For this reason, all "divide and conquer" based techniques, including the highly successful "branch and bound" algorithm are inapplicable. Cutting plane techniques are again very successful, but rely upon the ability to generate and evaluate essentially invalid solutions at times, which is not feasible with our problem. The greedy algorithm satisfies all of the constraints required to successfully evaluate potential solutions, and is also capable of quickly generating results. However it has been shown to be capable of producing highly sub-optimal output, and due to its nature is likely to produce solutions with large gaps between cities towards the end.

Genetic algorithms function by evaluating complete and valid solutions and have been shown to be able to produce good solutions to traveling salesman problems, along with many other optimisation problems. Amongst the problems they have been applied to, they have also been trialled for the purposes of generating narratives[153] and the ordering of textual descriptions [154]. For these reasons a genetic algorithm based implementation was created in order to generate an ordered discourse from the elements of the causation model.

While unconstrained depth first search is known to quickly become intractable due to its factorial increase in complexity, it was thought that the addition of domain specific constraints could reduce its complexity significantly enough that it would be capable of producing results in a reasonable amount of time. This technique has the advantage of knowing that if the algorithm completes it has found the optimal solution, which is not the case with a genetic algorithm. For this reason an implementation using this technique was also created.

### 7.1.2 Fitness Function

Previous work[97] has found that an increase in reading time, and an associated decrease in the ease of comprehension occurs when there is a discontinuity on one of the five situational axis. It is stated in [106] , that this occurs under the following circumstances:

1 **Spatiality**. The action or event expressed in the Statement is in a different spatial region from the content in Working Memory.

2 **Temporality**. The action or event expressed in the Statement involves a gap

or shift in the chronological timeline.

3 **Protagonist**. The protagonists in the Statement are not among the protagonists in Working Memory.

4 **Motivation**. The action expressed in the Statement is not part of an agent's plan in Working Memory.

5 **Causality**. The action or event expressed in the Statement does not causally flow from the content in Working Memory.

The function used to asses the fitness of a complete or partial discourse is based on these rules, and so specifies a quantitative assessment of each. For each axis, the function used will be stated, both in plain English and formal terms. The formal definitions share the following common notation:

$n$ is the index of the element to which the distance is being calculated

$e_i$ is the element in the discourse with index i

$d[i:j]$ is the subset of discourse starting from index i and ending at index j

$f_{name}$ is a named constant which acts as a factor

$p_{name}$ is a named constant which acts as a fixed penalty

Given the continuous model of space used here, the simplest and most easily quantifiable measure of the spatial discontinuity between two events is the Euclidean distance between the locations at which they occurred. This is used, scaled by a defined factor and given a defined upper bound for tuning purposes. In the event that an element does not have a position attached to it (as is the case with the mission started and successful process elements, for instance) then the distance in this axis is taken to be zero, as the event is considered to happening everywhere. Should the previous element not have a position attached to it, the position of the most recent element which does have an attached position is used.

$$S_n = \begin{cases} |position_n - position_y| * f_{space} & \text{if } e_n \text{ has a position} \\ 0 & \text{otherwise} \end{cases} \tag{7.1}$$

*where* :

$S_i$ is the distance along the space axis to the element with index i

$y$ is the index of the most recent element with a positon

$position_i$ is the postion of the element in the discourse with index i

While time, being one dimensional by definition, would appear to be simpler even than space for the purposes of measuring distance, this is not necessarily the case. Where as humans are quite used to being able to move with a reasonable amount of freedom in the three dimensions of space, this is not the case with time. In general humans are used to time moving at a specific rate, and more importantly in a specific direction, namely: forwards. Distance in time is defined here to be primarily the difference between the times of the two events in seconds. However, different scaling factors are used for forwards and backwards directions of time, and an additional fixed penalty is added when events are in reverse chronological order. In both cases, the maximum value is given an upper bound.

$$T_n = \begin{cases} 0 & \text{if } t_{n-1} = t_n \\ (t_n - t_{n-1}) * f_{timeForwards} & \text{if } t_{n-1} < t_n \\ (t_{n-1} - t_n) * f_{timeBackwards} + p_{timeBackwards} & \text{if } t_{n-1} > t_n \end{cases} \tag{7.2}$$

$where:$

$T_n$ is the distance along the time axis to the element with index i

$t_i$ is the time of the element in the discourse with index i

Unlike the quantitative measures used for the space and time axes, the measure used for the protagonist axis is highly qualitative. We use four possible cases for the distance on the protagonist axis, namely:

1 The new element has the same protagonist as the previous element;

2 The new element does not have the same protagonist as the previous element, but the protagonist has already featured in the discourse;

3 The new element does not have the same protagonist as the previous element, and the protagonist has not already featured in the discourse;

4 The new element does not have a specific protagonist.

Different penalties are provided for the second and third cases, and the distance in the first and last cases is taken to be zero.

$$P_n = \begin{cases} p_{knownProtagonist} & \text{if prot}_n \neq \text{prot}_{n-1} \wedge \text{prot}_n \in \text{d[0:n-1]} \\ p_{newProtagonist} & \text{if prot}_n \neq \text{prot}_{n-1} \wedge \text{prot}_n \notin \text{d[0:n-1]} \\ 0 & \text{otherwise} \end{cases} \tag{7.3}$$

$where:$

$P_n$ is the distance along the protagonist axis to the element with index i

$prot_i$ is the protagonist of the element in the discourse with index i

The motivation axis is again considered to be qualitative, but with the added possibility of overlap between values. Motivation is tracked using motivation frames, of which a process element may have any number. In addition, motivation frames refer to the desires an intention is contributing to the satisfaction of, but desires form a hierarchy with meta-desires, so if two intentions are not directly contributing to the same desire, they may still both be contributing to a meta-desire.

Motivation frames also carry a "level" indicating the order of the intentions within the plan. If the two elements have sequential levels for the same desire, they are then considered to be more directly related to each other in terms of their motivation than if they had non-sequential levels. Any process element referring directly to a desire is considered to have a motivation frame for that desire with a level of zero. Those which directly relate to a meta-desire are considered to have a motivation related only to this meta desire.

Given this application of motivation to the process elements, we consider there to be five distinct cases which define the potential difference in motivation of two elements:

1 One or both of the elements does not have a specific desire;

2 Both elements have a motivation frame which refers to the same desire, and the levels of the frames are sequential;

3 Both elements have a motivation frame which refers to the same desire, but the levels of the frames are not consecutive;

4 The elements do not have motivation frames which relate to common desire, but do have motivation frames which refer to desires which have the same meta-desire;

5 The elements have no commonalities across their motivation frames.

Distance along the motivational axis is considered to be zero in the first two cases, and differing penalties are provided for the final three.

$$M_n = \begin{cases} p_{noneConsequative} & \exists\text{x,y: x} \in m_n \wedge \text{y} \in m_{n-1} \wedge d_x{=}d_y \wedge |l_x\text{-}l_y| \neq 1 \\ p_{differentDesire} & \exists\text{x,y: x} \in m_n \wedge \text{y} \in m_{n-1} \wedge d_x \neq d_y \wedge \text{m}d_x{=}\text{m}d_y \\ p_{differentMetaDesire} & \forall\text{x,y: x} \in m_n \wedge \text{y} \in m_{n-1} \wedge d_x \neq d_y \\ 0 & \text{otherwise} \end{cases} \tag{7.4}$$

*where* :

$M_n$ is the distance along the motivation axis to the element with index i

$m_i$ is the set of motivation frames for the element with index $i$

$d_f$ is the desire motivation frame $f$ refers to

$l_f$ is the level of motivation frame $f$

$md_f$ is meta-desire of the desire motivation frame $f$ refers to

The final situational axis is causation, which has been found to be one of the most important in much of the literature:

> "Most researchers would claim, as we do, that the motivational and causal dimensions form the backbone of situations constructed during narrative comprehension." [107]

In particular, it is important for the purposes of comprehension that cause must precede effect in the discourse. For this reason we attach a penalty (which should ideally be large) when a process element is added to a discourse which does not already contain both the element which is considered to have caused it, and all of those which are considered to have contributed to it (see Section 6.1.2 for a description of these relationships). Aside from this penalty, we consider the causation axis to be essentially quantitative. Unlike the other quantitative axes, the measure is made *within* the discourse, however. Rather than a measure based on the position of the new element and the previous element in the causation model, instead we use the number of process elements which separate the new process element from its cause in the discourse, and scale this by a defined factor.

$$C_n = \begin{cases} p_{notAllContributions} & \text{if } ca_n \notin \text{d[0:n-1]} \vee (\exists\text{x: x} \in co_n \wedge \text{x} \notin \text{d[0:n-1]}) \\ (n - i(ca_n)) * f_{cause} & \text{otherwise} \end{cases} \tag{7.5}$$

*where* :

$C_n$ is the distance along the motivation axis to the element with index i

$ca_n$ is the element which is the cause of the element with index $n$

$co_n$ is the set of elements which contribute to the element with index $n$

$i(e)$ the index in the discourse of element $e$

| Gene value | 0.42 | 0.06 | 0.38 | 0.48 | 0.81 |
|---|---|---|---|---|---|
| Decodes as node | 3 | 1 | 2 | 4 | 5 |

Table 7.1: Example "random key" gene encoding.

The distance from the previous discourse to a particular element is than calculated by summing all of these values, thus:

$$distance_n = S_n + T_n + P_n + M_n + C_n \tag{7.6}$$

The complete fitness function for a complete discourse is calculated by summing this value for each element. So for a discourse with $n$ total elements, it is calculated thus:

$$fitness = \sum_{i=0}^{n} distance_i \tag{7.7}$$

Alternatively, as this value is generated in an ordered element by element fashion, it permits the calculation to be made in steps. This enables the generation of a discourse in an iterative fashion, with the evaluation of the distance to each new element being performed as it is added.

The actual values which were used for each of the constants referred to here can be found in Appendix D.

### 7.1.3 Genetic Algorithm Implementation

The genetic algorithm implementation used here for optimisation is the one published in *A Random-Key Genetic Algorithm for the Generalized Traveling Salesman Problem*[150]. The methodology was selected as it provides a general solution for traveling salesman problems and additionally is well enough documented to ease the development of a viable implementation. Other well documented methodologies (such as the general approach for ordering problems detailed in [152]) were trialled, but the approach detailed in [150] was experimentally found to give better results.

This approach uses "random keys" for gene encoding. This technique assigns each gene a float drawn from the interval $[0, 1]$. These are then decoded into references to the actual nodes of the problem by visiting them in ascending order of the genes. An example of this is shown in Table 7.1.

Given this methodology for gene encoding, an initial population can easily be created by repeatedly generating a list with a sufficient quantity of random numbers. In addition to the randomly generated values which make up the majority of the initial population, we also insure that a chronological ordering of the elements is included. This ensures that a good general candidate solution is already present when the algorithm begins, and so the algorithm can only improve upon this.

| Parent 1's Genes | 0.17 | 0.23 | 0.04 | 0.76 | 0.63 |
|---|---|---|---|---|---|
| Parent 1's Nodes | 2 | 3 | 1 | 5 | 4 |
| Parent 2's Genes | 0.53 | 0.96 | 0.44 | 0.57 | 0.99 |
| Parent 2' Nodes | 2 | 4 | 1 | 3 | 5 |
| Crossover Variable | 0.47 | 0.64 | 0.91 | 0.42 | 0.82 |
| Child's Genes | 0.17 | 0.96 | 0.44 | 0.76 | 0.99 |
| Child's Nodes | 1 | 4 | 2 | 3 | 5 |

Table 7.2: Worked example of "parameterized uniform crossover".

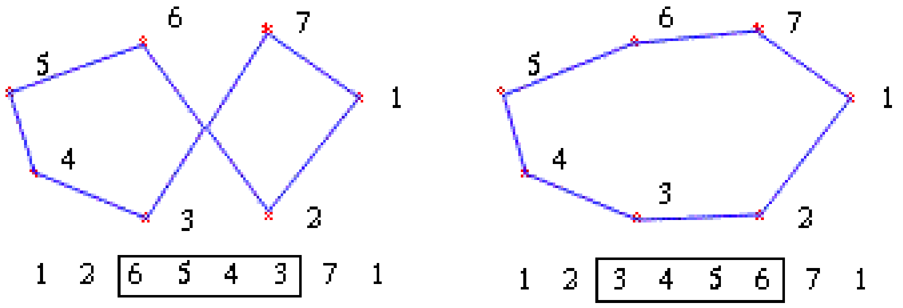


Figure 7.3: Example of "inversion" applied to a Euclidean TSP.

"Cloning" is used in order to ensure that the best candidates from each generation are always maintained. At the beginning of each cycle, the best solutions from the previous generation are copied directly into the new generation a pre-set number of times.

The random key technique allows the use of standard crossover operations to be employed, while making it almost certain that no invalid solutions will be generated as a result. The crossover strategy known as *parameterized uniform crossover*[155] is used. This approach randomly selects two parents from the best candidates currently in the population and generates a child by randomly choosing (with predefined weighting) either the first or second parent's value for a particular gene. The selection could be performed by using a random variable taken from the interval $[0, 1]$, if the value is lower than the pre-defined weighting the gene is taken from the first parent, otherwise it is taken from the second. An example of this is shown in Table 7.2.

The implementation eschews mutation as a method of ensuring variation in favour of immigration. This introduces a small number of new, randomly generated chromosomes into the population in each generation.

As a final step, a local improvement heuristic is used to add an additional possibility of improvement to the genetic algorithm. This "hybrid" approach of combining the guided random search of a genetic algorithm with a heuristic component has been proven to be highly successful in the past (see [156] for an example). As with

[150], the "2-opt" technique is used for this purpose. This technique attempts to find two edges of the tour that can be removed, and two edges that can be inserted in order to obtain a single tour with a lower cost than the original. The implementation used here operates by randomly selecting a section of the chromosome and "inverting" it. In the case of Euclidean TSPs, this has the potential to "straighten" a section of the candidate which "crosses" itself, and therefore create a more optimal solution (See Figure 7.3). Even though this TSP is very clearly not Euclidean in nature, this technique has been experimentally found to have a very good chance of finding improved solutions, and is computationally cheap to apply. This tactic is applied multiple times to a single chromosome in an attempt to perform as many improvements as possible.

Cloning, crossover and finally immigration are performed to produce a new generation. Each member of the population is then evaluated to find its fitness value. Improvement is then performed across the section of the population generated via crossover or immigration, re-evaluating each chromosome as it occurs. This process is repeated until either the best fitness value reaches a plateau and remains static for a pre-set number of generations, or a pre-set time limit is exceeded.

The input values for the genetic algorithm were found by testing it against a simpler problem, which allowed the rapid trial of multiple configurations. For this a simple asymmetric TSP was constructed using fractal landscapes (to provide a height map) and randomly placed cities. Given a starting city $a$ and destination city $b$, the distance between them was computed using a function which imposes an additional penalty for traveling "up hill," thus:

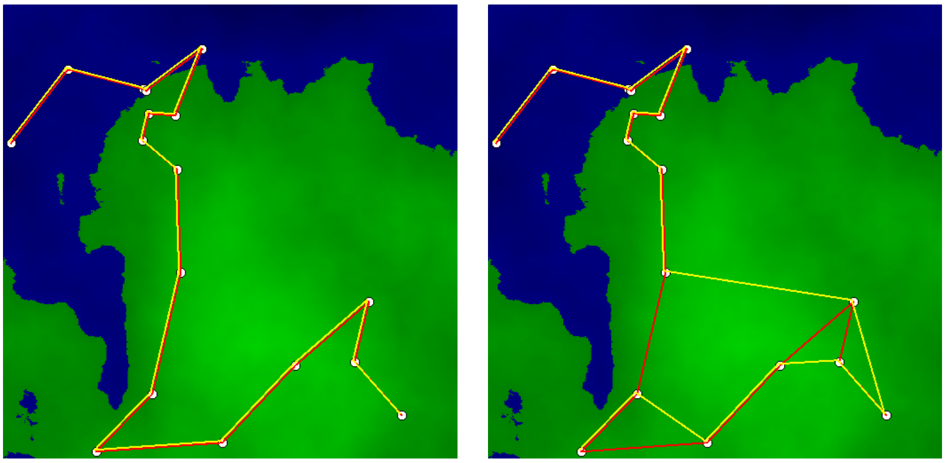
$$distance(a,b) = \sqrt{(x(a)-x(b))^2 + (y(a)-y(b))^2} + max(0, z(b)-z(a)) \tag{7.8}$$

For problems with a reasonably small number of cities, bounded depth first search was used to find the optimal solution for comparison purposes. Figure 7.4 shows some of the results from this. As can be seen from Figure 7.4a, under some cases the algorithm was found to produce slightly sub-optimal results, but this is expected from an algorithm which relies upon randomised search, such as genetic algorithms.

The actual parameters which were used in practice for the genetic algorithm are given in Appendix D.

### 7.1.4 Bounded and Constrained Depth First Search

In addition to the genetic algorithm, a bounded and constrained depth first search based implementation was also trialled as a potential technique for producing an



(a) Optimal solution found.

(b) Sub-optimal solution found.

Figure 7.4: Examples paths produced for simple asymmetric traveling salesman problem used for tuning of the genetic algorithm. Red lines indicate optimal solution produced via bounded depth first search, yellow lines indicated paths produced by the genetic algorithm.

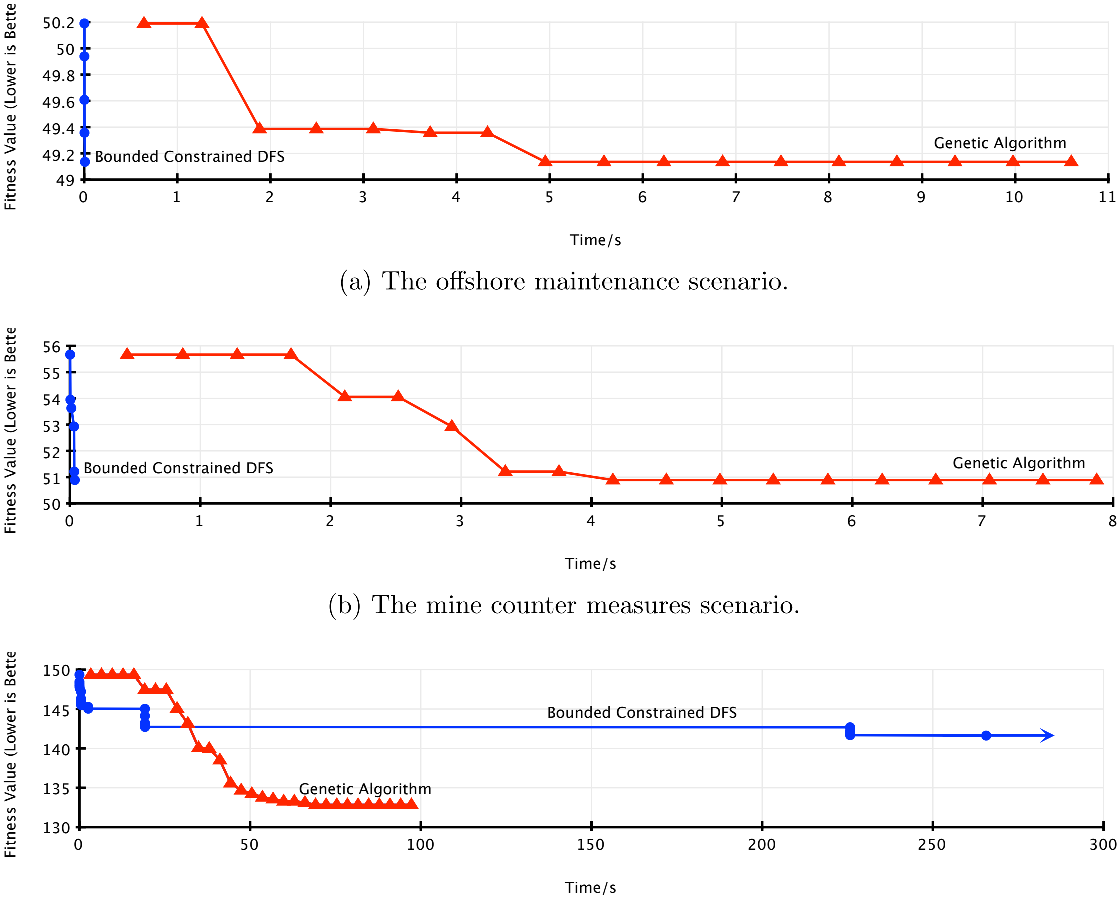
**Algorithm 7.1** The algorithm used to check for validity when adding a new element to the discourse using bounded constrained depth first search.

```
function isValid(PartialSolution $solution$, ProcessElement $element$)
    if $element.causedBy \notin solution$ then
        return $false$
    end if
    for each ProcessElement $contrib \in element.contributedToBy$ do
        if $contrib \notin solution$ then
            return $false$
        end if
    end for
    return $true$
end function
```

optimal discourse. While unmodified depth first search grows in complexity at a factorial rate and quickly becomes intractable, our implementation applies two techniques in an attempt to reduce this, and hence the run time of the algorithm.

The first of these is a bounding constraint. The solution is generated iteratively and evaluated at each stage. A record of the current best complete solution is maintained, and should any potential solution exceed this it is immediately discarded. This means that sub-optimal solutions can be rejected early without needing to construct and evaluate the entire discourse.

The second technique uses domain specific knowledge to disallow "invalid" solutions and ensure they are rejected as quickly as possible. This constraint only allows a *process element* to be appended to the discourse if its cause and all of the elements which contribute to it are already present (see Section 6.1.2 for a description of these relationships). A more formal description of this is given in Algorithm 7.1. This



(a) The offshore maintenance scenario.

(b) The mine counter measures scenario.

(c) The combination scenario.

Figure 7.5: Time taken to provide find a good solution with the bounded constrained depth first search and genetic algorithm based optimisation approaches for each of the mission described in Section 8.

directly excludes solutions which would otherwise carry the large penalty which is given by the fitness function when this constraint does not hold true. As with the bounding constraint, this condition is tested after each new element is added to the discourse.

It is ensured that the first discourse evaluated by this method consists of the (grouped) elements of the causation model in chronological order. This is known to be a reasonable solution, as the distance travelled along the chronological axis will be minimised, and all elements will be guaranteed to be placed later than their cause in the discourse. Beginning with a good solution in this way ensures that a maximum number of sub-optimal solutions will be rejected by the bounding constraint.

### 7.1.5 Optimisation Results

The graphs in Figure 7.5 shows the time required by the two optimisation algorithms to find solutions for the three scenarios which are described in Chapter 8. In the two simpler scenarios (offshore maintenance and mine counter measures), the **B**ounded

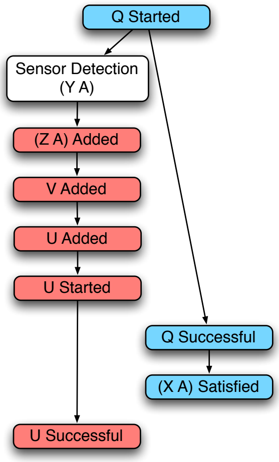


Figure 7.6: Chronological ordering of a subset of the example events from Figure 6.4. Ordering moves down the page.

**C**onstrained **D**epth **F**irst **S**earch (BCDFS) algorithm finds the optimal solution almost instantly, whereas the **G**enetic **A**lgorithm (GA) takes in the region of ten seconds stabilise on the same solution. In the significantly more complicated "combination" scenario, the GA requires almost two minutes to stabilise on a good solution, which represents a significant increase in run time. After running for two and a half times this time the BCDFS algorithm is still running, however, and has found a considerably less optimal solution.

These results show quite clearly that the fixed costs associated with the GA are considerably higher than the BCDFS, but the BCDFS increases in complexity much faster as the number of elements to be ordered increases. It should be noted that another factor which affects the performance of the BCDFS is the level of parallelism in the mission, as the constraints which are applied in an attempt to reduce its complexity are most effective when there are a minimum of distinct causal strands.

Clearly both solutions have merits, and are likely to be most effective under differing circumstances. The BCDFS provides near instant solutions to smaller missions, or larger ones with a smaller degree of parallelism. The GA, on the other hand, can be relied upon to find a good solution in most cases, but always requires a reasonable amount of time in order to do so. Performing some simple introspection on the nature of the causation model would likely provide a basis for the selection of the optimisation algorithm.

### 7.1.6 Potential Drawbacks of the TSP Model for Discourse

The traveling salesman based model used here to construct the discourse delivered to the user inherently ensures that each event is only expressed to the user once. This guarantees that needless repetition is avoided, but potentially beneficial redundancy is also avoided. This subsection will describe the possible issues raised by this.

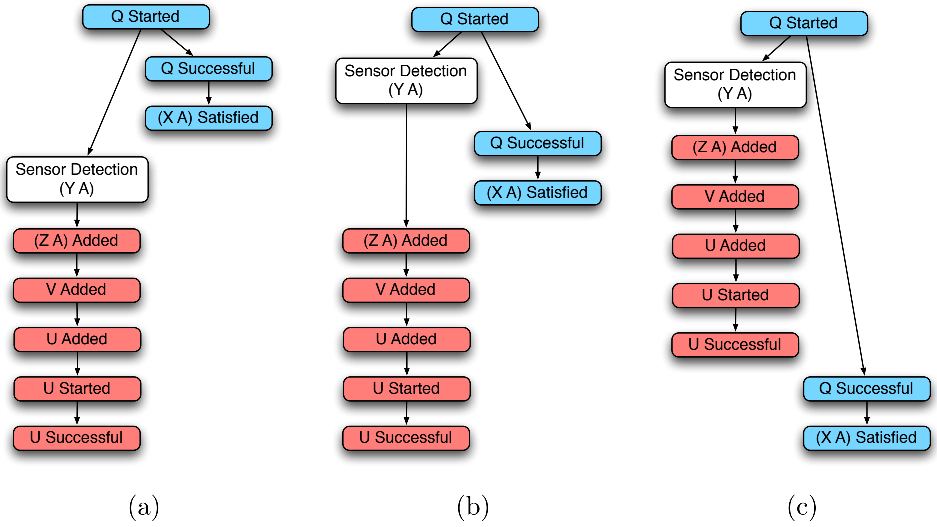


Figure 7.7: Alternative causation based orderings of the example events from Figure 7.6. Ordering moves down the page.

Figure 7.6 presents a subset of the events from the example causation model given in Figure 6.4, arranged in a chronological ordering (moving down the page). As can be seen in the diagram, breaks in both the causation and motivation of the narrative are created by this ordering. The user's attention is switched to a series of events which flow causally, but follow an entirely different motivation. Midway along this series of events, however, the user's attention is then switched back to the original motivation track, additionally breaking the chain of causation. Shortly afterwards, both motivation and causation are broken again as the user is returned to the second motivation track. While this removes breaks in the chronology of the events, the flow of motivation and causation is broken several times by this ordering of events. In the model used here, as well as that used by others (see Section 7.1.2), these latter two of the five situational axes (see 6.1.3) are considered to be the most important.

Figure 7.7 provides three alternative orderings of events which priorities causation and motivation over chronology. In the ordering shown in Figure 7.7a, the first motivation track is followed through to its conclusion, before the timeline is rewound to the point at which the sensor detection occurred. This is then followed through to the second motivation track. This minimizes breaks on both the motivational and causal axes, however it initially leaves the user with the misleading impression that no sensor detection occurred during the implementation of intention Q.

In the ordering shown in Figure 7.7b, the sensor detection is given to the user in situ with the enaction of the intention from which it resulted. This makes the consequences of the intention Q clear to the user, and again leeds to minimal breaks in both causation and motivation (since the sensor detection does not belong to a specific motivation track). However, the events which make up the second motiv-

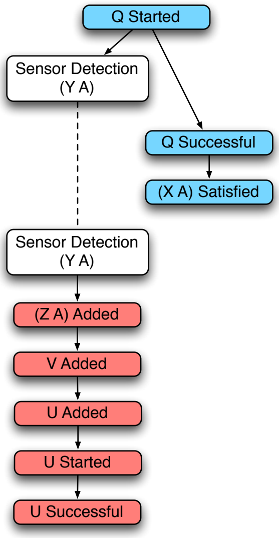


Figure 7.8: Causation based ordering of the example events from Figure 6.4 with repetition of pivotal event. Ordering moves down the page.

ation track are now separated from the sensor detection which caused them. This could be considered to be a particularly important causal connection, as it results in a direct change to the plan of the autonomous assets.

The ordering given in Figure 7.7c reunites the sensor detection with the desire and intention added events which it causes, but also results in the largest potential disruption to the causation and motivation of these three orderings.

Clearly none of these orderings is completely optimal, and previous work, such as that by Walker[157] has shown that humans expect some repetition where appropriate, as this can help avoid excessive cognitive demands.

Figure 7.8 features an alternative ordering of events in which the sensor detection is repeated. In this case, breaks in causation and motivation and minimised, while it is also made cleat to the user that the enaction of intention Q did result in a sensor detection. Furthermore, the additional of a new desire and the subsequent changes to the plan are not divorced from the event which caused them. So long as the user is informed that the sensor detection will be returned to (otherwise it is implied to have had no effect), this ordering has the potential to lead to the lowest cognitive load of all those considered. The creation of an of an ordering such as this is unfortunately impossible with the currently employed TSP based model. The TSP based model provides a useful simplification, though, which can act as a stepping stone to more complex models and techniques. One potential extended model which could be used to produce orderings with redundant utterances is discussed in Section 11.4.7, whilst some of the artifacts created by the current system are discussed in Chapter 8.

## 7.2 Further Discourse Development

After the process elements have been ordered to form the basic skeleton of the discourse, several additional operations are performed in order to complete this process. The first set of operations perform some additional simplifications and optimisations to reduce the amount of redundant information which is communicated to the user. One component of this is only possible after the process elements have been placed in order, hence the fact that it is performed at this stage. Secondly, stasis elements are added to the discourse in order to provide additional contextually relevant exposition.

The following subsections explain these processes in detail.

### 7.2.1 Further Simplifications

Some further processing is performed on the discourse in order to reduce the amount of redundant information and initial exposition which is fed to the user.

By default, each intention is mentioned to the user at least three times. One for each of when it is introduced, started and successful, and potentially again if the user told that it is “in progress”. In order to reduce this repetition, and ensure that the conditions and effect of the intention are communicated at the most relevant time (see the next section for an explanation of the mechanism behind this), a simplification is performed which replaces a set of *intention introduced* elements with a *plan introduced* element. This replacement is performed only if all of the *intention introduced* elements which refer to intentions whose primary motivation is to satisfy a particular desire (calculated using the motivation frames) are contiguous in the discourse.

The process described in Section 6.2.3 leads to the *mission started* process element having a large number of stasis elements attached to it. As is it always the first element in the discourse, the methodology described in the next section will lead to all of these always being delivered to the user. This leads to a large amount of exposition being delivered at the beginning of the mission. Even though all of this is relevant in the context of the causation model, in terms of the discourse produced it leads to an excess of stasis information being given to the user before any of the events of the mission have been revealed.

For this reason, in order to produce a more balanced discourse, each of the *AUV explanation*, *object property*, *object location*, and *capability* stasis elements is removed from the requires list of the mission started element. This leaves just the *meta-desire explanation* elements, meaning that the purpose of the mission will revealed at the beginning, but the additional information which causes the atomic desires to be instantiated will be revealed as it becomes relevant. In addition, the details regarding the nature and abilities of each AUV are also revealed as they become relevant.

### 7.2.2 Addition of Stasis Statements

As previously noted, all process elements have every potentially relevant stasis element attached to them. In practice the vast majority of these will be redundant, however, and so a heuristic is used to add only those which are judged to be required.

In general, when a process element is added to the complete discourse, it is first preceded by each of the stasis elements in its *relies upon* list, and then followed by each of the stasis elements in its *introduces* list. These are filtered, however, based on two principles. Firstly whether the information contained in the stasis element is currently "primed" in the mind of the user, in which case they do not need to be reminded of it. Secondly, the total number of times the information has been actively primed during the discourse. If this second value is above a pre set limit, then the user is considered to have learnt the information, and so does not need to be reminded of it.

In psychology, "priming" refers to the incidence of an earlier stimulus affecting the response to a later stimulus, due to the information from the first stimulus' recency in the subject's working memory. As an example (taken from [109]), subjects who read the two sentences:

1 Mark poured the bucket of water on the bonfire.

2 The bonfire went out.

Are likely to answer the question "Does water extinguish fire?" more quickly than those who have read the following two sentences:

1 Mark placed the bucket of water by the bonfire..

2 The bonfire went out.

For our purposes, we use the term "primed" to refer to information which is currently active in the user's working memory. We consider an element to be primed when an element which implies the same information is delivered to the user, or alternatively when a process element which relies upon the same information is. The mechanism behind the "implies" relationship is discussed in Section 6.1.2. An element is considered to remain primed for the delivery of a further number of elements, which is defined externally (see Appendix D for the value used in practice). The process is formalised in Algorithm 7.2.

**Algorithm 7.2** The algorithm used to add stasis elements to the discourse, whilst avoiding giving redundant information to the user.

```
Require: $skeleton$ the skeleton discourse, containing just the process elements
Require: $discourse$ the initially empty complete discourse
Require: $memoryRegisters$ the number of registers the user is assumed to have in
    their working memory
Require: $maxPrimes$ the number of times the user is assumed to be needed to be
    reminded of a particular piece of information before they are considered to have
    learnt it

for each Process Element $process$ in $skeleton$ do
    addStasisElements($process.reliesUpon$)
    $discourse \leftarrow discourse + process$
    addStasisElements($process.introduces$)
end for

function addStasisElements(StasisElement[] $elements$)
    for each Stasis Element $stasis \in elements$ do
        $primedCount \leftarrow 0$
        $isPrimed \leftarrow false$
        for each Element $element \in discourse$ do
            $index \leftarrow$ the index of $element$ from the end of the discourse
            if $element$ implies $stasis$ then
                $primedCount \leftarrow primedCount + 1$
                if $index \leq memoryRegisters$ then
                    $primed \leftarrow true$
                end if
            else if $element$ is a Process Element $\wedge stasis \in element.reliesUpon$ then
                $primedCount \leftarrow primedCount + 1$
                if $index \leq memoryRegisters$ then
                    $primed \leftarrow true$
                end if
            end if
        end for
        if $\neg primed \wedge primedCount < maxPrimes$ then
            $discourse \leftarrow discourse + stasis$
        end if
    end for
end function
```

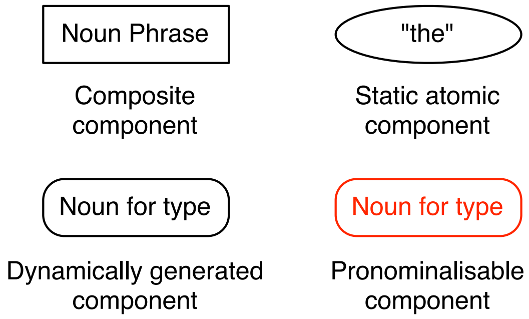


Figure 7.9: The legend for the syntax diagrams used in the following subsections.

## 7.3 Realisation of Natural Language

The next step in the creation of the discourse is the conversion of the process and stasis elements which make up the current representation into natural language statements. This process is carried out using Ehud Reiter's SimpleNLG[90] library, which is itself based on the principles laid out in his own (with Robert Dale) *Building Natural Language Generation Systems*[89].

The following subsections describe firstly the basic natural language structures which are passed into the system by the input files, then the potential for pronominalization, followed by the more complex fundamental structures representing beliefs, desires and intentions, and then finally the construction of the process and stasis statements. These sections contain many syntax diagrams, the legend for which can be found in Figure 7.9. It should be noted that these diagrams are based on the structures and relationships employed by the SimpleNLG library, and not those used traditionally in linguistics.

### 7.3.1 Basic NLG Structures

Just as the previous steps in the generation of the discourse required domain information in order to correctly process their input, so too does the natural language generation system. This natural language data is fused with the other domain information and supplied as part of the same XML input files which are used at every other stage of the *Glaykos* system (see Appendix B). Within these files, the following natural language information is included in the various structures in the world and module definition files:

- **Existent nouns**. These are nouns which are attached to a particular existent in the world definition. In particular, these provide the names for the AUVs themselves and the other objects in the world.

- **Type nouns**. These are generalised nouns which are attached to type definitions, for use when an existent of the type does not have a noun attached to it.

- **Type prepositions**. These are prepositions which are attached to particular type for use when an existent of this type is used as a modifier.
- **Predicate verbs**. These are verbs which are attached to predicate definitions for use when generating a verbal representation of either a belief or a desire from the predicate.
- **Belief verbs**. These are verbs which are attached to predicate definitions for the purpose of generating a specific verbal representation of a belief from the predicate.
- **Desire verbs**. These are verbs which are attached to predicate definitions for the purpose of generating a specific verbal representation of a desire from the predicate.
- **Intention verbs**. These are verbs which are attached to action definitions in the module (domain) definitions, for the purpose of generating a verbal representation of an intention.
- **Parameter roles**. These are attached to the parameters of both action and predicate definitions. They specify the role in a sentence each parameter will play. Valid roles are *subject*, *complement*, *modifier* and *none*.

In addition to the these, further nouns are generated during the processing of the mission for objects which are found in the simulation definition. As these are not known to the AUVs at the start of the mission, they are not given distinct names. For the purposes of identification, any existent which has been found to contribute to the generation of a desire and which does not have an attached existent noun is given a name based on the relevant type noun followed by a number. These numbers are awarded according to the order in which the objects were discovered. So, for example, the first mine discovered during a mine counter measures mission would be called "mine one", followed by "mine two" and "mine three," whereas rocks discovered during the same mission would be left as "a rock," as they do not make a further contribution to the mission.

Finally, another special class of noun is used for existents with a "location" type. Each of these is given a symbolic name for use within the planning system, but these are of little use for visually locating the position in the world. In order to compensate for this, whenever the noun for an existence with a location type is requested, the value returned is a phrase representing its Cartesian position within the world.

### 7.3.2 Pronominalization

Pronominalization is a process which transforms common or proper nouns, such as table or Kevin, into pronouns, such as he, she or it. The use of pronouns has some benefits to take into account in the creation of a discourse. To quote directly from [107]:

> "...there is evidence that when a protagonist is already in explicit focus and there are no competing referents, comprehension is impeded when the protagonist is referred to by a full noun specification rather than by a pronoun; this is called the repeated name penalty[158]. One explanation for this is that a full noun specification is a cue to the comprehender to introduce a new protagonist into the current model whereas a pronoun is a cue to attach the current model to the token representing the protagonist in Short Term Working Memory. Thus, the full noun specification clashes with the presence of a token representing the same referent in Short Term Working Memory."

The paper referenced in the previous quote[158] states specifically:

> "The experiments show that there is a single backward-looking [centre] that is preferentially realized as a pronoun, and that the backward-looking [centre] is typically realized as the grammatical subject of the utterance."

In order to create the most effective discourse possible, pronominalization is implemented as a component of the natural language generation system. This allows a particular reference to an AUV which forms the subject of a process or stasis statement to be replaced with the pronoun *it*. In order to take additional advantage of this and further reduce the complexity of the discourse we add an additional form of pronominalization which allows a particular predicate construct (either a belief, desire or intention) to be replaced with the pronoun *this*, together with any additional supporting syntax which may be required. As this refers to more complex and distinctive phrases, this second form of pronominalization is not limited to the subject of the statement.

Each process or stasis statement allows access to both their protagonist and what is considered to be their most relevant predicate. This second datum is particularly strictly defined, and must represent the principle predicate form which is used as part of the statement. These are passed into the next element in the discourse as part of the natural language realisation process. If any component of the statement generated by the second element is both eligible for pronominalization and matches either of the data passed in, then pronominalization occurs. In each of the syntax diagrams shown in the following sections (and listed in their entirety in

Appendix F) any component which is coloured red is considered to be eligible for pronominalization by this process.

As an example, this process will convert the following discourse section:

- The Inspection AUV started to approach site alpha.
- The Inspection AUV finished approaching site alpha.

Into the shorter, less complex, and (theoretically) more readable version:

- The Inspection AUV started to approach site alpha.
- It finished doing this.

As a result, this process has the potential to shorten the amount of time taken to communicate the discourse to the user, whilst simultaneously making the communication more effective and easier to understand.

### 7.3.3 Fundamental Structures

The majority of the stasis and process statements in some way make reference to at least one belief, desire, intention or meta-desire. The principle purpose of the basic natural language structures defined in Section 7.3.1 is to supply the data required for the generation of these. Each is made up of a predicate of one form or another, with a predicate head and list of parameters. The exception to this is the meta-desire, which always contains one predicate form, but may also contain others.

In general, when an instantiation of a particular belief, desire or intention is converted to a sentence structure, an initially empty sentence structure is created using the constructs supplied by the SimpleNLG[90] library. Next a verb phrase is constructed using the verb specified for the particular predicate type and attached to the sentence as its verb phrase. The parameters of the predicate are now iterated through. The noun is retrieved for the existent which occupies the parameter's position in the instantiated predicate. This is converted to a noun phrase, and if the parameter is defined to have a role of "subject" or "complement", it is passed to the sentence in this role. In the case that it is defined to be a "modifier", a preposition phrase is constructed, using the noun phrase as its complement and preposition defined for the type of the existent as the preposition itself. In the instance that the parameter role is stated to be "none", the parameter is discarded and not used in the creation of the sentence structure.

The most direct implementation of this is used to generate sentences which represent intentions. The syntax diagram for this is shown in Figure 7.10, and is essentially identical to the generalised description given above. In accordance with the methodology given in Section 7.3.2, the subject of the sentence (which will in

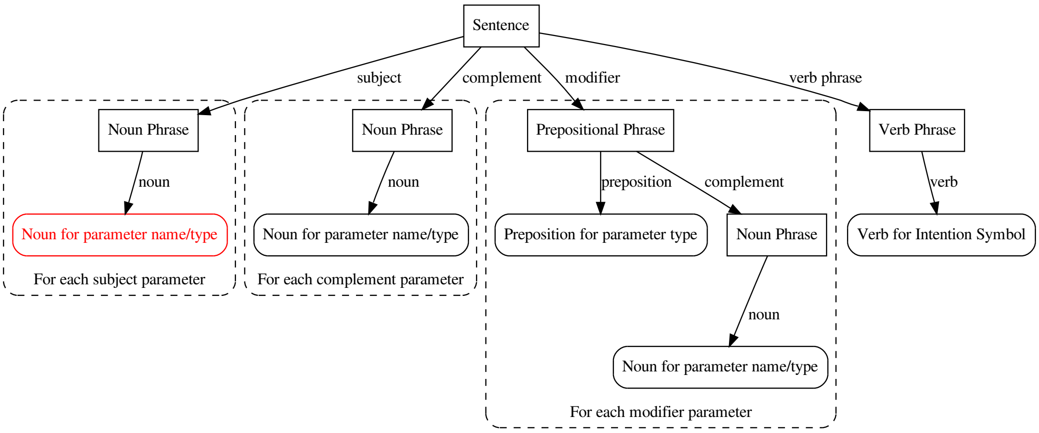

Sentence
subject
complement
modifier
verb phrase
Noun Phrase
Noun Phrase
Prepositional Phrase
Verb Phrase
noun
noun
preposition
complement
verb
Noun for parameter name/type
Noun for parameter name/type
Preposition for parameter type
Noun Phrase
Verb for Intention Symbol
For each subject parameter
For each complement parameter
noun
Noun for parameter name/type
For each modifier parameter

Figure 7.10: Syntactic structure for intentions.

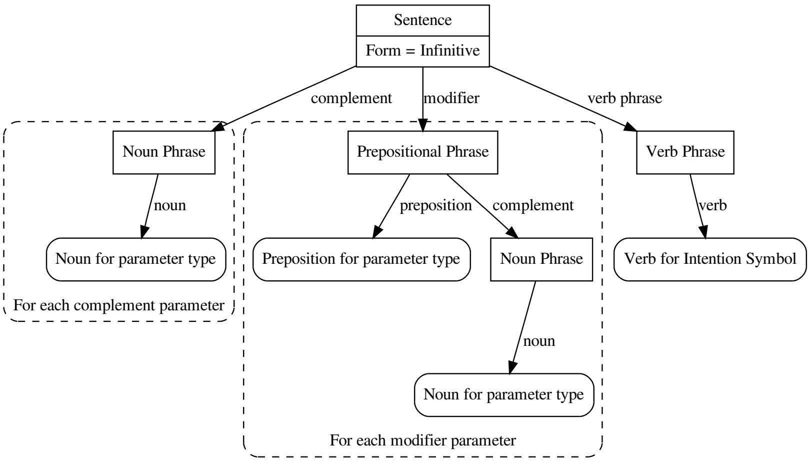


Figure 7.11: Syntactic structure for generalised intentions.

all cases be the name of an AUV) is eligible for pronominalization. This syntactic structure will produces sentences similar to:

> "The Inspection AUV examines Installation Alpha."

A second, simplified, expression of an intention is used by the *capability* stasis element, which describes in generalised terms what a particular capability allows an AUV to do. The syntax diagram for this form is shown Figure 7.11. It eschews subject completely, always uses type, rather than existent, nouns and reduces the sentence to the infinitive form. As expected, this form produces much terser and more generalised sentences, such as:

> "Examine installations."

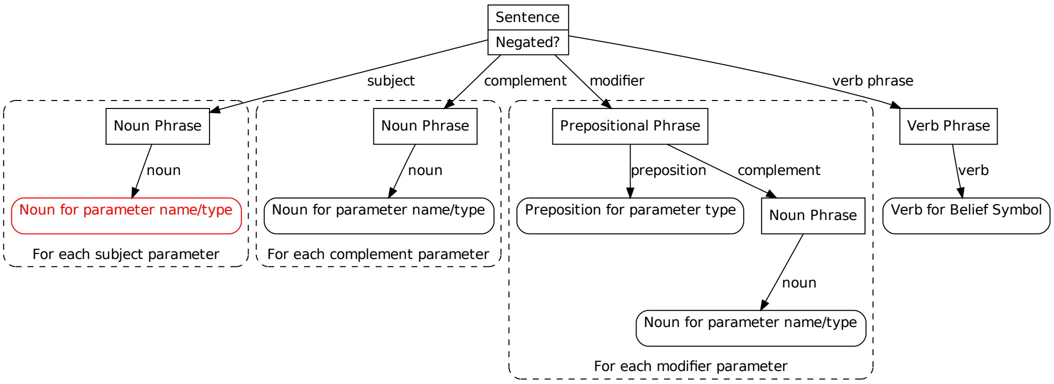

Sentence
Negated?
subject
complement
modifier
verb phrase
Noun Phrase
noun
Noun for parameter name/type
For each subject parameter
Noun Phrase
noun
Noun for parameter name/type
For each complement parameter
Prepositional Phrase
preposition
complement
Preposition for parameter type
Noun Phrase
noun
Noun for parameter name/type
For each modifier parameter
Verb Phrase
verb
Verb for Belief Symbol

Figure 7.12: Syntactic structure for beliefs.

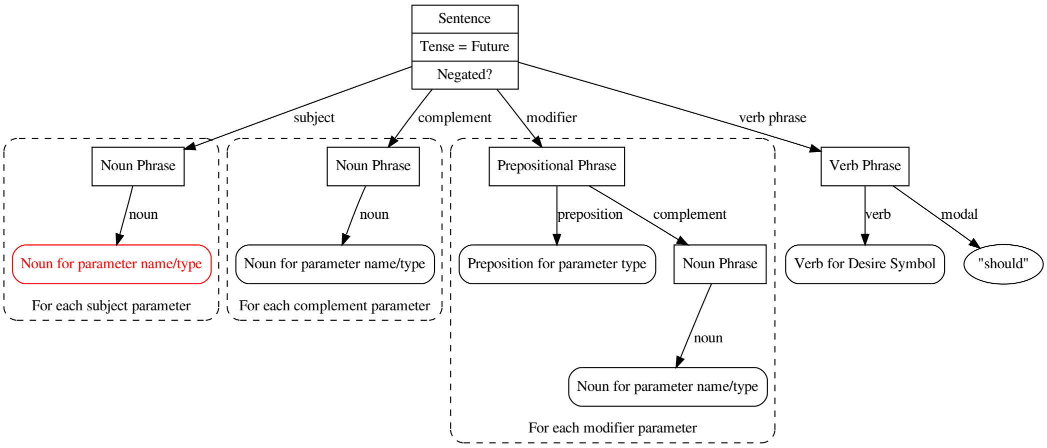


Figure 7.13: Syntactic structure for desires.

Unlike an intentions a belief has the potential to be negated. This flag, which is contained in the predicate representation, is passed to the sentence construct, the negation of which is handled internally by SimpleNLG. The syntactic structure used for the creation of belief forms is shown in Figure 7.12. It will produces sentences similar to:

> "Installation Alpha is functioning correctly."

The last basic predicate form is the desire. Its construction is almost identical to that used for beliefs, with the exception that it adds the additional model term "should" to the verb of the sentence, as shown in Figure 7.13. As a result, it produces sentences similar to:

> "Installation Alpha should be examined."

The final fundamental structure is the meta-desire, which has the potential to be

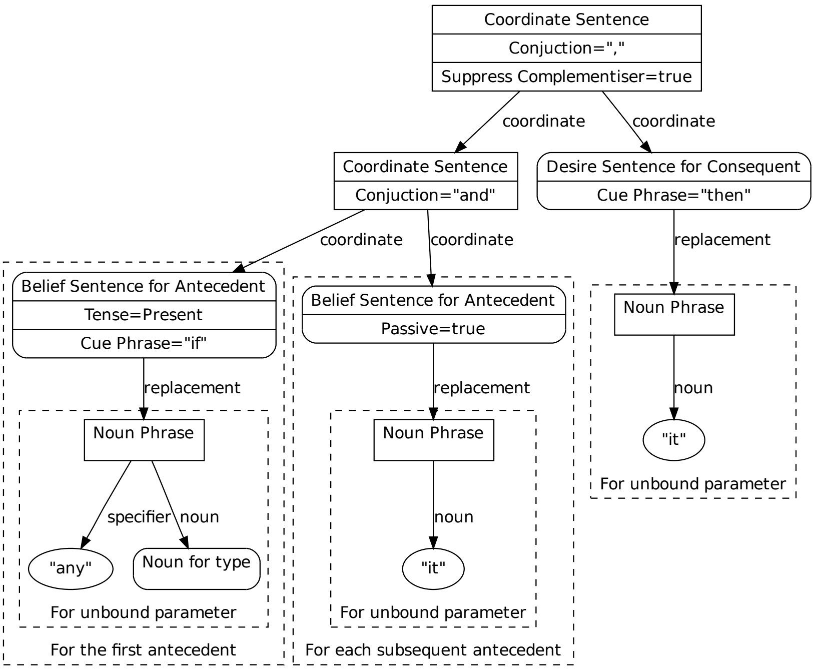


Figure 7.14: Syntactic structure for complex meta-desires.

significantly more complex than the other fundamental structures. The three forms a meta-desire can take, in ascending order of complexity, are:

1 **Static**. The meta-desire has no antecedents and no unbound parameters in its consequent;

2 **Dynamic, without antecedents**. The meta-desire has no antecedents, but one of more the parameters of its consequent is unbound;

3 **Dynamic, with antecedents**. The meta-desire has one or more antecedents, and zero or more unbound parameters in its consequent.

In the first case, the consequent of the meta-desire is a completely bound desire, and so it is treated exactly as one when transformed into a sentence representation.

The second case is again relatively simple. As this represents a logical "for all" case, all unbound parameters are simply given nouns consisting of the specifier "all" followed by the type noun for the parameter, which is pluralised. Otherwise, the sentence is generated in the same fashion as a standard atomic desire. So, as an example, this might produce a sentence such as:

> "All installations should be examined."

The final case introduces by far the most complexity, as it represents a logical "implication", or "if/then," case. As a reflection of this, the current implementation supports only a single unbound variable, though this variable may be repeated multiple times in the parameters of the antecedent(s) and consequent. A belief sentence is generated for each of the antecedents. Any unbound parameter is replaced with a noun phrase made up of the specifier "any" and the noun for the type of the parameter in the case of the first antecedent, and with the pronoun "it" for the remaining antecedents. In addition, the sentence for the first antecedent is given the cue phrase "if." The antecedents are then added to a coordinate sentence, which uses "and" as its conjunction. The consequent is treated as a desire sentence, with any unbound variable again being pronominalized to "it", and given the cue phrase "then." The structures for the antecedents and consequent are then added to a coordinate sentence which uses a comma as its conjunction, as shown in Figure 7.14. An example sentence which could be generated in this way might be:

> "If any installation is malfunctioning, then it should be repaired."

This section has described the basic structures which represent the underlying concepts behind the mission. It is the ways in which these interrelate and affect each other throughout the course of the mission which must be communicated to the user, however. This is carried out by the process and stasis statements, the conversion to natural language of which will be described in the next section.

### 7.3.4 Process and Stasis Statements

For the most part, the elements used in the discourse are converted to natural language using predefined syntax structures based upon the classes contained in the SimpleNLG library. These syntax structures are built using a combination of the wrapper types provided by SimpleNLG (such as "sentence", "verb" and "noun phrase"), the basic and fundamental structures described in Sections 7.3.1 and 7.3.3, and finally static text. The generation is performed in a similar fashion to the fundamental structures described in Section 7.3.3, however where as these are also entirely dynamically generated, the process and stasis statements are created using a higher degree of predefined structure.

Figure 7.15 shows a simple example of this methodology in the form of the syntactic structure used for the *desire satisfied* process element. This consists of a sentence with a noun phase subject which refers to the desire in question, together with a verb which states that is has been satisfied. A significantly more complex example can be found in Figure 7.16, which shows the *conditions* stasis element. This element represents the requirements of an intention, and so must express each of the other intentions which are required to have completed in order to make it possible, and also those which must not be attempted until after it has completed. As such, its syntax structure has a high potential for variation and expressivity,

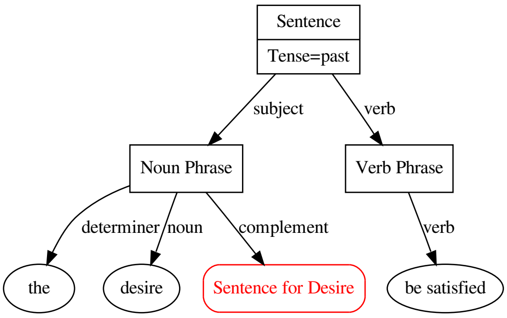


Figure 7.15: Syntactic structure for "Desire Satisfied" process statement.

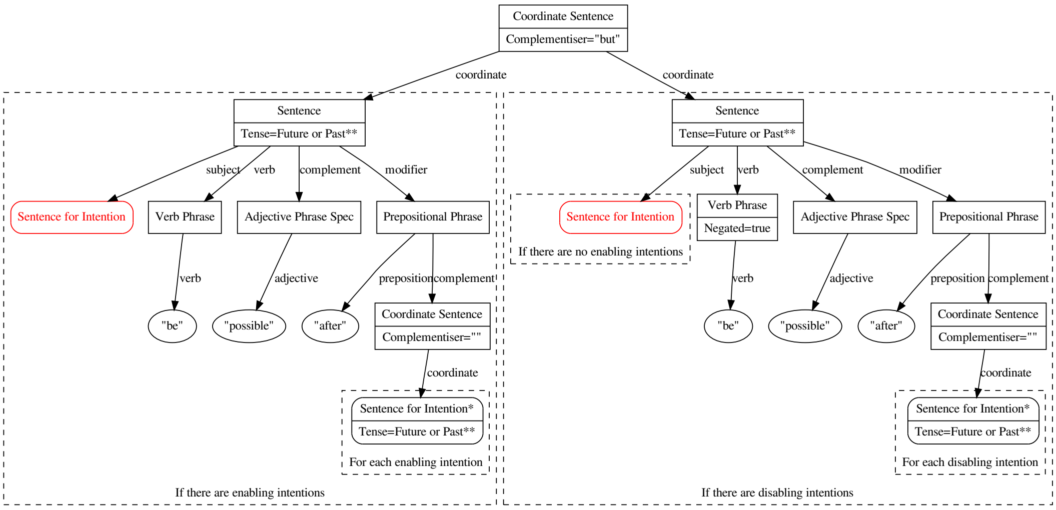


Figure 7.16: Syntactic structure for "Conditions" stasis statement.

but can still be dynamically created from the relatively small amount of data which passed in by the input files.

There are, however, some exceptions to this methodology, which are as follows:

- **Mission started**. This process element was originally converted to natural language using an entirely static syntax structure which produced the phrase "The mission started." This was found be slightly incongruous in practice, however, so this element is simply not expressed to the user.

- **Intention introduced**. This element is expressed to the user as the appropriate intention sentence, with the tense set to future. This produces phrases such as "The Inspection AUV will examine Installation Alpha."

- **Intention in progress**. This stasis element is expressed to the user in a similar fashion to the intention introduced process element, however with the tense set to present. As such, it produces phrases such as "The Inspection AUV is examining Installation Alpha."

- **Object property**. This element is expressed using the appropriate belief sentence, producing phrases such as "Installation alpha is functioning correctly."

- **Sensor mode explanation** and **actuator mode explanation**. These two stasis elements have the potential for large variation, making them difficult to represent with a predefined syntax structure. As a consequence, they are instead represented as "canned text," which is supplied to the system in the sensor and actuator description files (see Section B.1). This text is supplied in two forms: verbose, which is used the first time a particular sensor or actuator mode is expressed to the user; and curt, which is used for each subsequent occasion.

There are also two further exceptions, which do use a pre-defined syntax structures but in a slightly more irregular fashion. These are:

- **Capability**. This element is likely to contain a large amount of information which has the potential to stall the discourse if it is repeated to the user numerous times. In order to combat this two differing syntax structures are employed. The first, or complex, version is used the first time a particular capability is expressed to the user. The second, or simple, version is used for each further expression of the capability, regardless of whether it refers to the same AUV. The syntax structures for these can be found, respectively, in Figures 7.17 and 7.18.

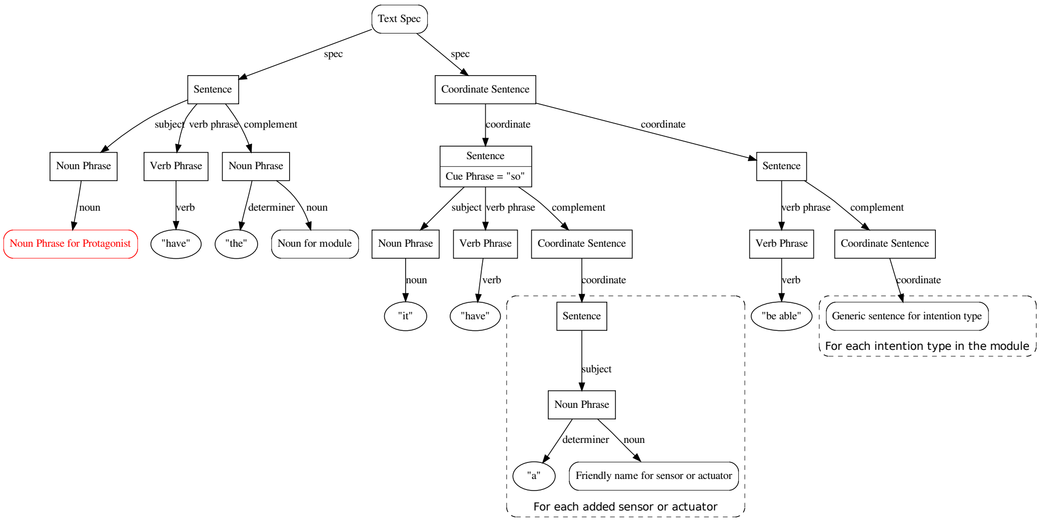


Figure 7.17: Syntactic structure for the complex version of the "Capability" stasis statement.

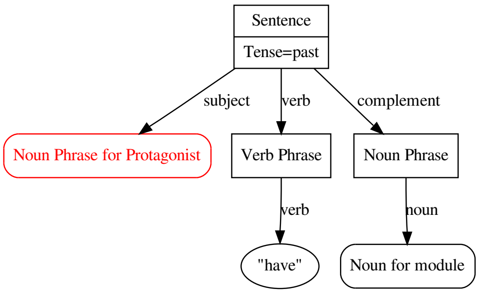


Figure 7.18: Syntactic structure for the simplified version of the "Capability" stasis statement.

- **AUV explanation**. The principle purpose of this stasis statement is to identify the named AUV and show the user the location at which it started the mission. As a result, it only needs to be expressed once, and so this stasis statement is ignored upon repetition.

The complete list of syntax structures used for all process and stasis elements can be found in Appendix F.

## 7.4 Generation of Additional Script Elements

The ultimate aim of the current version of the *Glaykos* system is the creation of an automated debrief, based on the graphical user interface described in Section 4.3. The system requires the mission logs, in order to accurately represent the movement and actions of the agents which carried out the mission, together with a set of scripts which instruct the system on how the mission should be replayed and what additional information should be given to the user. These scripts are:

- The **audio** script, which specifies audio files which should be played at certain times during the debrief;
- The **subtitle** script, which specifies the subtitles which should be displayed along the bottom of the screen;
- The **time** script, which describes the relationship between playback time and mission time during the debrief;
- The **camera** script, which describes the movement of the virtual camera;
- The **focus** script, which specifies which vehicles or other objects in the world should be highlighted at which times during the playback;
- The **graphics** script, which specifies the additional graphics which should be shown to the user on the heads-up-display.
- The **pause** script, which will cause the system to pause at certain predefined times and wait for the user to push the space bar before continuing.

The generation of these scripts is based entirely on the information contained in the discourse created in Section 7.1, and its natural language representation generated in Section 7.3.

The remainder of this section will describe these scripts in more detail, together with methodology used to generate them.

### 7.4.1 The Audio, Time, Subtitle and Pause Scripts

For the purposes of the *Glaykos* system, the audio script is used to verbally narrate the natural language representation of the discourse to the user. This audio narration is used as the backbone for the timing of the entire debrief. The time script instructs the GUI as to which time during the mission should be shown to the user at each time during the debrief. For *Glaykos*, its contents are based almost entirely on the timings produced for the audio script. However, the contents of the audio script is affected by the time scripts, as additional statements are added to the audio script in order to help keep the user situated when large changes in time occur. Due to the interdependency, the two scripts are generated simultaneously.

The principle rule the time script attempts to maintain is that the event each process statement refers to should be shown to the user at the same time the process statement begins to be narrated by the audio script. This must take into account the fact that a process statement may refer to an event which happened simultaneously, or very close, to that described by the previous process statement, it may refer to one which happened later in the mission, or it may refer to one which happened earlier in the mission. In the first case, the playback of the mission must be slowed down in order to ensure that the narration occurs with the correct timing. In the later two cases the mission can be shown in real time, but a rewind or fast-forward must also be performed.

The narrative audio itself is created using the FreeTTS[159] text-to-speech system, which provides an open source speech synthesiser. A sound file is created for the natural language version of each statement in the discourse, and the length in seconds of each audio file recorded. The discourse is then added to the audio script one process statement at time, with this being preceded and followed by the appropriate stasis statements. Different pauses are added between the statements, depending on the type of the statements which are being separated:

- **Stasis to stasis**, which is used between stasis statements which precede or follow a single process statement;

- **Stasis to process**, which is used between the stasis statements which precede a process statement and the process statement itself;

- **Process to stasis**, which is used between a process statement and the stasis statements which follow it.

The use of different lengths of pause in this way allows the process statements to be emphasised and separated in a reasonably subtle manner. A further type of pause is used between the sets of process and stasis statements, during which any fast forwarding or rewinding is performed. If the amount of time passed during the fast-forward or rewind is higher than a pre-set threshold, an additional statement is added to the discourse in order to keep the user appraised of the system's actions. This statement takes the form: "Fast-forwarding by xxx," or "Skipping backwards[2] by xxx," where xxx is a "friendly" representation of the time, such as "20 seconds," or "1 and a half minutes."

The subtitle script is used to ensure that the information contained in the narration reaches the user, even when the synthesised voice is unclear. To facilitate this, it is synchronised exactly with the audio script.

Appendix G gives a complete example discourse, together with the timings which would be used for the audio and time scripts. Additionally, the values which were used in practice for each of the parameters used here are listed in Appendix D.

### 7.4.2 The Camera Script

The camera script controls the position and view of the virtual camera during the debrief, and so has the potential to be very complex. The handling of it for the current version of *Glaykos*, however, is very simple, as it was decided that a static "birds eye view" should be used for the mission debriefs. There were two reasons for this. Firstly, it became apparent that implementing a methodology capable of dynamically moving the camera in order to follow the events of the mission required more development time than was available. Secondly, it seemed likely that the automated camera control this would lead to had the potential to be highly contentious. Some users might find it distracting (as is sometimes the case with other visual media, such as films[3]), or become frustrated by the feeling that parts of the mission were being "hidden" from them. Alternatively, unless the system is implemented perfectly, the result could be that the user's attention is drawn to the wrong events during the debrief, and as a result their confusion is increased.

The camera position and orientation used for the "birds eye view" is calculated using the extent of the mission in the north and east directions. To get this, the logs are explored to find the highest and lowest north and east positions inhabited by the AUVs and objects which are contained in the mission. The camera's position on the north and east axes is then calculated to be the exact centre of this. Its height above the mission area is then calculated using the field of view of the camera in order to ensure that all of the AUVs and objects will be in view at all times during the mission debrief. This is represented pictorially in Figure 7.19.

The major downside to this approach is that for missions which cover a large area the camera will be positioned at quite a distance from the events which are being viewed, making the AUVs and objects appear small and hard to differentiate.

[2] "Skipping backwards by" is used in place of "Rewinding by" as FreeTTS is unable to correctly pronounce the word "Rewinding."

[3] *Requiem for a Dream*[160] is one example of a film which has received both praise[161] and criticism[162] for its cinematography.

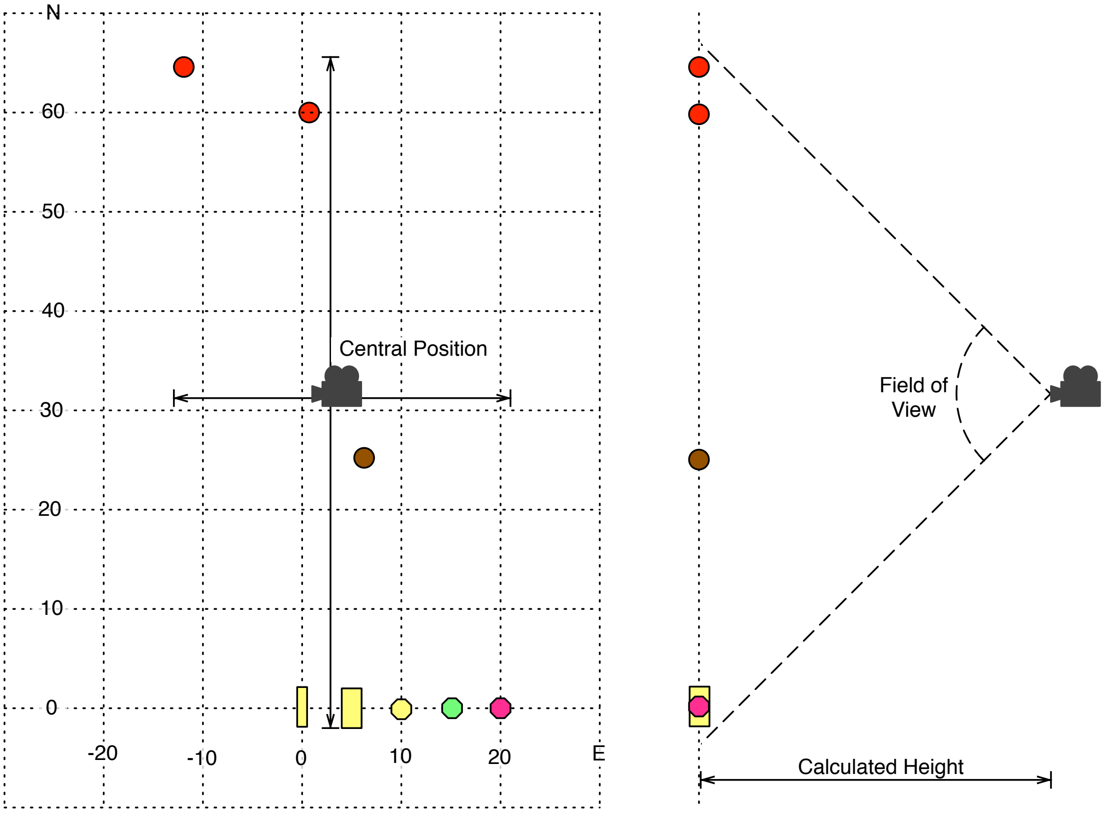


Figure 7.19: Top and side elevations, showing camera position calculated to make all elements visible.

As a result the interactions between the AUVs and the world objects also become difficult to observe. Missions which have been tested using the current version of the system have been carried out over reasonably small areas, in order to avoid this deficiency.

### 7.4.3 The Focus Script

The focus script is used to tell the GUI which elements should become focused, or selected, during the course of the debrief. It is used by *Glaykos* to indicate which protagonist is responsible for a particular event, and thus keep the user as situated as possible on the protagonist situational axis (see Sections 2.10 and 7.1.2). In addition, it is used to highlight relevant objects in the world. Both conceptually, and in terms of implementation, *Glaykos*' handling of the focus script is extremely simple.

The *intention introduced*, *intention started*, *intention succeeded* and *sensor detection* process elements actively involve a protagonist. As a consequence, for each of these elements, the protagonist is highlighted for the extent of the process statement and the stasis statements which precede and follow it. If the next process element in the discourse has the same protagonist, then the focus is maintained. Alternatively, if it has a different protagonist, the focus is cancelled concurrently with the end of

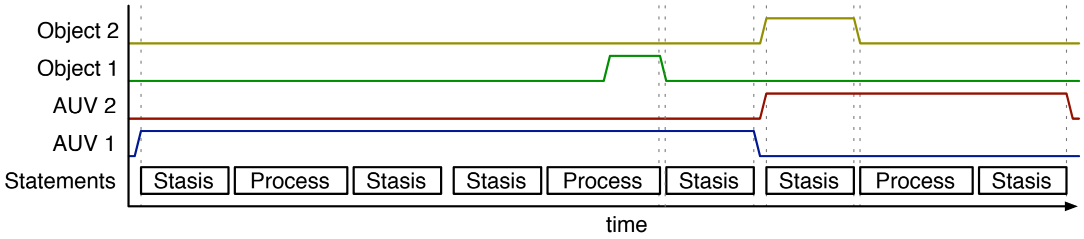


Figure 7.20: Illustration of how focus of particular object is handled in a hypothetical script. The first two process statements refer to AUV 1, whereas the third refers to AUV 2. The second process statement is a *sensor detection* which refers to Object 1, where as the fifth stasis statement is an *object property* which refers to Object 2.

the last stasis element which follows the process statement.

In the case of the *object property* and *object location* stasis statements, information is being passed to the user regarding an object in the world (which is not one of the AUVs). In order to directly draw the attention of the user to the object which is being referred to, this object is focused for the length of the stasis element.

The *sensor detection* process element presents an additional special case, as it also contains direct references to other objects in the world, to the extent that the syntax structure used to convert it to a process statement (see Figure F.8) contains actual object location and object property elements. Again the objects these phrases refer to should be highlighted, but not for the entire length of the sensor detection statement. Since the point at which the phrases describing the objects occurs during the audio clip is not known, a "rule of thumb" is used. Each of the phrases is assumed to take the same amount of time to deliver to the user, and the introduction to the statement ("The search AUV's side scan sonar detected that...") is assumed to take the same amount of time again. Based on this, the relevant objects are focused for the corresponding section of the statement.

This strategy is illustrated in Figure 7.20, while Appendix G gives a complete example discourse, with accompanying focus script elements, which illustrates some of the ways in which this methodology is applied.

### 7.4.4 The Graphics Script

The graphics scripts is used to control the Heads Up Display (HUD) which provides additional information to the user in the form of animated two dimensional graphics. The script is designed to be able to represent extremely complex graphics and the transitions between them, and so has the potential for a high degree of complexity. Fortunately, the internal format of the script was designed to reflect the XDOT output format which can be produced by the GraphViz[131] software package. As a result, GraphViz can be used to generate "key frames" for the animated graphics, and the HUD itself will automatically generate the transitions between them, based on the names of the shapes, nodes, and text labels which make up the graph.

The heads up display graphics are used by *Glaykos* to display the motivation of the statement which is currently being delivered to the user, and hence keep the user as well located on the motivation situational axis as possible (see Sections 2.10 and 6.1.3 for descriptions of the situational axes). The graphs themselves are based on the *Action Resource Formalism* described in *Plan Merging in Multi-Agent Systems*[82]. This methodology models time and causation as flowing upwards, however, rather than the more traditional right or down (see Section 2.8 for a discussion of this). For our graphs we model these as flowing in one of the the more traditional directions (depending on the type of the graph), in order to provide as much familiarity for the user as possible.

The *desire* view style is used for illustrating the motivation as it relates to process statements which describe the addition and satisfaction of desires. Specifically, this is the *meta-desire description* stasis statement, and the *desire added*, *desire satisfied*, *meta-desire satisfied* and *mission succeeded* process statements. The desires in a mission make up a tree structure, with the successful completion of the mission at the root, a set of meta-desires below this, and a set of desires below each of these (see Figure 6.2 for an example). This creates a tree with a constant depth of three, but a variable breadth which changes both within and between missions. As a consequence of this, the desire views represent causation as flowing from left to right, as this represents the most efficient use of space.

A common scheme is used for the display of desire views, which is:

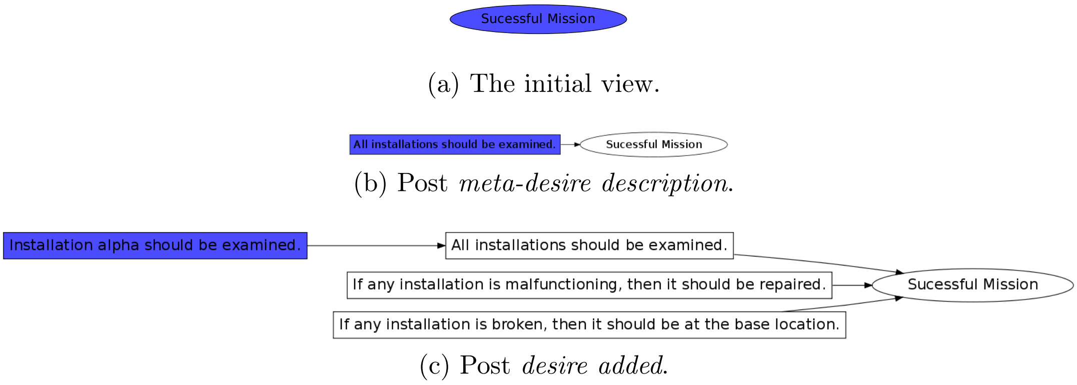
1 If the HUD is currently hidden, it is faded back in showing the same graph which was in use when it was faded out.

2 If the graph which is currently shown is an intention view, it is transitioned into a desire view, in the state at which it was last seen.

3 The graph is transitioned to its new appearance.

In the case of the *meta-desire explanation* and *desire added* statements, the new appearance adds the new desire to the graph, highlighted with a blue background. Figure 7.21 shows some examples of this.

In the case of the *desire satisfied*, *meta-desire satisfied* and *mission succeeded* process statements, the background colour of the element which has been satisfied is changed to green in order to represent the satisfaction. Figure 7.22 shows a series of views which represent this. Namely, these are the three process statements which will always come last in a discourse describing a successful mission: the satisfaction of the last desire, the meta-desire satisfaction which is triggered by this, and the successful conclusion of the mission.

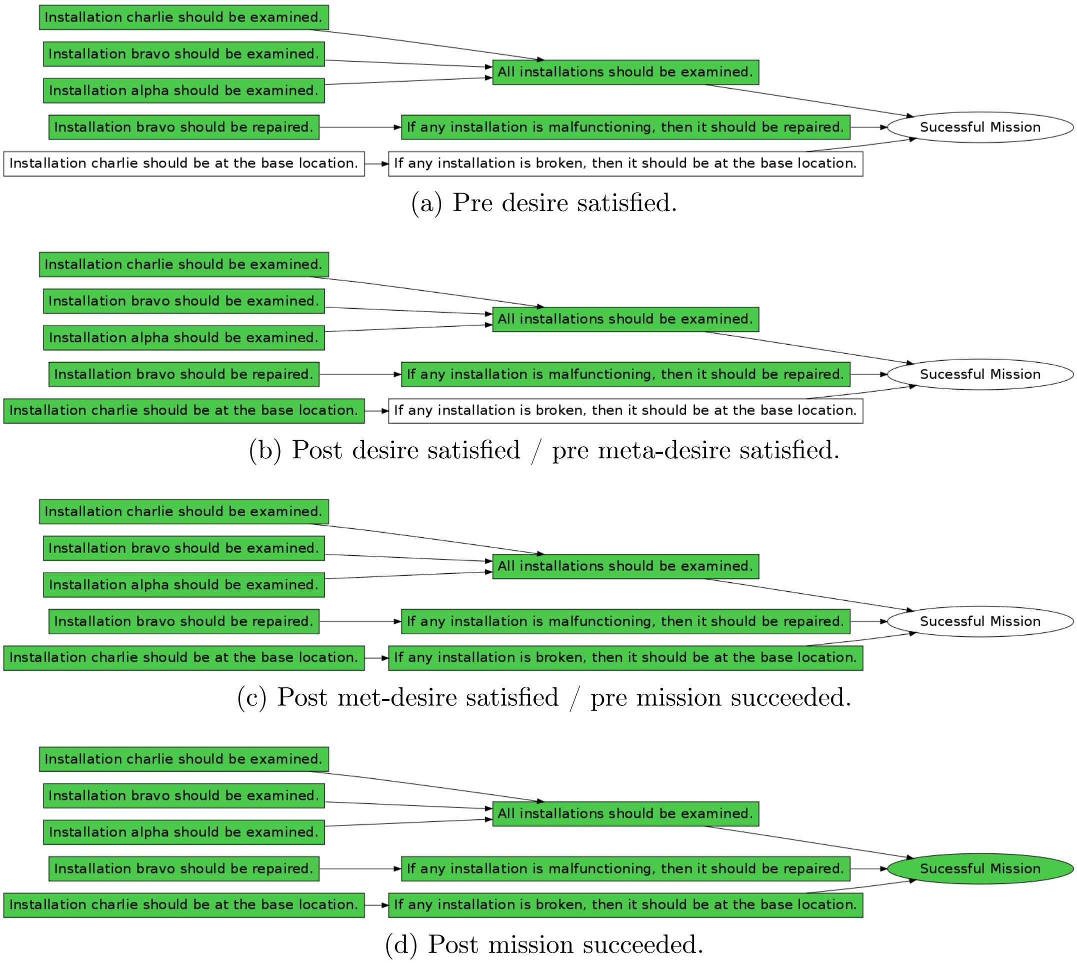

Sucessful Mission
All installations should be examined.
Sucessful Mission
Installation alpha should be examined.
All installations should be examined.
If any installation is malfunctioning, then it should be repaired.
If any installation is broken, then it should be at the base location.
Sucessful Mission

(a) The initial view.

(b) Post *meta-desire description*.

(c) Post *desire added*.

Figure 7.21: Example views for addition of meta-desires and desires. Note: text in the actual HUD appears white.


Installation charlie should be examined.
Installation bravo should be examined.
Installation alpha should be examined.
Installation bravo should be repaired.
Installation charlie should be at the base location.
All installations should be examined.
If any installation is malfunctioning, then it should be repaired.
If any installation is broken, then it should be at the base location.
Sucessful Mission
Installation charlie should be examined.
Installation bravo should be examined.
Installation alpha should be examined.
Installation bravo should be repaired.
Installation charlie should be at the base location.
All installations should be examined.
If any installation is malfunctioning, then it should be repaired.
If any installation is broken, then it should be at the base location.
Sucessful Mission
Installation charlie should be examined.
Installation bravo should be examined.
Installation alpha should be examined.
Installation bravo should be repaired.
Installation charlie should be at the base location.
All installations should be examined.
If any installation is malfunctioning, then it should be repaired.
If any installation is broken, then it should be at the base location.
Sucessful Mission
Installation charlie should be examined.
Installation bravo should be examined.
Installation alpha should be examined.
Installation bravo should be repaired.
Installation charlie should be at the base location.
All installations should be examined.
If any installation is malfunctioning, then it should be repaired.
If any installation is broken, then it should be at the base location.
Sucessful Mission

(a) Pre desire satisfied.

(b) Post desire satisfied / pre meta-desire satisfied.

(c) Post met-desire satisfied / pre mission succeeded.

(d) Post mission succeeded.

Figure 7.22: Example states for desire satisfaction, meta-desire satisfaction and mission succeeded views.

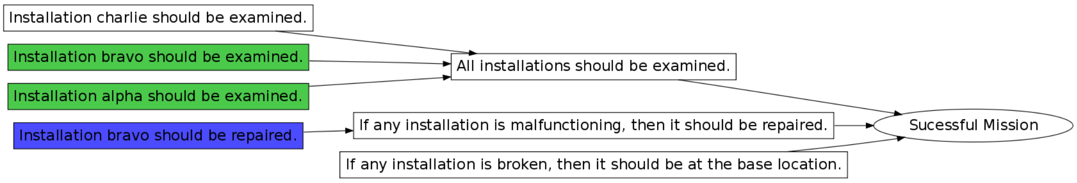

Installation charlie should be examined.
Installation bravo should be examined.
Installation alpha should be examined.
Installation bravo should be repaired.
All installations should be examined.
If any installation is malfunctioning, then it should be repaired.
If any installation is broken, then it should be at the base location.
Sucessful Mission

Figure 7.23: Addition of a new desire into a view which already contains satisfied desires.

It is important to note that the desire views always represent the current state of the mission from the user's point of view. So, desires which have been declared to be satisfied maintain their green background, even if the actual playback of the mission rewinds to a point before the satisfaction of the desire. All desires also remain visible in the graph if the playback moves back to a point before they were added. The representation of satisfied desires is also maintained for views which show the addition of a new desire, as illustrated in Figure 7.23. This methodology priorities a higher level view of the mission, and is designed to maintain consistency and stabilise the user's situational model of the progress of the mission as a whole as much as possible.

Intention views are used to visualise motivation for all process statements which relate directly to intentions. Each intention view is a more linear graph which connects the *mission successful* node to the relevant intention or intentions, via the relevant meta-desire, the relevant desire and all of the other intentions in the causal chain between the desire and the intention(s) in question. As result, this graph has an effective width of one (though some intentions may be in parallel) and a variable height. In order to most effectively use the available space, these views model causation as flowing from the top of the screen to the bottom. This also serves to effectively distinguish the intention views from the desire views.

As with the desire views, a common progression scheme is used for the display of intention views. Desire views display the progression of a single tree structure, whereas the intention views must display the progression of the plan for each desire. As a result the scheme is somewhat more complex:

1 If the HUD is currently hidden, it is faded back in showing the same graph which was in use when it was faded out.

2 If the graph which is currently shown is a desire view, it is transitioned into an intention view, in the state at which it was last seen.

3 If the graph which is currently shown relates to a different motivation than than the one of the graph which is to be shown:

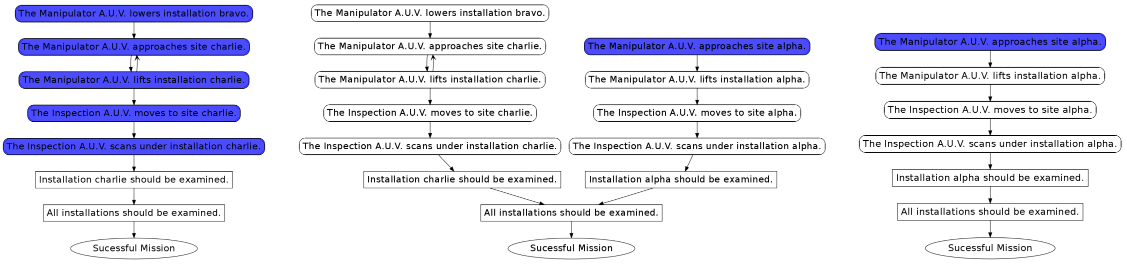
   (a) The graph is transitioned to an intermediate state which shows both the original view and the target view, in the state in which it was last seen.

   (b) The graph is transitioned to one which removes the original view.

4 The graph is transitioned to its new appearance.

An example of the progression described in part 3 of this scheme is shown in Figure 7.24.

The final appearance of the intention view depends on the type of statement which is being displayed. These are as follows:


The Manipulator A.U.V. lowers installation bravo.
The Manipulator A.U.V. approaches site charlie.
The Manipulator A.U.V. lifts installation charlie.
The Inspection A.U.V. moves to site charlie.
The Inspection A.U.V. scans under installation charlie.
Installation charlie should be examined.
All installations should be examined.
Sucessful Mission
The Manipulator A.U.V. lowers installation bravo.
The Manipulator A.U.V. approaches site charlie.
The Manipulator A.U.V. lifts installation charlie.
The Inspection A.U.V. moves to site charlie.
The Inspection A.U.V. scans under installation charlie.
Installation charlie should be examined.
The Manipulator A.U.V. approaches site alpha.
The Manipulator A.U.V. lifts installation alpha.
The Inspection A.U.V. moves to site alpha.
The Inspection A.U.V. scans under installation alpha.
Installation alpha should be examined.
All installations should be examined.
Sucessful Mission
The Manipulator A.U.V. approaches site alpha.
The Manipulator A.U.V. lifts installation alpha.
The Inspection A.U.V. moves to site alpha.
The Inspection A.U.V. scans under installation alpha.
Installation alpha should be examined.
All installations should be examined.
Sucessful Mission

(a) Initial view, with multiple intentions selected.

(b) Intermediate view.

(c) Final view, with single intention highlighted.

Figure 7.24: Example transition between intention views.

- **Intention Introduced**. The intention in question is highlighted with a blue background. It is assumed, due to the way that the discourse is ordered, that this intention will not be present in the previous view, and so it will also be seen fading into the view.
- **Plan Introduced**. In a similar manner to the intention introduced element, every intention in the plan is highlighted, as per the example in Figure 7.24a. Again, it is assumed that these intentions will not have been visible in the previous view, and so they will be shown fading in.
- **Intention Started**. Again, the intention is question is highlighted with a blue background, as per the example in Figure 7.24c.
- **Intention Successful**. In this case, the intention in question is removed from the view, showing that it is no longer part of the plan. It is also ensured that the intention is highlighted in blue in the previous view.

As with the desire views, causal connections are represented in the intention views by arrows with a solid triangular tip. Invalidation connections (see Section 6.1.1) are additionally represented in the view by an arrow with a hollow triangular tip, pointing backwards compared to the causal connections. An example of this is visible in Figure 7.24a, between the second and third intentions from the top.

This methodology provides the user with a slightly lower level and less abstract view of the motivation of the mission than that provided by the desire views.

The appearance of each of the views is timed that so that transition starts as the system begins to read the relevant process element, and the final view becomes visible before the system finishes reading it. The exact timings vary depending on the amount of time required to read the process statement, but where possible each individual view (including any intermediate transitions) is kept visible for at least one second.

The heads-up display which shows these views is faded out after the conclusion of the current process statement if the next process element does not require it, otherwise it remains visible. Appendix G gives a complete example discourse, with references to the elements of the graphics script which are displayed at each point. The views themselves are also shown.

## 7.5 Summary

This chapter has discussed the generation of the discourse constructed to deliver the data contained in the vector situation model, which is described in the previous chapter, to the user. The problem of serializing the essentially parallel structure of the elements which make up vector situation model into a concrete ordering is treated as an optimisation problem. Novel techniques based on the study of how narrative can be used to effectively structure discourse for human understanding are used to form a fitness function so that a genetic algorithm and bounded constrained depth first search can be used to find a potentially optimal ordering.

An outline of the structure of the discourse results. This is then simplified where possible and embellished with the addition of relevant expository information. This leads to a complete "text base," in the parlance of *How does the mind construct and represent stories?*[106], which is then transformed into the "surface code" via natural language generation techniques.

The surface code is converted to spoken language using a speech synthesiser. The timings taken from this are then used to generate a set of further scripts to create a full audio visual mission replay using the graphical user interface described in Chapter 4.3, including the control of time within the mission and additional graphics which are displayed to the user using the Heads-Up-Display (HUD). The results in an debrief, automatically generated in a novel fashion, and suitable for efficiently providing the user with a complete picture of the events of the mission.

The next chapter will describe the scenarios which were used for testing the *Glaykos* system, and the discourses which were generated as a result.

# Chapter 8

# Scenarios and Generated Discourse

This chapter describes the three mission scenarios which were created for use in the testing of the *Glaykos* system. The first two of these scenarios are based on relevant real world applications of autonomous underwater vehicles, whilst the third was created with the aim of producing a more complex mission with increased parallelism, making it much harder to follow. In the following sections, each of these scenarios will be described in more detail. Firstly, the goals and initial world state for each of the scenarios will be described. Next the implementation of each mission will be discussed, and the degree of parallelism within in the mission will be given[1]. Excerpts from the discourse produced for each of the scenarios are also provided. Finally, further excepts of the discourse generated by the *Glaykos* system will be given, to illustrate the potential "critical incidents" within the discourse, which have the potential to cause cognitive difficulties for the user. These potentially critical incidents will be broadly classified thusly:

- **Planner based**. These are cases in which the planning system (see Section 4.1) has caused the AUVs to behave in a way which may be found to be unintuitive for human users;

- **Narrative order based**. These are cases in which the *Glaykos* system itself has produced an ordering of events during the mission replay which may cause human users some difficulty, such as is discussed in Section 7.1.6;

- **Realiser based**. These are cases in which the natural language generation system (see Section 7.3) has produced language which may cause human users some difficulty. Incidents of this type are likely to be the result of errors in the NLG system which produced language problems which were not detected until after testing began.

Finally, the last section of this chapter will give a brief summary.

[1] Measured by the number of AUVs operating simultaneously at each time within the mission

## 8.1 Offshore Maintenance

The first scenario deployed as part of the experiments used offshore maintenance as its backdrop. Offshore maintenance in this instance refers to the maintenance of offshore oil installations, typically at depths of over 1000 metres. These kinds of mission are usually carried out by Remotely Operated Vehicles (or ROVs) deployed by support ships at extremely high cost. Recently, new technology has allowed the development of prototype autonomous solutions to a subset of these problems[8]. This first scenario projects this technology forward to a point at which multiple AUVs can be deployed to perform both inspection and repair tasks.

### 8.1.1 Initial State and Mission Goals

In the scenario there are three AUVs. The *Manipulator AUV* is capable of lifting an installation and potentially moving it to another location. The *Inspection AUV* is capable of examining an installation, but requires it to be lifted off the ocean floor in order to do so. Finally, the *Intervention AUV* is able to repair a malfunctioning installation (but must first move close to it).

In the mission area there are three installations:

- Installation Alpha, which is functioning correctly.
- Installation Bravo, which is “malfunctioning”.
- Installation Charlie, which is “broken”.

The AUVs are aware of the existence and position of each of the installations, but not of their status. The high level goals (meta-desires) of the mission are:

- All installations should be examined.
- Any installation which is “malfunctioning” should be repaired.
- Any installation which is “broken” should be moved to the base location.

The existence and position of the base location mentioned in the third goal is defined in the AUVs’ definition of the world. The physical layout of the mission, including the starting positions of all of the AUVs and objects, is shown in Figure 8.1.

### 8.1.2 Implementation

Focusing on the actual actions of the AUVs, the salients facts regarding the implementation of the offshore maintenance mission are as follows:

- The Manipulator AUV lifted Installation Alpha so that it could be examined;

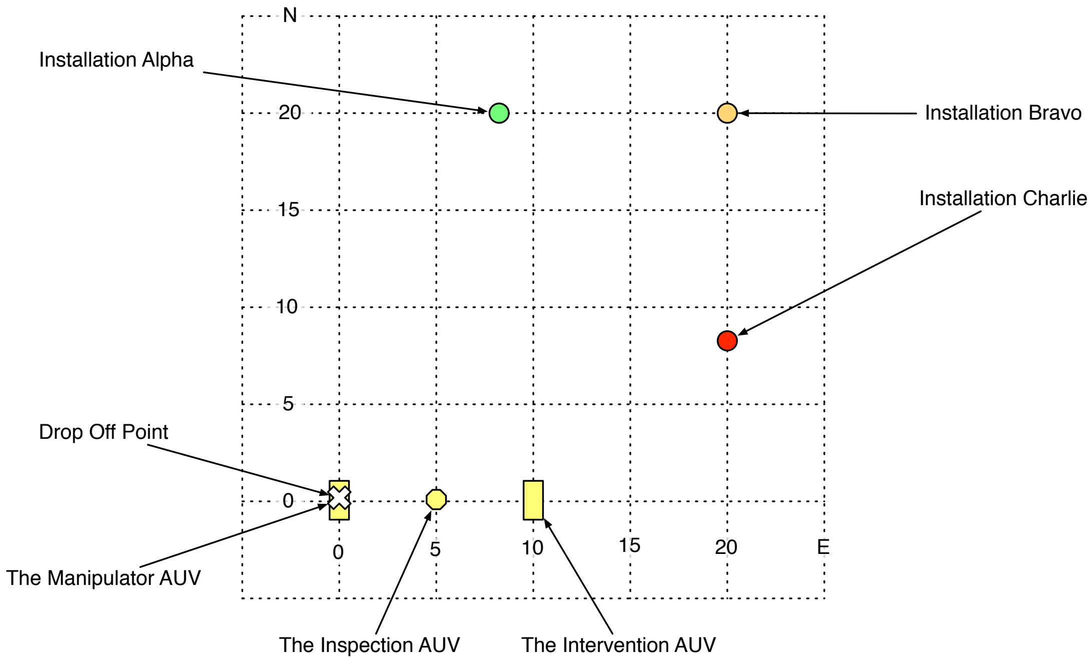


Figure 8.1: The initial physical layout of the Offshore Maintenance Mission.

- The Inspection AUV examined Installation Alpha;
- Installation Alpha was found to be working correctly;
- The Manipulator AUV lifted Installation Bravo so that it could be examined;
- The Inspection AUV examined Installation Bravo;
- Installation Bravo was found to be “malfunctioning”;
- The Intervention AUV repaired Installation Bravo;
- The Manipulator AUV lifted Installation Charlie so that it could be examined;
- The Inspection AUV examined Installation Charlie;
- Installation Charlie was found to be “broken”;
- The Manipulator AUV moved Installation Charlie to the base location.

As the actions of one AUV are often dependent on another (in particular the *Inspection* and *Intervention AUVs*), this mission is carried out in a largely serial fashion, but with a large number of causal connections between the events. The parallelism of the mission is illustrated in Figure 8.2.

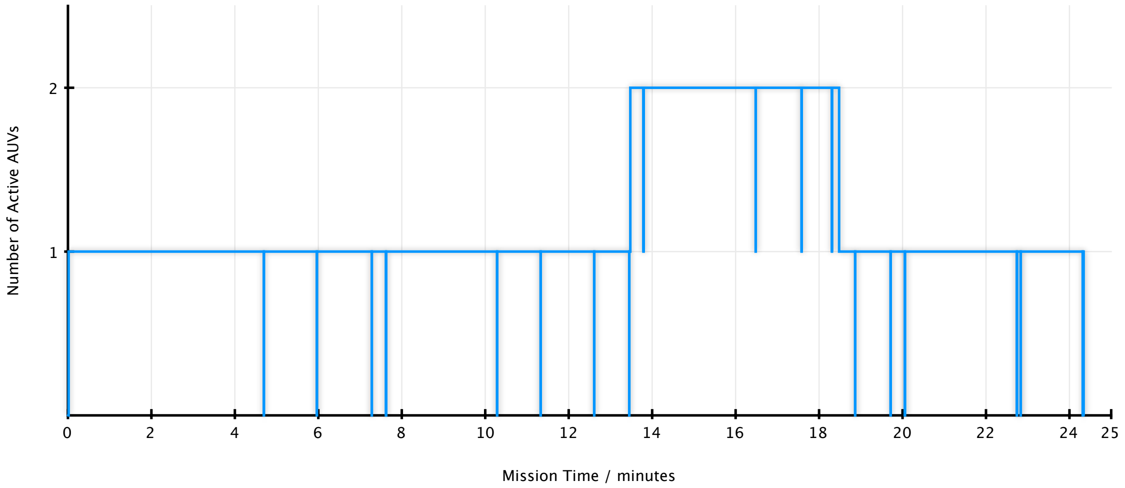


Figure 8.2: The graph of activity during the Offshore Maintenance Mission.

### 8.1.3 Discourse Excerpt

The section provides a short excerpt from the end of the discourse generated for the offshore maintenance scenario, featuring the dialogue which is given to the user. Each line of dialogue is preceded by its index within the discourse, the index of the process element to which it is linked in parentheses, and a symbol representing the nature of the statement ( "+" for process statements and "-" for stasis and supporting statements). In this case, the events are delivered to the user in chronological order, as they represent the close of the final causal thread which leads to the completion of the mission:

- 110 (57) + [The Manipulator A.U.V.] finished moving to the base location.
- 111 (58) - The Manipulator A.U.V.'s releasing installation charlie at the base location means that the desire that installation charlie should be at the base location will be satisfied.
- 112 (58) + It started to do this.
- 113 (58) - This was possible after the Manipulator A.U.V. grabbed installation charlie and the Manipulator A.U.V. moved to the base location.
- 114 (59) + It finished releasing installation charlie at the base location and put its lifting arm in passive mode.
- 115 (60) + The desire that installation charlie should be at the base location was satisfied.
- 116 (61) + The desire that if any installation is broken, then it should be at the base location was satisfied.
- 117 (62) + The mission succeeded.

The entire discourse for this scenario is given in Appendix G, complete with its complete timing an additional elements. A longer excerpt, together with selected graphics transitions, is provided for the combination scenario in Section 8.1.3.

### 8.1.4 Potentially Critical Incidents in Discourse

This section will discuss the potentially critical incidents in the discourse, as described in the introduction to the this chapter. Statements which are shown in red are those which are the focus of the critical incident.

There are four potentially critical incidents in this discourse, the first of which is realiser based:

- 22 (11) + The Inspection A.U.V. started to move to site alpha.
- 23 (11) - This was possible after the Manipulator A.U.V. lifted installation alpha.
- 24 (12) - Skipping forward by 1 minute and 11 seconds.
- 25 (12) - It finished moving to site alpha.
- 26 (13) - It has the examine installations module and so it has an installation scanner and is able to scan under installations.

In this critical incident, the natural language generation system substituted the pronoun "it" for the phrase "The Inspection AUV" at two positions in the dialogue at which this is at best ambiguous and at worst misleading. It may not be clear to the user as to whether the the two instances of "it" in statements 25 and 26 refer to the Inspection AUV, as per statement 22, or the Manipulator AUV, as per statement 23. The ordering of the statements would tend to suggest the second (incorrect) option, whilst the context of the statements suggests the first (correct) option.

The second critical incident is the result of the *Glaykos* system attempting to rearrange the events in order to fit its model of a *narrative order*:

- 50 (26) + It started to scan under installation bravo and put its installation scanner in active mode.
- 51 (27) - Skipping forward by a quarter of a minute
- 52 (27) + It finished doing this and put its installation scanner in passive mode.
- 53 (28) + The desire that installation bravo should be examined was satisfied.
- 54 (29) + In order that installation charlie should be examined, the Manipulator A.U.V. started to lower installation bravo.

- 55 (30) - Skipping forward by 2 minutes and 37 seconds
- 56 (30) + It finished doing this and put its lifting arm in passive mode.
- 57 (31) + It started to approach site charlie.
- 58 (31) - This was possible after the Manipulator A.U.V. lowered installation bravo.
- 59 (32) - Skipping backward by 3 minutes and 12 seconds
- 60 (32) + The Inspection A.U.V.'s installation scanner detected that installation bravo is malfunctioning.

In this case, the potential for user confusion arrises for the fact that an event in the mission has been moved out of chronological order so that the description of one causal path can be completed, and a series of parallel events established, before another is begun. This has the potential in decrease the user's cognitive load, but may also give the user the impression that no problems were found with installation bravo, only to then confound this assumption by rewinding back to the point at which installation bravo's malfunction was discovered. The discourse then continues to the entire of the causal path which results in installation bravo being repaired, without interruption.

The third critical incident in the offshore maintenance scenario is also narrative order based:

- 85 (45) - Skipping forward by 43 seconds
- 86 (45) + [The Inspection A.U.V.] finished moving to site charlie.
- 87 (45) - The Inspection A.U.V.'s scanning under installation charlie means that the desire that installation charlie should be examined will be satisfied.
- 88 (45) + It started to do this and put its installation scanner in active mode.
- 89 (45) - The status of the installation will be detected.
- 90 (45) - This was possible after the Inspection A.U.V. moved to site charlie.
- 91 (46) + It finished scanning under installation charlie and put its installation scanner in passive mode.
- 92 (47) + The desire that installation charlie should be examined was satisfied.
- 93 (48) + The desire that all installations should be examined was satisfied.
- 94 (49) - Skipping backward by 25 seconds

- 95 (49) + The Inspection A.U.V.'s installation scanner detected that installation charlie is broken.

Aside from the fact that this series of events refers to installation charlie, rather than bravo, and that no actions which are occurring in parallel are established, this incident is essentially identical to the previous incident. The discourse then follows through with the causal path which results in installation charlie being taken to the base location, and subsequently the completion of the mission. The fourth, and final, critical incident occurs during this delivery:

- 99 (52) - Skipping forward by 11 seconds
- 100 (52) + In order that installation charlie should be at the base location, the Manipulator A.U.V. started to lower installation charlie.
- 101 (53) - Skipping forward by 2 minutes and 37 seconds
- 102 (53) + It finished doing this and put its lifting arm in passive mode.
- 103 (53) - It released the grabbed object.
- 104 (54) + The Manipulator A.U.V. started to grab installation charlie and put its lifting arm in active mode.
- 105 (54) - This was possible after the Manipulator A.U.V. lowered installation charlie.
- 106 (55) + It finished grabbing installation charlie.
- 107 (56) + It started to move to the base location.
- 108 (56) - This was possible after the Manipulator A.U.V. lowered installation charlie.

In this case the planner is the cause of the issue. In order to ensure that reasonably optimal plans are created, without increasing the complexity of the world description, the actions to grab an installation in order to lift it, and to grab an installation in order to move it were separated. This results in the AUV needing to put the installation down, only to immediately grab it again in order to move it , for reasons which are unlikely to be obvious to the user. The excerpt given in Section 8.1.3 follows on directly from this incident.

Table 8.1 summarises the potentially critical incidents in the offshore maintenance scenario. As previously noted, a representation of the entire discourse generated for the offshore maintenance scenario, complete with all of the addition elements described in Section 7.4, can be found in Appendix G.

| Name | Category | Affected Process Statements |
|---|---|---|
| Offshore 1 | Realiser | 12-13 |
| Offshore 2 | Order | 32 |
| Offshore 3 | Order | 49 |
| Offshore 4 | Planner | 52-56 |

Table 8.1: Potentially critical incidents in the discourse for the offshore maintenance scenario.

## 8.2 Mine Counter Measures

Mine Counter Measures, or MCM, is one of the most prevalent uses of AUVs. In particular, it is one of the scenarios most often used for military demonstrations of AUV capabilities. In the standard scenario, there is an area of ocean floor which is thought to contain one or more underwater mines, which should be "neutralised", together with zero or more other objects, which are mine like in appearance, but non-threatening. This latter class of objects is to be detected, labelled as such, and then essentially ignored.

### 8.2.1 Initial State and Mission Goals

Three classes of vehicle are used during the mission. The first is designed to quickly explore the area, performing a broad sensor sweep in order to identify any mine like objects. The second is designed to inspect these objects more closely and determine whether they are a mine or not. The final class of vehicle is designed to navigate back to identified mines and destroy them, generally via an explosive charge.

In the scenario used here, there are two mines and a single "rock" within the search area. There is a single *Search AUV* of the first class, a single *Classification AUV* of the second class, and three *Destroy AUV*s. At the start of the mission the AUVs are not aware of any of the objects in the world. The physical layout of the mission is as shown in Figure 8.3.

The high level goals (meta-desires) of the mission are:

- The mission area should be searched.
- All detections should be inspected.
- All mines should be destroyed.

### 8.2.2 Implementation

Focusing on the actual actions of the AUVs, the salients facts regarding the implementation of the mine counter measures mission are as follows:

- The Transit AUV searched the mission area;

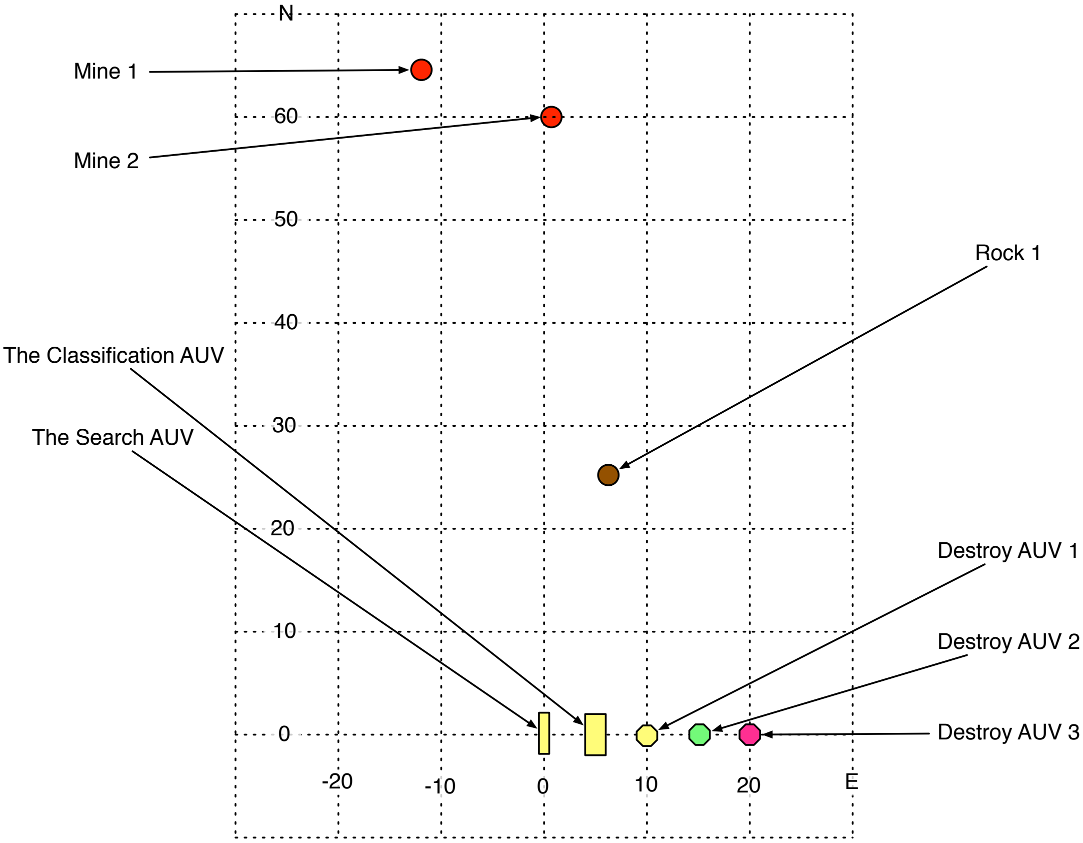


Figure 8.3: The initial physical layout of the Mine Counter Measures Mission.

- The Transit AUV discovered detection 1;
- The Inspection AUV examined detection 1;
- The Inspection AUV discovered a rock;
- The Transit AUV discovered detection 2;
- The Inspection AUV examined detection 2;
- The Inspection AUV discovered mine 1;
- Destroy AUV 1 destroyed mine 1;
- The Transit AUV discovered detection 3;
- The Inspection AUV examined detection 3;
- The Inspection AUV discovered mine 2;
- Destroy AUV 2 destroyed mine 2.

In this mission, all of the AUVs' actions are essentially independent, and do not depend on the actions of any other AUV. As a result the mission is carried

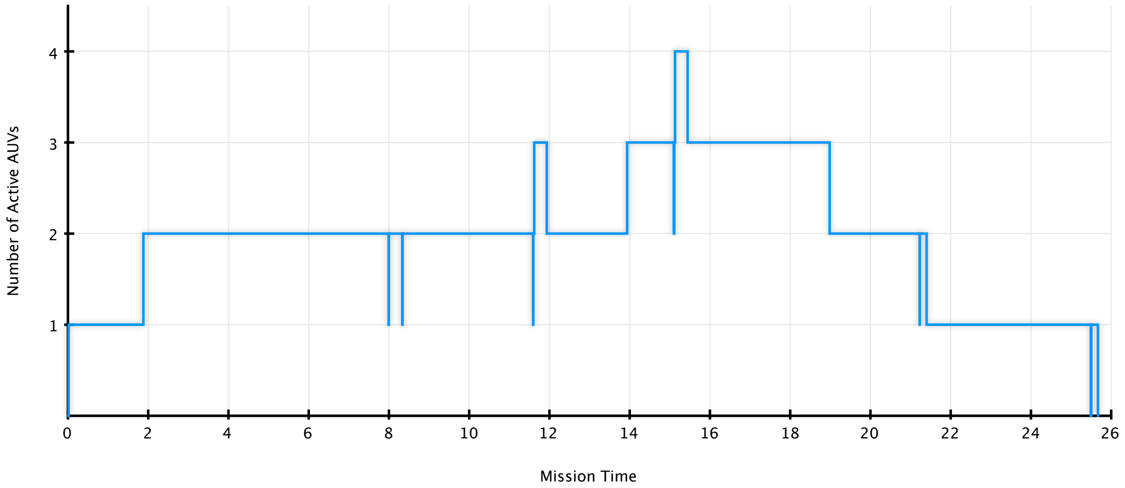


Figure 8.4: The graph of activity during the Mine Counter Measures Mission.

out in a reasonably parallel fashion, with a smaller number of causal links than the offshore maintenance scenario. The parallelism graph which represents this is shown in Figure 8.4.

### 8.2.3 Discourse Excerpt

The section provides a short excerpt from the start of the discourse generated for the mine counter measures scenario, featuring the dialogue which is given to the user. As before, each line of dialogue is preceded by its index within the discourse, the index of the process element to which it is linked in parentheses, and a symbol representing the nature of the statement ( "+" for process statements and "-" for stasis and supporting statements).

As this example shows the very start of the debrief, the high level goals of the mission are first established, followed by the more concrete goals which specific tasks are assigned to. In this instance, only the search task requires the creation of a plan at the start of the mission. The implementation of this plan begins immediately, with statement 8 ensuring the user is aware of the specific motivation behind the particular action being carried out. A detection is quickly discovered, and a plan created to ensure its classification in line with the high level goals:

- 1 (1) - It is desired that the mission area should be searched.
- 2 (1) - It is desired that all detections should be examined.
- 3 (1) - It is desired that all mines should be destroyed.
- 4 (2) + The desire that the mission area should be searched was introduced.
- 5 (3) + A plan was created to satisfy this.

- 6 (4) - The Transit A.U.V. has the search for detections module and so it has a side-scan and is able to search.

- 7 (4) - Its searching the mission area means that the desire that the mission area should be searched will be satisfied.

- 8 (4) + In order that the mission area should be searched, it started to do this and put its side-scan in active mode.

- 9 (4) - In active mode the sidescan will detect objects on the ocean floor.

- 10 (5) - Skipping forward by 1 minute and 40 seconds

- 11 (5) + The Transit A.U.V.'s side-scan detected that detection one is at location north 25 east 8 depth 13.

- 12 (6) + The desire that detection one should be examined was introduced.

- 13 (7) + A plan was created to satisfy this.

A longer excerpt, together with selected graphics transitions, is provided for the combination scenario in Section 8.1.3.

### 8.2.4 Potentially Critical Incidents in Discourse

The narrative order based potentially critical incidents in the mine counter measures scenario are very similar to those found in the offshore maintenance scenario. The first example is shown below:

- 34 (19) + [The Inspection A.U.V.] started to examine detection two at location north 60 east 1 depth 10 and put its camera in active mode.

- 35 (20) - Skipping forward by a quarter of a minute

- 36 (20) - It finished doing this and put its camera in passive mode.

- 37 (21) + The desire that detection two should be examined was satisfied.

- 38 (22) - Skipping backward by 26 seconds

- 39 (22) + The Inspection A.U.V.'s camera detected that mine one is at location north 60 east 1 depth 10.

Two further examples of this type of ordering can be found later in the discourse. Once again, the original causal thread is completed before the discourse moves to the second, potentially leaving the user with the mistaken impression that no discovery was made (see Section 7.1.6).

Also found in this discourse were two incidents which are attributed to the planning system's need for actions to be completed:

- 45 (27) - It has the destroy mines module and so it has a bomb and is able to destroy mines.

- 46 (27) - Its destroying mine one means that the desire that mine one should be destroyed will be satisfied.

- 47 (27) + It started to do this and put its bomb in active mode.

- 48 (27) - When put in active mode, the bomb will destroy any mine within five metres.

- 49 (27) - This was possible after Destroy A.U.V. 1 approached location north 60 east 1 depth 10.

- 50 (28) + It finished destroying mine one and put its bomb in passive mode.

In the context of a real mission, this is clearly an erroneous statement; a bomb cannot be placed in "passive" mode, and a vehicle which has deployed its bomb is extremely unlike to the able to communicate this, or any other, fact after the action of activating the bomb has been carried out. The planning system as it stands, however (see Section 4.1), requires actions to be completed in order that their effects be carried through in to the world state, allowing the relevant mission goals to be marked as complete.

The second example of a narrative order based incident also includes a second incident of a more complex nature within it:

- 52 (29) - Skipping backward by 2 minutes and 41 seconds

- 53 (29) - The Transit A.U.V. has the search for detections module.

- 54 (29) - Its searching the mission area means that the desire that the mission area should be searched will be satisfied.

- 55 (29) + In order that the mission area should be searched, it finished doing this and put its side-scan in passive mode.

- 56 (29) - In passive mode the sidescan will detect nothing.

- 57 (30) + The desire that the mission area should be searched was satisfied.

- 58 (31) + The desire that the mission area should be searched was satisfied.

- 59 (32) - Skipping backward by 5 minutes and 11 seconds

- 60 (32) + The Transit A.U.V.'s side-scan detected that detection three is at location north 65 east -11 depth 10.

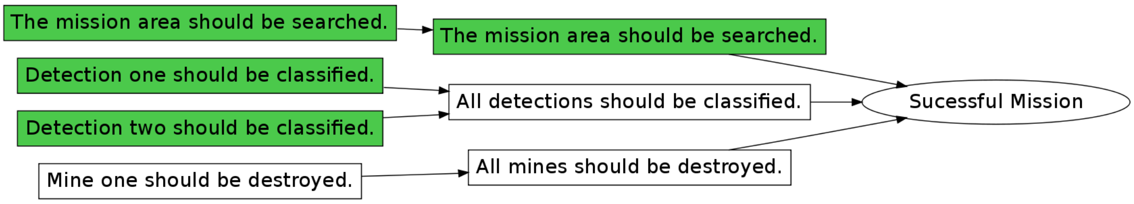

The mission area should be searched.
Detection one should be classified.
Detection two should be classified.
Mine one should be destroyed.
The mission area should be searched.
All detections should be classified.
All mines should be destroyed.
Sucessful Mission

Figure 8.5: The desire visualization for the point in the mission at which the search task is completed, showing a similar repetition to that which is found in the dialogue.

Aside from the fact that this example deals with the search, rather than the classify, task, it is extremely similar in nature to the first incident discussed in this section. Contained within it it, however, is an apparently repeated statement (statements 57 and 58). In fact the first of these is a *desired satisfaction* process statement, while the second is a *meta-desire satisfaction* process statement (see Section 6.1.2). The repetition is a result of the meta-desire's static nature. This is illustrated further in Figure 8.5, which shows that the desire visualization (see Section 7.4.4) for this stage of the mission also includes the repetition. The interaction between the dialogue and visualization systems in the current implementation makes repeated statements such is this difficult to collapse into a single statement. As such, it is not possible to classify this incident as specifically being the result of narrative order, planning or realisation, and it has been given the classification of "compound" in order to represent this.

The third example of a narrative incident is almost identical to the first, aside from the fact that it also incloses the completion of the classification task:

- 70 (38) + It finished examining detection three at location north 65 east -11 depth 10 and put its camera in passive mode.

- 71 (39) + The desire that detection three should be examined was satisfied.

- 72 (40) + The desire that all detections should be examined was satisfied.

- 73 (41) - Skipping backward by 27 seconds

- 74 (41) + The Inspection A.U.V.'s camera detected that mine two is at location north 65 east -11 depth 10.

Similarly, the final incident is essentially identical to the previous planner based incident:

- 85 (47) + It finished destroying mine two and put its bomb in passive mode.

Table 8.2 summarises the potentially critical incidents in the mine counter measure scenario.

| Name | Category | Affected Process Statements |
|---|---|---|
| MCM 1 | Order | 22 |
| MCM 2 | Planner | 28 |
| MCM 3 | Compound | 31 |
| MCM 4 | Order | 32 |
| MCM 5 | Order | 41 |
| MCM 6 | Planner | 47 |

Table 8.2: Potentially critical incidents in the discourse for the mine counter measures scenario.

## 8.3 Combination

The "combination" scenario is a combination of the offshore maintenance and mine counter measures scenarios. As such it contains all of the objects and AUVs from these two missions, operating at the same time in the same space. This scenario is highly unlikely in real life. It is possible that both tasks would need to be carried out in the same space, however the MCM mission would almost certainly be carried out first, in order to avoid the inherent danger to the other vehicles caused by operating in a potential mine field. This scenario makes the assumption that both missions are time critical and asks the participant to disregard the unlikelihood of them being carried out simultaneously.

### 8.3.1 Initial State and Mission Goals

The mission was designed to provide more challenging debrief for domain experts, who in particular would be very familiar with the mine counter measures scenario. In order to increase its complexity, the positions of the objects in the world have also be altered, in order to ensure that in some cases important events happen almost simultaneously, but in different areas of the map, and with different motivations behind them.

This mission inherits all of the high level goals of the previous two missions, together with all of their AUVs and objects. As previously mentioned, some of the start positions have been altered, however. The resultant physical layout of the mission is as shown in Figure 8.6.

### 8.3.2 Implementation

As with the goals, the salient facts regarding the implementation of this mission are the union of those for the offshore maintenance and mine counter measures missions, which can be found in sections 8.3.2 and 8.2.2.

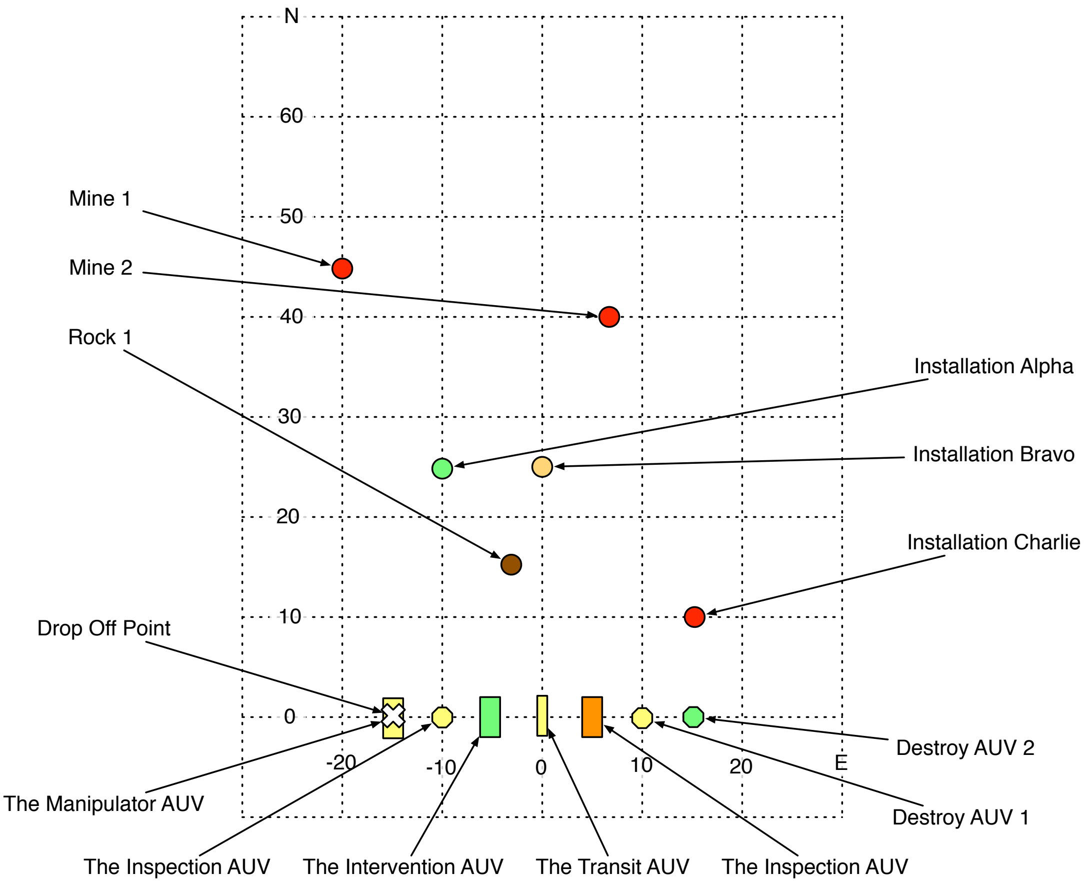


Figure 8.6: The initial physical layout of the Combination Mission.

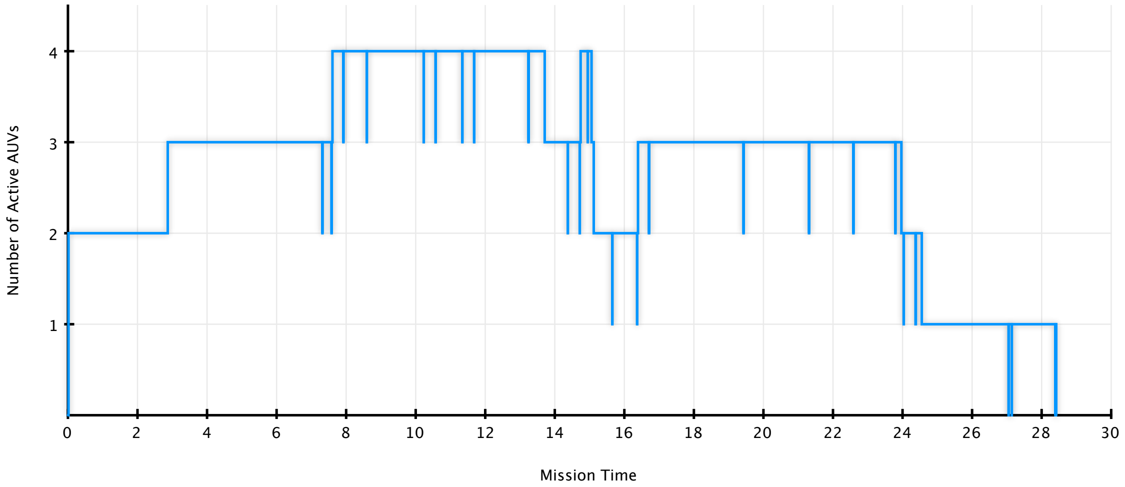
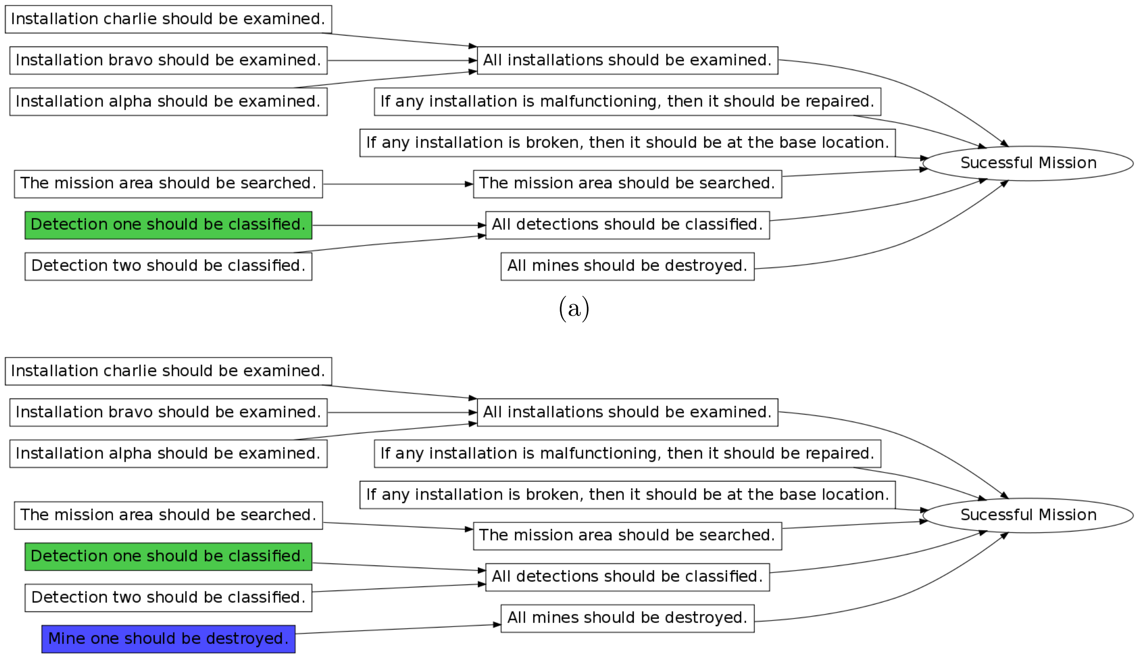


Figure 8.7: The graph of activity during the Combination Mission.


Installation charlie should be examined.
Installation bravo should be examined.
Installation alpha should be examined.
All installations should be examined.
If any installation is malfunctioning, then it should be repaired.
If any installation is broken, then it should be at the base location.
The mission area should be searched.
The mission area should be searched.
Sucessful Mission
Detection one should be classified.
All detections should be classified.
Detection two should be classified.
All mines should be destroyed.
Mine one should be destroyed.

(a)

(b)

Figure 8.8: The graphics progression for the introduction of the desire that mine one should be destroyed in the combination scenario.

As a result of the combination of the two original scenarios, this mission has has a very high degree of parallelism, which is shown in Figure 8.7.

### 8.3.3 Discourse Excerpt

The section will provide an excerpt from the discourse generated for the combination scenario. Due to the high level of parallelism in this scenario, its discourse is most apt to show some of the capabilities of the *Glaykos* system. As before, each line of dialogue will be preceded by the number of the statement in the discourse, the index of the relevant process statement in parentheses and a symbol representing the nature of the statement ( "+" for *process statements* and "-" for *stasis* and *supporting statements*). In addition, for illustrative purposes the graphics progression which accompanies the dialogue will also be given for selected statements.

The excerpt begins with the discovery of the first mine in the mission, the changes which are made to the plan in response, and the start of the implementation of this new plan:

- 41 (23) + [The Classification A.U.V]'s camera detected that mine one is at location north 40 east 8 depth 13.
- 42 (24) + The desire that mine one should be destroyed was introduced. (See Figure 8.8 for graphics)

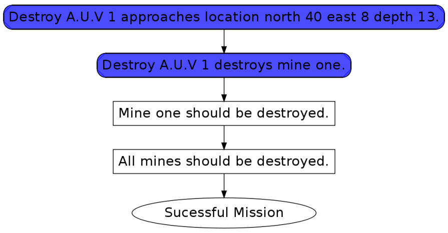

Destroy A.U.V 1 approaches location north 40 east 8 depth 13.
Destroy A.U.V 1 destroys mine one.
Mine one should be destroyed.
All mines should be destroyed.
Sucessful Mission

Figure 8.9: The graphics displayed for the introduction of the plan to destroy mine one in the combination scenario.

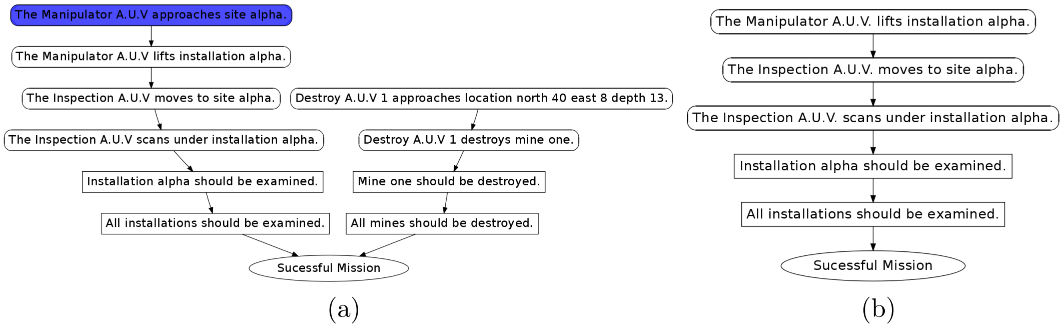

The Manipulator A.U.V approaches site alpha.
The Manipulator A.U.V lifts installation alpha.
The Inspection A.U.V moves to site alpha.
Destroy A.U.V 1 approaches location north 40 east 8 depth 13.
The Inspection A.U.V scans under installation alpha.
Destroy A.U.V 1 destroys mine one.
Installation alpha should be examined.
Mine one should be destroyed.
All installations should be examined.
All mines should be destroyed.
Sucessful Mission
(a)
The Manipulator A.U.V. lifts installation alpha.
The Inspection A.U.V. moves to site alpha.
The Inspection A.U.V. scans under installation alpha.
Installation alpha should be examined.
All installations should be examined.
Sucessful Mission
(b)

Figure 8.10: The graphics progression for the completion of the Manipulator AUV's approach to site alpha, which also moves between different motivations via a dual display.

- 43 (25) + A plan was created to satisfy this. (See Figure 8.9 for graphics)
- 44 (26) + In order that mine one should be destroyed, Destroy A.U.V 1 started to approach location north 40 east 8 depth 13.

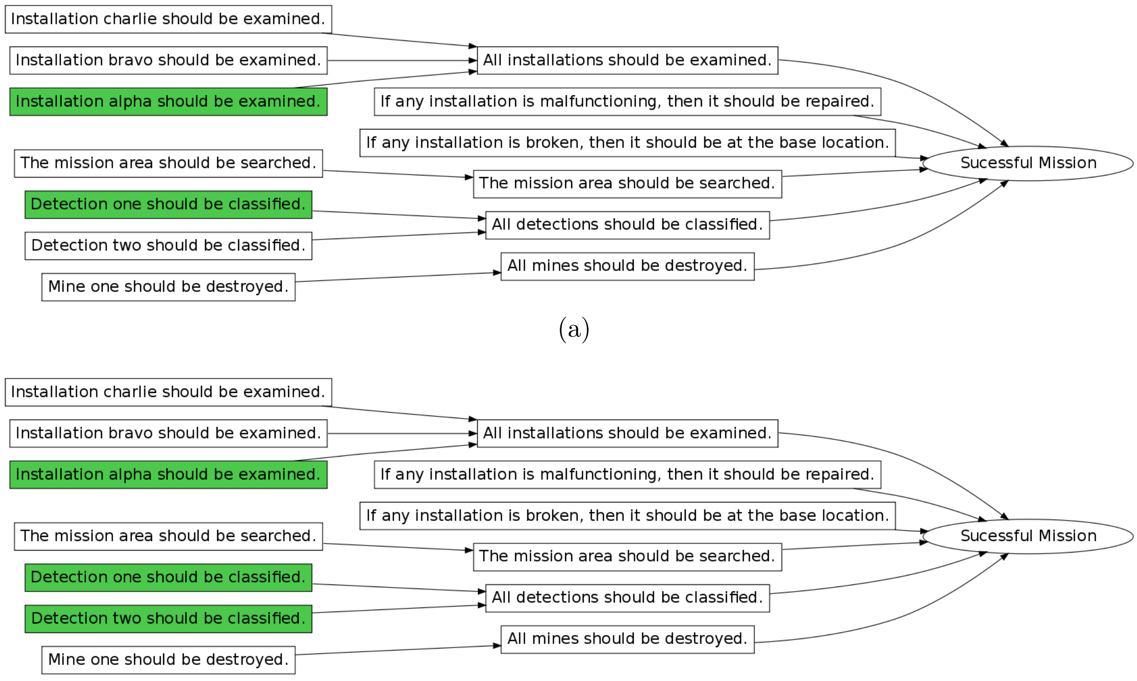
While the Destroy AUV moves towards its target, the system rewinds slightly in order to follow the simultaneous actions of the AUVs which are working on the goals from the offshore maintenance scenario. It can be seen here that the system notifies the user when shifting between different motivations, such as at the beginning of statement 46:

- 45 (27) - Skipping backward by 24 seconds
- 46 (27) + In order that installation alpha should be examined, the Manipulator A.U.V finished approaching site alpha. (See Figure 8.10 for graphics)
- 47 (28) - It has the lift objects module and so it has a lifting arm and is able to grab objects, to lift objects, to lower objects and to release objects at the new location.
- 48 (28) + It started to lift installation alpha and put its lifting arm in active mode.
- 49 (28) - In active mode the lifting arm will grab the nearest object.

- 50 (29) - Skipping forward by 1 minute and 9 seconds
- 51 (29) + The Manipulator A.U.V finished lifting installation alpha.
- 52 (30) + The Inspection A.U.V started to move to site alpha.
- 53 (31) - Skipping forward by 1 minute and 36 seconds
- 54 (31) + It finished doing this.
- 55 (32) - It has the examine installations module and so it has an installation scanner and is able to scan under installations.
- 56 (32) + It started to scan under installation alpha and put its installation scanner in active mode.
- 57 (32) - In active mode the installation scanner will detect the status of an installation.
- 58 (33) + The Inspection A.U.V's installation scanner detected that installation alpha is working correctly.
- 59 (34) - Skipping forward by 13 seconds
- 60 (34) + It finished scanning under installation alpha and put its installation scanner in passive mode.
- 61 (34) - In passive mode the installation scanner will detect nothing.
- 62 (35) + The desire that installation alpha should be examined was satisfied.
- 63 (36) + In order that installation bravo should be examined, the Manipulator A.U.V started to lower installation alpha.

Having described a set of events for the offshore maintenance task, the narrative now switches back to the mine counter measures task, fast forwarding slightly to find the classification AUV finishing a task:

- 64 (37) - Skipping forward by 37 seconds
- 65 (37) + In order that detection two should be classified, the Classification A.U.V finished moving above location north 15 east -4 depth 15.
- 66 (38) + It started to examine detection two and put its camera in active mode.
- 67 (39) + Its camera detected that a rock is at location north 15 east -4 depth 15.


Installation charlie should be examined.
Installation bravo should be examined.
Installation alpha should be examined.
All installations should be examined.
If any installation is malfunctioning, then it should be repaired.
If any installation is broken, then it should be at the base location.
The mission area should be searched.
Sucessful Mission
The mission area should be searched.
Detection one should be classified.
All detections should be classified.
Detection two should be classified.
All mines should be destroyed.
Mine one should be destroyed.
Installation charlie should be examined.
Installation bravo should be examined.
Installation alpha should be examined.
All installations should be examined.
If any installation is malfunctioning, then it should be repaired.
If any installation is broken, then it should be at the base location.
The mission area should be searched.
Sucessful Mission
The mission area should be searched.
Detection one should be classified.
All detections should be classified.
Detection two should be classified.
All mines should be destroyed.
Mine one should be destroyed.

(a)

(b)

Figure 8.11: The graphics progression for the satisfaction of the desire that detection two should be examined

- 68 (40) - Skipping forward by a quarter of a minute
- 69 (40) + It finished examining detection two and put its camera in passive mode.
- 70 (41) + The desire that detection two should be classified was satisfied. (See Figure 8.11 for graphics)

As the mine counter measures task has a high level of parallelism with it, other tasks are being carried out simultaneously to this. The narrative rewinds slightly in order to focus on the events surrounding the third detection discovered by the search AUV:

- 71 (42) - Skipping backward by 26 seconds
- 72 (42) + The Search A.U.V's side-scan detected that detection three is at location north 45 east -20 depth 15.
- 73 (43) + The desire that detection three should be classified was introduced.
- 74 (44) + A plan was created to satisfy this.
- 75 (45) + In order that detection three should be classified, the Classification A.U.V started to move above location north 45 east -20 depth 15.
- 76 (46) - Skipping forward by 3 minutes

- 77 (46) + It finished doing this.
- 78 (47) + It started to examine detection three and put its camera in active mode.
- 79 (47) - Rocks and mines will be identified.

This example discourse has been taken for a section of the mission which contains a high degree of parallelism, leading to the shifts between tasks which are shown. These are the result of the system attempting to keep causally linked events together, whilst simultaneously trying to avoid numerous large shifts forwards and backwards in time.

In total, 34 of the statements which make up the complete discourse have been given here, whilst the complete discourse contains 180. This example has also limited the number of graphics progressions provided for brevity, and not given the exact timing used for the delivery of each element for readability. The complete discourse for the offshore maintenance scenario is given in Appendix G, which provides every element of the discourse, together with all of the timings.

### 8.3.4 Potentially Critical Incidents in Discourse

As with the goals and implementation, the set of potentially critical incidents in the discourse for the combination scenario is largely the union of those for the other two scenarios. For obvious reasons, all of the planner based incidents are present in near identical form. Equivalents to all but one of the narrative order based incidents are also present, as the change in ordering has eliminated incident MCM 3 (see Table 8.2). Likely due to differences in ordering, the realiser based incident found in the offshore maintenance scenario (see Table 8.1) is also not present. Rather than providing a exhaustive and repetitive record of the incidents found in the discourse, this section will reference back to the incidents from the other scenarios where appropriate, and describe the one incident which shows a serious difference from it's equivalent.

Table 8.3 gives a summary of the incidents found in this discourse, with the equivalent to incident 3 being marked in red due to differences. Where as the previous narrative order based incidents have taken the form discussed in Section 7.1.6, in which a discovery is revealed after the completion of the event which triggered it, the separation is much greater in this instance:

- 78 (47) + It started to examine detection three and put its camera in active mode.
- 79 (47) - Rocks and mines will be identified.

| Name | Equivalent | Category | Affected Process Statements |
|---|---|---|---|
| Combination 1 | MCM 1 | Order | 23 |
| Combination 2 | MCM 3 | Compound | 53 |
| Combination 3 | MCM 4 | Order | 58 |
| Combination 4 | MCM 2 | Planner | 64 |
| Combination 5 | Offshore 2 | Order | 72 |
| Combination 6 | MCM 6 | Planner | 90 |
| Combination 7 | Offshore 3 | Order | 98 |
| Combination 8 | Offshore 4 | Planner | 101-104 |

Table 8.3: Potentially critical incidents in the discourse for the combination scenario. The equivalence value for incident 3 is marked in red in order to indicate that it shows differences to its equivalent in the mine counter measures scenario.

- 80 (48) - Skipping forward by 11 seconds
- 81 (48) + The Classification A.U.V finished examining detection three and put its camera in passive mode.
- 82 (49) + The desire that detection three should be classified was satisfied.
- 83 (50) + The desire that all detections should be classified was satisfied.
- ...
- 98 (58) - Skipping forward by 12 seconds
- 99 (58) + The Classification A.U.V's camera detected that mine two is at location north 45 east -20 depth 15.

A total of fourteen statements, covering seven process statements, are delivered to the user between the completion of the causal strand which the led to the detection occurring and user being informed of the detection. As a result, this is almost certainly the most extreme of all of the narrative order based potentially critical incidents.

## 8.4 Summary

Three missions were used in order to create debriefs for use in the experiments described in Chapter 9. The first of these was an offshore style inspection and repair scenario with a low degree of parallelism, but a high degree of inter-agent dependency. The second was a short military mine counter measures mission with a higher degree of parallelism, but a lower degree of inter-agent dependency. Lastly, a "combination" mission was created in order to provide a more complex scenario for expert participants. This combined all elements of the two other scenarios, and so featured a large degree of both parallelism and inter-agent dependency.

This chapter has described each of these scenarios in terms of the overall mission goals and initial world state, together with the pertinent facts from the implementation of the mission. An excerpt was provided from each discourse, together with descriptions of the potentially critical incidents which in the generated discourse which had the potential to give the participant some difficulty. Additionally, the extended excerpt provided for the combination mission also provided selected graphics transitions as well as the dialogue.

The next part of this thesis will describe the methodology which was employed to the test the effectiveness of these debriefs, along with the results of these tests and a discussion of this work as a whole.

# Part III

# Resolution: Experimentation and Discussion

> “
>
> He reached out and pressed an invitingly large red button on a nearby panel. The panel lit up with the words *Please do not press this button again.*
>
> ”
>
> *“The Hitchhiker’s Guide to the Galaxy” by Douglas Adams*

The third part of this thesis is modeled after the third act found in many forms of narrative, and sometimes referred to as the “resolution.” In the resolution, the protagonist must face the antagonist in a final confrontation, and in doing so either succeed or fail in their aims.

Mirroring this, Part 3 contains three chapters, which are geared towards the testing of the system described in the previous part. Firstly, in Chapter 9 the experimental methodology is discussed, together with the scenarios which are used for experimentation and the expectations for the outcome. Next, Chapter 10 gives the myriad results of these experiments. Chapter 11 follows this to discuss these results and comment on the effectiveness of the created system, whilst also taking a step back to consider the work described here as a whole, and finally discusses potential future work which follows on from the system described herein, further developing the framework described in Chapter 3.

# Chapter 9

# Experimental Methodology

This chapter describes a set of experiments which were designed to informally prove the *Glaykos* system, and then carried out with human participants. These experiments were designed to compare the effectiveness of the generated debriefs both to the current state of the art and to a control case consisting of the same data, but delivered in strict chronological order and without many of the narrative inspired simplifications.

Human participants were recruited for the initial set of tests via emails sent to the mailing lists of the Ocean Systems Laboratory and SeeByte Ltd, as well as from the larger community in Edinburgh via an advert posted on a popular social networking website. The intention was that the participants should come from as wide a background as possible. In particular, it was considered beneficial that at least a portion of the participants should have little to no experience of AUVs and the other underlying technologies employed in this project. This would help avoid any preconceived notions about how a debrief should be carried out. These tests deployed a relatively soft incentive: participants were offered their choice from a selection of home made cakes.

A second batch of tests was planned, in which participants would have been recruited via emails sent across the entire of the university with a financial incentive. Unfortunately this had to be cancelled due to time constraints.

Based on the information gained in a background information survey (see Appendix H.1), the participants were split into three groups. The first, relatively small, group consisted of those with very high levels of experience of Autonomous Underwater Vehicles and related control systems. The remaining participants were initially split equally into two groups, balanced such that the average scores for the experience based sections of the background information survey for each group were as closely matched as possible.

The remainder of this chapter is divided into three sections. The first of these describes the actual experiments which were carried out and the second describes the results which are expected from these experiments, in broad terms. Finally, the third section summaries the chapter.

## 9.1 The Experiments

In total, three distinct experiments were carried out. Two of these were designed at the inception of the experimental phase, while the last was constructed in response to deficiencies discovered during the initial batch of experimentation. Each of these experiments will now be described.

### 9.1.1 Experiment 1

The first experiment was designed to compare the output of the Narrative Debriefing System to a **base case**, which delivered the same information, but in a strict chronological order, without any of the markup related to the motivation of the agents which the *Glaykos* System creates. The output created by the *Glaykos* System will henceforth be referred to as the **experimental case**. In both cases, the optional pause script (see Section 4.3.4) was used to add a pause after the delivery of each process statement and its accompanying stasis statements,

Each participant viewed both cases, so two distinct mission scenarios were used to ensure that there was a minimum of crossover between the information contained in the two debriefings. The two utilised scenarios were based upon offshore maintenance and mine countermeasures, two of the most common applications which require the use of autonomous underwater vehicles as something more than a mobile sensor platform. The scenarios themselves were described in more detail in Sections 8.1 and 8.2. Before each debrief, each user viewed an appropriate set of instructions for the scenario and case they were about to view. Examples of these can be found in Appendix H.2.

After each debriefing, the participant was asked to fill in a short questionnaire. In this they were asked to give a simple subjective assessment of how well they think they understood the mission. Next they were asked to explain the scenario in their own words (in order to gain a more objective assessment of how well they understood the mission). Finally they were asked to state in which ways the debriefing made the mission easy to understand, and in which ways it made it harder to understand. In the case of the questionnaire given after the second debriefing, an additional question is added, which asked the participant which of the two debriefing methodologies they preferred. Examples of these surveys can be found in Appendix H.4.

The intention is not to gauge the participants' short term recall of the mission they have observed, rather it is to ascertain their actual understanding of why the mission proceeded as it did, and their long term retention of the events within the mission. This being the case, the ideal situation would be to have the participant fill in the questionnaire some time after they had observed the debriefing, ideally a time in the order of a day. Additionally, the two cases themselves would ideally be viewed around a week apart. This would introduce a considerable increase in the complexity of the logistics required to carry out the experiment, however, and could potentially lead to some participants being unavailable for the second case. It might also lead to the surveys not being filled in at all, or being left a longer time than that specified.

As an alternative, the decision was made to use three Pixar short films to separate the two debriefs from their accompanying questionnaire, and also to separate the two scenarios. These were selected as they are all around (a logistically ideal) five minutes in length. They are also generally well regarded and considered enjoyable to watch, making the introduction of boredom on the part of the participant unlikely. Additionally, each contains a short, yet complete narrative, with (in the case of those selected) no dialogue. This ensures that the details of the debrief are removed from the participant's short-term and working memories, meaning they are forced to rely upon their long-term memory. Furthermore, the lack of dialogue means that the user is unlikely to have been "primed" with any words or phrases. As an additional benefit, this approach also means that each participant has essentially the same "experience" and timings for the duration of the experiment.

The participants were divided into two groups, the first viewing the offshore maintenance mission with the *experimental case*, followed by the mine countermeasures mission with the *base case*, and the second with the two cases reversed. This was an attempt to firstly remove any bias which might come from seeing one of the cases first, and secondly to make it possible to directly compare the participants' opinions of the two cases applied to the same scenario.

The experimental schedule was as follows:

1 Participant reads the relevant instructions for Offshore Maintenance scenario debrief with *experimental case* for group 1 or the *base case* for group 2.

2 Participant asks any questions.

3 Participant views debrief for Offshore Maintenance scenario debrief with relevant case (*experimental* 9m 28s, *base* 11m 10s).

4 Instructions are removed.

5 Participant watches Pixar short film "Geri's Game" (4m).

6 Participant fills in survey 1.

7 Participant watches Pixar short film "For the Birds" (3m 25s).

8 Participant reads the instructions for Mine Countermeasures scenario debrief with *base* (group 1) or *experimental* (group 2) *case*.

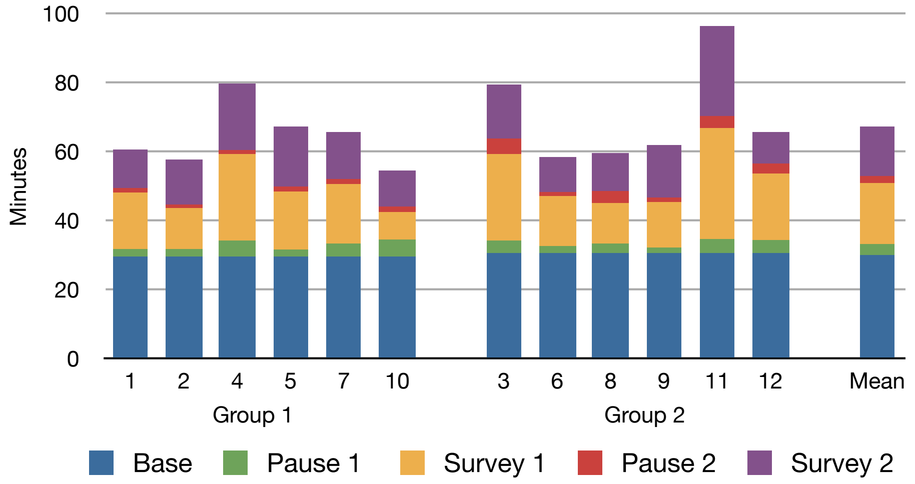


Figure 9.1: The timing for each of the participants in Experiment 1.

9 Participant asks any questions.

10 Participant views debrief for Mine Countermeasures scenario debrief with relevant case (*base* 8m, *experimental* 7m 25s).

11 Instructions are removed.

12 Participant watches Pixar short film "One Man Band" (4m 33s).

13 Participant fills in survey 2.

The total time for the experiment was the base time taken for the two scenarios plus the three short films (29m 27s for group one and 30m 33s for group 2) combined with the participants' pauses for the two scenarios and the time taken to fill in the two surveys. The time taken by each of the twelve participants, together with the mean time, is recorded in Figure 9.1. The average time taken for this experiment was slightly less than one hour and ten minutes.

### 9.1.2 Experiment 2

After an initial batch of experimentation was conducted under the conditions described above, it become clear that while some interesting data was being collected, there were some fundamental flaws with the comparison being performed.

- The *base case* and the *experimental case* are very similar in nature, both taking the form of an automated replay using synthesised speech. As such, there were reduced grounds for comparison.
- The *base case* does not accurately reflect the current state of the art for the debrief of AUV missions, and so a comparison to this is not being made.

- Using the total time the user took to view the debrief as a measure of how long the user took to process each unit of information provided unsatisfactory results, as:
  - The results are distorted; some users took a long time to push the space bar to continue after the first unit of information was delivered, as they then reread the instructions to ensure they were doing doing the right thing. Others mistakenly pushed the space bar an additional time at the end of the debrief, preventing it from completing and extending the time.
  - The results are occluded; all that is known is the total time taken, not the pause taken after each piece of information. This makes it impossible to tell if the pause actually increases after statements which are proceeded by leaps in time or causation.

With these factors in mind, a new set of experimental conditions was created. First of all, the *base case* was replaced with a new test. This test, called the **user case**, gave the user complete control of the debrief, using the system described in section 4.3.3. No contextual information was provided to the user regarding the motivation of the AUVs or the causal relationships between the events. The only overlay provided simply displayed the name of the AUV or object which the user currently had selected. However, the instructions given to the participant at the start of the debrief were augmented to include more details regarding the scenario in question. An example of these instructions can be found in AppendixH.3. This creates a significantly larger difference between the two cases viewed by the participant, and more closely resembles the debriefing functionality provided by the current state of the art, SeeByte Ltd's SeeTrack software package[1] .

Secondly, the scripted playback system was augmented to record the amount of time taken by the user each time the system pauses. This will allow accidental pauses by the user to be ignored, as well as allowing excessively long pauses at the beginning of the debrief to be discounted.

With these changes made, the new experimental schedule was as follows (times in brackets are the minimum required):

1 Participant reads the relevant instructions for Offshore Maintenance scenario debrief with *experimental* (group 3) or *user* (group 4) *case.*

2 Participant asks any questions.

[1] In fact the functionality provided by the *user case* still exceeds that which is provided by SeeTrack (see Section 2.5), as the temporal control is more limited and only available in certain versions of the product, and the spatial control is limited to a two dimensional, top-down viewpoint. This difference is considered to be minor when compared to the difference between the *experimental* and *user cases*, however.

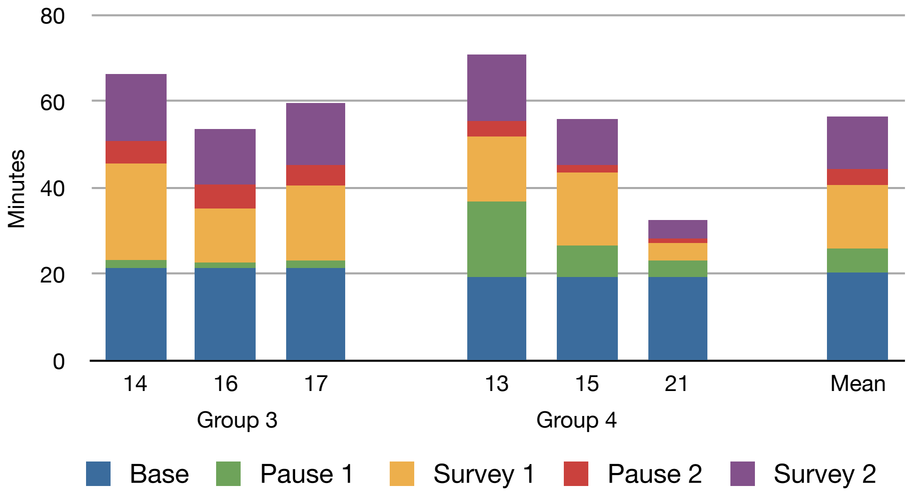


Figure 9.2: The timing for each of the participants in Experiment 2.

3. Participant views debrief for Offshore Maintenance scenario debrief with relevant case (*experimental* 9m 28s, *user* 0m 0s).
4. Instructions are removed.
5. Participant watches Pixar short film “Gerry’s Game” (4m).
6. Participant fills in survey 1.
7. Participant watches Pixar short film “For the Birds” (3m 25s).
8. Participant reads the instructions for Mine Countermeasures scenario debrief with *user* (group 3) or *experimental* (group 4) *case*.
9. Participant asks any questions.
10. Participant views debrief for Mine Countermeasures scenario debrief with relevant case (user 0m 0s, *experimental* 7m 25s).
11. Instructions are removed.
12. Participant watches Pixar short film “One Man Band” (4m 33s).
13. Participant fills in survey 2.

The total time for the experiment was the basic time taken for the two scenarios plus the three short films (21m 27s for group three and 19m 23s for group 4) combined with the participants’ pauses for the two scenarios and the time taken to fill in the two surveys. The time taken by each of the six participants, together with the mean time, is recorded in Figure 9.2. The average time taken for this experiment was slightly less than one hour.

### 9.1.3 Experiment 3

Experiment 3 was in part conceived at the same time as experiment 1, but performed last. A number of the experimental participants are domain experts in the area of Autonomous Underwater Vehicles, some in specific areas which are directly relevant to this project. For this reason it was felt that more complex scenarios would be required in order to carry out a similar level of testing. A particular area of concern is that the "Mine Counter Measures" scenario is very commonly used for demonstrations and explanations with AUVs and so is likely to be highly familiar to these participants. This is less likely to be the case with the "Offshore Maintenance" scenario, as this is a custom task which was specifically created for the testing of this project. This comparison of a familiar scenario to an unfamiliar one adds an additional and unwanted variable.

With this in mind, a new and more complicated scenario was created by combining the the original two (with some minor alterations). This new scenario was described in Section 8.3. No second additional scenario was created, so none was available to provide a second case for comparison. The availability of these participants was quite limited, however, so time likely would not have been available to test with a second scenario of equivalent complexity and length. This being this case, each participant was given either the *user* or *experimental case* and no other. The weighting was shifted from a 50:50 split to favour the user case, as this generates the most raw data.

The experimental methodology was identical to the first half of experiments 1 and 2 (times in brackets are the minimum required):

1 Participant reads the relevant instructions for Combination scenario debrief with *experimental* (group 5) or *user* (group 6) *case*.

2 Participant asks any questions.

3 Participant views debrief for Combination scenario debrief with relevant case (*experimental* 13m 29s, *user* 0m 0s).

4 Instructions are removed.

5 Participant watches Pixar short film "Gerry's Game" (4m).

6 Participant fills in survey 1.

The total time for the experiment was the basic time taken for the two scenarios plus the three short films (17m 29s for group five and 4m for group six) combined with the participants' pauses for the scenario and the time taken to fill in the survey. The time taken by each of the five participants, together with the mean time, is recorded in Figure 9.3. The average time taken for this experiment was around than thirty minutes.

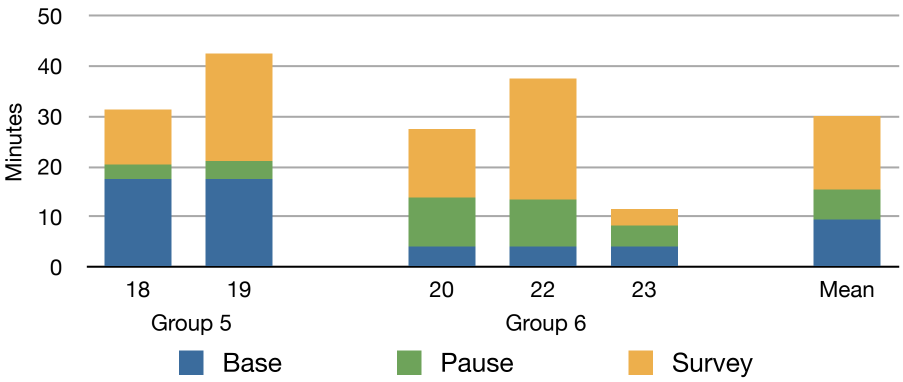


Figure 9.3: The timing for each of the participants in Experiment 3.

## 9.2 Expectations

The experiments carried out here are some ways intended to mirror those performed by Graesser et al. in *Advanced Outlines, Familiarity and Text Genre on Retention of Prose*[97] and Pennington and Hastie in their work[98, 99, 100, 101]. These found that when information was presented narratively, reading times were faster (in the first case) and information was more easily absorbed and retained (in both cases).

As the *experimental case* structures its information narratively, it was expected to produce lower pause times than the *base case*. Furthermore it is expected for it to give participants a better understanding of the mission than either the *base* or *user cases*. Reflecting this, it would also be expected that the participants' subjective assessment would prefer the *experimental case* over the *base case*.

The debrief presented here is a monologue, which is not the eventual goal of the framework *Glaykos* is in tended to become a component of (see Chapter 3). In this framework, user interaction is intended to be a duologue, as is suggested by the design rules discussed in Section 2.6. Especially in comparison to the *user case*, the author therefore suspects that that may be a negative reaction to the lack of user interaction afforded by the *experimental case*. Despite this, it is still felt that users will prefer the *experimental case* over the *user case*, due to the extra information it provides.

Given the *Glaykos* system's complete understanding of the mission, the author would expect it to be able to replay the events in an efficient order for user understanding, without the need to replay the same time period an excessive number of times. Thus it is expected that the total amount of mission replay to be lower for the *experimental case* than for the *user case*.

Again as stated in the human computer interaction rules presented in Section 2.6, every user is different. A key component of the framework presented in Chapter 3 is a persistent user profile, which is designed to allow the system to adapt itself to each user's preferences over time. The system presented here does not have the ability to adapt itself to its user, but neither does it have the opportunity. The current *Glaykos* system produces one potential narrative ordering of events, but this is unlikely to be the ideal ordering for every possible user. Given this fact, the author expects the orderings the participants use to recount the events of the mission to show some variation.

As such, the author's expectations can be summed up, thus:

1 In experiment 1 (*experimental* vs *base case*, non-expert participants), pause times will be less for the *experimental case* than the *base case.*

2 In experiments 2 (*experimental* vs *user case,* non-expert participants) and 3 (*experimental* vs *user case,* expert participants), pause times will increase with distance moved along the causal dimension (see Section 7.1.2).

3 In experiment 1, users will prefer the *experimental case* to the *base case.*

4 In experiments 2 and 3, there will be a preference for the *experimental case* over the *user case.*

5 In experiment 2 in particular, users may have some negative reaction to the lack of control afforded by the *experimental case.*

6 In all experiments, the participants' accuracy in recounting the events of the mission will be higher for the *experimental case* than either the *base case* or the *user case.*

7 In experiments 2 and 3, the user will need to replay more time in order to understand the mission fully (as compared to the *experimental case*).

8 There will be considerable variation in the ways participants choose to repeat the events of the mission.

Of these expectations, the author would consider participant accuracy (6) and playback exploration (7) to be the most important, as these are the factors which will remain of primary importance should the system develop into a more complete systems which holds a duologue with the user, rather than the current monologue.

## 9.3 Summary

This chapter has described a set of experiments designed to test the *Glaykos* system detailed in Part II. In total, three experiments were used to test the system, two of which were part of the initial design and a third which was added when the deficiencies were revealed in the first. This compared the output of the *Glaykos* system to a simpler *base case*, which maintained a strict chronological ordering and lacked many of the narrative influenced optimisations present in the *Glaykos* generated *experimental case.* This was found to be a flawed basis for comparison, however. The other two experiments compared the *experimental case* to a more interactive debrief which attempted to mimic the current state of the art. The second of these was carried out with domain expert participants.

Finally, a set of predictions was made about the outcome of these experiments. In total eight predictions were made, but the most important were considered to be that the level of participant accuracy would be higher for the *experimental case*, and that the *experimental case* will be more efficient in terms of the number of repetitions of the same time period required to gain an understanding of the mission. As well as positive, negative predications were also made, such as the expectation that uses would be frustrated by the lack of control afforded to them by the *experimental case*.

The results collected while carrying out these experiments, and their ramifications as to the validity of the expectations described here, will be described in the following chapter.

# Chapter 10

# Results

This chapter presents the results obtained from the experiments described in Chapter 9. These are divided into ten sections. The first eight of these present the results pertinent to each of the expectations expressed in Section 9.2, in order. The penultimate section then discusses additional observations which arose during and after the experiments themselves. Lastly the final section presents a summary of the chapter.

## 10.1 Were Pause Times Less for the *Experimental Case* than the *Base Case* in Experiment 1?

During the *experimental* and *base cases*, information was delivered one event at a time, with the system pausing after each. The Participant was given the following instructions:

> "After each event and its accompanying information has been delivered, the system will pause and wait for you to push the space bar before continuing.

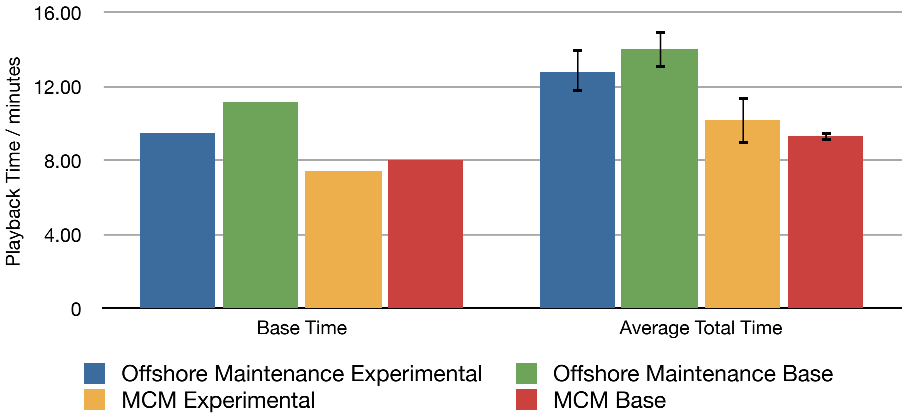


Figure 10.1: Base playback time (without users' pauses) and average playback time with users' pauses in experiment one. The error bars show the standard deviation.

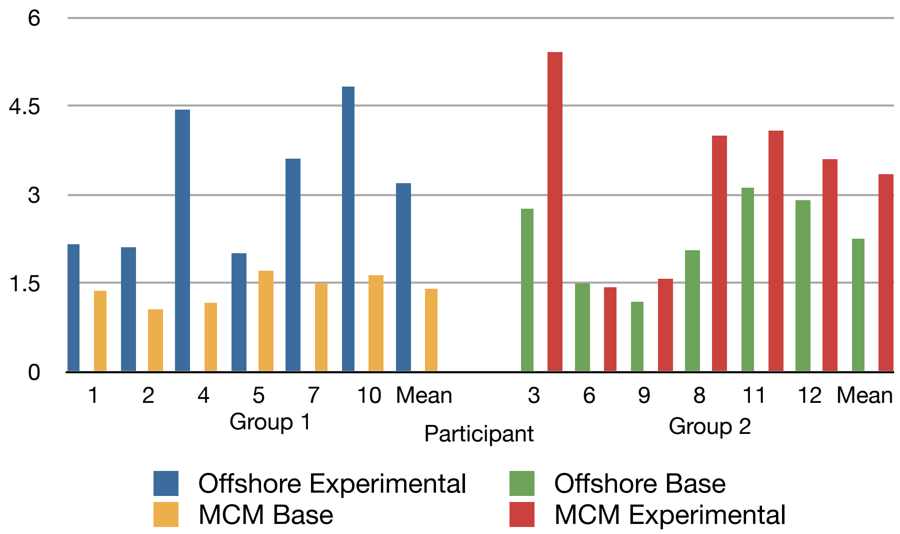


Figure 10.2: The average individual pauses between events for each scenario and case during experiment one.

> Please take any time you feel you need to process the information and relate it to the information you have previously been given before pushing the space bar."

As the only data which experiment one made available in this class is the total amount of time each Participant took to complete each debriefing, only two forms of result can be produced. The first and simplest of these is the average time taken to complete the debrief for each scenario and case, which is shown in Figure 10.1, alongside the base time each debrief takes without any pauses at all, for the sake of comparison. In each case the base time for the *experimental case* is shorter, as the *experimental case* typically requires less additional domain information to be imparted (according to the model used in Section 7.2.2). After the user pauses have been factored in, the total time is still shorter for the offshore maintenance scenario (through the difference is smaller), but not for the mine counter measures scenario.

The reason for this can be found in the second result. As the base playback time is known, as is the total number of pauses the average individual pause can be calculated. These data are shown in Figure 10.2. For both scenarios the pause time is longer for the *experimental case*, in all but one of the individual cases as well as the mean. This is particularly distinct for the mine counter measures scenario, where the mean pause time for the *experimental case* is found to be more than twice that for the *base case*.

This does not intrinsically indicate whether the difference is statistically significant, however. To test this, Student's t-test was applied to the data. A total of five comparisons were made. The first four of these compared the results within[1] and

[1] Comparing the sets of results for the two scenarios experienced by an individual group.

|  | Offshore | MCM |
|---|---|---|
| Offshore | 0.1566 | 0.0183 |
| MCM | 0.0462 | 0.0130 |

Table 10.1: Student's t-test results for comparisons between and within groups. Categories on the columns represent data from the *experimental case*, while categories on the columns represent the *base case*. All tests were homoscedastic with two tails.

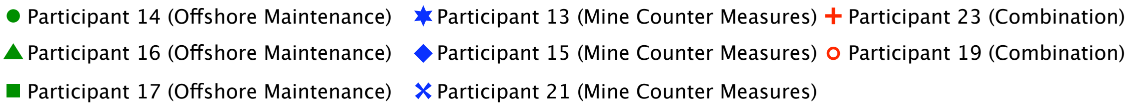

● Participant 14 (Offshore Maintenance)
▲ Participant 16 (Offshore Maintenance)
■ Participant 17 (Offshore Maintenance)
✶ Participant 13 (Mine Counter Measures)
◆ Participant 15 (Mine Counter Measures)
✕ Participant 21 (Mine Counter Measures)
+ Participant 23 (Combination)
○ Participant 19 (Combination)

Figure 10.3: Legend for scatter plots shown in Figures 10.4.

across[2] groups. The results of these tests are shown in Table 10.1. Since the $p$ value output from the t-test is considered to indicate statistical significance for values less than 0.05 it can be seen that three out of the four tests indicated statistical significance. This result becomes even more pronounced when the data is combined into a single set, wherein a p value of 0.0036 results, indicated an extremely high level of statistical significance. This indicates that expectation 1 was shown to be false.

There could be several reasons for this. It is possible that on average the participants found the *experimental case* more confusing. A reasonable number of participants indicated that they found the non-chronological ordering of the *experimental case less* confusing, however (see Section 10.9.3). Furthermore, while many participants disagreed with this, the majority of the participants indicated that they found the additional graphics helpful, and none disagreed with this. Taking these facts into account, it would appear that the most likely reason for the increased pause times lies with the additional graphics the participants found so helpful in the *experimental case*. As they remained on screen during the pauses, they made additional text available to the participants during the pauses. If the Participant chose to read this text it would almost certainly increase the length of the pause.

## 10.2 Did Pause Times Increase with Distance Moved Along the Causal Dimension in Experiments 2 and 3?

Figure 10.4 shows the individual pause times for each participant graphed as a scatter plot (See Figure 10.3 for the legend). It is not the absolute length of each pause we are interested in, but rather the relative length for that participant. To this end, the data set for each participant has been converted to a standard normal

[2] Comparing the results for the same scenario between the two groups.

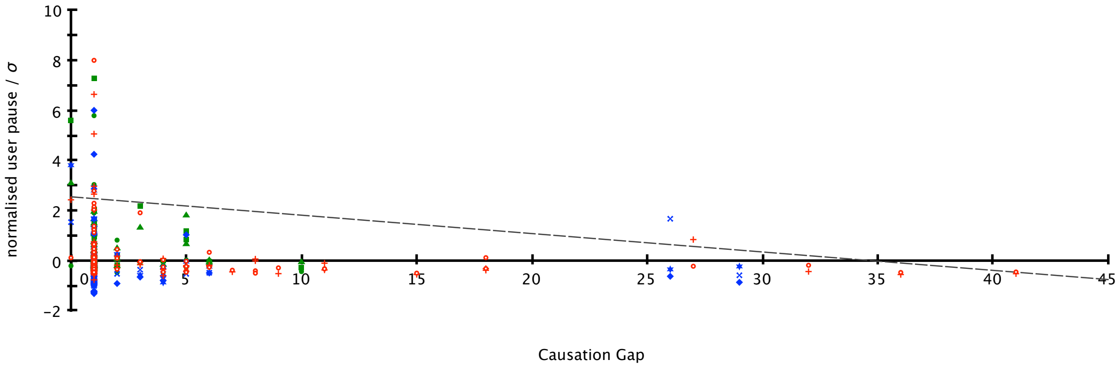


Figure 10.4: Scatter chart to show correlation between user pause times and causation gap between events in the *experimental case* for experiments two and three. The data set for each participant has been converted to the standard normal form. The legend for the different symbols can be found in Figure 10.3.

distribution by subtracting the mean and dividing by the standard deviation, for the purposes of this graph. There is still no clear pattern in these data which is visible to the naked eye, a fact which is reenforced by the line of best fit, which corresponds to a negative, but very low, correlation of $-0.0735$. This indicates that expectation 2 was shown to be false.

Given that the work of both Graesser[97] and Pennington and Hastie[98, 99, 100, 101] has shown that there is a penalty in comprehension associated with discontinuity on the causal axis, the most likely explanation for this complete lack of correlation is that the pause times used as a measure here are not in fact equivalent to the reading time measure used in the literature.

## 10.3 Did Users Prefer the *Experimental Case* to the *Base Case* in Experiment 1?

During experiments 1 and 2, at the end of the second survey the participant is asked to specify a preference for one the two debriefing methodologies they have experienced. This forms the basis for the first and simplest of the subjective assessments. Secondly, the third question on each of the surveys asks the participant to what extent they agree or disagree with the following statements (the abbreviation used for each of these statements in the following graphs is shown in brackets after each):

- The mission was easy to follow (**follow**).
- The chronological ordering of events was clear (**time**).
- The causal relationship between the events was clear (**cause**).
- It was obvious which vehicle each of the events concerned (**protagonist**).
- The spatial relationships in the scene were clear (**space**).

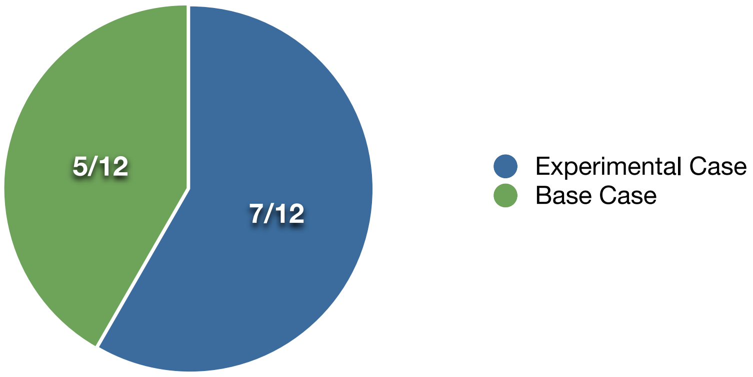


Figure 10.5: Participant preference for Experiment 1.

- It was obvious how each event contributed to and affected the mission as a whole (**contribution**).
- The intention behind each event was clear (**intention**).

The user's options are assigned numerical values as follows:

- Strongly disagree = -2
- Disagree = -1
- Neither agree nor disagree = 0
- Agree = 1
- Strongly agree = 2

As Figure 10.5 shows, Experiment 1 produced a slight preference for the *experimental case*. Though the difference is small (around 16%), a preference was found here, which stands in favour of the potential "optimisations" added to this case.

The results of the more in depth subjective assessment of the scenarios can be found in Figure 10.6. The *experimental case* does not fair so well in this comparison. On average, the *base case* gains higher scores almost uniformly with the mine counter measures scenario. The *experimental case* fairs a little better in the offshore maintenance scenario, getting higher scores on average in four of the seven classes, but by smaller margins than where it looses in two of the remaining three. This reflected in the combined means, which show similar values for each category, with the exception of time.

As these data are ordinal in nature, the Mann-Whitney-U (MWU) test is the most appropriate measure of statistical significance. Table 10.2 shows the output of the MWU test for the same set of comparisons which were performed in Section 10.1. The first two rows give the significance of the results taken from between groups,

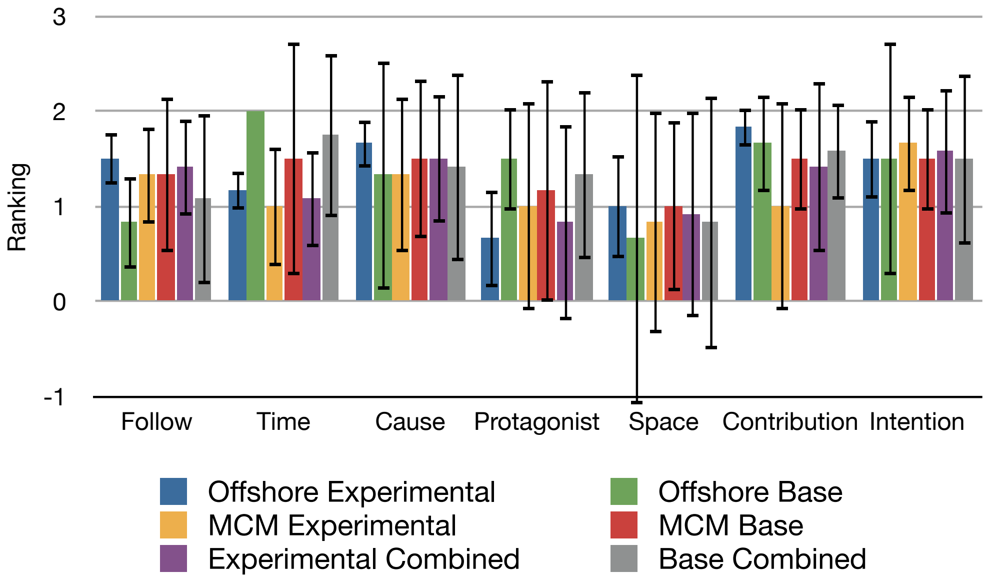


Figure 10.6: Subjective assessments for experiment 1. The error bars indicate the standard deviation.

| | Follow | Time | Cause | Protag. | Space | Contrib. | Intention |
|---|---|---|---|---|---|---|---|
| Offshore | 0.171 | 0.005 | 0.847 | 0.116 | 0.865 | 0.523 | 0.674 |
| MCM | 0.859 | 0.097 | 0.652 | 0.665 | 0.993 | 0.423 | 0.575 |
| Group 1 | 0.789 | 0.091 | 0.847 | 0.309 | 1.000 | 0.241 | 0.784 |
| Group 2 | 0.336 | 0.006 | 0.719 | 0.423 | 0.867 | 0.212 | 0.673 |
| Combined | 0.398 | 0.001 | 0.867 | 0.168 | 0.976 | 0.868 | 0.972 |

Table 10.2: The Mann-Whitney-U $P$ value results for the between groups, within groups, and combined analysis for Experiment 1.

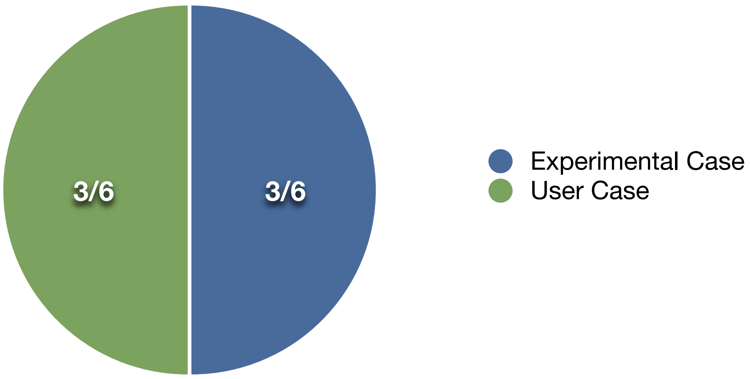


Figure 10.7: Participant preference for Experiment 2.

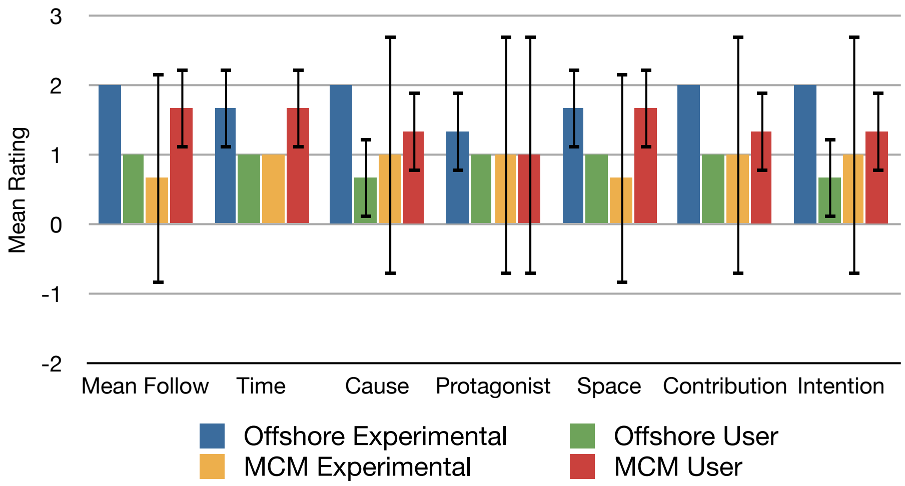
so the first row compares the *experimental case* for the offshore scenario from group 1 to the *base case* for the offshore scenario with group two. The second two rows describe the results from within the individual groups, comparing the different cases, but with different scenarios as well. The fifth column then gives the t-test for the data set which results when the two groups are combined. Across this entire data set, the only category which shows any statistical significance (i.e. the P value was less than 0.05) is *time*. Although this is not constant across all of the comparisons, it does suggest that the participants found the flow of time significantly easier to follow in the *base case*.

Taken overall this shows a small, but not statistically significant, preference for the *experimental case*, which would tend to disprove Expectation 3.

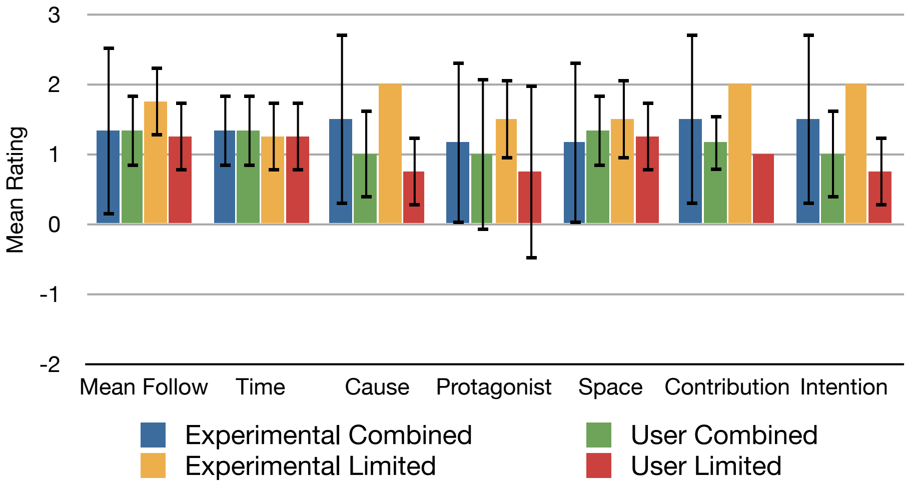
## 10.4 Did Users Prefer the *Experimental Case* to the *User Case* in Experiments 2 and 3?

The participant preference measure for experiment 2 (as shown in Figure 10.7) is split down the middle, with half of the participants preferring each case. One interesting factor comes into play when looking at the statement agreement of the participants who indicated that they preferred the *user case*, however. Of the three of them, two gave significantly higher scores to the *user case*. The participants indicated that although they felt the *experimental case* left them better informed, they simply did not like it as much. The primary reason for this being that they disliked the fact that they were not in control when observing the *experimental case*. This specific complaint is covered in more detail in Section 10.5.

The statement agreement results for the offshore maintenance scenario in experiment 2 are extremely encouraging (see Figure 10.8a). In all classes the *experimental case* scores higher than the *user case*. Additionally, the maximum score for any class with the *user case* never exceeds the minimum value for the same class with



(a) The mean rating for each case and scenario in Experiment 2. The error bars represent the standard deviation.



(b) The ratings which result from combining the data for both scenarios, together with the limited set which excludes the two outliers (participants 13 and 17). The error bars represent the standard deviation.

Figure 10.8: Subjective assessment for experiment 2.

| | Follow | Time | Cause | Protag. | Space | Contrib. | Intention |
|---|---|---|---|---|---|---|---|
| Offshore | 0.025 | 0.114 | 0.034 | 0.317 | 0.114 | 0.025 | 0.034 |
| MCM | 0.346 | 0.114 | 0.814 | 1.000 | 0.346 | 0.814 | 0.814 |
| Group 1 | 0.317 | 1.000 | 0.114 | 0.814 | 1.000 | 0.114 | 0.114 |
| Group 2 | 1.000 | 1.000 | 0.500 | 0.480 | 1.000 | 0.480 | 0.500 |
| Combined | 0.476 | 1.000 | 0.097 | 0.665 | 0.859 | 0.091 | 0.097 |
| Limited | 0.186 | 1.000 | 0.011 | 0.343 | 0.495 | 0.008 | 0.011 |

Table 10.3: The Mann-Whitney-U $P$ value results for the combined analysis of Experiment 2.

the *experimental case*.

The results for the mine counter measures scenario are somewhat more turbid, with the *user case* tending to score higher. There are two likely outliers in this data, however. Participant 13 confessed to having come directly from a long haul flight after the completion of the experiment, and had an extremely negative reaction to the *experimental case* with this scenario, which significantly reduced the scores received by the *experimental case*. Participant 17 is an army officer, with considerable battle space command experiment, and so was able to quickly and accurately comprehend the *user case*'s presentation of the mine counter measures scenario. Due to their lack of experience with the exact technologies which are relevant to this project (AUVs, automated planning, etc.), they were mistakenly placed in this group, when they should have been placed in one of the groups for experiment 3, which considers expert users.

Figure 10.8b presents the mean values which are produced by combining the two scenarios for each case, together with a second set of "limited" means, which excludes the two potential outliers entirely. The overall means show little differentiation, with similar values and often high standard deviations across all of the categories. In the case of the limited means, however, the *experimental case* received higher scores across the board.

Once again, the Mann-Whitney-U test is the most appropriate measure of statistical significance for these data. The same batch of tests was performed as in Section 10.4, with the test being applied within and between groups, together with the combined data sets for each case. Additionally, the limited data set without the data from participants 13 and 17 was also tested. In the case of the between groups tests, as well as the test for the mine counter measure scenario, no statistically significant difference was found for any of the categories. However, the Mann-Whitney U test did show that the *experimental case* did receive statistically higher ratings for the follow, cause, contribution and intention categories. Likewise, for the limited data set the cause, contribution and intention categories showed high statistical significance in favour of the *experimental case*.

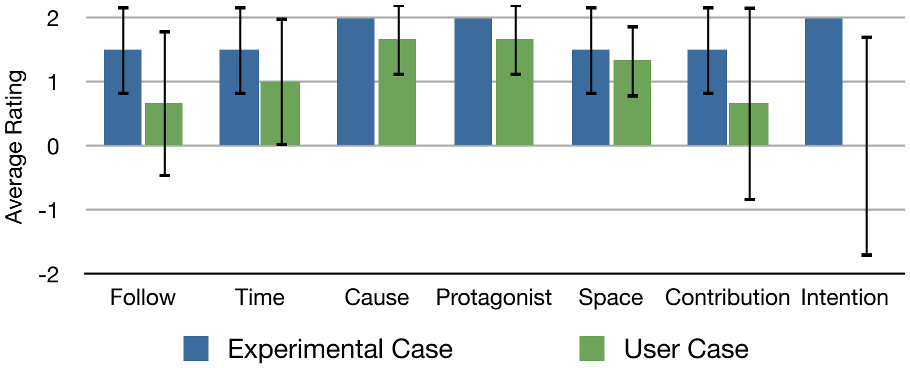


Figure 10.9: Subjective assessment for the Combination scenario during experiment 3.

| | Follow | Time | Cause | Protag. | Space | Contrib. | Intention |
|---|---|---|---|---|---|---|---|
| Combination | 0.361 | 0.543 | 0.414 | 0.414 | 0.739 | 0.543 | 0.182 |

Table 10.4: The Mann-Whitney-U $P$ value results for the combined analysis of Experiment 3.

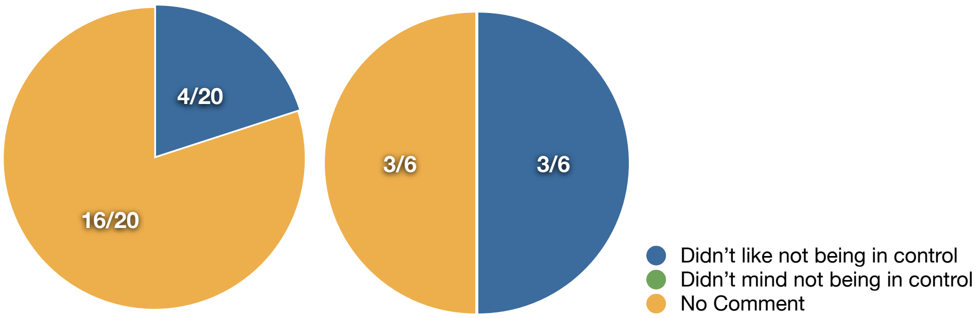
In the case of experiment 3, each participant only experiences one scenario, and so there is no direct participant preference result. The statement agreement results are still generated, however, and these results are displayed in Figure 10.9. The *experimental case* again performs very well here, scoring higher than the *user case* in every class. It should be noted here that the scenario is significantly more complex than those used previously, but the participants used in experiment 3 are all domain experts. This being the case it is not surprising that the *user case* received higher scores than it generally did in experiment 2.

As only a single comparison is possible with this data, only one set of Mann-Whitney U tests were performed. These are shown in Table 10.4. In this case no statistical significance is evident, likely as a result of the small number of data points.

Based in these results, expectation 4 could tentatively be considered to have been proven correct, as the only statistically significant results were in favour of the *experimental case*. However, the vast majority of the data showed no statistical significance.

## 10.5 Did Users Have Some Negative Reaction to the Lack of Control Afforded by the *Experimental Case*?

The main place in which the *experimental* and *base cases* differ from the *user case* is that they restrict the user's control of the debrief significantly. As shown in Figure



(a) For all participants. (b) For Experiment 2 participants.

Figure 10.10: Participant opinion of the lack of control in the *base* and *experimental cases*.

10.10a, 100% of the participants who expressed an opinion regarding this stated that they didn't like it, though these only represented 20% of the sample group. Of those who also experienced the more control heavy *user case*, this reached 50%, however, as shown in Figure 10.10b. Interestingly, the additional participant who expressed a problem with the lack of control was part of experiment 1, and so did not have any experience of the *user case*:

> "It had not occurred to me that the events could be delivered in non-chronological order, until I saw [the *experimental case*]. If this is going to happen then I want to be in control of it."
>
> *Participant 8*

There was clearly some negative reaction to the lack of control afforded by the *experimental case*, though this opinion was not expressed by a majority of participants. A majority is not what was expected, however, only that the lack of control would illicit some negative reaction, and this would be stronger in the case of the participants in Experiment 2. For any future experiments performed with this system, it would seem prudent to directly compel the user to express an opinion on this issue. Nevertheless, expectation 5 is considered to have been proved correct.

## 10.6 Was the Participants' Accuracy Higher for the *Experimental Case* than Either the *Base* or *User Cases*?

In this section we explore the effect each of the different cases and scenarios had on Participant accuracy in recounting the events of the mission. "Accuracy" is measured here using the d-prime, or sensitivity index, measure, which combines the hit and false alarm rates for each participant, adjusted for the baseline of the population (see *Rhodes et al.*[163] for another example of its use). The hit and false alarm rates are calculated based on each participant's answer to the following question in the post-debriefing questionnaire:

- 5. Please give a brief overview of the events which occurred during the mission and how they related to each other and the mission as a whole.

A hit is awarded for each of the pertinent facts which where given for the three scenarios (see Sections 8.3.2, 8.2.2 and 8.3.2). For the hit to be awarded, the fact in question must be either directly given, or strongly implied. So, for example, the following text:

> "Installations were lifted by the manipulator AUV so that the inspection AUV could inspect them. This was done for each of Installations Alpha, Bravo and Charlie."

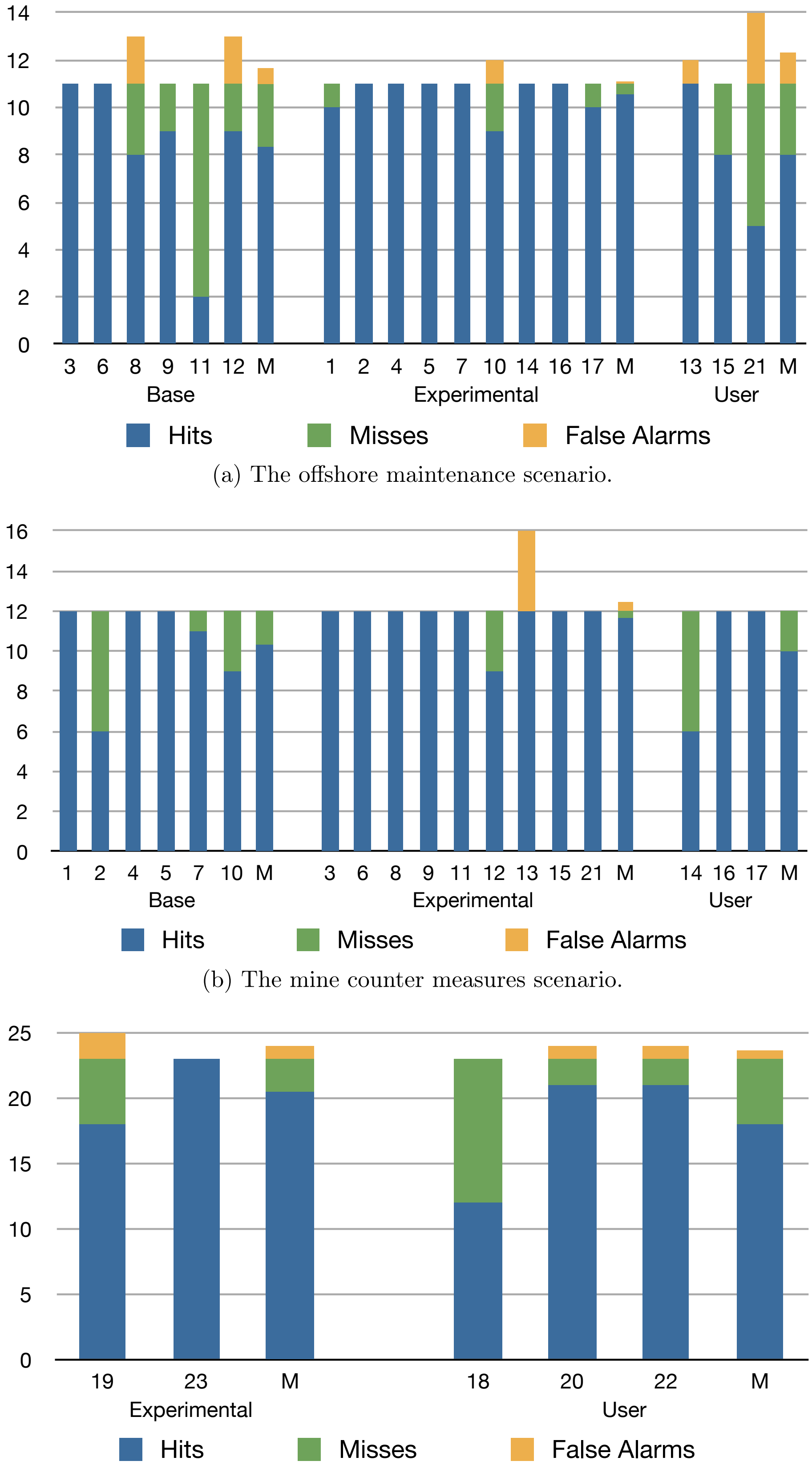
Would be taken to give a hit for each of the following pertinent facts:

- The Manipulator AUV lifted Installation Alpha so that it could be examined;
- The Inspection AUV examined Installation Alpha;
- The Manipulator AUV lifted Installation Bravo so that it could be examined;
- The Inspection AUV examined Installation Bravo;
- The Manipulator AUV lifted Installation Charlie so that it could be examined;
- The Inspection AUV examined Installation Charlie.

False alarms were given when any participant stated an event which did not actually occur during the mission. In the case that an action is ascribed to the wrong AUV, both a hit and a false alarm is given.

The graphs in Figure 10.11 have the number of hits, misses and false alarms for each of the participants, together with the mean for each group. It can be seen that the average hit rate is higher for the *experimental case* than it is for either the base or uses cases in every scenario. This is not sufficient to establish a significant difference, however, especially given that the rates for false alarms are much more even.

Figure 10.12 gives the average of the d-prime measure for each case and scenario, together with combined averages for all three scenarios, and a "limited" average which does not include the combination scenario. This limited measure is provided is there are significantly fewer data points for the combination scenario, making it much harder to trust the characterisation of its baseline. It can be seen once



(a) The offshore maintenance scenario.

(b) The mine counter measures scenario.

(c) The combination scenario.

Figure 10.11: The hits, misses and false alarms for each of the participants in each of the scenarios. "M" indicates the mean for the preceding group.

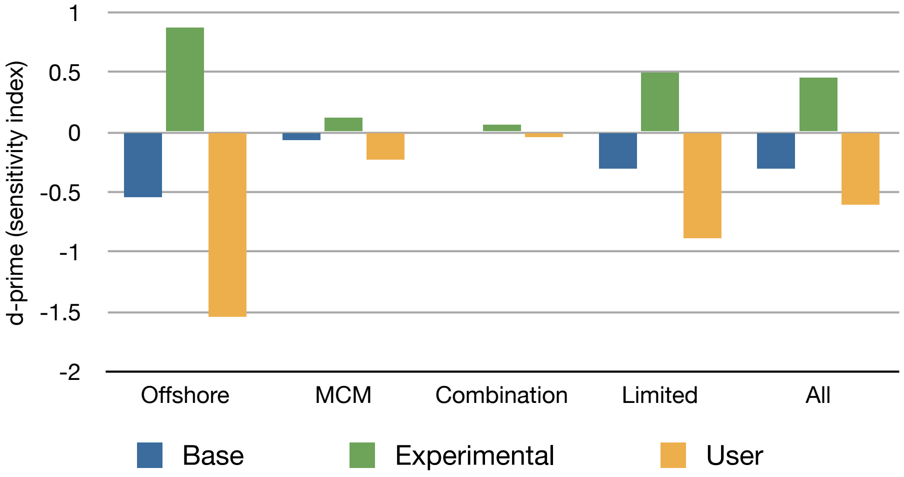


Figure 10.12: The average d-prime figures for each scenario and case. "All" averages all scenarios for each case, while "limited" excludes the combination scenario.

| | Offshore | MCM | Combination | Limited | All |
|---|---|---|---|---|---|
| Experimental vs. Base | 0.0314 | 0.8023 | n/a | 0.0948 | 0.1187 |
| Experimental vs. User | 0.0282 | 0.7364 | 0.9449 | 0.0588 | 0.0795 |

Table 10.5: The t-test results for comparisons *experimental* vs. *base* and *experimental* vs. *user*. Groups are the same as in Figure 10.12.

again that the average d-prime is higher for the *experimental case* than either the *base* or *user case* in every instance (though by only a small amount in the case of the MCM and combination scenarios). The individual d-prime figures were tested for statistical significance using Student's t-test, the results of which are shown in Table 10.5. This shows that participants accuracy was significantly higher for the *experimental case* with the offshore maintenance scenario, but not in any of the other tests.

There is an additional factor to take into account here, however. Participant 13's status as a potential outlier was previously discussed in Section 10.3, and a strong case can be made for giving the participant a similar status here. It can be seen from Figure 10.11b that they were the only participant to have any false alarms in any case for the mine counter measures scenario. In actual fact, the participant uniquely included an additional AUV which was not present in the mission in their recounting of the events of the mission, making the false alarms both unusually numerous and

| | Offshore | MCM | Combination | Limited | All |
|---|---|---|---|---|---|
| Experimental vs. Base | 0.0314 | 0.1994 | n/a | 0.0096 | 0.0196 |
| Experimental vs. User | 0.0282 | 0.2341 | 0.9449 | 0.0101 | 0.0200 |

Table 10.6: The t-test results for comparisons *experimental* vs. *base* and *experimental* vs. *user*, without the inclusion of the data from Participant 13. Groups are the same as in Figure 10.13.

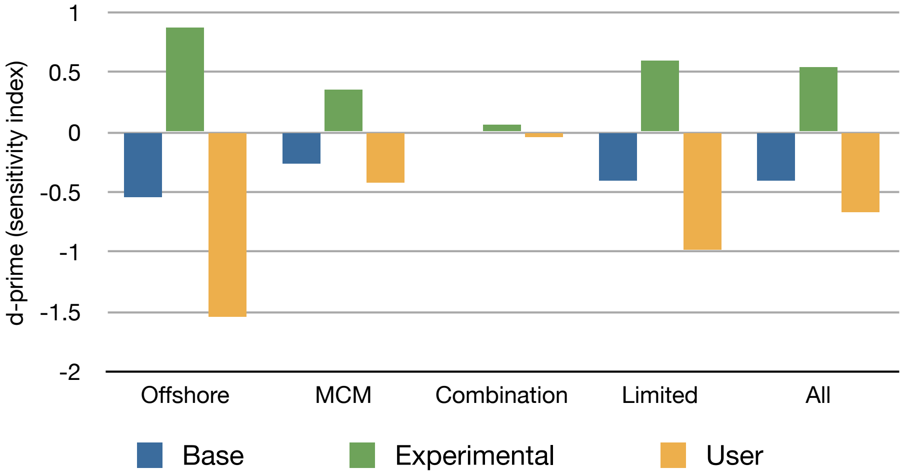


Figure 10.13: The average *d*-prime figures for each scenario and case, without the inclusion of the data from Participant 13. "All" averages all scenarios for each case, while "limited" excludes the combination scenario.

strong.

Figure 10.13 and Table 10.6 reproduce the previous data with Participant 13 removed. A noticeable difference can be seen in the affected graphs, with the MCM scenario in particular now showing a a more distinct difference between cases. The t-test data in Table 10.6 shows that the difference for the MCM scenario is still not significant, however, and of course the values for the combination scenario remain unchanged. Neither of these is surprising, however, as the MCM scenario gave the participants the least trouble of the three and there are too few data points for the combination scenario to be able to establish statistical significance. Both combined sets are now showing statistical significance, however, to a higher degree than the offshore maintenance scenario does alone.

It is considered to be valid to treat Participant 13 as an extreme outlier and remove the relevant data from these comparisons. In this case, both combined data sets show a significantly higher degree of accuracy for participants who viewed the *experimental case* compared to those who viewed either the *base* or *user case*. Consequently, expectation 6 is considered to have been proven correct.

## 10.7 Did the User Need to Replay More Time in Order to Understand the Mission Fully in Experiments 2 and 3?

Participants in experiments 2 and 3 had the *user case* as one of their debriefings. This gave them complete spatial and temporal control of the playback, allowing them to experience the events which made up the mission in any order they chose, together with the following task:

> "You should use these controls to explore the mission to gain as full an understanding of it as you think is possible. Please pay particular attention to any apparent temporal, spatial and causal relationship between the different events, as well as any motivation behind each."

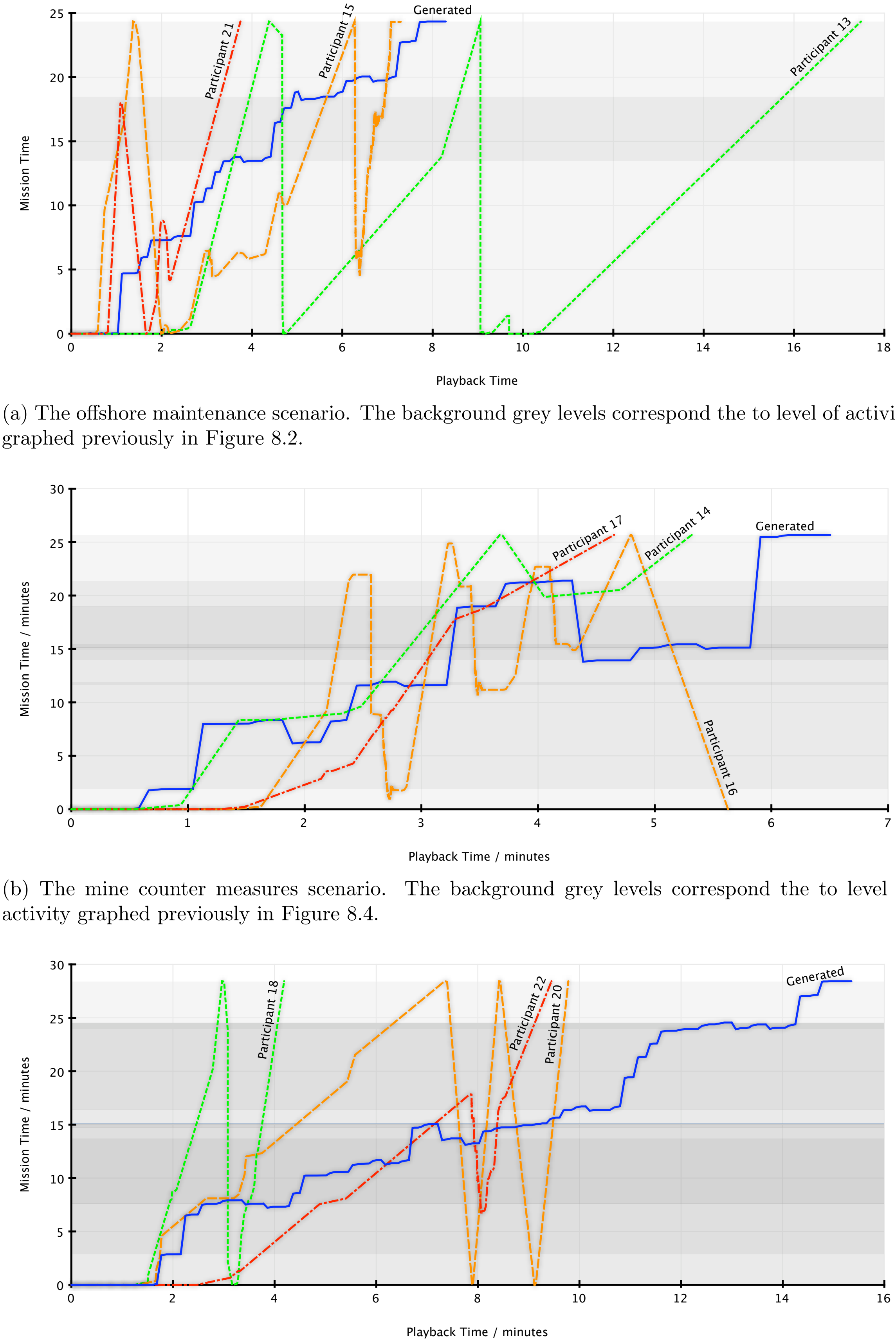
During the experiment, the system recorded their every action. In this section, this data will be used to ascertain whether expectation 7 from Section 9.2 proved correct.

The complete temporal path each participant takes though the mission is graphed, together with that taken by the automated approach. This is used to calculate the second temporal measure: the amount of temporal redundancy the Participant required, in terms of re-watching the same time period multiple times. This measure is further subdivided into the total amount of mission time, and the mission time which is viewed forwards, or chronologically.

For example, consider a situation in which the participant watches the mission up to 5 minutes, then rewinds back to the 4 minute mark, and then continues watching until the 10 minutes mark (and the end of the mission). The total amount of mission time they have viewed is 12 minutes (5 + 1 + 6). We represent these values as a ratio to the actual length of the mission for ease of comparison between scenarios of different lengths, which in the example gives us a value of 1.2. This value, which will be referred to as the "playback ratio", can be calculated for the playback generated for the *experimental case*, providing a relative measure of the efficiency of the automated and manual mechanisms.

The graphs in Figure 10.14 show the temporal path taken through the scenarios by each of the participants, as well as those automatically generated for the use of the *experimental case*. As the system which produced the generated path had complete knowledge of the mission, it can be seen to move the temporal position to very specific times, and then tends to linger at these times for more extended periods. The human participants, on the other hand, have no such knowledge and so are forced to "search" for the key events which make up the mission. The same part of the mission may have to be reviewed several times in order to ensure that all pertinent events have been seen and understood. An earlier event might have been missed, and then only found after one of the events it triggered is traced back to its cause.

A reasonably canonical example can be found in the trace produced by Participant 15 (see Figure 10.14a). The Participant quickly runs through the entire mission, likely in order to gain a quick overview. Next, they rewind back to the very beginning and then proceed more slowly, stopping at some events, and in some cases reviewing them multiple times. After reaching the end of the mission for a second time, they appear to be unhappy with their understanding on several particular events (or groups of events) and so rewind once more in order to view these in detail. Finally they proceed to the end of the mission and indicate that they are sat-



(a) The offshore maintenance scenario. The background grey levels correspond the to level of activity graphed previously in Figure 8.2.

(b) The mine counter measures scenario. The background grey levels correspond the to level of activity graphed previously in Figure 8.4.

(c) The combination scenario. The background grey levels correspond the to level of activity graphed previously in Figure 8.7.

Figure 10.14: The complete temporal path taken by the participants and the generated playback for each scenario.

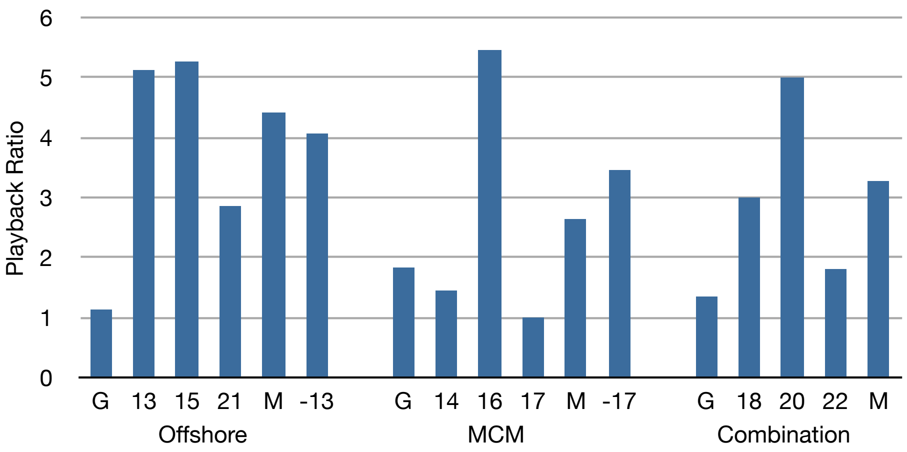


Figure 10.15: The playback ration for each of the participants in the *user case*, divided by scenario. "G" indicates the ratio for the debrief generated by the *Glaykos* system, "M" the mean for the participants in a scenario, and "-X" the mean with participant X removed.

|   | All | No 13 | No 17 | No 13/17 |
|---|---|---|---|---|
| p | 0.0319 | 0.0411 | 0.0102 | 0.0145 |

Table 10.7: The results of the t-test comparison between the generated and users' playback ratios with and without potential outliers.

isfied with their understanding of the mission. Also canonical in this example is that the user took less (playback) time than the generated path in order to accomplish this.

The graphs in Figure 10.15 reduce those in Figure 10.14 to the more discrete playback ratio previously mentioned. All but two of the participants required a higher degree of redundancy than that used by the generated path. Even so, the mean playback ratio for the participants is higher than the ratio for the generated debrief in all three scenarios.

The higher means are not sufficient to prove statistical significance, however, and there are not enough data points for each individual scenario for statistical tests to be performed on each. The ratios for each scenario are nor directly comparable due to the differing levels of parallelism in each scenario, making a direct combined measure more difficult. To remedy this, each ratio was reduced to the standard score for its scenario (by subtracting the mean and dividing by the standard deviation) and Student's t-test performed on resulting sets of data. The result of this is provided in Table 10.7, which shows that the difference is statistically significant. Also provided are the results of the t-test for the data sets which remove the two potential outliers in the data.

Participant 17 is, as previously mentioned (in section 10.4), an army officer with real world battle space management experience. In this context, they are required

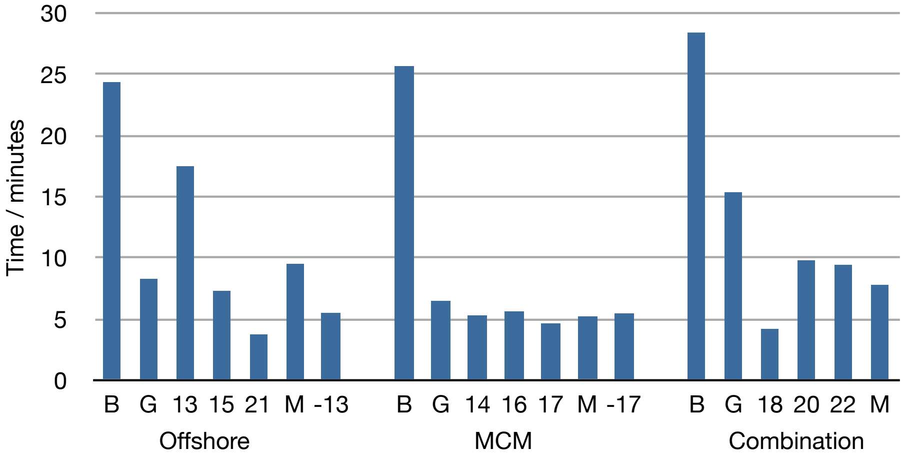


Figure 10.16: The playback time for each of the participants in the *user case*, divided by scenario. "B" indicates the actual time for the mission, "G" the time for the debrief generated by the *Glaykos* system, "M" the mean for the participants in a scenario, and "-X" the mean with participant X removed.

to be able to simultaneously keep track of the assignments and actions of a large number of assets, some of whom are soldiers whose lives potentially depend on their commander having this exact skill. This being the case, a relatively simple mine counter measures mission with sparsely occurring events presented little to no challenge. The Participant simply set the playback to a reasonable speed and viewed the mission once. Clearly, again as previously noted, Participant 17 should have been placed in one of the groups of participants used during experiment 3. For this reason, they can to some extent be considered an outlier.

Participant 13 has also been considered an outlier is previous sections, due their extreme tiredness at the time of the experiment, which was not indicated until afterwards. Participant 13's playback ratio was high, but not unusually so.

In ether case, the mean for the participant's playback ratio is higher than that for the generated debrief with and without the data for the outliers, as shown in Figure 10.15. Likewise, the removal or either, or both, of the outliers does not change the significant nature of the difference, as shown in Table 10.7. This being the case, expectation 7 is considered to have been proven correct.

For the sake of completeness, the other simple measure which can be extracted from these data is the amount of playback time each of the participants needed to view the mission for. This is shown in Figure 10.16, together with the length of the actual mission and the time taken for the debrief created for the *experimental case*. In all cases except one, the user took less time to become satisfied with their understanding of the mission than the generated version required to complete its explanation. On average, the users took around 70% of the time needed by the generated path. Initially, this appears to be a fairly negative result. However, one of the complaints made against the speech synthesis system (see Section 10.9.3) was

|   | All | No 13 | No 17 | No 13/17 |
|---|---|---|---|---|
| p | 0.1373 | 0.0005 | 0.1334 | 0.0005 |

Table 10.8: The results of the t-test comparison between the generated and users' playback times with and without potential outliers.

that it slowed down the rate of the playback unnecessarily. Users were able to read the subtitles in a fraction of the time it took for the text-to-speech system to read them aloud.

This data was again reduced to standard scores and Student's t-test performed, the results of which are given in Table 10.8. This shows that for the data taken as a whole the difference between the the participant's playback time and the system's is not statistically significant. However, this is another case in which Participant 13 can be considered an outlier (being the only participant who took more time than the generated debrief). Indeed, the further tests in given in Table 10.8 show that the removal of Participant 13 gives a very high level statistical significance indeed. Again for the sake of completeness, results are also given with participant 17 removed, which show little to no difference. The conclusion drawn from this is that the participants took significantly less time to debrief themselves than the *Glaykos* system required, but at least of portion of the blame for this can be laid at the feet of the text-to-speech system.

## 10.8 Was There Considerable Variation in the Ways Participants Chose to Repeat the Events of the Mission?

Question 5 of each feedback survey asks the Participant:

> "Please give a brief overview of the events which occurred during the mission and how they related to each other and the mission as a whole. Feel free to use whichever ordering you think is most appropriate."

This section details how the participants chose to order the events of the mission. Across the data taken from the Participant surveys, three distinct ordering methodologies were found:

**Chronological**, in which the events are listed in a strict chronological ordering. An example of chronological ordering might be:

> "The Transit started scanning the mission area. It found a potential target and the inspection AUV was dispatched to examine it. While it was moving to the target, the transit AUV found a second target. The

> inspection AUV arrived at the first detection and found it to be a rock. It started to move to the second detection...”

**Causal**, in which the events are given an ordering which tends to prioritise causal connections over temporal ones. An example of causal ordering might be:

> “The Transit AUV detected a target. The Inspection AUV found that it was a rock. The Transit AUV found a second target. The inspection AUV examined it and found it to be a mine. Destroy AUV 1 destroyed the mine...”

**Motivational**, which takes a higher level view, and orders the events according to the particular goal they satisfy. An example of motivational ordering might be:

> “The Transit AUV scanned the mission area. It found three detections, each of which was examined by the Inspection AUV. The second and third detections were found to be mines, while the first was a rock. The Destroy AUVs destroyed each of the mines.”

The pie charts in Figure 10.17 show the compiled raw data for each of the cases for the offshore maintenance and mine counter measures scenarios. The data for the combination scenario will be considered separately. Causal ordering seems to be the most popular ordering methodology for the offshore scenario, while motivational is given greater significance for the MCM scenario.

It should also be noted that the *experimental case* shows a preference towards a causal ordering, which reflects the ordering in which it was delivered. This would tend to suggest that this ordering helps to build a strong causal model in the participant’s mind, which then carries over into their ordering of events. Conversely, however, the *base case* shows some preference away from the chronological ordering in which it was delivered, with a fairly even split between causal and motivational orderings. This suggests that the order in which the actions are delivered to the user does not have a direct relationship to the order in which they choose to recount them.

Also evident in the results is that the *user case* tends to show a bias away from causal ordering. This would seem logical, as this case provides the least causal information, even compared to the *base case*, which does provide some causal information, even though it does not use a causal ordering.

The data for the combination scenario is shown in Figure 10.18. This shows a large bias towards a motivational ordering with this methodology being used in four fifths of all cases. This may be a reflection of the fact that this scenario is very complex and contains a large number of events, leading participants to use the ordering strategy which provides the most simplification. A second variable is

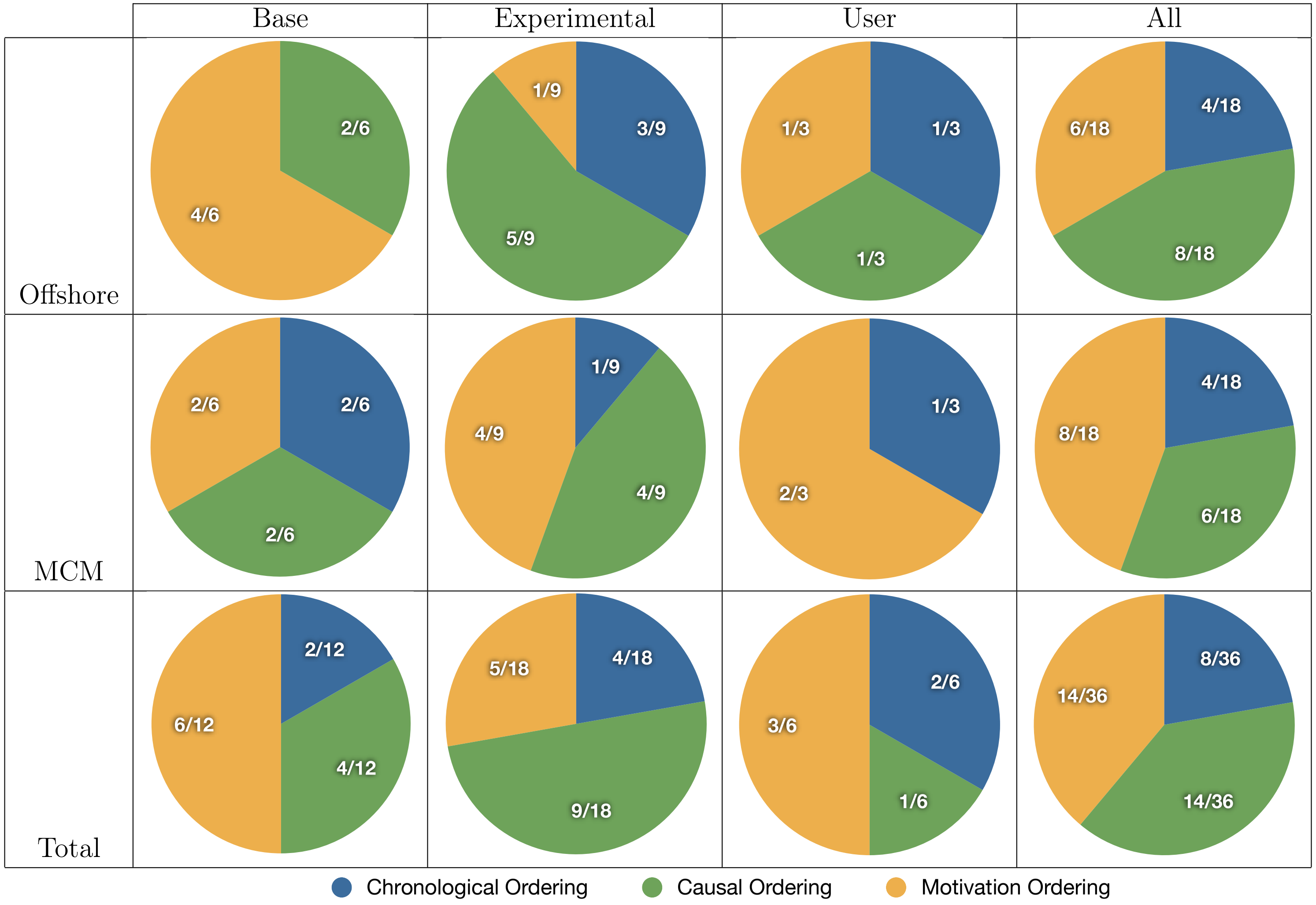


Figure 10.17: The different orderings used by participants for each case with the offshore maintenance and mine counter measures scenarios.

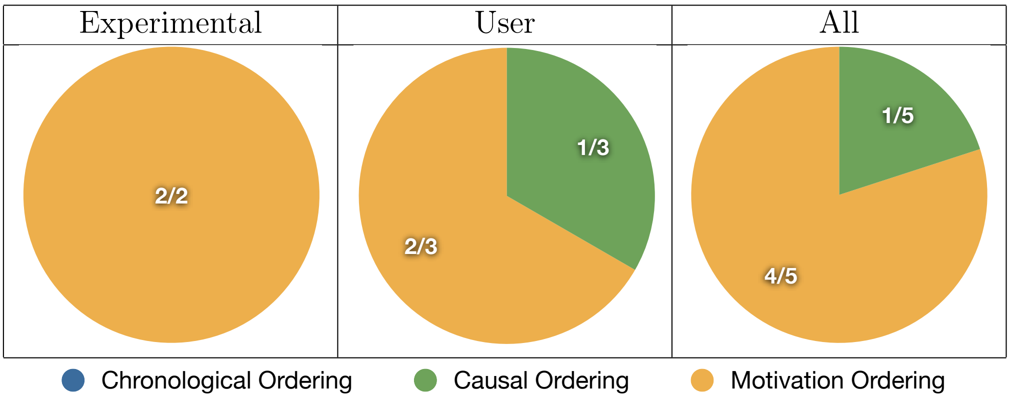


Figure 10.18: The different orderings used by participants for each case with the combination scenario.

| | Chronological | Causal | Motivational | Total |
|---|---|---|---|---|
| Chronological | 2 | 1 | 1 | 4 |
| Causal | 1 | 4 | 3 | 8 |
| Motivational | 1 | 1 | 4 | 6 |
| Total | 4 | 6 | 8 | 18 |

Table 10.9: Distribution of different combinations of orderings used by participants who viewed two scenarios. Rows represent the ordering used in the first survey, columns the second.

also present for the combination scenario, however, in that the participants who viewed were all deemed to be domain experts. This also raises the possibility that a motivational ordering is the preference of domain experts, to whom scenarios such as these are more commonplace.

Something else which might seem clear from the imbalance between the data for the offshore maintenance and mine counter measures scenarios is that not all of the participants who viewed both of these scenarios used the same ordering for both. On the contrary, 45% of these participants changed the ordering they used between the two scenarios and cases. The distribution of the different combinations of orderings used is shown in Table 10.9, which considers only the orderings used, not the particular case or scenario they were used in relation to. This shows that the participants showed no clear preference for any one combination of orderings across the three differing scenarios.

Expectation 8 is difficult to prove via grounded statistical means. What is clear from Table 10.9, though, is that every combination of orderings is represented, and no one ordering emerges as dominant. Based upon this, expectation 8 is taken to have been proved correct.

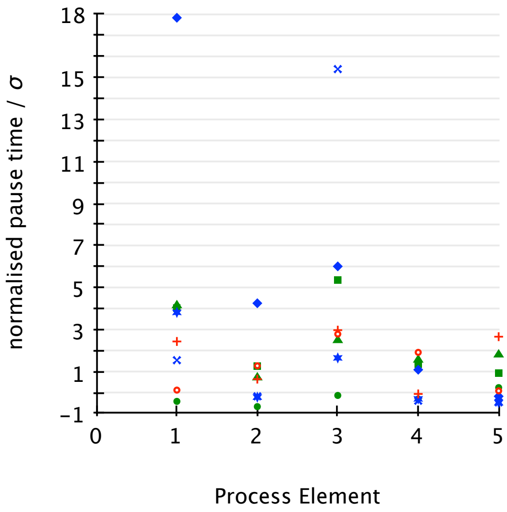


Figure 10.19: The normalised pauses for each of the participants who experienced the *experimental case* in Experiments 2 and 3. Symbols are as per Figure 10.3.

## 10.9 Additional Observations

This section discusses several further observations which came to light either during or after the collection and processing of the experimental data. It contains three subsections. The first of which attempts to gauge the degree of trouble, if any, the potentially critical incidents discussed in Chapter 8 caused the experimental participants. The second subsection takes the data used to process the playback ratios for each participant in Section 10.7 and flips the axes in order to examine which times in the mission each user focussed on in particular. Lastly, the third subsection considers the unprompted opinions expressed by the participants during the open questions at the end of the surveys.

### 10.9.1 Effects of Potentially Critical Incidents

This section will examine the pause times between process statements from participants who were debriefed by the *experimental case* in Experiments 2 and 3 and look for any correlation to the process statements which were designated as potentially critical incidents in Chapter 8.

Is it is not the absolute length of each pause which is important here, but rather the relative length of each for the particular participant. Consequently, each dataset was first reduced to standard normal form. The majority of the participants paused for particularly long periods of time during the first few pauses, however, and this was found to distort the data in some cases. Figure 10.19 shows the first five pauses (in standard normal form) for each of the participants, which illustrates the large range of the data, which is particularly extreme in the case of some participants. In order to mitigate against the effect of this, the first five pauses were excluded from

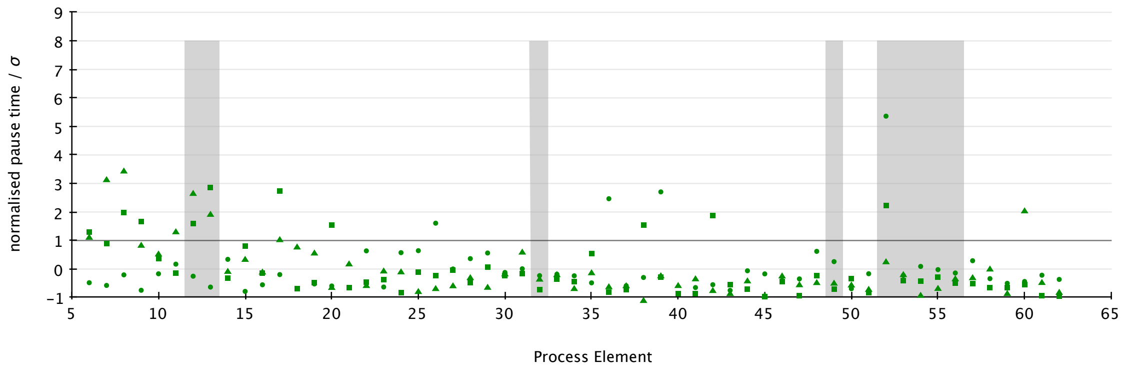


Figure 10.20: The normalised pauses for each participant from Experiment 2 in the offshore maintenance scenario, with the first five pauses excluded. Shaded areas correspond to potentially critical incidents (see Section 8.3.4). Symbols are as per Figure 10.3.

each data set before conversion to standard normal form. In general, any pause which is found to be more than one standard deviation above the mean (1 on the y axis of the graphs used here), which also coincides with a critical incident will be considered worthy of note, and those more than 2 standard deviations above the mean considered significant (corresponding to a p value of 0.05).

As can be seen in Figure 10.20, some noise is present in the data for the offshore maintenance scenario, but this is to be expected, as complete consistency is highly unlikely. Of the pauses corresponding to potentially critical incidents in this debrief (see Section 8.3.4), those related to the first and fourth are notable or significant.

The first incident was realiser based which resulted in two ambiguous uses of the pronoun “it”. That it gave the participants pause, then, is not surprising.

The fourth incident is planner based, resulting from an apparently redundant action which is performed to ensure the planner does not create highly suboptimal plans. This incident covers a large number of process elements, as the apparently redundant action is referred to multiple times. Only the first of the process elements results in significant pauses, however, suggesting that after any initial confusion the participants accept the strange behaviour.

The other two potentially critical incidents in this debrief are narrative order based and did not result in any significant pause. In fact, in five out the six cases the pause time was below the mean.

In the case of the mine counter measures mission (see Figure 10.21), only a single notable pause coincides with a potentially critical incident, the third in the mission (see Section 8.2.4). This is a particularly extreme example of an narrative order related incident, in which the debrief completes the survey of the mission area before rewinding a full five minutes to the Transit AUV’s discovery of a potential target. Coupled with this is the fact that it also follows another potential critical

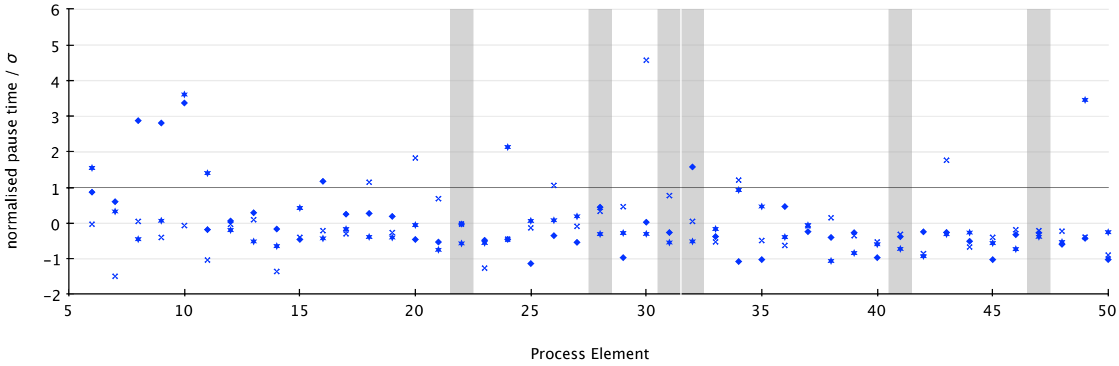


Figure 10.21: The normalised pauses for each participant from Experiment 2 in the mine counter measures scenario, with the first five pauses excluded. Shaded areas correspond to potentially critical incidents (see Section 8.2.4). Symbols are as per Figure 10.3.

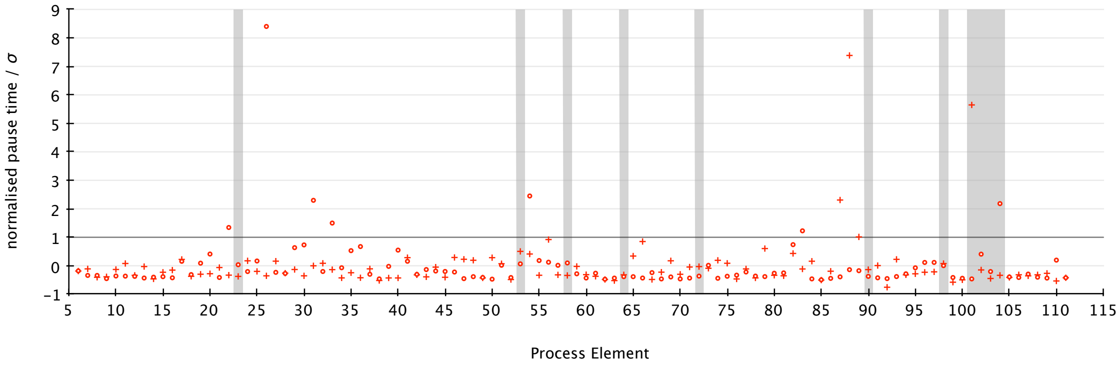


Figure 10.22: The normalised pauses for each participant from Experiment 3 in the combination scenario, with the first five pauses excluded. Shaded areas correspond to potentially critical incidents (see Section 8.3.4). Symbols are as per Figure 10.3.

incident involving a repeated statement which may appear redundant. Even so, this only produced a notable pauses in a single participant and no significant pauses.

Figure 10.22 shows the normalised pauses for the combination scenario. Once again, only a single critical incident (see Section 8.3.4) aligns with any of the heightened pauses, the eighth and last, which is equivalent to the fourth and last from the offshore maintenance scenario. This incident caused significant pauses in both of these scenarios, though in the case of the combination scenario they are split between the first and the last of the process elements which make up the incident. As before, this incident is planner based. Also notable are the results for incident 3 in the combination scenario. Like incident 3 in the mine counter measures scenario, this is a fairly extreme example of a narrative order based incident, however, no notable or significant pauses are present for the incident in question.

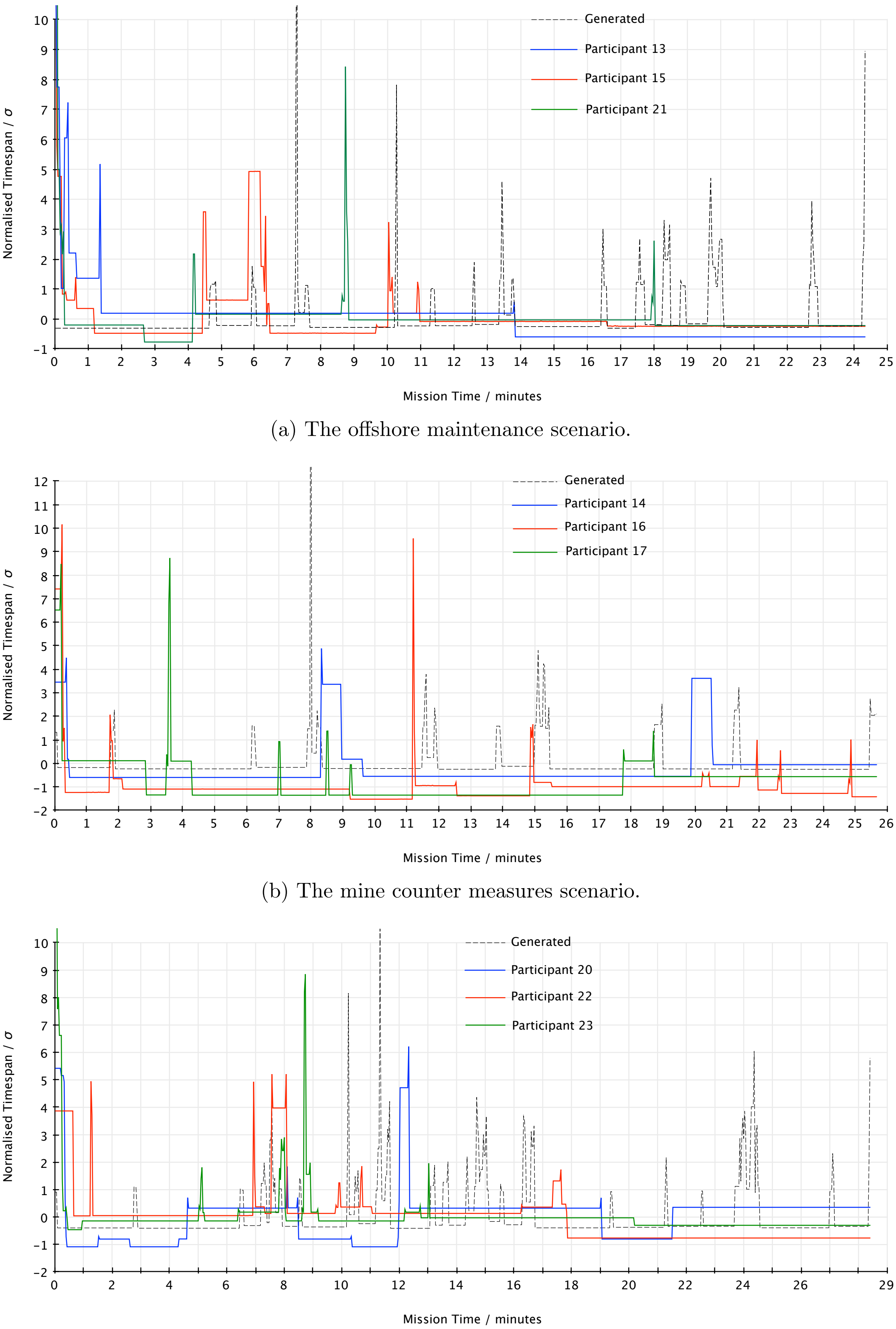
A total of four potentially critical incidents produced notable or significant pauses in these results. The first of these was incident 1 in the offshore maintenance scenario. This was a realiser based incident produced by an error and it seems clear that fixing this should be considered a priority. The second of these, which was also mirrored and reenforced by the fourth, was a planner based critical incident. As such, the issue itself does not reside in the *Glaykos* system, and should the system be moved to a more advance control system issues such as this should no longer be present. Lastly, the third critical incident which coincides with a notable pause is narrative order based. Half of the eighteen potentially critical incidents in the three scenarios were narrative order based, while only this single incident produced any notably longer pause. Of the four correlations, this was the weakest. It was only seen in a single participant and the length of the pause was considered only notable and not significant. This suggests that the narrative order based incidents are of less concern than those which were realiser or planner based. Nevertheless, projected future work (see Section 11.4.7) is intended to give the *Glaykos* system additional functionality and allow it to avoid incidents such as these, should it be required.

### 10.9.2 Focus in the *User Case*

This subsection takes the same data used in Section 10.7 and flips the axes, converting the data into histograms in an attempt to find which particular times during the scenarios the participants who viewed the *user case* focussed on. The amount of time taken by the different participants varied by a great deal in some cases, and uniformly differed to the amount of time taken by the *Glaykos* system. In order to directly compare the focus of the different participants to the each other, and to the generated debrief, the data has again been reduced to standard normal form.

Figure 10.23 shows the normalised histograms for each of the scenarios. Considering the offshore maintenance scenario in isolation initially, looking at the data by eye, there appears to be very little similarity between the times focussed upon by the individual participants and the generated debrief. There appears to be some overlap between the generated debrief and participant 15 around the five and six minutes marks, but other than this, the peaks in the data seem largely uncorrelated. The same also appears to be true for much of the other scenarios, with some exceptions.

Table 10.10 shows the (linear regression based) correlation between each pair of histograms in Figure 10.23a. As expected, there is very little correlation shown here between the human participants and the generated debrief, with Participant 15 showing the highest correlation of the three. The correlation between the individual participants is much higher, however, which is not as expected. A potential answer for this is the fact that the participants had fewer peaks in their histograms, as compared to the generated debrief, and this results in a higher correlation. This makes the correlation an unhelpful measure, as it is the times which were specifically focussed on in which we are interested, rather than those which were not.



(a) The offshore maintenance scenario.

(b) The mine counter measures scenario.

(c) The combination scenario.

Figure 10.23: The normalised timespan each participant spent viewing each time in each scenario.

|  | 13 | 15 | 21 |
|---|---|---|---|
| Generated | -0.0040 | -0.0791 | -0.0210 |
| 13 |  | 0.5125 | 0.5422 |
| 15 |  |  | 0.5589 |

Table 10.10: The correlation between the normalised histograms of the participants and the generated debrief in the offshore maintenance scenario (see Figure 10.23a).

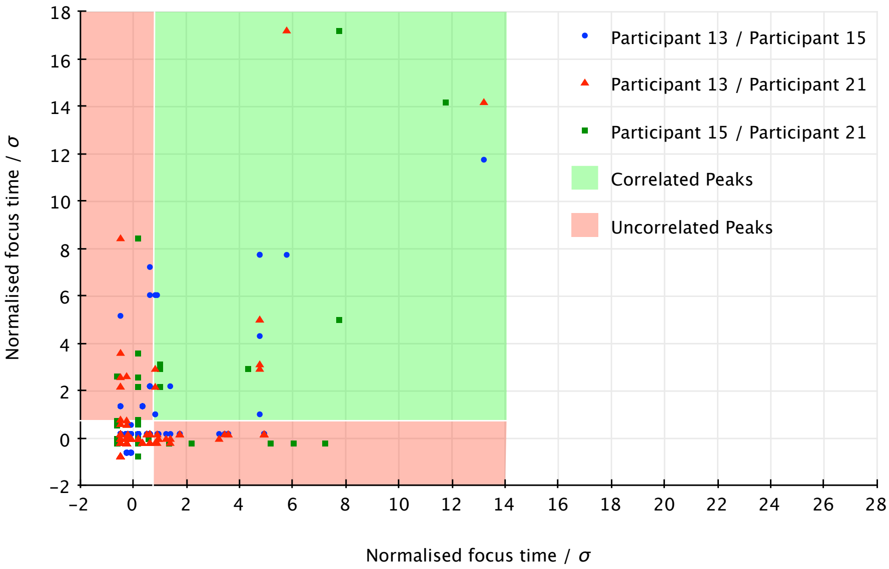


Figure 10.24: Scatter chart showing the relationship between the normalised timespan spent at each time by the human participants using the *user case* for the offshore maintenance scenario.

|  | 13 | 15 | 21 |
|---|---|---|---|
| Generated | 0.0360 | 0.000 | 0.000 |
| 13 |  | 0.1667 | 0.1860 |
| 15 |  |  | 0.1667 |

(a) The offshore maintenance scenario.

|  | 14 | 16 | 17 |
|---|---|---|---|
| Generated | 0.0180 | 0.0513 | 0.0270 |
| 14 |  | 0.1500 | 0.1455 |
| 16 |  |  | 0.2308 |

(b) The mine counter measures scenario.

|  | 20 | 22 | 23 |
|---|---|---|---|
| Generated | 0.0140 | 0.0539 | 0.0350 |
| 20 |  | 0.1587 | 0.2162 |
| 22 |  |  | 0.2258 |

(c) The combination scenario.

Table 10.11: The fraction of shared peaks between each combination of histograms for each scenario.

Figure 10.24 displays the the data for each pair of participants as a scatter plot showing that the majority of the points are clustered towards the lower left of the chart. The interesting data, that which represents the peaks in the individual histograms, makes up a lesser degree of the chart. This interesting data can be split up into two subsets:

1 Peaks which correlate between histograms;

2 Peaks which do not.

A value of of 0.75 standard deviations above the means has been taken to represent a peak, as this encompasses the vast majority of the peaks found for the data sets across the three scenarios. Figure 10.24 divides the data for the offshore maintenance scenario up thusly, showing the area which represents the correlated peaks in green and the areas which represent uncorrelated peaks in red. In order order to attach an actual measure to this, the following calculation is made:

$$c_{x,y} = \frac{\sum peak(x) \wedge peak(y)}{\sum peak(x) \vee peak(y)} \tag{10.1}$$

Or more simply: the number of correlated peaks in the two histograms divided by the total number of peaks in the two histograms.

The sub-tables of Table 10.11 show the calculated values for each of these comparisons. Once again, it can be seen the the individual participants are much more correlated with each other than they are with generated debriefs. A set of homoscedastic t-tests where run between the values for human versus *Glaykos* comparisons

| Scenario | Offshore | MCM | Combination |
|---|---|---|---|
| p-value | 0.0003 | 0.0082 | 0.0023 |

Table 10.12: T-Test results for comparisons of human versus human, and human versus generated debrief, as per Table 10.11.

and human versus human comparisons[3]. The results of these tests are shown in Table 10.12. This shows that in every scenario, the participants correlated with each other significantly more than they correlated with the generated debrief, in terms of particular times in the mission upon which were focussed. That said, they each disagree more than they agree, by around a factor of four in each case.

Two relevant things are known about the debriefs produced by the *Glaykos* system:

1 The times it focusses on correspond more or less exactly with the significant events in each scenario (in terms of accomplishing the goals of the mission);

2 It pauses at each for a reasonable amount of time.

This means that each of the peaks in the histograms for the generated debriefs correspond to the times of the actual events the *Glaykos* system was designed to convey to the user. There could be several reasons why users did not choose to focus on the same periods of time. They could simply have mean misguided, or more inclined to watch the significant events in motion, rather than pausing on them. Likewise, there could have been other spacial factors the users choose to focus upon, such as times at which the individual vehicles were in closer physical proximity to each other. As the intent of each participant was not recorded during the *user case* it is not possible to say for certain, however.

Possibilities for further experimentation will be described in Section 11.4.1, among which possibilities to further improve the quality of the data collected by the *user case* will be discussed.

### 10.9.3 Specific Opinions

The last three questions of each of the surveys were very open ended:

6 In what ways do you think the playback made the events during the mission and their relationship easy to understand?

7 In what ways do you think the playback made the events during the mission and their relationship confusing?

8 Do you have any other comments?

[3] Essentially, comparisons between the values in the top row of each sub-table in in Table 10.11, and the other values in the sub-table.

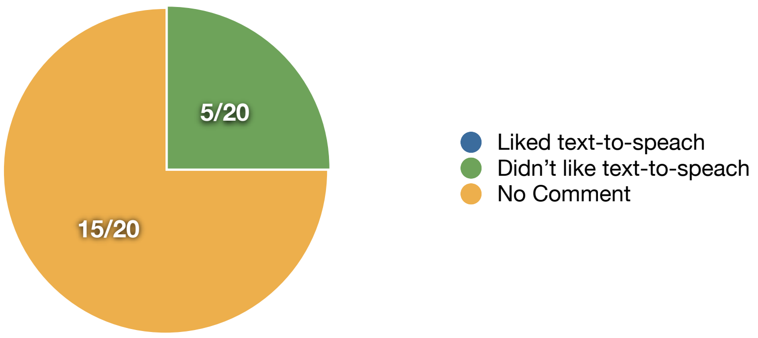


Figure 10.25: Participant opinion of the text-to-speech voice synthesis.

These provided a forum for the participants to express their feelings about the debriefs, without any other prompting. From this, opinions regarding three key aspects emerged. Each of these will now be covered in more detail in the following subsections.

### Opinion of Text-to-Speech

One fact which immediately becomes clear on observing Figure 10.25, and this is that 100% of the participants who commented on the speech syntheses system expressed a negative opinion. A relatively low quality speech synthesiser was used, and this explains some of the negative comments, for example:

> "The computer voice for the audio is hard to understand at first, and tends to clip words at the ends of sentences short so it is helpful to have the captions."
>
> *Participant 5*

> "The voice synthesiser was not clear on several words and phrases, which was distracting."
>
> *Participant 16*

However, others felt that the speech itself was frustrating, as is slowed down the rate at which they were able to view the mission. This opinion was often expressed vocally, but not actually recorded in the survey.

### Opinion of Non-Linear Ordering

One of the key differences between the *base* and *experimental cases* is the fact that the *experimental case* eschews chronology in favour of an ordering more closely tied to the causal relationships between the events. This proved to be far more contentious than the speech synthesis, with the majority of participants making

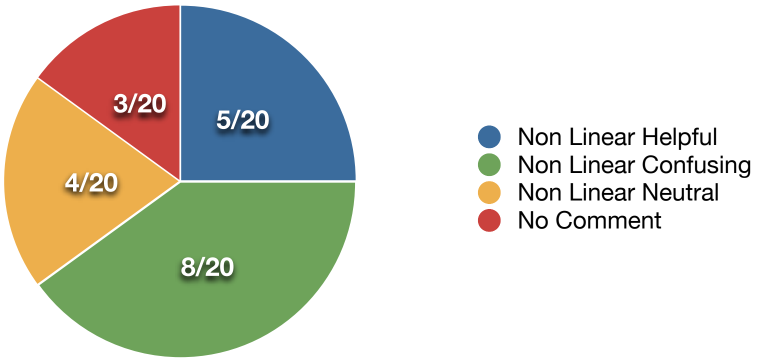


Figure 10.26: Participant opinion of non-linear ordering.

comment upon it, as shown in Figure 10.26. The largest portion of these were in the negative camp, expressing opinions such as:

> "The skipping backwards and forwards made the overall timeline of events harder to follow."
>
> *Participant 6*

However, a number also found that the non linear ordering made the events of the mission easier to follow and understand:

> "It addressed each step in the order of the mission aims, and not necessarily chronologically. This was not confusing, but emphasised how logical the progression of each segment occurred."
>
> *Participant 5*

> "The playback made the events easy to understand by going back in the timeline to examine events happening at the same time."
>
> *Participant 7*

A smaller number also commented on the non linear ordering, but did not feel that made any significant contribution, positive or negative, to their understanding of the mission:

> "I was undecided about skipping backwards and forwards through time, it didn't confuse, but I'm not sure if it contributed either."
>
> *Participant 3*

One very interesting opinion was put forward by the first participant:

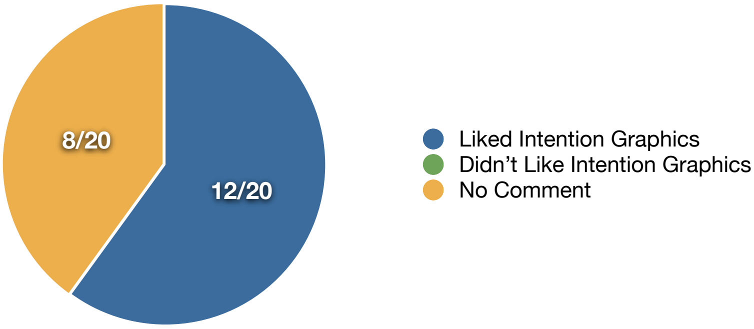


Figure 10.27: Participant opinion of intention graphics.

> "Stepping backwards and forwards in time was confusing, when concurrent action would probably be fairly easy to understand in this scenario. If it became more complicated with more units doing different tasks then this way would probably not get any more confusing, but concurrent action would become too busy to keep track of in detail."
>
> *Participant 1*

Although this is generally a negative opinion of the non-chronological ordering, it offers the constructive point that it might also lead to more clarity in complex missions. As the experiments had already started, a question regarding this was not added to the surveys, however the majority of participants who were informally asked if they thought the *experimental case* would scale better to more complex scenarios indicated that they thought it would.

### Opinion of Intention Graphics

Another difference between the *experimental* and *base cases* (as well as the *user case*, for that matter), is that the *experimental case* presents the user with a graph or flow chart which represents the motivation behind the event which is currently being described. In contrast with the speech synthesis, 100% of the participants who commented on this liked it, as shown in Figure 10.27 Additionally, a greater percentage of participants made comment. Here are some examples:

> "The extra graphical information was very helpful, as it provided a summary, and made the relationships between each element of the mission to the rest of the mission extremely clear."
>
> *Participant 6*

> "In addition the graphical version of the plan and its changes helped make the "big picture" more obvious."
>
> *Participant 23*

## 10.10 Summary

This chapter has described the results obtained from the experiments detailed in the previous chapter. These results were separated into the nine specific areas. The first eight of these described the results pertinent to each of the expectations given in Section 9.2, in order. The final section then discussed some additional observations which arose during and after the experiments themselves.

The length of the user's pause times was found to be longer for the *experimental case* then the bases case, contrary to expectation. Also contrary to expectation, the length of user's pauses was not found to be correlated to the distance moved along the causal dimension between delivered process statements. User's were found to prefer the *experimental case* to the *base case*, but not by a statistically significant amount. Likewise, the user's subjective assessments favored the *experimental case* over the *user case*, but this was only significant for selected categories.

As expected, some users did have a negative reaction to the lack of control afforded by the *experimental case*, particularly those who also viewed the *user case.* Also as expected, user's were found to have a significantly higher level of accuracy in the *experimental case* than either the *base* or *user cases.* User's were also found to require significantly more of the mission to be be repeated in order to be satisfied by their understanding of the scenario than was required by the debrief generated by the *Glaykos* system. Lastly, a wide variety of orderings was used by user's to recount the events of the mission, as was expected.

The additional observations were divided into three sub-sections. The first described the effects of the potentially critical incidents in each debrief (see Section 8) on the pause times of the user. This found that while the realiser based and some of the planner based incidents did appear to have an effect, those connected to narrative order did not. Next the times during the mission which users focussed on when using the *user case* were found to have no correlation to those focussed on by the generated debriefs, but some limited correlation to each other.

Finally, selected unprompted opinions expressed by the participants in the post experimental surveys were discussed. Some aspects received universally negative feedback (when any feedback was given), such as the speech synthesis used by the *base* and *experimental cases.* The additional motivation based graphics added to the heads up display during the *experimental case*, on the other hand, received acclaim from more than half of the participants, with no negative comments. Finally, the non chronological ordering used in the *experimental case* proved to be contentious, with participants being reasonably split between disliking this, finding it helpful, explicit ambivalence and not feeling it worthy of comment.

Some further discussion of these results, and how they reflect upon the work described in this thesis as a whole can be found in the following chapter, together with some discussion of possibilities for future work.

# Chapter 11

# Discussion and Future Work

This, the final chapter of this thesis, provides a discussion of the work carried here as a whole, with particular focus on the results which were provided in the previous chapter and their ramifications as to the performance of the system created during this work, and the knowledge which has been gained. Following on from this, potential avenues of future work are then discussed, both in terms of the big picture and smaller, more atomic, extensions. The chapter as a whole is then summarised, and final thought is given.

## 11.1 The Journey So Far

A discussion provides the author with the possibility to take a step backwards and considered their work as a whole. To take a review of their entire metaphorical garden and not just the individual plants and flower beds which comprise it.

This work is intended to look at the interface between autonomous underwater vehicles and their human operators, and advance the state of the art in this area. It began by looking backwards, at the work which has come before, the solutions which have been found, and the problems which have been encountered.

As befits such an interdisciplinary field, a wealth of relevant previous work was found, though in a highly disparate collection of areas. These ranged from work which helped clarify both the domain and the problem itself, through general guiding principles and potentially helpful techniques and technologies, to specific solutions for similar and related problems.

In particular, it became obvious that although autonomous platforms have a high degree of utility, and the potential for a great deal more, there is some deficit of trust between them and their operators in both the military and offshore domains. Fortunately there is cause to think that repeated use of a system can help reduce this deficit, though systems will inevitably become more complex as autonomy increases, and thus this represents a moving target. Be that as it may, the assumption was made that simulation can at least partially stand in lieu of the real system. It is also clear that the level of communication between the system and its user must be of the highest possible standard for this to be the case, however. Thus, the operator's debrief of the mission, be it carried out with real or simulated vehicles, must be as effective as possible.

Inspiration was drawn from the specific field of situational awareness as it relates to autonomous and automatic systems, and the more general field of human computer interaction. This yielded various principles which could be applied to this work, as well as several examples of state of the art user interfaces which could supply inspiration. Some additional attention was given to the fields of visualisation and natural language generation, as these have the potential to increase the quality of the human computer interaction still further. Evidence has shown that a narrative based structure can often be the best way of expressing information in order for the human mind to process it as quickly and effectively as possible, with a minimum of cognitive load. As a result, relevant work from the fields which attempt to study narrative, either for analytical or generative purposes, was also reviewed. A wealth of useful information was found here, as to formal structures which have been developed to represent narrative at different levels, and potential methods to create a metric to separate a "good" attempt at a narrative from a "bad" one.

Effective debriefing is only one part of the puzzle, however, and must sit amongst other related technologies in order to provide a complete user experience. A framework was designed in order to specify this set of technologies (Chapter 3) and provide the context for the debriefing system. This framework prioritises communicating with the user at the highest level possible, in terms of receiving data from the user, requesting details when required and delivering information back to the user. Further priorities include employing simulation to provide effective validation and using iterative improvement in order to ensure that the created mission accurately matches the user's specification.

As no one element of this framework can stand completely alone, three specific elements were implemented in order to support the specified aim of creating an effective debriefing system. Chapter 4 described these. Firstly a prototype planner based control system was created in order to enable real or simulated AUVs to carry out operations in partially unknown environments (Section 4.1).

The second element was a system which enabled autonomous missions to be simulated at speeds much faster than real time, enabling simulated testing (and through this verification) to be carried out as quickly as possible (Section 4.2).

The final element was a graphical user interface capable of replaying such missions (either simulated or real) with complete control over the temporal and spatial aspects, together with animated heads-up display graphics. This allows either the system or the user to define exactly how the replay is delivered and which additional information is supplied along with it (Section 4.3).

These three elements together form the method for generating input for, and then consuming the output of, the system which automatically generates mission debriefs. With the required supporting technologies in place, such a system could then be created. Previous work (see Section 2.10) suggested that structuring data in a narrative fashion would be the most effective methodology for delivering the debrief to the user. Thus, a strategy was pursued which attempted to reverse the process which is believed to occur in the human mind when a narrative is understood. Firstly the mission logs are analysed and transformed into a highly detailed "situation model" consisting of every event and data element from within the mission, together with every possible connection between them (Chapter 5). This is known as the "bitmap situation model", owing to the fact all detail is represented explicitly, as in a bitmap image. From this, a second situation model is created, containing less numerous but more complex elements designed to model the causation and motivation within the mission, and thus more closely match the situation model believed to be used by humans (Chapter 6). Continuing the image based metaphor, this structure is known as the "vector situation model" in reference to vector images which encode their detail more implicitly through more involved structures.

Crucially, the elements of a situation model are not arranged in any particular order; they represent their information in a more abstract and parallel structure. The next step, then, is to select an order for the elements so that they might be delivered to the user as part of a coherent discourse (Chapter 7). This is treated as an optimisation problem, and a genetic algorithm is used to select the most efficient ordering, attempting to minimise the distance travelled along the five situational axes of space, time, protagonist, motivation and causation. This creates a skeleton discourse which is then annotated with additional expository elements as required, and then translated into natural language for delivery to the user. From this the automated discourse is created, firstly a speech synthesiser is used to create audio narration, with which the timing for replay of the mission is created. Additional graphics are created based on the vector situation model in order to provide the user with additional information and context regarding the flow of causation and motivation throughout the mission.

Thus, a debrief is created which is intended to deliver the events of the mission to the user as quickly and efficiently as possible. This discourse is ordered in a way designed to reduce cognitive load and avoid needless repetition, whilst keeping the user situated along the axes of space, time, protagonist, motivation and causation.

In order to test this system, two specific comparisons were made (Chapter 9). Firstly, the created system (the *experimental case*) was compared to a similar automated debrief which used a chronological ordering of events (the *base case*). Additionally the *base case* did not provide the additional graphics and cues intended to help the user find their location along the five situational axes. Secondly, the created system was compared to a more manual debrief, designed to be similar in nature to the current state of the art (the *user case*).

In the course of testing the first comparison was found to be inadequate, however. Simple though it was, the *base case* still represented a significant improvement over the state of the art and was not significantly different enough to the experimental system in order to provide a satisfactory comparison. As a result of this, the second comparison, which was originally intended to only be used with expert subjects was extended to other participants as well.

As a result, three distinct experiments were carried out. The first used non-expert participants and the first comparison (*experimental* vs *base case*) with two different mission scenarios. The second again used non expert participants with the same two missions as before, but employed the second comparison (*experimental* vs *user case*). The final experiment used expert participants, and in order to reflect this a more complex mission scenario was created by combining the original two. As a result each expert only experienced a single mission scenario with either the *experimental* or *user case* and no direct comparison could be made.

In each experiment data was collected in two ways. Firstly, the user's behaviour was monitored. In the *user case*, this provided a wealth of information as a large amount of interaction was required. In the *base* and *experimental cases* interaction was more limited however, and the amount of time the user elected to pause between elements was all that could be recorded. Secondly, participants filled in a questionnaire after each experiment which elicited their subjective assessment of each case, as well as asking them to recount aspects of the mission, allowing a more objective assessment to be made.

## 11.2 The Present Location

When considering the results, it is import to note, first of all, that the system described here was never meant to represent any kind of complete solution, or finished product. Rather, it is intended to act as a stepping stone towards a complete system for the creation and validation of complex missions to be carried out by multiple autonomous assets. In other words, a system which conforms to the framework described in Chapter 3.

Among other things, this framework rejects the notion of a "one size fits all" solution for every operator. A "user profile" component is included in order for the system to adapt to each user and attempt to reason inductively regarding their preference and meaning. The results in Chapter 10 strongly support the need for this component, in particular those which relate to the users' opinion of the non-chronological ordering employed by the *experimental case* (Section 10.9.3), and the ordering selected by the participants to recount the events of the mission (Section

10.8). This first element proved quite contentious, with many participants stating that they found this confusing, or simply didn't like it. A notable number of participants professed to finding this ordering beneficial, however. Others stated ambivalence, feeling that it neither helped nor hindered, while some apparently did not consider the matter to be worthy of comment. Clearly the preference, and the potential benefit of this approach, rests entirely with the individual. Furthermore, the novelty of this debriefing strategy, almost by definition, makes it quite unconventional, and so the possibility that a negative reaction to change on the part of some of the participants is being observed cannot be discounted.

The ordering used by the participants when asked to describe the events of the mission in "the ordering [they felt was] most appropriate" was again split. Some correlation was found to the particular scenario being described, with the more complex "combination" scenario never being recounted in a strict chronological order and instead leaning towards the higher level "motivational" approach. Some correlation was also found to the method by which the scenario was delivered to the participant. Those who had experienced the mission in causal order were more likely to use a causal ordering than those who had not. Conversely, those who had experienced the information-light but control-heavy *user case* were the least likely to employ a causal ordering. This potentially suggests that their understanding of the causal relationships between the events of the mission was the most lacking.

Of those participants who viewed two scenarios, almost half (45%) elected to use a different ordering when describing different missions, with the nine potential combinations of orderings each being represented (see Table 10.9). This leads us to the conclusion that the "correct" ordering of the events in a mission is a function of both the user and the mission itself.

There is, however, some evidence of a difference between "what the user wants" and "what the user needs". Most of the participants in Experiment 2 who indicated they preferred the *user case* to the *experimental case* also indicated that they felt they were better informed by the *experimental case*. Furthermore, the results relating to participant accuracy show a significant increase for those who viewed the *experimental case* (see Section 10.6). This is massively out of proportion with the marginal level of preference the participants assigned to it. This is an important point, and should be stressed. It shows that just because the user does not like something does not mean that it is not beneficial to them. It is even a possible that there is a direct relationship between the two, as the user may be forced to concentrate more when a particular discourse pushes them outside of their comfort zone. This is speculation, however, and is neither directly supported these results, nor within the scope of this work. Regardless of this, it is clear that some form of persuasion may be required in order to help convince the user to accept what may be a more effective debriefing strategy, rather than falling back to what they find more comfortable.

Another tenet of the framework described in Section 3 is that the interaction between the system and the user should be a duologue, rather than the monologue created by the current version of the *Glaykos* system, which creates a monologue, as it is (as previously mentioned) intended to act as a first step towards a more complex and complete system. It is felt that the creation of an initial backbone which is able to analyse the potential orderings of a mission is a key step which is required for the implementation of the more complex user interaction which would be required to support a duologue with the user. To use an analogy: the system has been given the means to form an opinion before it is given the means to debate the merit of this opinion with the user.

The collected results, however, show a clear need for a duologue of some sort. Some users reacted negatively to the lack of control afforded by the *experimental case*, one of whom did not even require any contact with the control-heavy *user case* in order to prompt this (see Section 10.5).

It also seems possible that the use of a speech synthesiser contributed to the perceived lack of control. The intention behind using a speech synthesiser was to increase the efficiency and ease in the communication with the user. Speech is the native method of communication for most humans, and it was thought that some benefit could be gained from using it. In particular, it was thought that it would allow the user to concentrate on the scene they were watching, rather than having to glance backwards and forwards between the scene and textual subtitles. In reality, the speech synthesizer decreased the potential rate of information throughput, as users were able to read the subtitles in a fraction of the time it took the text-to-speech system to read them aloud. The quality of the speech synthesiser may also have been a factor here, with several users stating that they found it distracting and difficult to understand (see Section 10.9.3).

Users did, in contrast, express a marked preference for the additional contextual information provided by the *experimental case* (see Section 10.9.3), which also lead to a significantly higher rate of accuracy among participants than either the *base* or *user cases* (see Section 10.6). The results in Section 10.7 additionally show that the *Glaykos* system has the potential to reduce the amount of exploration required by the user in order to be sure that all of the key events in the mission have been viewed and understood. These results also reinforce the negative conclusion regarding the speech synthesis, since even though the participants required more exploration in order to gain an understanding with which they were satisfied, without the presence of a narration they required less time in order to do so.

Section 9.2 made 8 predictions regarding the outcome of the results. Of these, three were found to be incorrect:

1. In experiment 1 (*experimental* vs *base case*, non-expert participants), pause

times will be less for the *experimental case* than the *base case.*

2. In experiments 2 (*experimental* vs *user case,* non-expert participants) and 3 (*experimental* vs *user case,* expert participants), pause times will increase with distance moved along the causal dimension (see Section 7.1.2).

3. In experiment 1, users will prefer the *experimental case* to the *base case.*

One was found to be correct, though questionable/marginal:

4. There will be a preference for the *experimental case* over the *user case.*

Finally, four were found to be correct:

5. In experiment 2 in particular, users may have some negative reaction to the lack of control afforded by the *experimental case.*

6. In all experiments, the participants' accuracy in recounting the events of the mission will be higher for the *experimental case* than either the *base case* or the *user case.*

7. In experiments 2 and 3, the user will need to replay more time in order to understand the mission fully (as compared to the *experimental case*).

8. There will be considerable variation in the ways participants choose to repeat the events of the mission.

While prediction 2 did not hold true, neither did the converse, and no correlation (either positive or negative) was found. Given that the literature[97, 98, 99, 100, 101] has shown that there *is* a penalty in comprehension for a discontinuity on the causal axis, it seems likely that the paused times seen here do not represent an accurate manifestation of this. This being the case, it seems likely that this also reflects upon prediction 1, which was also incorrect and likely influenced by the presence of the on screen motivation graphics. Predictions 3 and 4, on the other hand, are highly subjective in nature, and reflect the user's own preference, rather than the actual efficacy of the demonstrated system. The dichotomy between "what the user wants" and "what the user needs" has already been discussed.

Though the list of (non-questionably) correct predictions does not constitute a majority, it does contain those which were considered to be the most important (predictions 6 and 7); these are objective measures which have the most relevance to increasing the user's knowledge of the mission in the most efficient manner possible. These results also have an additional interaction, as they show that the experimental case not only requires less repetition of events in order impart knowledge of the mission the the user, but that this knowledge is also more complete. This demonstrates that this work both advances the current state of the art, and further increases the knowledge which can be leveraged for further development in this area. That predictions 5 and 8 also proved correct shows that every user is different and must be adapted to, again echoing the need for the system to adapt to the user.

## 11.3 At the End of the Road

Though the clear differences between users make it difficult to make concrete predictions, it may be that the ideal autonomous mission debrief would be provided by a system which fuses the level of control provided to the user by the *user case* with the contextual information provided by the *experimental case*. The analogy of a guided tour of a museum is used in order to describe this. In order for the analogy to hold, the museum in question must be a slightly peculiar one, however. Specifically: one which provides no placards with information about its exhibits.

In this analogy, the current state of the art (which is both mimicked and to some extent surpassed by the *user case* used in the testing described herein) would correspond to a visitor entering the museum and proceeding to explore unaided. The visitor is able to wander through the exhibits at will and in any order they choose. They have complete control over their experience. However, they only have the layout of the museum to guide them, and so may view the exhibits completely out of contextual order. They may view an exhibit which shows a solution before that which illustrates the problem it was intended to solve. Worse, they may never find the problem at all, or be unaware of the significance of the solution.

The currently implemented system would then be likened to a traditional guided tour. In this, the visitor is given a predesigned tour, over which they have no control. The tour is designed to give the visitor all the information they require, together with any additional pertinent details for background and context. The visitor can be reasonably sure that they have gained the most from their experience. It is also likely that the information was delivered in an order which emphasised the important relationships between the exhibitions, and contained a minimum of repetition. However, they may be frustrated by both their lack of freedom of exploration and the tour's potential "lowest common denominator" approach.

Finally, the ideal system would be likened to the more modern "interactive tour." In this, the user is provided with a digital audio player and a set of headphones. The player contains a preset "ideal tour", on which it guides the user by playing a series of audio clips. The clip for each exhibit on the main tour "suggests" the next exhibit the user should view. The user is free to ignore this suggestion, however, and may view the exhibits in the order which makes the most sense to them. The clips can also suggest potential branches with additional information. The user can also wander freely, selecting which information they receive next by typing numerical codes on the exhibits into the keypad on the audio player. Additionally, the device could potentially appraise the user of what they have missed, ensuring once again that the user can be confident they have obtained the most from their experience (though this is not functionality which is present in the current generation of interactive tours).

This approach aids the user as much as possible, and makes the knowledge contained in the debrief easily discoverable, whilst also ceding control of the dialogue flow. In this way the three sets of human computer interaction guidelines given in Section 2.6 are more closely followed, particularly those given in *Rules of Etiquette, or How a Mannerly AUI should Comport Itself to Gain Social Acceptance and be Perceived as Gracious and Well-Behaved in Polite Society*[72].

The following section will describe some of the first steps which might be taken along a potential path to this ideal solution.

## 11.4 First Steps

The individual sections in Chapter 4 of this thesis contained Future Work sections which detailed potential extensions with a narrow focus on the particular subsystem they described. This section deals with the evolution of the *Glaykos* system towards the idealised implementation described in the previous section, and the potential extensions which may constitute it. The remainder of this chapter is divided into sections which discuss each of these in turn, numbering seven in total.

### 11.4.1 Further Experiments With the Present Implementation

While the experiments which were described in Chapter 9 produced some results which were both interesting and encouraging, it is also the case that much of the statistical analysis was hamstrung by the small numbers of participants in some of the experimental groups, especially when extreme outliers arose (such as in Section 10.6). This being the case, the primary recommendation should further experiments be carried out would be to maximize the number of participants, but also to split them into a smaller number of groups. Ideally, there should be at least ten participants in each experimental group.

A further recommendation would be that further experiments follow the pattern of Experiment 3 (see Section 9.1.3) and concentrate on results obtained using the combination scenario, is this is undoubtably the scenario which places the largest amount of stress on the participants, and thus is the most likely to produce a differential across the experimental cases.

In addition to the experimental and user cases being used, it is also recommended that the base case be reintroduced, along with a fourth case which sits between the base and experimental cases. This case would use a purely chronological ordering, as in the base case, but also add all of the "optimisations" (such as the intention graphics) used by the experimental case. This would reduce the number of variables which are varied between cases, and in particular allow the effect of the non-chronological ordering to be examined in isolation. In addition it is also recommended that the participants be asked to provide narration of their intent during the user case, so that this may also be analysed.

### 11.4.2 Mid and Post Mission Analysis

At present the *Glaykos* system is designed to create a debrief based on data obtained from a full mission. In the contexts in which the system is used here, this mission has always been a faster than real time simulation, and the debrief intended for pre-mission verification. The system additionally has the potential to be applied to mid and post mission analysis, however.

In order to apply the system in its current form to post mission analysis, few modifications would need to be made. All that would be required is a control system capable of controlling multiple autonomous assets to the successful conclusion of the mission[1], and additionally generate mission logs in the form understood by the *Glaykos* system. In this respect the system described in Section 4.1 is not suitable at present. This system was only ever designed to act as a functional prototype, and incubator for more advanced systems. The only real world tests that have been performed have been with a single robot, and then without the planning layer which would be required for complex missions. Progress is being made towards more advanced and complete systems, some of which is described in Section 2.3.

Applying the *Glaykos* system to the analysis of missions while they are in progress represents a more significant technical challenge. The constraints applied by underwater acoustic communications mean that contact with an AUV is rarely real time, and often carried out in unpredictable bursts with large amounts of time between them. Some previous work has explored methodologies[68] for maintaining operator awareness under these circumstances. This work was mainly concerned with the navigational state of a single AUV controlled from the executive layer, however. More complex missions carried out with multiple AUVs controlled from a higher level could potentially benefit from the additional insight the *Glaykos* system is capable of providing. This would require some changes to the system to allow it to decompose the mission in something close to real time as it is received, whilst simultaneously communicating updates to the operator.

[1] That is: the control system should be capable, under ideal circumstances, of successfully completing the mission upon which it has been deployed.

### 11.4.3 Dealing with Failure

At present the system is designed to deal only with "perfect" missions. This is unrealistic, as it does not allow for failures of any kind, or take into account the extremely low bandwidth of underwater communication.

The model used in the current version of the system has no vocabulary for expressing the latency in communication which is routinely found in the underwater domain. It lacks, therefore, a mechanism for explaining the reason for a potentially prolonged gap between cause and effect which is likely to result.

Also lacking is a vocabulary for expressing partial or total mission failure, both of which are distinct possibilities in such a hazardous domain. Furthermore, a mission which contains some sort of failure is more likely to be in need of explanation than one which does not. In fact, providing an explanation for mission failure has the potential to be the most important use of the system. Clearly, an ideal system should contain this vocabulary and the means to use it.

The first step towards an implementation of this would be extensions to the bitmap and vector situation models. The former would need additional constructs to represent failures of different types, as well as the time taken for communication to occur. The latter would require additional process statements to represent failure itself and more advanced plan manipulation. In particular, the natural language vocabulary used to represent these concepts would have to be very carefully chosen. Communication latency would also lead to the AUVs being out of synchronisation for longer periods of time, and the system would need methods of representing differences in knowledge and plan between the AUVs.

### 11.4.4 Generalisation

Many users elected to use a "motivational" ordering of events when recounting the mission they had viewed (see Section 10.8). This ordering groups events according to the high level motivation behind them, resulting in a generalisation of repetitive groups of events (e.g. "There were three installations, each of these was examined"). This is particularly appropriate for missions which contain repeated groups of similar actions. A large MCM mission, for example, might contain thirty detections, each of which would be inspected in a near identical manner. Fifteen of these might then be found to be mines, each of which would then be destroyed in a nearly identical manner. Repetition such is this is to be avoided, as the previous two sentences go some way to illustrating.

The ability to find patterns such as this and generalise them would be a valuable addition to the system. It would likely help the user find valuable abstraction in the cause and effect of a mission, and lead to the creation of concise mission debriefs, which might otherwise be long and tedious. Generalisation could also be used to provide “iterative deepening,” a technique which initially presents the user with a high level summary and then allows the more detail to be added selectively in stages.

In order to perform this sort of generalisation, a pattern matching system would be required to find the repetitions of similar sets of results and group them together. Additional techniques would also be required in order to express this generalisation to the user, both in terms of discourse ordering and natural language generation.

### 11.4.5 Control of the “Camera”

The present system makes no attempt to control the “camera”, instead adopting a static “birds eye view” of the mission. There are two reasons for this. Firstly, the problem was considered to be difficult and likely to require more development time than was available. Secondly, automated camera movement had the potential to be a contentious feature, which would be highly susceptible to the effects of user preference and therefore introduce an unwanted additional variable.

One downside to the use of a static viewpoint is that the user is able to see all the events which are occurring at the time being viewed, some of which might be part of a different causal thread which has not yet been communicated to the user.

The use of a controlled camera would allow the system to limit the user’s view to only those events which are currently relevant. In the case of the duologue based system described in Section 11, the ability to control the camera would allow the system to “show” an event to the user, rather than simply telling them where to find it. This could further allow the system to adapt to user preference by providing “levels of control.” Potential examples of this might include:

- **Autopilot**, in which the system utilises the user’s preferences to construct an “ideal” debrief and then communicates this with minimal interaction from the user.
- **Co-pilot**, in which the user tells the system what they would like to see and the system shows it to them.
- **Manual pilot**, in which the user takes complete control and the system tells them what they are looking at, what other events it is connected to, and potentially makes suggestions (in a similar manner to those provided by websites such as amazon.com and youtube.com).

### 11.4.6 Speed and Performance

One of the key areas in which this system could be improved is by decreasing the amount of time required to process a mission. We identify two key weak points in this area: the use of an ontology based knowledge base as the “bitmap situation model"; and the use of uninformed algorithms (genetic and bounded, constrained depth first search) in order to find the optimal ordering of events.

The use of an ontology based knowledge base was motivated by the ability it affords to quickly create complex searches using SPARQL (a query language similar to SQL), which would otherwise have to be created completely by hand. This lead to a valuable decrease in development time, at the expense of runtime performance. Converting the "bitmap situation model" to pure objects and coding these searches by hand would likely result in a significant decrease in the time required to construct the "vector situation model." Additionally, the degree of development required would be reduced, as the nature of the required searches is now known.

The use of a more domain specific informed algorithm could likewise increase the runtime performance of finding the ideal ordering of events. While genetic algorithms and bounded constrained depth first search are powerful tools, they are general problem solvers. Like ontologies, they operate at a very high level of abstraction and as a result require a high resource footprint. A more specific and targeted algorithm or heuristic could likely reduce this.

Also possible is that no concrete ordering strategy is required. If the *Glaykos* system is to form more of a duologue with the operator (as discussed in the previous chapter), then a more fluid approach might potentially be taken. After the processing and simplification steps described in Section 6.2.4 have been performed, more of the ordering could become implicit, and the user could simply be provided with options ordered by a more local heuristic whenever a choice becomes appropriate.

### 11.4.7 The "Well Rested Traveling Salesman Problem"

One change which could potentially aid the methodology described in the previous section is a slight shift away from the "traveling salesman problem" (TSP) based analogy which is used in the creation of the "ideal" ordering of events. In the TSP, each "city" is visited exactly once. As a result, even optimal solutions to the problem may contain "trips" between two very distant "cities." This is potentially a problem in our domain, as our intention is to decrease the cognitive load required by the user to construct an accurate situation model. As our cognitive model predicts that a longer "trip" is likely to correspond to an increase in cognitive load, it is possible that a slightly longer "circuit" which consists of shorter but more numerous "trips" could be considered a more optimal solution than a shorter circuit with longer individual "trips."

This updated model makes two changes to the standard (asymmetric, for our purposes) traveling salesman problem:

- It is possible to visit the same "city" more than once;

- There is an upper bound on the length of an individual "trip."

This has been named the "well rested traveling salesman problem." It is possible that this change in analogy can be used to help ensure that each newly introduced event has a strong link to at least one other node in the user's situation model, such as in the situation described in Section 7.1.6. However, more complex problem solvers might then be required in order to process it.

Additional influence may also be drawn from the various other studies into effective means of discourse ordering which have been performed, such as those by *Karamanis*[164], *Karamanis et al.*[165, 166], Althaus[167], and Lapata[168]. These have the potential to help shape the metric which is used for ordering, whilst also potentially extending to the higher level methodology which is employed.

## 11.5 Summary

This chapter has served two complementary functions. Firstly, it has looked backwards and discussed all of the work contained in this thesis. Particular focus was given to the results explained in Chapter 10, how these compare to the expectations from Chapter 9. Though not all of the expectations proved correct, a (bare) majority did, and crucially these included those which were considered to be most important. As result of this, the *Glaykos* system was considered to constitute an advance upon the current state of the art, and this work a source of knowledge which can be deployed by further research in this area.

Secondly, as befits a project intended to act as one part in a larger framework, this chapter looked forward to the potential extensions which can directly use this work as their foundation. In total, seven were discussed, with some representing optimisations or extensions to *Glaykos* itself, and others further components of the framework described in Chapter 3 which build directly on top of the *Glaykos* system.

## 11.6 Final Thought

The *Glaykos* system is intended to act as a first step towards improving the quality of communication between autonomous systems and their operators, and through this building trust between the two. This chapter has shown this first step to be firmly placed, on solid ground and, perhaps most crucially, there are other steps beyond it, easily reached.

# Part IV

# Epilogue: Appendices

> “But it ain’t all buttons and charts, little albatross. You know what the first rule of flyin’ is? Well I suppose you do, since you already know what I’m about to say.”
>
> *“Serenity” by Joss Whedon*

This, the fourth and final part of this thesis, stands in lieu of the concluding epilogue found in many traditional narratives. Like many epilogues, it serves to supply additional information which provides context to that which has come before.

Contained within are eight appendices, which provide additional details which are of some relevance to, but are not required for the understanding of, the main body of this work. The first of these describes the additional tools which have been used in the creation of the software components described herein. The second and third appendices described the on disk data formats used to described the domain and problem information used for planning and simulation, and the output logs of the control system, respectively. The fourth appendix described the specific parameters used to tune the various algorithms which are used. The fifth gives the SPARQL queries which are used by the algorithms in Chapters 5 and 6 to access the bitmap situation model ontology. The fifth provides a complete list of the syntactic structures used by the natural language generation system described in Section 7.3. The seventh provides the complete script for an example narrative debrief. Finally the eighth appendix gives examples of the surveys and instructions which were given to the experimental participants, as described in Chapter 9.

# Appendix A

# Tools

This appendix describes the tools which have be used to aid the design and implementation of this project. Some have been implemented by the author as by products of this or other projects, some have been implemented by others based in the Oceans Systems Laboratory, and others have been implemented by further third parties. It is noted in the text which of these three categories each item fits into.

## A.1 Languages

This section describes the programming and representational languages which have been used as part of these project, either as part of the direct implementation, or as some form of data storage.

### Java

Java [169] is a high level, object orientated programming language designed, implemented and maintained by Sun Microsystems, California. For the reasons stated below, it is also the programming language which has been selected for the implementation of this project. A complete reference to the language (including tutorials and API) can be found in [170], where as [171] gives a more informal (though less up to date) description of the language and the philosophy behind its implementation.

Although Java is syntactically very similar to C++, it operates at a higher level. User memory allocation is eliminated, the java runtime managing the destruction of objects as required. It also adheres more closely to the object orientated model, thus no creation of truly global variables and methods is possible. This leads to a more stable system where segmentation faults and memory leaks are highly unlikely. As errors obviously occur during the development process, Java also aides debugging by providing a detailed output of the “Exception” which caused the error and forced the runtime to exit. Java is also the language the author is most proficient and comfortable with.

Unlike many programming languages, Java does not compile to native code as standard (although there are several open source and commercial native Java compilers available). Instead it is compiled to an intermediate stage common to all systems, and then interpreted at runtime by a platform specific interpreter known as the *Java Virtual Machine*, or the *Java Runtime Environment*. This implementation scheme is often cited in as both the language's main strength and its major weakness. Being interpreted at this level provides Java with an extremely effective garbage collector to handle with memory allocation, a very robust threading model, easy integration of modular code from multiple projects and an extremely stable runtime. It also means that compiled Java code can run on any system with an installed JRE, without the need for porting or cross compilation.

As previously mentioned, this runtime implementation is also often seen as the source of Java's main weakness. As Java code is not executed at a native level it runs at a slower speed than it might otherwise. Additionally, as the JVM is required for Java runtime, running Java programs tend to have quite a large memory and resource footprint (relative to similar programs running from binary executables). However, since the implementation proposed in this project is intended to run on workstations, rather than embedded platforms with limited resources, this issues are not seen as serious problems, and it is felt that the strengths of the language more than make up for them when working with this kind of high level architecture.

## XML

XML [172] (eXtensible Markup Language) is an ISO standard model for storing data with markup. As an example, a subset of XML is HTML (HyperText Markup Language) which is used for storing pages on the world wide web. It is be used for the majority of the data storage in this project, with the exceptions noted below. Additionally, a set of extremely comprehensive tools for integrating it with Java are readily available.

## Groovy

Groovy [124] is a dynamically typed scripting language which is implemented with the same technology as Java, allowing direct integration between the two. Java code can be called directly from Groovy and scripts and classes defined in Groovy can be directly called within Java code, to the extent that Groovy code can be compiled at runtime and used as part of a Java program. For this reason Groovy is used as a potential language to define the behaviours used by the planner based control system described in Chapter 4.1, as it allows for additional behaviours to be added to the system after the main program has been finalised.

### OWL And SPARQL

OWL [173] (Web Ontology Language) is the W3C standard for representing ontologies and knowledge bases backed by them. It is based on RDF [174] (Resource Description Framework). SPARQL[175] (SPARQL Protocol and RDF Query Language) is a query language, similar in form to SQL, which is used to search such ontologies. Respectively, they are used in this project for Ontology definition and storage, and ontology query and search.

### PDDL

PDDL[46, 47] (Planning Domain Description Language) is, as previously noted, the state of the art planning language used each year as the basis of the International Planning Competition. It is used in this project as either the direct representation, or the underlying basis for all planning related constructs.

## A.2 Applications

This section describes the standalone application which have been used in the development of this project. The differentiation between the items in this section and those in the next is that although all of them have released source code, none of this code is used as part of the project, only the application itself is used.

### GraphViz

GraphViz[131] is a graph plotting and visualisation program developed by AT&T's Bell Labs, based on various published graph plotting algorithms, many of which can be found in [176]. It allows definition of graphs in a simple and highly flexible language, which is then processed and the graph output either in the same format with plotting information added, or as one of the various image types the application supports. It is used in this project as part of the graphical user interface visualisation system. Additionally, it was used to generate some of the diagrams in this thesis.

### Protégé

Protégé[177] is a free and open source ontology editor and knowledge base framework, developed at Stanford University. It is used in this project for the creation and editing of the ontology which describes the bitmap situation model, and for additional reference back to the ontology structure during development.

### Eclipse

Eclipse[178] is an Integrated Development Environment (or IDE) primarily for programming in Java, and also built using the Java language. It is built and maintained by IBM, and currently supported as an open source project. It provides a large amount of flexibility and extensibility, thus with extra plug-ins it can be used to writing XML and Groovy with full syntax highlighting and code completion. It is used in this project for all Java, Groovy and XML editing.

## A.3 Released Source Code

This section describes the source code released by various projects which is used directly by different parts of this project.

### Jena and ARQ

Jena[179] is an open source Java implementation of the OWL syntax and semantics. It allows for the input of an ontology defined in OWL (or various other formats) and the creation of individuals and properties in a knowledge base which conforms to the specified ontology. It then allows for query and search of this knowledge base, either by using a programatic API, or through SPARQL queries, the functionality for which is provided by the ARQ[180] package. These are used in this project for adding entries to the knowledge base used as the bitmap situation model, and searching the same.

### Pellet

Pellet[181, 182] is an open source OWL reasoner with greater functionality and speed than those provided by default in Jena. It is used in the this project to reason over the knowledge base, and most particularly provide consistency checking.

### SimpleNLG

SimpleNLG[90] is a simple, but very powerful, natural language generation system created by Ehud Reiter, based upon his book *Building Natural-Language Generation Systems*[89]. It is used in this project to transform predicate based planning terminology (and other elements) into plain English.

### FreeTTS

FreeTTS[159] is an open source speech synthesis program developed by Sun Microsystems. It is used in this project to read explanations and information aloud to the user.

### JME and LWJGL

JME[126] (Java Monkey Engine) is a high performance game engine programmed entirely in the Java language, using the LWJGL[127] (Light Weight Java Game Library) OpenGL bindings in order to enable hardware accelerated graphics capabilities. The projected is supported and maintained by the video game developer NCSoft, but freely available on an open source licence. JME is used in this project for all 3D graphics which are displayed to the user.

### FengGUI

FengGUI[129] is an OpenGL based library for overlaying a 2D "Heads Up Display" over the top of 3D graphics. It is used in this project to visually provide the user with additional information during mission debriefing.

### AUV Simulator

The AUV Simulator is a hydrodynamic model of the RAUVER AUV. It takes thruster values as input, converts these to axis force vectors and then calculates the resultant velocity and position of the simulated AUV. It is used in this project to provide the dynamics of the AUVs during mission simulations.

### AutoPos

The AutoPos is a waypoint based autopilot developed in the Ocean Systems Laboratory for the control of AUVs. It takes a position in three dimensional space as a command, and then uses a Proportional Integral Derivative (or PID) control system to set the value of the vehicle's thrusters in real time in order to bring it to its destination as quickly and exactly as possible. It is used in this project for the waypoint based control of real and simulated AUVs.

### Timing Framework

The Timing Framework [130] is an open source timing and animation framework, developed by Chet Haase at Sun Microsystems, as part of the Swing Labs Java Desktop Technology project. Elements of it are used in this project to aid the animation of visualisations.

### Groovy

The source code used to implement the Groovy[124] dynamic programming language is freely available under an open source licence. This code is used in this project in order to enable runtime compilation of Groovy code into Java compatible .class files for use as dynamically loaded behaviours in the planner based control system.

# Appendix B

# Input Files

This appendix describes the elements of the system described in Chapter 3 as *domain information*, *mission goal*, *world data* and *simulation data*. These are implemented across several XML files, with a dependency structure which will be explained.

## B.1 Sensor and Actuator Descriptions

These files form part of the *domain information*, and provide data about sensor and actuator systems which an AUV may be equipped with. Presently, the files' primary purpose is to supply the simulation system with the information it requires to create reasonable facsimiles of these systems for the purpose of fast simulation. Additionally, this information is used to visualise the sensor and actuator during mission playback. Sensor systems detect the state of the world, and we will first consider the description of one of these. An example of such a file (with some elements curtailed for brevity) is shown below:

```
<data>
 <meta type="sensor" name="under_scanner"/>
 <sensor name="under_scanner" default="passive">
  <description friendly="..." curt="..." verbose="..."/>
  <mode name="passive">
   <description friendly="..." curt="..." verbose="..."/>
   <simulation active="false"/>
  </mode>
  <mode name="active">
   <description friendly="..." curt="..." verbose="..."/>
   <simulation active="true">
    <detects class="property" type="under_status"/>
    <geometry range="7" vertical="50" horizontal="50"/>
    <attitude roll="180" pitch="0" yaw="0"/>
   </simulation>
  </mode>
 </sensor>
</data>
```

The top level `data` tag is common to all XML files designed to be loaded by the "Data Stream" resource loading framework which is used by the *Glaykos* system. Below this the "meta" tag provides "Data Stream" with information about the contents of this file for indexing purposes. The sensor description proper begins on line three with the `sensor` tag, which contains attributes for the name of the sensor and its default state.

Contained below this in the hierarchy is the description tag. This contain information used by the output processing system to convey the purpose of this sensor to the user.

Below this are one or more `mode` tags, which represent the modes this sensor can be put in. The first of these in this example represents the `passive` mode, which is the default for this sensor. Below this we find another `description` tag, which is used to communicate the the effect of switching the sensor to the this mode to the user. After this is the information required for simulation: in this case simply that the sensor is not active in this mode.

The second mode this sensor can occupy is `active`, and in this case the information required for simulation is more in depth. Again we find the simulation tag, this time with its active attribute set to true. Below this is a single `detects` tag (through the file format allows for multiple of of these), which specifies which world state information the sensor is able to detect. In this case the sensor detects properties of the type `under_status`. A sensor might also detect the existence of a particular type, in which case the `detects` tag would be similar to the following:

```
    <detects class="kind" type="mine"/>
```

The last two tags contain information used define the geometry of the sector within which the sensor is able to detect. The first of these is the `geometry` tag, which contains attributes to represent the range, vertical angle and horizontal angle of the sector. Below this is the `attitude` tag, which defines the direction in which the sensor points, in this case it is rotated through 180 degrees in order to point directly upwards.

Next we consider the description of an actuator system. Actuators are similar to sensors, but actively attempt to change the state of the world rather than simply detecting it. An example is show below:

```
<data>
 <meta type="actuator" name="arm"/>
 <actuator name="arm" default="passive" type="grab">
  <description friendly="..." curt="..." verbose="..."/>
  <mode name="passive">
   <description friendly="..." curt="..." verbose="..."/>
   <simulation active="false"/>
  </mode>
  <mode name="active">
   <description friendly="..." curt="..." verbose="..."/>
```

```
11      <simulation active="true">
12        <effect type="grabbable"/>
13        <geometry range="2"/>
14      </simulation>
15    </mode>
16  </actuator>
17 </data>
```

Again, the file begins with information required by the “Data Stream” system. On line three the actuator description begins, and follows a very similar format to the sensor definitions. The first difference is that the `actuator` tag also contains an `effect` tag with a `type` attribute, used to specify the nature of the actuator. This is required as actuator systems are able to affect the world in a myriad of fashions and as such it is not possible to create a completely generic representation for their simulation. This example defines a “grab” actuator, which is modeled after a manipulator arm and allows the AUV lift objects in the world and carry them to other locations. Also implemented in simulation is a “destroy” actuator, modeled after an explosive device, and a “visible” actuator, modeled after an intervention arm. These require differing geometric representations, though the same notation is used for the two in the definition.

It is intended that in future (more advanced) versions of the system depicted in this thesis, this definition will include `real` as well as `simulation` tags, which will contain the protocol information required to control the real sensor and actuator systems during a non-simulated mission.

## B.2 Module Descriptions

Modules definitions complete the domain information. However, while the sensor and actuator definitions contain information which is most pertinent to lower level systems, the module definitions contain the data which is required for the functioning of the planner and other more high level systems. They essentially replicate the the domain description used by PDDL (see [46]), but in a more modular and extensible form. An example of a module definition is shown below:

```
<data>
 <meta type="module" name="examine_installation"/>
 <requires type="module" name="auv"/>
 <module name="examine_installation">
  <type name="installation" extends="grabbable" noun="installation"/>
  <type name="status"/>
  <constant type="status" name="good" noun="working␣correctly"/>
  <constant type="status" name="bad" noun="malfunctioning"/>
  <predicate name="scanned_under" verb="is">
   <parameter name="what" type="instalation" role="subject"/>
  </predicate>
  <predicate name="under_status" verb="be␣examined">
   <parameter name="what" type="instalation" role="subject"/>
   <parameter name="value" type="status" role="complement"/>
  </predicate>
  <action name="scan_under" verb="scan␣under">
   <parameter name="who" type="auv" role="subject"/>
   <parameter name="where" type="location" role="none"/>
   <parameter name="what" type="installation" role="complement"/>
   <precondition>
    (and (at ?who ?where) (lifted_at ?what ?where) (has ?who examine_installation))
   </precondition>
   <end>
    (scanned_under ?what)
   </end>
   <control file="ScanUnderControlScript.groovy"/>
  </action>
  <sensor name="under_scanner"/>
 </module>
 <requires type="sensor" name="under_scanner"/>
</data>
```

The definition begins with the standard enclosing `data` tag, which is again followed by a `meta` tag, allowing the contents of this file to be indexed by the system. This is in turn by a `requires` tag, which tells the same indexing system that this module first requires a resource of type `module`, named `auv` to be loaded.

Next is the `module` tag itself, which has a single attribute providing its name. Contained within this tag are the definitions of the types and predicates which can be used to describe the world and the goal of the mission, as well as the actions which the planner uses to attempt to satisfy the goal. The module definition may contain:

- Zero or more type definitions;
- Zero or more constant definitions;
- Zero or more predicate definitions;
- Zero or more action definitions.

The first, and simplest, of these definitions is provided by the `type` tag, and is used to define the types of objects which can be present in a mission. Each of these must provide a `name` attribute, and may optionally provide an `extends` attribute in order to create a type hierarchy (as in the `installation` type shown in the example). In addition to this information, each type may also supply additional information to be used by the Natural Language Generation (NLG) system. In the example, the `installation` type also supplies a `noun` which can be used to describe objects of this type, and a relevant preposition may also be supplied. See Section 7.3.1 for a description of the natural language information which can be attached to the domain and how it is used. This ability to attach arbitrary additional formation to the elements of the domain is one of the main motivations for using XML to define the domain.

PDDL allows instances of objects to be defined in the domain definition to be used as constants, and this also possible using *Glaykos*' XML input format. Two constants are defined in the example using the `constant` tag. Each is required to provide attributes which define the `type` and `name` of the constant. As with type definitions, they may also have NLG information attached to them, as is the case in the example.

The next set of definitions is for the predicates.. As in PDDL, these consist of a predicate and a set of typed parameters. The predicate itself is defined by the `name` attribute of the `predicate` tag. Contained within this are zero or more `parameter` tags with `name` and `type` attributes. As is the case with the type definitions, additional information used by the NLG system is also attached to the definition, providing a verb to be used when converting the predicate into a sentence, and the role each parameter plays in such a sentence.

Action definitions are similar in nature to predicate definitions, as they also defined by a predicate and a set of typed parameters. This being the case, the `action` tag carries a `name` attribute and contains `parameter` tags with `name` and `type` attributes. Again these elements may also have NLG information attached to them of a similar form to that used by the predicate definitions.

An action definition also requires a precondition and an effect, however. The precondition is defined by the `precondition` tag, and is provided in raw PDDL. Throughout the input files used by *Glaykos*, complex terms such as this are always provided in raw PPDL, as this allows existing parser technology to be leveraged and simplifies the task of parsing them. As no additional mark-up is required to be added to these terms, this does not decrease the effectiveness of the system. The effects of the action are split into those which happen at the start and those which happen at the end, in order aid the control system (see Section 4.1.2). These are again raw PDDL terms, and are enclosed between a set of `start` and a set of `end` tags, respectively. The action used in the example only applies effects on completion, however.

The final component of the action description is an element not present in the PPDL definition; the `control` tag. This supplies the control systems with the data it needs to allow the actual action to be carried out with either a real or simulated vehicle. In this case the information is contained in a class written in the *groovy*[124] scripting language which will be compiled at runtime. Alternatively, the information can be contained in a precompiled Java class. In this case a `java` attribute would be used in place of the `file` attribute seen here, and this would supply the fully qualified name of the class which defines the behaviour. For example:

```
    <control java="uk.ac.hw.ece.osl.auv.ScanUnder/>
```

sensors and actuators

The final element in this definition is a single `requires` tag. This signals to “Data Stream” that another resource is required. In this case the requirement is specified at the end of the file in order to ensure that the requirement is satisfied after the module definition has been processed, as one of the types defined by the module is required by the sensor definition.

## B.3 World Descriptions

World definitions provide the problem definition to the planning system, together with some of the information required by the simulation system. The world definition implements the World Data component described in Chapter 3. An example is shown below:

```
<data>
 <meta type="world" name="grabbing"/>
 <requires type="module" name="grab"/>
 <requires type="module" name="examine_instalation"/>
 <requires type="colour␣scheme" name="x11"/>
 <world name="grabbing">
  <auv type="auv" name="auv1" model="rauver" colour="yellow" noun="Manipulator␣AUV">
   <start name="auvStart1" north="0" east="0" depth="9" noun="manipulator␣AUV␣start"/>
   <module name="grab"/>
  </auv>
  <auv type="auv" name="auv2" model="nessie" colour="yellow" noun="Inspection␣AUV">
   <start name="auvStart2" north="0" east="5" depth="9" noun="inspection␣AUV␣start"/>
   <module name="examine_instalation"/>
  </auv>
  <item name="o1">
   <start name="o1Location" north="20" east="8" depth="15" noun="site␣A"/>
   <contains type="instalation" name="o1instalation" noun="instalation"/>
  </item>
  <item name="o2">
   <start name="o2Location" north="8" east="20" depth="15" noun="site␣B"/>
   <contains type="instalation" name="o2instalation" noun="instalation"/>
  </item>
  <location name="drop_off" north="0" east="0" depth="15" noun="drop␣off␣location"/>
  <goal>
   (forall (?what - instalation)
           (and (scanned_under ?what)
                (imply (under_status ?what bad)
                       (at ?what drop_off)
                )
           )
   )
  </goal>
 </world>
</data>
```

The file again begins with the enclosing `data` tag. Within this are: first of all, the `meta` and `requires` tags used by the “Data Stream” system to load the correct resources; and secondly the world definition itself. The the only other attribute in the `world` tag itself is the `name` of the world, used for reference purposes.

Within the `world` tag are the definitions for:

- One or more AUVs;
- One or more item definitions;
- Zero or more location definitions;
- Zero or more area definitions;
- Zero or more pipe definitions;
- Exactly one goal definition.

In this example there are two AUVs. The `auv` tag contains the following attributes:

- The `type` of object used to represent the AUV in the planning system;
- A `name` for the AUV, this is used both for the name of the object used by the planning system and for reference in the simulation system;
- The `model` used for the geometry of the AUV during simulation and playback;
- The `colour` of the key geometry in this model (see Section 4.3.1);
- A `noun`, which is used to represent this AUV in natural language constructs.

Within the `auv` tag is exactly one tag containing the `start` position of the vehicle, and zero or more `module` tags. The `start` tag contains the Cartesian coordinates and water depth of the AUVs start point, together with a noun which is used in natural language structures which refer to this location. This will generate a `location` object with an automatically generated name in the planner's world state, as well as providing a physical starting position to the simulation system.

Each of the `module` tags tells the simulation and playback systems that the AUV has a particular module (and any associated sensors and actuators), and prompts the planning system to add a proposition indicating that the AUV has a capability with the same name.

Item, pipe and area definitions are used to provide information to the system about physical objects in the world which are know to the AUVs at the beginning of the mission. In the example are two `item` definitions, which describe discrete physical objects. These enclose further tags which describe the `location` of the item and the existents or properties they contain. This provides all the information needed by the planning system and data server in order to deal with the item. Areas and pipes are similar in nature to items, but contain multiple locations and cannot be labeled with planner terms.

This example contains one `location` definition, but world definitions may also contain `area` and `pipe` definitions. All three of these create relevant objects in the planning systems initial world state, and are also added to the control system's Data Server component (see Chapter 4.1.2).

Lastly the goal definition is provided in pure PDDL between the `goal` tags. This is parsed in by the PDDL parser and passed to the planning system.

## B.4 Simulation Descriptions

Simulation descriptions contain data about the world that the user does not wish the planning system to know about. That is, elements which should be simulated, but are intended to be discovered by the AUVs during the mission, rather than being known at mission start. It also contains information used purely by the simulation system, such as data used for the dynamics of the world objects and triggers for simulated failures. An example simulation definition, which corresponds to the world description example used in the previous section, is shown below:

```
<data>
 <meta type="simulation" name="grabbing1"/>
 <requires type="world" name="grabbing"/>
 <simulation name="grabbing1" world="grabbing" seed="13111981">
  <auv name="auv1">
   <power totalKWh="100" charge="100" thrusterW="25" />
  </auv>
  <auv name="auv2">
   <power totalKWh="100" charge="100" thrusterW="25" />
   <failure system="thrusters" mode="true" condition="time" value="
       360000"/>
  </auv>
  <item name="o1" default="default">
   <property>
    (under_status o1instalation bad)
   </property>
   <state name="default" colour="gray"/>
   <state name="scanned" colour="red"/>
  </item>
  <item name="o2" default="default">
   <property>
    (under_status o2instalation good)
   </property>
   <state name="default" colour="gray"/>
   <state name="scanned" colour="green"/>
  </item>
 </simulation>
</data>
```

The definition begins with the standard enclosing `data` tag, followed by the `meta` and `requires` tags, which are used to ensure correct resource loading. Next is the `simulation` tag itself. This contains the name of the simulation, which is used purely for reference, the name of the world the simulation operates in (and adds additional data to) and finally a `seed` attribute, which holds the random seed used for any random elements which are used as part of the simulation. Within the `simulation` tag are definitions for:

- Zero or more `auv` tags, containing additional data for the AUVs defined in the world definition.
- Zero or more `item` tags, either providing additional data regarding the items defined in the world definition, or adding new items to the world.

The additional data which may be added to the AUV definitions consists of:

- Zero or one `power` tags, describing the power characteristics of the simulated AUV.
- Zero or more `failure` tags, describing failures which may happen to the AUV during the course of the mission.

Both of these are evidenced in our example files. The `failure` tag in this example would cause a failure to occur in the `auv2`'s thrusters after a preset length of time. Failures are described in more detail in Section 4.2.4.

In the instance that an `item` tag is intended to add a new item to the world (one which is unknown to the AUVs at the start of the mission) it should contain all of the required data. Otherwise, it needs only to describe the additional data which is not included in the world definition. In our example, additional properties are added to each of the installations which describe to current status of each.

In addition, the item blocks in the simulation definition should add one or more state tags to the item. These describe the physical appearance of the object (though at this stage, only its colour is supported). The state in which the the item starts the simulation is defined by the default attribute in the item tag. The state of an item can be changed on interaction with the sensors and actuators of the AUVs. In our example, each of the items is designed to change colour in order to represent its status after it has been scanned.

# Appendix C

# Output Log Files

Three types of log file are output from the simulation system (or any real vehicles deploying an appropriate control system). In addition, the simulation system outputs an additional log file, which describes the changes in position and state of each of the simulated objects in the world. In this chapter, each of these files and the log events they contain will be discussed in an appropriate amount of detail.

The log events in the the output files are represented as XML, which is converted to and from an object representation for use in memory. Each log event derives from a single base class, which contains basic functionality for reading from and writing to file, as well as the following information:

- The type of event;
- The name of the agent which is the source of the event;
- The time at which the event happened.

This being the case, each event is enclosed in tags which have the form:

```
<event type="position" agent="auv1" time="0">
</event>
```

The remainder of this chapter is divided into four sections, each of which describes the contents of one of the types of log file.

## C.1 Position Logs

These logs store the navigational information of each agent involved in the mission. Contained within are numerous instances of a single event. Despite this, they are likely to be largest of the types of log file, as in the current system the event is recorded five times a second for the length of the mission. An example position is shown below:

```
<event type="position" agent="auv1" time="0">
```

```
  <local-position north="0.0" east="-15.0" depth="5.0" altitude="0.0"/>
  <orientation roll="0.0" pitch="0.0" yaw="0.0"/>
  <local-speed north="0.0" east="0.0" depth="0.0" altitude="0.0"/>
  <rotation-speed roll="0.0" pitch="0.0" yaw="0.0"/>
  <body-speed x="0.0" y="0.0" z="0.0"/>
  <global-position latitude="-15.0" longitude="0.0" depth="5.0" altitud
</event>
```

The event is enclosed in the previously mentioned "event" tags. Following this is all of the data required to give a complete navigational description of the AUV, namely:

- Its position in local frame, measured in metres along the north, east and depth axes, relative to a local origin, together with an altitude value relative to the seabed;
- Its orientation in terms of roll, pitch and yaw.
- The speed at which it is moving along the north, east and depth axes.
- The speed at which is it rotating.
- The speed at which it is moving in its own body frame (surge, sway and heave).
- Its global position in geodetic coordinates.

## C.2 Platform Logs

These logs detail the instructions sent to the underlying vehicle platform during the run of the mission. Specifically, these logs contains the waypoints which are requested, and any requested mode changes for the sensors and actuators attached to the the vehicle. An example "waypoint" event is shown below:

```
<event type="waypoint" agent="auv1" time="1000">
  <waypoint number="1" mode="0" speed="1.0" x="1" y="1" z="1"
     yaw="1" depthMode="true">
    <local-coordininate north="0.0" east="-15.0" depth="15.0"
       altitude="0.0"/>
    <direction-request roll="0.0" pitch="0.0" yaw="
       11.309932708740234"/>
    <position-tolerence x="0.5" y="0.5" z="0.5"/>
    <direction-tolerence roll="5.0" pitch="5.0" yaw="5.0"/>
  </waypoint>
</event>
```

The "waypoint" event contains a single `waypoint` tag, which holds all of the data which is required to instruct the vehicle to move to a new location. Within the tag itself is the number of the waypoint (new waypoints are only carried out when the number changes), the mode of the waypoint, the speed the vehicle should move at, a set of axis enable values (which tell the vehicle which control axes should be actively controlled) and finally a flag which indicates whether the waypoint is relative to the water surface (depth mode) or sea floor (altitude mode). With this tag are further tags which represent the request in terms of position and orientation, as well as the `tolerance` (the margin for error within which the waypoint is considered to have been achieved) for the same.

The second type of event recorded in these logs is the "sensor and actuator mode" event. As the control system uses named states to control the sensors and actuators, these are all that is recorded in the log files. Each event of this type contains one or more "sensor" or "actuator" tags, which specify the name of the sensor or actuator, together with the name of its new state. An example "sensor and actuator mode" event is shown below:

```
<event type="sensor␣and␣actuator␣mode" agent="auv1" time="439200">
  <actuator name="arm" mode="active"/>
</event>
```

## C.3 Planner Logs

These logs contains all the information pertaining to the operation of the planning system during the mission. The majority of the events which happen throughout the course of a mission are related either directly or tangentially to the beliefs of the agents carrying out the mission. Additional events describe the changes in the desires and intentions of the agents.

### C.3.1 Belief Change Events

A belief change event contains the following data, further to that defined in the base log event type:

- A list of existents which have been added to the world;
- A list of beliefs which have been updated;
- A flag declaring whether this belief change triggered a replan in the source agent's planning system.

Belief change events are sub-divided into those which have an internal source and those which have an external source. The "internal source belief change" event subclass does not add any additional data to the event and is used to provide a clean hierarchy. The "Initial Beliefs" class further subclasses this without adding any additional data, but is used to provide a clear naming scheme. It is used to specify the beliefs of an agent right at the beginning of a mission, before any other events have occurred. An example of this might be:

```
<event type="initial␣beliefs" agent="auv1" time="0">
  <added name="arm" type="actuator"/>
  <added name="grab" type="capability"/>
  <added name="auv1" type="auv"/>
  <added name="auvStart1" type="location"/>
  <updated predicate="start" status="true">
    <parameter name="auv1" type="auv"/>
    <parameter name="auvStart1" type="location"/>
  </updated>
  <updated predicate="has" status="true">
    <parameter name="auv1" type="auv"/>
    <parameter name="grab" type="capability"/>
  </updated>
  <replan value="false"/>
</event>
```

This is a very simple example, as the actual recorded events can be extremely large (depending upon the complexity of the mission), however all the previously mentioned elements of a "belief change" event are evident.

The "intention status" again subclasses the "internal source belief change", this time referring to the effects produced by carrying out an intention. It adds additional data to the event, specifying the intention which caused the change. It is itself subclassed into "intention stated" and "intention succeeded" events, which delineate the changes caused when an intention started and when it completed. The two events are identical aside from their name, so we give an example only of the "intention succeeded" event:

```
<event type="intention␣suceeded" agent="auv2" time="500000">
  <updated predicate="examined" status="true">
    <parameter name="o1detection" type="detection"/>
  </updated>
  <replan value="true"/>
  <intention predicate="examine">
    <parameter name="auv2" type="auv"/>
    <parameter name="o1location" type="location"/>
    <parameter name="o1detection" type="detection"/>
    <requires predicate="above" status="true">
      <parameter name="auv2" type="auv"/>
      <parameter name="o1location" type="location"/>
    </requires>
    <requires predicate="at" status="true">
      <parameter name="o1detection" type="detection"/>
      <parameter name="o1location" type="location"/>
    </requires>
    <requires predicate="has" status="true">
      <parameter name="auv2" type="auv"/>
      <parameter name="classify" type="capability"/>
    </requires>
    <expects predicate="examined" status="true">
      <parameter name="o1detection" type="detection"/>
    </expects>
  </intention>
</event>
```

This example updates a single belief and causes the planning system to initiate a replan. In additional, it can be seen that the intention is included in its entirety, complete with its requirements and expectations, instead of just a symbolic representation. This ensures that the event is self contained to some extent, and relies on a minimum of other events or external data in order to carry its meaning.

The final belief change which is considered to have an internal source is the “fault status” event, which is designed to convey the changes in belief caused by the diagnosis of a real or simulated fault. In additional to the standard data carried by the “belief change” event, this event also contains:

- The name of the affected system;
- A name given to the diagnosis of the fault.

As no simulated faults are used in any of out final testing, and the *Glaykos* system described in Part II is not yet equipped to cope with faults, no example is included.

Belief change events which come from an external event contain an extra data element, namely a string representing the source of the data. There are two types of “external source belief change” events, “sensor” events which contain belief changes generated by the agent’s own observations, and “communication” events which contain belief changes observed by other agents and then communicated to this agent. With “sensor” events, the source name corresponds to the sensor which made the detection, such as in the following example:

```
<event type="sensor" agent="auv1" time="112400">
  <added name="o1location" type="location"/>
  <added name="o1detection" type="detection"/>
  <updated predicate="at" status="true">
    <parameter name="o1detection" type="detection"/>
    <parameter name="o1location" type="location"/>
  </updated>
  <replan value="true"/>
  <source name="sidescan"/>
</event>
```

This example show the result of a detection made by auv1's "sidescan" sensor, which results in the creation of a detection and a location, as well as the belief that the detection is at the location.

With "communication" events the source string corresponds to the name of the agent the communication was received from. "Communication" events also contain an additional "id" value, which is used to link to the received communication to the point at which is was sent by its source. An example of a communication event, which corresponds the same information contained in the example sensor event, is shown below:

```
<event type="communication" agent="auv2" time="375600">
  <added name="o2detection" type="detection"/>
  <added name="o2location" type="location"/>
  <updated predicate="at" status="true">
    <parameter name="o2detection" type="detection"/>
    <parameter name="o2location" type="location"/>
  </updated>
  <replan value="true"/>
  <source name="auv1"/>
  <id value="375800"/>
</event>
```

The "communication sent" event is used to tie this event to the point in the source agent's timeline at which it made this communication. This event contains only the "id" of the communication, and is exemplified below:

```
<event type="communication␣sent" agent="auv2" time="481000">
  <id value="481000"/>
</event>
```

The use of this event allows the post mission analysis systems to cope both with the potential time lag associated with acoustic communications and the potential for none synchronised clocks being used by the different agents.

"Data associations" are structures which link the existences used by the planning system (specifically those with a "location" type) to locations in the real world. The "data association change" event is used to record whenever these are changed, which can be when the initial data is added at the start of a mission, or as the result of either a "sensor" or "communication" event. This event is a direct subclass of the base "log event" type, and contains only a list the updated associations, which are represented as key/value pairs. An example, which once again corresponds to the previous “sensor” event, is shown below:

```
<event type="data␣association␣change" agent="auv1" time="112400">
  <association>
    <key name="o1location" type="location"/>
    <value north="25.0" east="8.0" depth="13.0" altitude="2.0"/>
  </association>
</event>
```

The final log event which relates to the beliefs of the agents is “frozen/thawed” events, which is used to convey the results of the mechanism described in Section 4.1.1, which allows the planning system to discount agents which have become uncommunicative. As with the “fault status” event, this is not used in test simulations and so no example is provided.

### C.3.2 Desire Change Events

“Desire change” events describe the changes in the desires of an agent throughout the mission. Along with the data included in the “log event” base class, each contains a complete list of the current desires of the agent (rather than a list containing only the additions). An example might be:

```
<event type="desire␣changed" agent="auv1" time="112600">
  <desire predicate="examined" status="true">
    <parameter name="o1detection" type="detection"/>
  </desire>
  <desire predicate="searched" status="true">
    <parameter name="area1" type="area"/>
  </desire>
</event>
```

The “initial desires” events performs a similar function to the “initial beliefs” event, recording the desires of the agent at the start of the mission. It extends the basic “desire change” events to also add the meta-desires of the agent to the log. Meta-desires are discussed in more detail in Section 5.3.2. An example of an “initial desires” event, with included meta-desires, is shown below:

```
<event type="initial␣desires" agent="auv1" time="0">
  <meta-desire>
    <right predicate="scanned_under" status="true">
      <parameter name="?what" type="instalation"/>
    </right>
  </meta-desire>
  <meta-desire>
    <left predicate="under_status" status="true">
```

```
      <parameter name="?what" type="instalation"/>
      <parameter name="bad" type="status"/>
    </left>
    <right predicate="at" status="true">
      <parameter name="?what" type="instalation"/>
      <parameter name="drop_off" type="location"/>
    </right>
  </meta-desire>
  <desire predicate="scanned_under" status="true">
    <parameter name="o2instalation" type="instalation"/>
  </desire>
  <desire predicate="scanned_under" status="true">
    <parameter name="o1instalation" type="instalation"/>
  </desire>
</event>
```

### C.3.3 Intention Change Events

"Intention change" events are used to record the changes in the intentions of the agents as the mission progresses. Aside from the data contained in the base "log event", these contain:

- A list of the intentions which constitute the *entire* plan;
- A correspondence list, which provides the previous index of each of the intentions in the plan (or "-1" if the intention is new);
- A list of the indexes of the intentions which will be carried out by *this* agent; and
- The event (if any) which is currently being carried out by this agent.

These first two elements allow the exact state of the plan to be tracked across the entire mission, and ensure that different intentions with the same predicate and parameters (a possibility) are not confused. The second two elements make it possible the track the actions and intentions which are specific to the agent in question. An example of an "intention change" event is show below:

```
<event type="intention␣change" agent="auv2" time="112600">
  <intention predicate="search">
    <parameter name="auv1" type="auv"/>
    <parameter name="area1" type="area"/>

    ...
  </intention>
  <intention predicate="move_above">
    <parameter name="auv2" type="auv"/>
    <parameter name="o1location" type="location"/>

    ...

  </intention>
  <intention predicate="examine">
    <parameter name="auv2" type="auv"/>
    <parameter name="o1location" type="location"/>
    <parameter name="o1detection" type="detection"/>

    ...

  </intention>
  <correspondence value="0"/>
  <correspondence value="-1"/>
  <correspondence value="-1"/>
  <local value="1"/>
  <local value="2"/>
  <current value="1"/>
</event>
```

As with the “intention succeeded” event shown in Section C.3.1, this event contains the intentions in their entirety for the sake of completeness. As this would result in an extremely long example in this case, the intentions have been foreshortened by removing their requirements and expectations in order to make the event easier to view. In this example their are three intentions in the plan, the first of which has been carried over from a previous plan, and the second two are newly added. The second two intended to be carried out by this agent, and of these it has already begun the first.

## C.4 World Object Logs

This file contains the log for the state and position of simulated objects in the world. Two types of event are used to record this. The first records the position of the object and is identical to that used for the position of the AUVs, as described in Section C.1. The seconds is used to record changes in the state of the object (represented visually as changes in colour). The event contains a single `state` tag specifying the new state of the object, as shown in the example below:

```
<event type="world␣object␣state␣changed" agent="o1" time="0">
```

```
  <state name="default"/>
</event>
```

# Appendix D

# Operational Parameters

## D.1 Base Genetic Algorithm Parameters

Number of genes in the population.

$population = 1000$

Maximum time for the GA to run for (in milliseconds).

$runTime = 128000000$

Number of generations after which to quit if the fitness doesn't improve.

$cutoff = 20$

Number of genes to use for crossover.

$genePool = 200$

Number of genes to clone.

$vips = 16$

Number of new genes to introduce in each generation.

$imegration = 80$

## D.2 Fitness Function Parameters

The gain applied to the distance along the space axis.

$f_{space} = 0.1$

The maximum distance along the space axis.

$m_{space} = 1.0$

The gain applied to time when moving forwards.

$$f_{timeForwards} = 0.004666667$$

The gain applied to time when moving backwards.

$$f_{timeBackwards} = 0.006666667$$

The additional penalty applied to time when moving backwards.

$$p_{timeBackwards} = 0.2$$

The maximum distance along the time axis.

$$m_{time} = 5.0$$

The cost of moving to a different protagonist, whom the user does know about.

$$p_{knownProtagonist} = 0.2$$

The cost of moving to a different protagonist, whom the user doesn't know about.

$$p_{newProtagonist} = 1$$

The penalty if both elements have a motivation frame which refers to the same desire, but the levels of the frames are not consecutive.

$$p_{noneConsequative} = 0$$

The penalty if the elements do not have motivation frames which relate to common desire, but do have motivation frames which refer to desires which have the same meta-desire.

$$p_{differentDesire} = 0.7$$

The penalty if the elements have no commonalities across their motivation frames.

$$p_{differentMetaDesire} = 1$$

The penalty for when an element is added to a discourse which does not already contain both the element which is considered to have caused and it, and all of those which are considered to have contributed to it.

$$p_{notAllContributions} = 100.0$$

The gain otherwise applied to the causal axis (measured within the discourse between this element and that which caused it).

$$f_{cause} = 0.2$$

The maximum value for the causal axis (excepting the previously stated penalty).

$$m_{cause} = 6.0$$

## D.3 Script Generation Parameters

The number of short term memory registers the user is assumed to have (a high value reduces repetition).

$$memoryRegisters = 30$$

The maximum number of a times a fact will be implied.

$$maxPrime = 3$$

The length of the pause between the initial stasis statements (those in the *relies upon* set) in milliseconds.

$$introPause = 200$$

The length of the pause between the initial stasis statements and the process statement in milliseconds.

$$prePause = 200$$

The length of the pause after the process statement in milliseconds.

$$postPause = 200$$

The length of the pause between the final stasis statements (those in the *introduces* set) in milliseconds.

$$outroPause = 200$$

The length of the pause after all of the statements in milliseconds.

$$afterPause = 500$$

# Appendix E

# SPARQL Queries

This appendix gives the SPARQL queries which are used by the algorithms in Chapters 5 and 6 to access the bitmap situation model ontology. These are subdivided into sections named for the section of the main body of this thesis in which they are used.

## E.1 Processing Existences, Beliefs, Desires, Intentions and Data Associations (Section 5.3.1)

The following SPARQL query is used to find times at which existents are added to the world model. It returns a set of tuples. For each time in each agent's timeline at which no existent is added to the world, the tuple will contain just the name of the agent and a reference to the time. For each time at which an existent is added to the world state, the tuple will also contain a reference to this existent. In the event that multiple existents are added at the same time, one tuple will be returned for each. These are ordered, first by the name of the agent who's timeline they refer to, and then by the time.

```
SELECT DISTINCT ?time ?name ?existent ?timeObject ?agentname
WHERE {
    ?agent         data:name             ?agentname ;
                   data:timeLine         ?timeObject .
    ?timeObject    data:time             ?time .
    OPTIONAL {
        ?timeObject    data:event           ?event .
        ?event         ref:type             data:BeliefChange ;
                       data:addExistences   ?existent .
        ?existent     data:nameSymbol      ?name .
    }
}
ORDER BY ?agentname ?time
```

The following query is used to find times at which beliefs in the world model are updated, together with the events which caused the change. Data is returned in a similar format to the above query.

```
SELECT DISTINCT ?time ?belief ?timeObject ?statement ?status ?name
```

```
WHERE {
    ?agent         data:name         ?name ;
                   data:timeLine     ?timeObject .
    ?timeObject    data:time         ?time .
    OPTIONAL {
        ?timeObject   data:event               ?event .
        ?event        ref:type                 data:BeliefChange ;
                      data:updatedBeliefs      ?belief ;
                      data:order               ?order .
        ?belief       data:concerningStatment ?statement ;
                      data:status              ?status ;
                      data:order               ?index .
    }
}
ORDER BY ?name ?time ?order ?index
```

The following SPARQL query performs the same function for changes in desire:

```
SELECT DISTINCT ?time ?desire ?timeObject ?name
WHERE {
    ?agent         data:name           ?name ;
                   data:timeLine       ?timeObject .
    ?timeObject    data:time           ?time .
    OPTIONAL {
        ?timeObject   data:event          ?event .
        ?event        ref:type            data:DesireChange ;
                      data:desireAdded    ?desire .
                }
}
ORDER BY ?name ?time
```

The following SPARQL query performs the same function for intention changes:

```
SELECT DISTINCT ?time ?event ?timeObject ?name
WHERE {
    ?agent         data:name         ?name ;
                   data:timeLine     ?timeObject .
    ?timeObject    data:time         ?time .
    OPTIONAL {
        ?timeObject   data:event        ?event .
        ?event        ref:type          data:IntentionChange .
    }
}
ORDER BY ?name ?time
```

The following SPARQL query performs the same function for data associations:

```
SELECT DISTINCT ?time ?association ?timeObject ?ename ?name
WHERE {
    ?agent         data:name         ?name ;
                   data:timeLine     ?timeObject .
    ?timeObject    data:time         ?time .
    OPTIONAL {
        ?timeObject   data:event                  ?event .
        ?event        ref:type                    data:DataAssociationChange ;
                      data:changedAssociation     ?association .
        ?association  data:associatedExistence   ?existent .
        ?existent     data:nameSymbol             ?ename
    }
}
ORDER BY ?name ?time
```

## E.2 Resolving Communications Events (Section 5.3.5)

The following SPARQL query is used to retrieve every belief which is changed by a communications event, the communication event which precipitated the change locally, and the remote event which caused the original change on the source vehicle. This is used to connect the

```
SELECT DISTINCT ?comms ?updated ?event
WHERE {
    ?comms         ref:type                    data:CommunicationEvent ;
                   data:communicationID        ?id ;
                   data:updatedBeliefs         ?updated ;
                   data:fromAgent              ?src .
    ?updated       data:concerningStatment     ?statement ;
                   data:status                 ?status .
    ?sent          ref:type                    data:CommunicationSent ;
                   data:communicationID        ?id ;
                   data:atTime                 ?timeObj .
    ?timeObj       data:ofAgent                ?src ;
                   data:believed               ?updated2 .
    ?updated2      data:concerningStatment     ?statement ;
                   data:status                 ?status ;
                   data:updateEvent            ?event .
}
```

## E.3 Building the Motivation model (Section 6.2.2)

The following SPARQL query is used to to retrieve a reference to each intention, the time it was added to the plan, the time it was started, the time it was completed and the name of the agent which carried it out:

```
SELECT DISTINCT ?intention ?sTimeVal ?eTimeVal ?aTimeVal ?agentName
WHERE {
    ?sEvent     ref:type                    data:IntentionStarted ;
                data:concerningIntention    ?intention ;
                data:atTime                 ?sTime .
    ?eEvent     ref:type                    data:IntentionSuceeded ;
                data:concerningIntention    ?intention ;
                data:atTime                 ?eTime .
    ?aEvent     ref:type                    data:IntentionChange ;
                data:intentionAdded         ?intention ;
                data:atTime                 ?aTime .
    ?sTime      data:time                   ?sTimeVal .
    ?eTime      data:time                   ?eTimeVal .
    ?aTime      data:time                   ?aTimeVal ;
                data:ofAgent                ?agent .
    ?agent      data:name                   ?agentName .
}
```

The following SPARQL query is used to retrieve the intentions which have a requirement on at least one other intention which is carried out by the same agent, and the intentions which satisfy these requirements. Also retrieved are the requirements which are satisfied, and the beliefs which satisfied them, but this information is mainly use for debugging purposes:

```
SELECT DISTINCT ?intention ?intention2 ?requirement ?belief
WHERE {
    ?event        ref:type                    data:IntentionStarted ;
                  data:concerningIntention    ?intention .
    ?intention    data:requires               ?requirement .
    ?requirement  data:satisfiedBy            ?belief .
    ?belief       data:updateEvent            ?event2 .
    ?event2       ref:type                    data:IntentionStatus ;
                  data:concerningIntention    ?intention2 .
    FILTER (?event != ?event2)
}
```

The following SPARQL query has the same purpose as the previous query, however it finds the instances in which an intention's requirements are satisfied by the intention of another agent:

```
SELECT DISTINCT ?intention ?intention2 ?requirement ?belief
WHERE {
    ?event          ref:type                    data:IntentionStarted ;
                    data:concerningIntention    ?intention .
    ?intention      data:requires               ?requirement .
    ?requirement    data:satisfiedBy            ?belief .
    ?belief         data:updateEvent            ?event2 ;
                    data:concerningStatment     ?statement ;
                    data:status                 ?status .
    ?event2         ref:type                    data:CommunicationEvent ;
                    data:dataSource             ?event3 .
    ?event3         data:concerningIntention    ?intention2 ;
                    data:updatedBeliefs         ?belief2 .
    ?belief2        data:concerningStatment     ?statement ;
                    data:status                 ?status .
}
```

The following SPARQL query is similar to the previous two, but is used to find intentions which are invalidated by other intentions of the same agent:

```
SELECT DISTINCT ?intention ?intention2 ?requirement ?belief
WHERE {
    ?event          ref:type                   data:IntentionStarted ;
                    data:concerningIntention   ?intention .
    ?intention      data:requires              ?requirement .
    ?requirement        data:satisfiedBy                       ?belief .
    ?belief         data:updatedBy             ?belief2 .
    ?belief2        data:updateEvent           ?event2 .
    ?event2         ref:type                   data:IntentionStatus ;
                    data:concerningIntention   ?intention2 .
    FILTER (?intention != ?intention2)
}
```

The following SPARQL query again finds invalidations, but those created by the intentions of other agents:

```
SELECT DISTINCT ?intention ?intention2 ?requirement ?belief
WHERE {
    ?event          ref:type                   data:IntentionStarted ;
                    data:concerningIntention   ?intention .
    ?intention      data:requires              ?requirement .
    ?requirement    data:satisfiedBy           ?belief .
    ?belief         data:updatedBy             ?belief2 .
    ?belief2        data:updateEvent           ?event2 ;
                    data:concerningStatment    ?statement ;
                    data:status                 ?status .
    ?event2         ref:type                    data:CommunicationEvent ;
                    data:dataSource             ?event3 .
    ?event3         data:concerningIntention    ?intention2 ;
                    data:updatedBeliefs         ?belief3 .
    ?belief3        data:concerningStatment     ?statement ;
                    data:status                 ?status .
}
```

The following SPARQL query retrieves each desire, along with its meta-desire, the time which it was added to the mission and the intention which solved it:

```
SELECT DISTINCT ?intention ?desire ?added ?meta
WHERE {
    ?desire       ref:type                    data:Desire ;
                  data:satisfiedBy            ?belief ;
                  data:superDesire            ?meta ;
                  data:desireAddEvent         ?event .
    ?event        data:atTime                 ?time .
    ?time         data:time                   ?added .
    ?belief       data:updateEvent            ?event2 .
    ?event2       ref:type                    data:IntentionStatus ;
                  data:concerningIntention    ?intention ;
                  data:atTime                 ?time2 .
    ?time2        data:time                   ?satisfied .
}
```

# Appendix F

# Complete List of Syntactic Structures

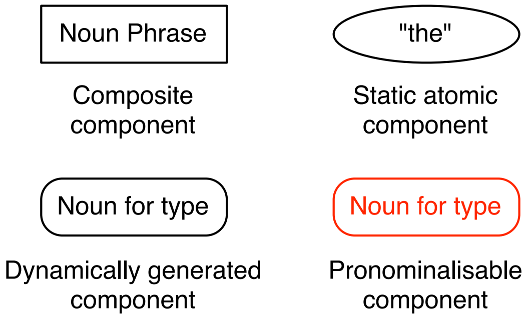


Figure F.1: The legend for the syntax diagrams used in the following sections.

## F.1 Process Statements

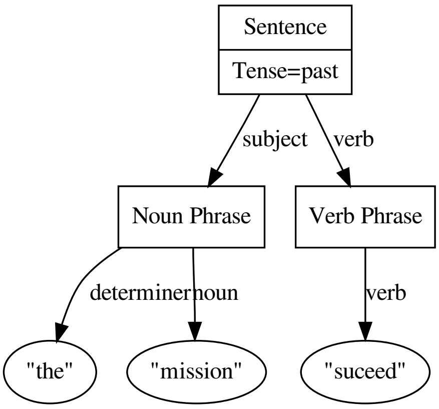


Figure F.2: Syntactic structure for "Mission Succeeded" process statement.

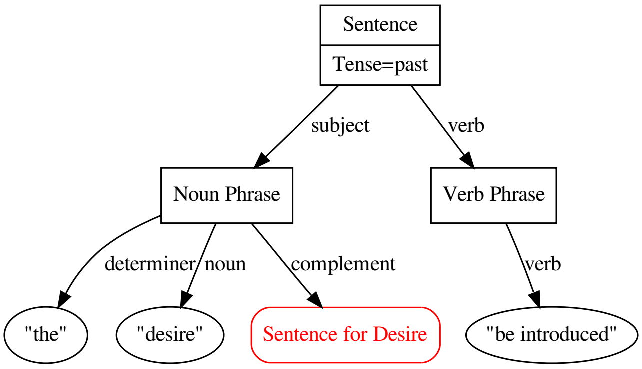


Figure F.3: Syntactic structure for “Desire Introduced” process statement.

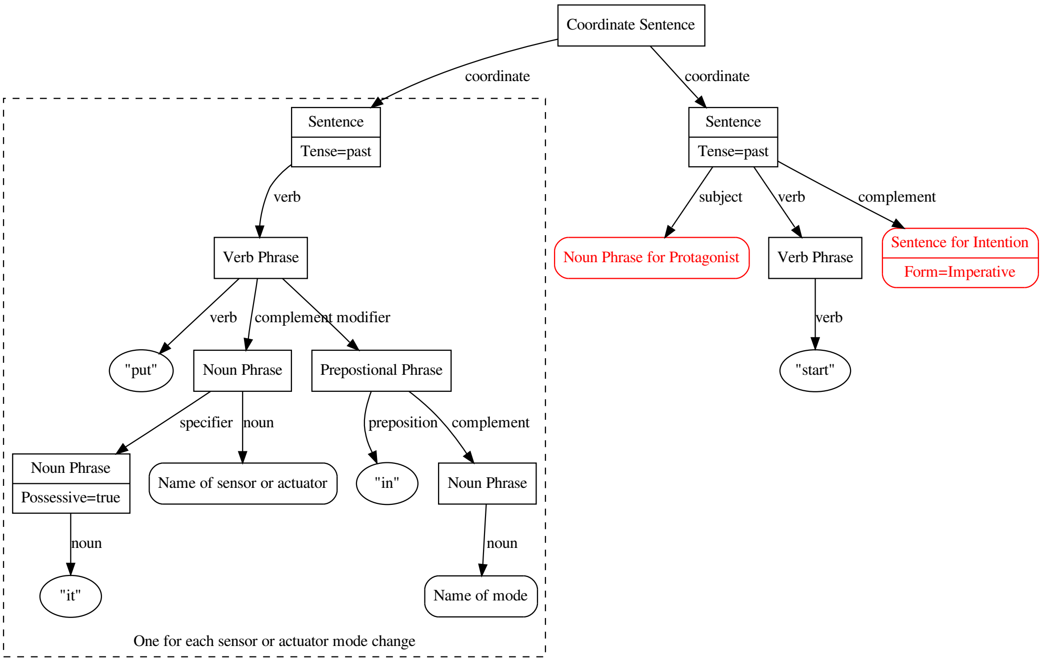


Figure F.4: Syntactic structure for “Intention Started” process statement.

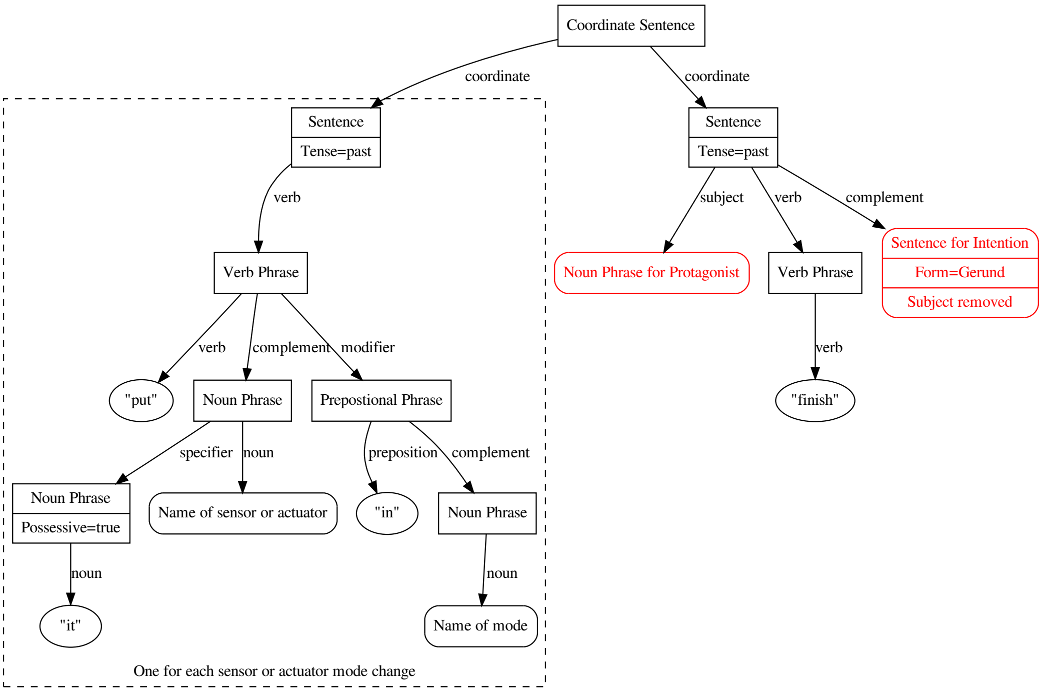


Figure F.5: Syntactic structure for "Intention Successful" process statement.

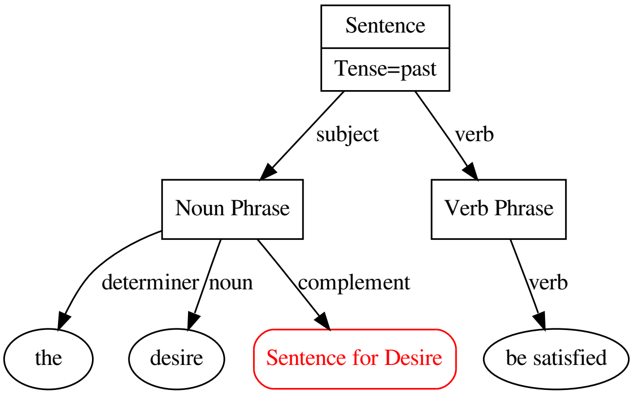


Figure F.6: Syntactic structure for "Desire Satisfied" process statement.

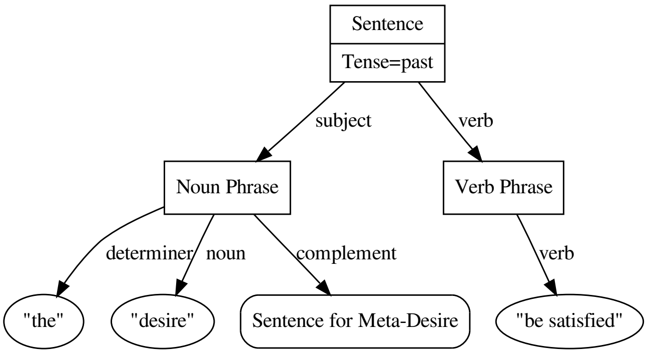


Figure F.7: Syntactic structure for “Meta-Desire Satisfied” process statement.

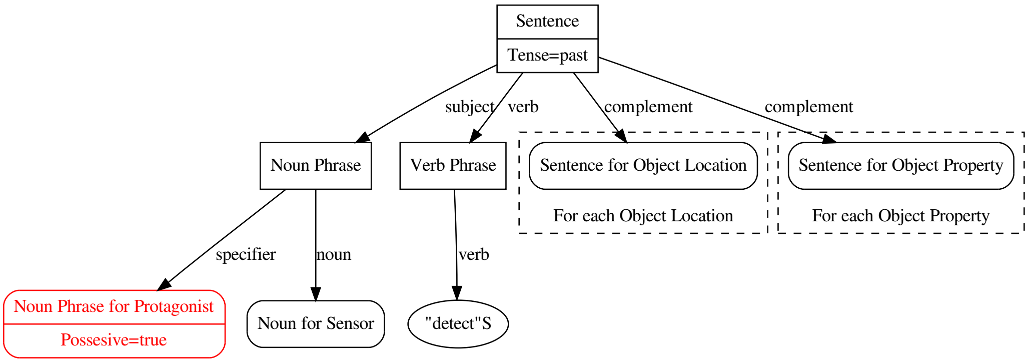


Figure F.8: Syntactic structure for “Sensor Detection” process statement.

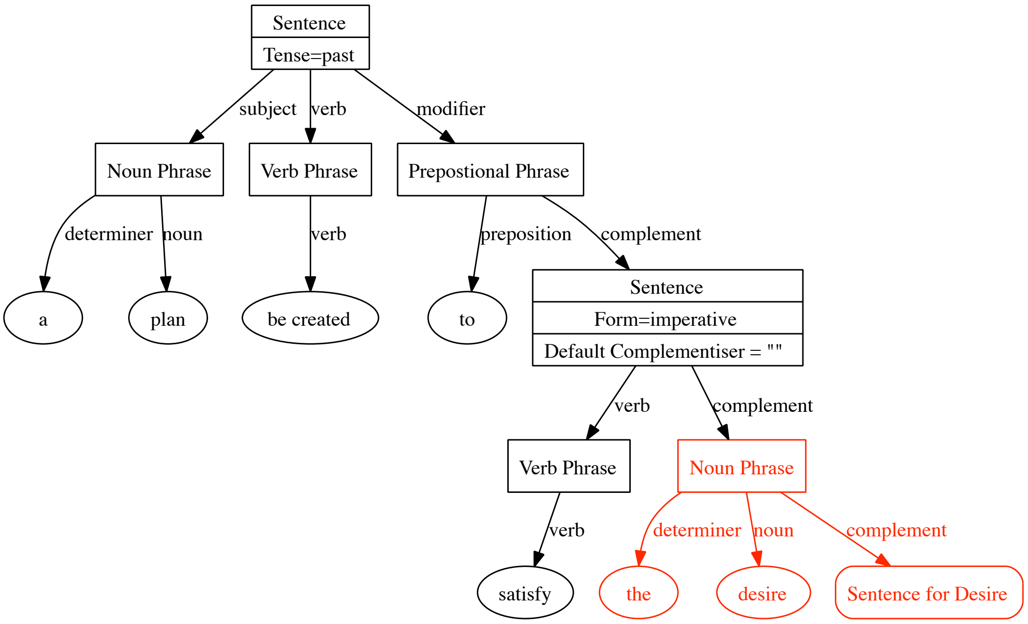


Figure F.9: Syntactic structure for the "Plan Introduced" process statement.

## F.2 Stasis Statements

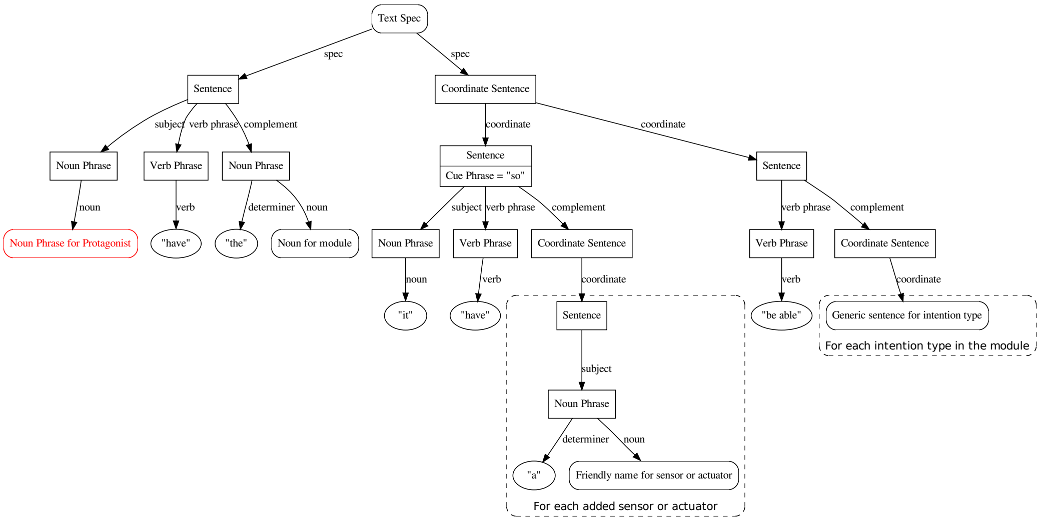


Figure F.10: Syntactic structure for the complex version of the "Capability" stasis statement.

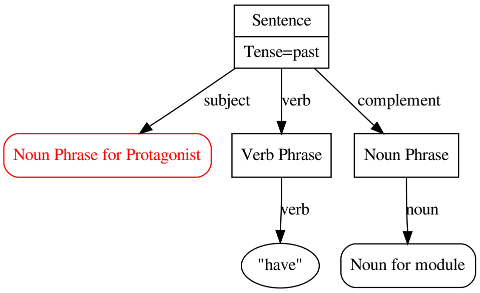


Figure F.11: Syntactic structure for the simplified version of the “Capability” stasis statement.

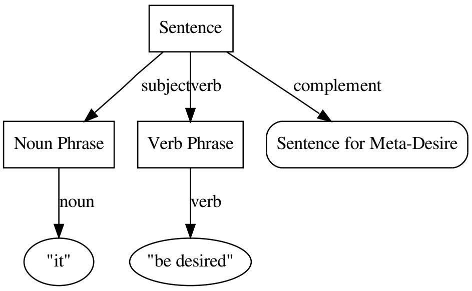


Figure F.12: Syntactic structure for “Meta-Desire Explanation” stasis statement.

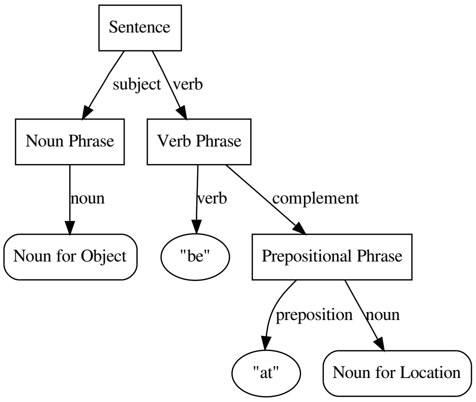


Figure F.13: Syntactic structure for “Object Location” stasis statement.

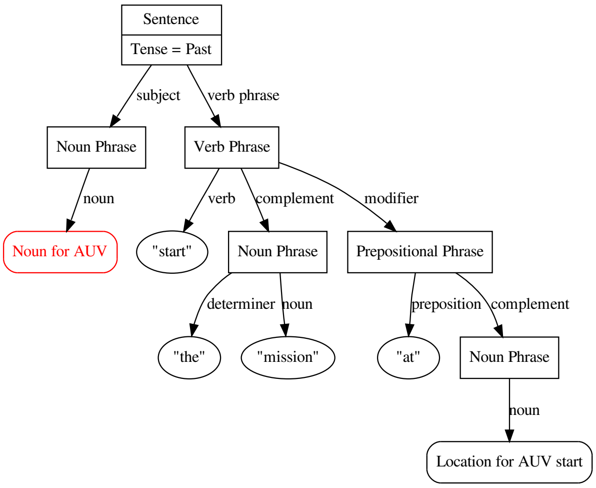
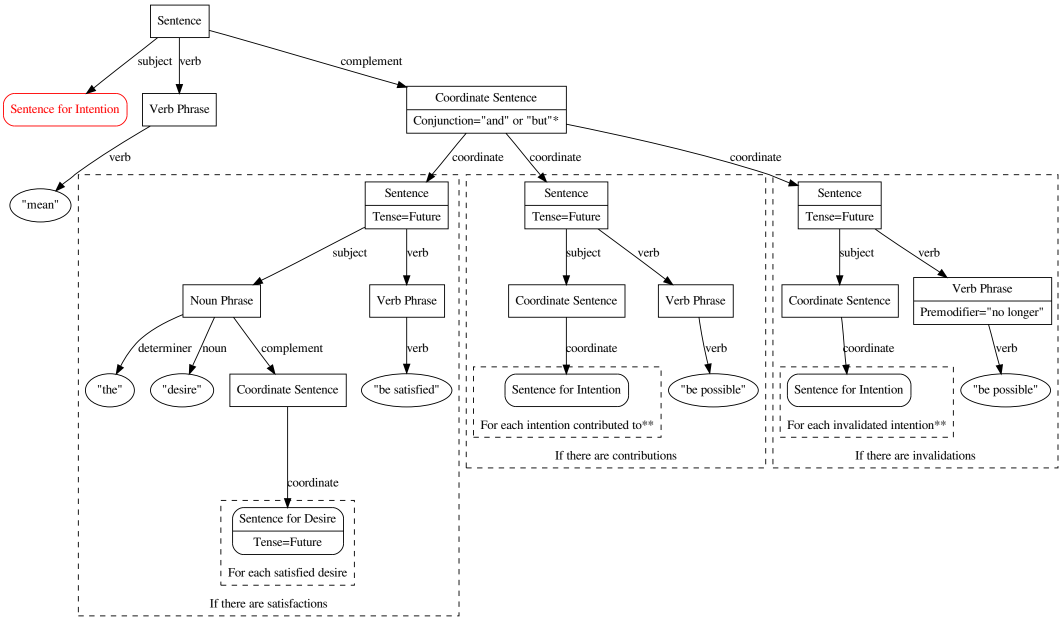


Figure F.14: Syntactic structure for "AUV Explanation" stasis statement.



*Conjunction is "but" if there are invalidations, and is "and" otherwise
**Which has already been mentioned in the dialog

Figure F.15: Syntactic structure for "Effect" stasis statement.

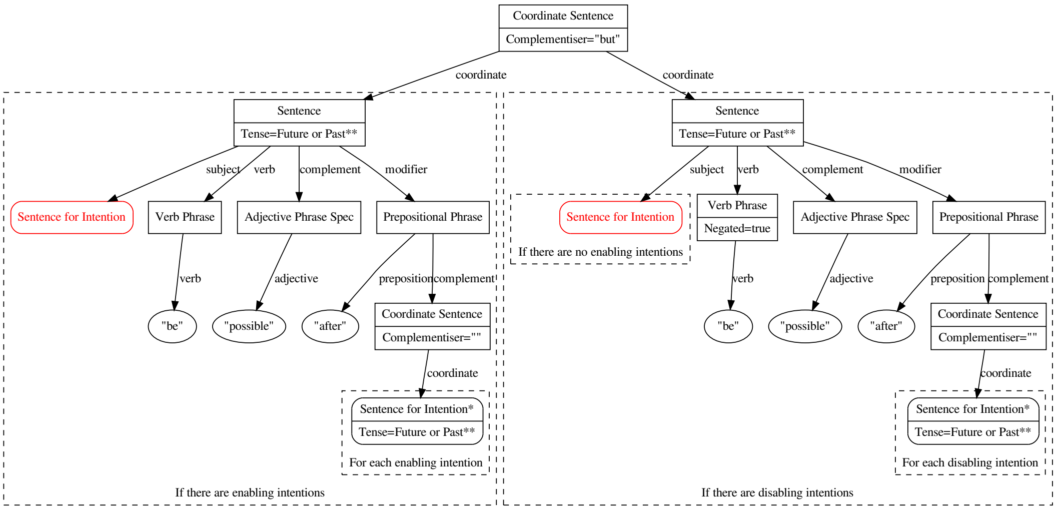


Figure F.16: Syntactic structure for “Conditions” stasis statement.

# Appendix G

# Example Narrative Script

00:00:00.00 VISUAL: BLANK

00:00:00.20 SPEECH: "It is desired that all installations should be examined."

00:00:00.20 VISUAL: graph D0 (Figure G.1)

00:00:00.70 VISUAL: graph D0 (Figure G.1)

00:00:01.70 VISUAL: graph D1 (Figure G.2)

00:00:03.66 SPEECH: "It is desired that if any installation is malfunctioning, then it should be repaired."

00:00:04.16 VISUAL: graph D1 (Figure G.2)

00:00:05.16 VISUAL: graph D2 (Figure G.3)

00:00:08.59 SPEECH: "It is desired that if any installation is broken, then it should be at the base location."

00:00:09.09 VISUAL: graph D2 (Figure G.3)

00:00:10.09 VISUAL: graph D3 (Figure G.4)

00:00:13.71 PAUSE

00:00:13.91 SPEECH: "Installation alpha is at location north 20 east 8 depth 15."

00:00:13.91 FOCUS: o1

00:00:18.16 UNFOCUS: o1

00:00:18.36 SPEECH: "The desire that it should be examined was introduced."

00:00:18.86 VISUAL: graph D3 (Figure G.4)

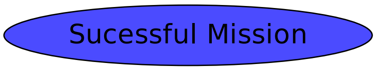


Figure G.1

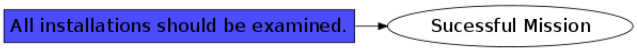


Figure G.2

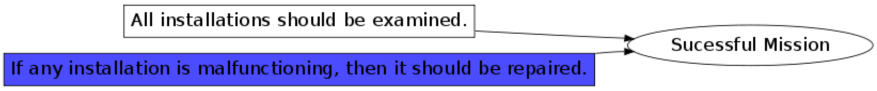

All installations should be examined.
If any installation is malfunctioning, then it should be repaired.
Sucessful Mission

Figure G.3

00:00:19.86 VISUAL: graph D4 (Figure G.5)

00:00:21.87 PAUSE

00:00:22.07 SPEECH: "A plan was created to satisfy this."

00:00:22.57 VISUAL: graph D4 (Figure G.5)

00:00:23.57 VISUAL: graph I0 (Figure G.6a)

00:00:25.10 PAUSE

00:00:28.30 SPEECH: "Installation bravo is at location north 20 east 20 depth 15."

00:00:28.30 FOCUS: o2

00:00:32.82 UNFOCUS: o2

00:00:33.02 SPEECH: "The desire that it should be examined was introduced."

00:00:33.52 VISUAL: graph I0 (Figure G.6a)

00:00:34.52 VISUAL: graph D5 (Figure G.5)

00:00:35.52 VISUAL: graph D5 (Figure G.5)

00:00:36.52 VISUAL: graph D6 (Figure G.7)

00:00:37.02 PAUSE

00:00:37.22 SPEECH: "A plan was created to satisfy this."

00:00:37.72 VISUAL: graph D6 (Figure G.7)

00:00:38.72 VISUAL: graph I1 (Figure G.6b)

00:00:40.25 PAUSE

00:00:43.45 SPEECH: "Installation charlie is at location north 8 east 20 depth 15."

00:00:43.45 FOCUS: o3

00:00:47.78 UNFOCUS: o3

00:00:47.97 SPEECH: "The desire that it should be examined was introduced."

00:00:48.47 VISUAL: graph I1 (Figure G.6b)

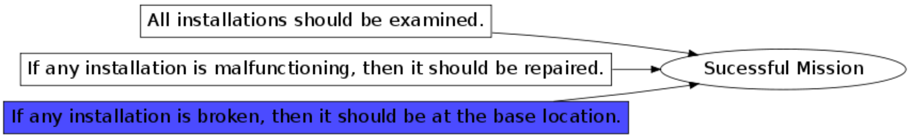

All installations should be examined.
If any installation is malfunctioning, then it should be repaired.
If any installation is broken, then it should be at the base location.
Sucessful Mission

Figure G.4

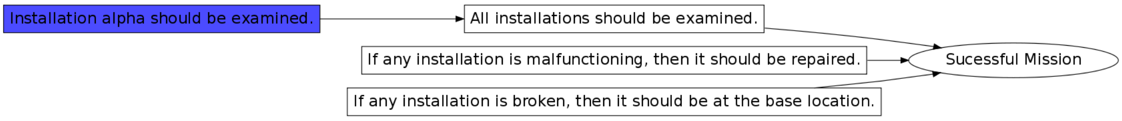

Installation alpha should be examined.
All installations should be examined.
If any installation is malfunctioning, then it should be repaired.
If any installation is broken, then it should be at the base location.
Sucessful Mission

Figure G.5

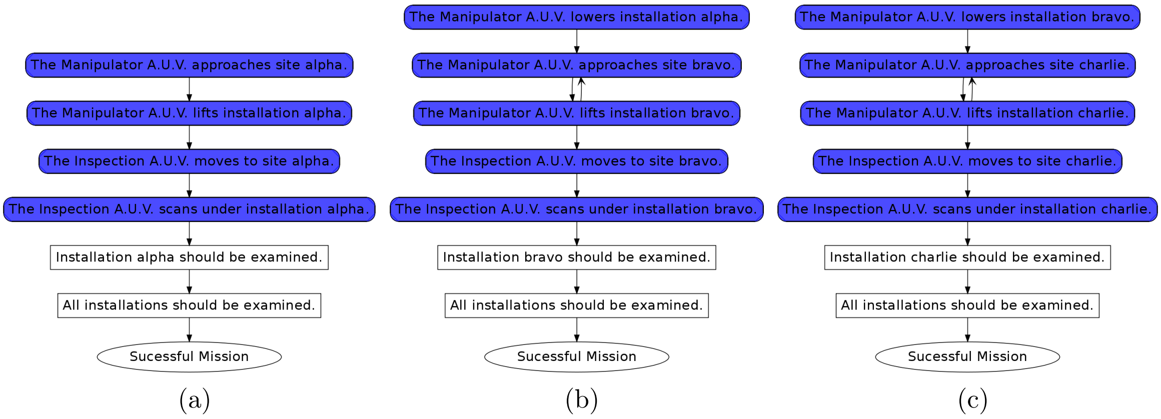

The Manipulator A.U.V. approaches site alpha.
The Manipulator A.U.V. lifts installation alpha.
The Inspection A.U.V. moves to site alpha.
The Inspection A.U.V. scans under installation alpha.
Installation alpha should be examined.
All installations should be examined.
Sucessful Mission
(a)
The Manipulator A.U.V. lowers installation alpha.
The Manipulator A.U.V. approaches site bravo.
The Manipulator A.U.V. lifts installation bravo.
The Inspection A.U.V. moves to site bravo.
The Inspection A.U.V. scans under installation bravo.
Installation bravo should be examined.
All installations should be examined.
Sucessful Mission
(b)
The Manipulator A.U.V. lowers installation bravo.
The Manipulator A.U.V. approaches site charlie.
The Manipulator A.U.V. lifts installation charlie.
The Inspection A.U.V. moves to site charlie.
The Inspection A.U.V. scans under installation charlie.
Installation charlie should be examined.
All installations should be examined.
Sucessful Mission
(c)

Figure G.6

00:00:49.47 VISUAL: graph D7 (Figure G.7)

00:00:50.47 VISUAL: graph D7 (Figure G.7)

00:00:51.47 VISUAL: graph D8 (Figure G.8)

00:00:51.97 PAUSE

00:00:52.17 SPEECH: "A plan was created to satisfy this."

00:00:52.67 VISUAL: graph D8 (Figure G.8)

00:00:53.67 VISUAL: graph I2 (Figure G.6c)

00:00:55.20 PAUSE

00:00:55.20 FOCUS: auv1

00:00:55.40 SPEECH: "In order that installation alpha should be examined, the Manipulator A.U.V. started to approach site alpha."

00:00:55.90 VISUAL: graph I2 (Figure G.6c)

00:00:56.90 VISUAL: graph I3 (Figure G.9a)

00:00:57.90 VISUAL: graph I3 (Figure G.9a)

00:00:58.90 VISUAL: graph I4 (Figure G.9b)

00:01:02.28 PAUSE

00:01:03.28 SPEECH: "Skipping forward by 4 minutes and 38 seconds"

00:01:07.92 SPEECH: "It finished doing this."

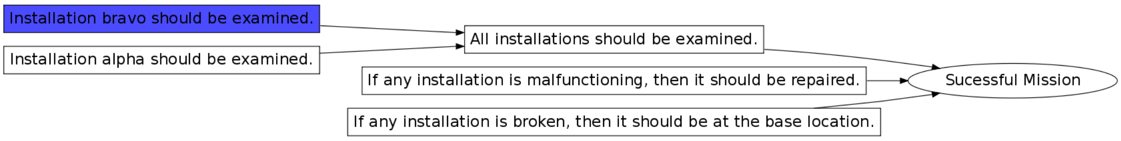

Installation bravo should be examined.
Installation alpha should be examined.
All installations should be examined.
If any installation is malfunctioning, then it should be repaired.
If any installation is broken, then it should be at the base location.
Sucessful Mission

Figure G.7

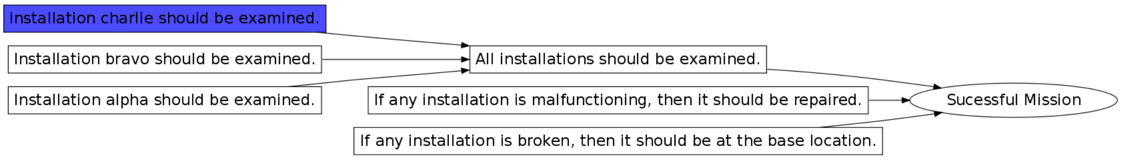

Installation charlie should be examined.
Installation bravo should be examined.
Installation alpha should be examined.
All installations should be examined.
If any installation is malfunctioning, then it should be repaired.
If any installation is broken, then it should be at the base location.
Sucessful Mission

Figure G.8

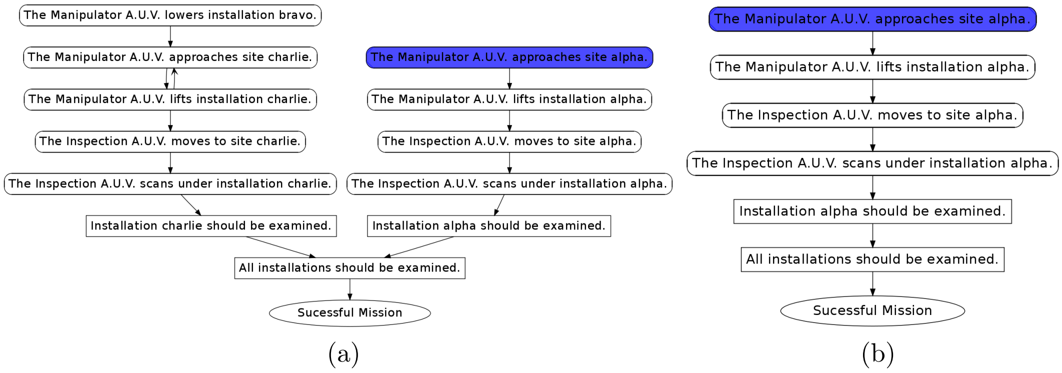

The Manipulator A.U.V. lowers installation bravo.
The Manipulator A.U.V. approaches site charlie.
The Manipulator A.U.V. lifts installation charlie.
The Inspection A.U.V. moves to site charlie.
The Inspection A.U.V. scans under installation charlie.
Installation charlie should be examined.
The Manipulator A.U.V. approaches site alpha.
The Manipulator A.U.V. lifts installation alpha.
The Inspection A.U.V. moves to site alpha.
The Inspection A.U.V. scans under installation alpha.
Installation alpha should be examined.
All installations should be examined.
Sucessful Mission
(a)
The Manipulator A.U.V. approaches site alpha.
The Manipulator A.U.V. lifts installation alpha.
The Inspection A.U.V. moves to site alpha.
The Inspection A.U.V. scans under installation alpha.
Installation alpha should be examined.
All installations should be examined.
Sucessful Mission
(b)

Figure G.9

00:01:08.42 VISUAL: graph I4 (Figure G.9b)

00:01:09.42 VISUAL: graph I5 (Figure G.10a)

00:01:09.92 PAUSE

00:01:10.12 SPEECH: "It has the lift objects module and so it has a lifting arm and is able to grab objects, to lift objects, to lower objects and to release objects at the new location."

00:01:20.38 SPEECH: "It started to lift installation alpha and put its lifting arm in active mode."

00:01:20.88 VISUAL: graph I5 (Figure G.10a)

00:01:21.88 VISUAL: graph I6 (Figure G.10b)

00:01:24.90 SPEECH: "In active mode the lifting arm will grab the nearest object."

00:01:28.42 SPEECH: "This was possible after the Manipulator A.U.V. approached site alpha."

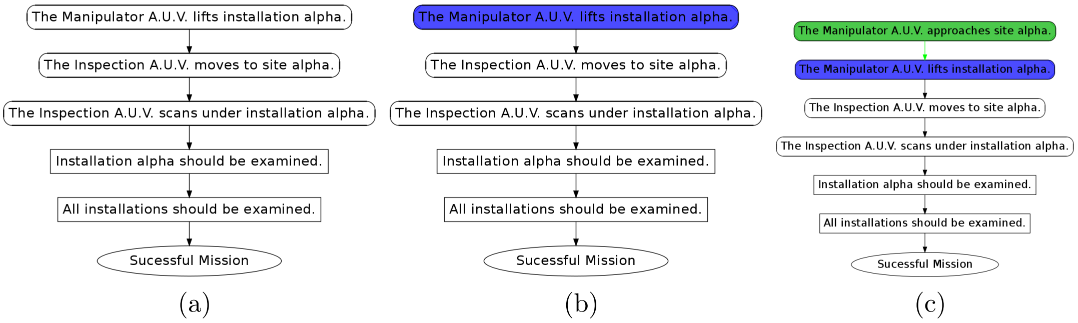
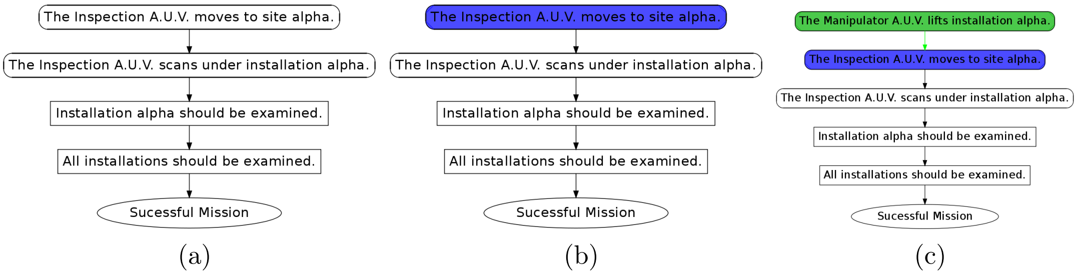

The Manipulator A.U.V. lifts installation alpha.
The Inspection A.U.V. moves to site alpha.
The Inspection A.U.V. scans under installation alpha.
Installation alpha should be examined.
All installations should be examined.
Sucessful Mission
(a)
The Manipulator A.U.V. lifts installation alpha.
The Inspection A.U.V. moves to site alpha.
The Inspection A.U.V. scans under installation alpha.
Installation alpha should be examined.
All installations should be examined.
Sucessful Mission
(b)
The Manipulator A.U.V. approaches site alpha.
The Manipulator A.U.V. lifts installation alpha.
The Inspection A.U.V. moves to site alpha.
The Inspection A.U.V. scans under installation alpha.
Installation alpha should be examined.
All installations should be examined.
Sucessful Mission
(c)

Figure G.10


The Inspection A.U.V. moves to site alpha.
The Inspection A.U.V. scans under installation alpha.
Installation alpha should be examined.
All installations should be examined.
Sucessful Mission
The Inspection A.U.V. moves to site alpha.
The Inspection A.U.V. scans under installation alpha.
Installation alpha should be examined.
All installations should be examined.
Sucessful Mission
The Manipulator A.U.V. lifts installation alpha.
The Inspection A.U.V. moves to site alpha.
The Inspection A.U.V. scans under installation alpha.
Installation alpha should be examined.
All installations should be examined.
Sucessful Mission

(a) (b) (c)

Figure G.11

00:01:28.42 VISUAL: graph I6 (Figure G.10b)

00:01:29.42 VISUAL: graph I7 (Figure G.10c)

00:01:31.84 VISUAL: graph I7 (Figure G.10c)

00:01:32.84 VISUAL: graph I6 (Figure G.10b)

00:01:33.34 PAUSE

00:01:34.34 SPEECH: "Skipping forward by 1 minute and 5 seconds"

00:01:38.44 SPEECH: "It finished lifting installation alpha."

00:01:38.94 VISUAL: graph I6 (Figure G.10b)

00:01:39.94 VISUAL: graph I8 (Figure G.11a)

00:01:41.44 UNFOCUS: auv1

00:01:41.44 PAUSE

00:01:41.44 FOCUS: auv2

00:01:41.64 SPEECH: "The Inspection A.U.V. started to move to site alpha."

00:01:42.14 VISUAL: graph I8 (Figure G.11a)

00:01:43.14 VISUAL: graph I9 (Figure G.11b)

00:01:45.36 SPEECH: "This was possible after the Manipulator A.U.V. lifted installation alpha."

00:01:45.36 VISUAL: graph I9 (Figure G.11b)

00:01:46.36 VISUAL: graph I10 (Figure G.11c)

00:01:49.03 VISUAL: graph I10 (Figure G.11c)

00:01:50.03 VISUAL: graph I9 (Figure G.11b)

00:01:50.53 PAUSE

00:01:51.53 SPEECH: "Skipping forward by 1 minute and 11 seconds"

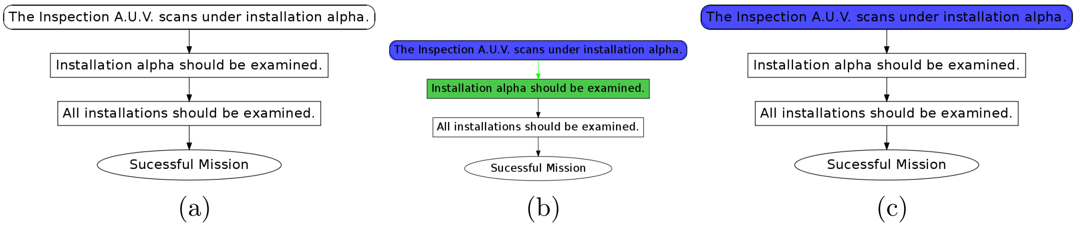
00:01:55.80 SPEECH: "It finished moving to site alpha."

00:01:56.30 VISUAL: graph I9 (Figure G.11b)

00:01:57.30 VISUAL: graph I11 (Figure G.12a)

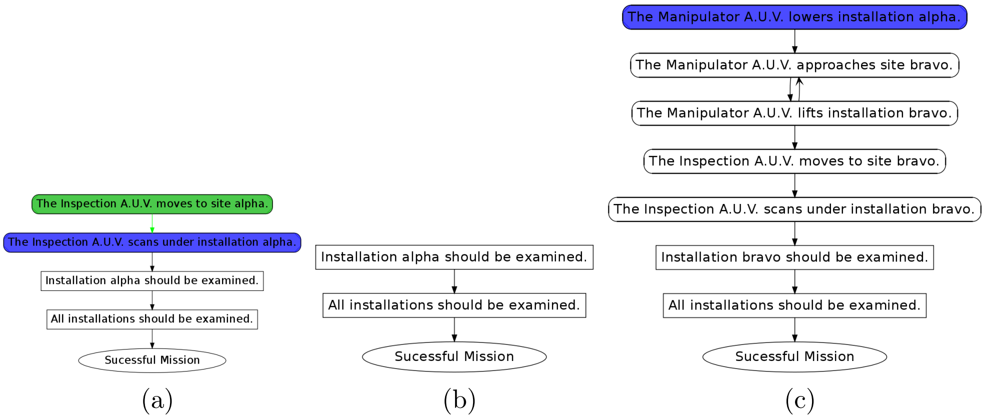
00:01:58.40 PAUSE

00:01:58.60 SPEECH: "It has the examine installations module and so it has an installation scanner and is able to scan under installations."


The Inspection A.U.V. scans under installation alpha.
Installation alpha should be examined.
All installations should be examined.
Sucessful Mission
The Inspection A.U.V. scans under installation alpha.
Installation alpha should be examined.
All installations should be examined.
Sucessful Mission
The Inspection A.U.V. scans under installation alpha.
Installation alpha should be examined.
All installations should be examined.
Sucessful Mission

(a) (b) (c)

Figure G.12


The Inspection A.U.V. moves to site alpha.
The Inspection A.U.V. scans under installation alpha.
Installation alpha should be examined.
All installations should be examined.
Sucessful Mission
Installation alpha should be examined.
All installations should be examined.
Sucessful Mission
The Manipulator A.U.V. lowers installation alpha.
The Manipulator A.U.V. approaches site bravo.
The Manipulator A.U.V. lifts installation bravo.
The Inspection A.U.V. moves to site bravo.
The Inspection A.U.V. scans under installation bravo.
Installation bravo should be examined.
All installations should be examined.
Sucessful Mission

(a) (b) (c)

Figure G.13

00:02:05.47 SPEECH: "The Inspection A.U.V.'s scanning under installation alpha means that the desire that installation alpha should be examined will be satisfied."

00:02:05.47 VISUAL: graph I11 (Figure G.12a)

00:02:06.47 VISUAL: graph I12 (Figure G.12b)

00:02:11.14 VISUAL: graph I12 (Figure G.12b)

00:02:12.14 VISUAL: graph I11 (Figure G.12a)

00:02:13.11 SPEECH: "It started to do this and put its installation scanner in active mode."

00:02:13.61 VISUAL: graph I11 (Figure G.12a)

00:02:14.61 VISUAL: graph I13 (Figure G.12c)

00:02:17.27 SPEECH: "In active mode the installation scanner will detect the status of an installation."

00:02:22.03 SPEECH: "This was possible after the Inspection A.U.V. moved to site alpha."

00:02:22.03 VISUAL: graph I13 (Figure G.12c)

00:02:23.03 VISUAL: graph I14 (Figure G.13a)

00:02:25.29 VISUAL: graph I14 (Figure G.13a)

00:02:26.29 VISUAL: graph I13 (Figure G.12c)

00:02:26.29 VISUAL: graph I13 (Figure G.12c)

00:02:26.79 VISUAL: BLANK

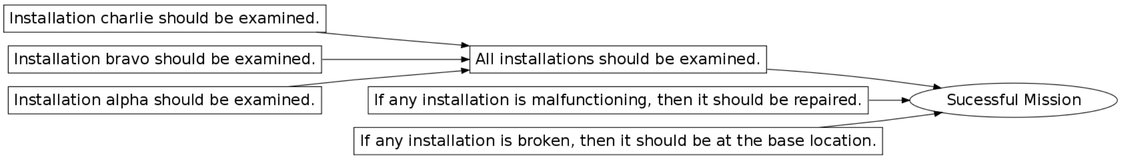

Installation charlie should be examined.
Installation bravo should be examined.
Installation alpha should be examined.
All installations should be examined.
If any installation is malfunctioning, then it should be repaired.
If any installation is broken, then it should be at the base location.
Sucessful Mission

Figure G.14

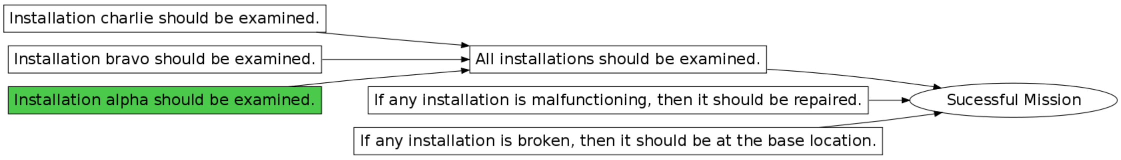

Installation charlie should be examined.
Installation bravo should be examined.
Installation alpha should be examined.
All installations should be examined.
If any installation is malfunctioning, then it should be repaired.
If any installation is broken, then it should be at the base location.
Sucessful Mission

Figure G.15

00:02:26.79 PAUSE

00:02:26.99 SPEECH: "Its installation scanner detected that installation alpha is working correctly."

00:02:31.98 PAUSE

00:02:32.98 SPEECH: "Skipping forward by 13 seconds"

00:02:36.61 VISUAL: BLANK

00:02:36.81 VISUAL: graph I13 (Figure G.12c)

00:02:36.81 SPEECH: "It finished scanning under installation alpha and put its installation scanner in passive mode."

00:02:37.31 VISUAL: graph I13 (Figure G.12c)

00:02:38.31 VISUAL: graph I15 (Figure G.13b)

00:02:42.41 SPEECH: "In passive mode the installation scanner will detect nothing."

00:02:46.27 UNFOCUS: auv2

00:02:46.27 PAUSE

00:02:46.47 SPEECH: "The desire that installation alpha should be examined was satisfied."

00:02:46.97 VISUAL: graph I15 (Figure G.13b)

00:02:47.97 VISUAL: graph D9 (Figure G.14)

00:02:48.97 VISUAL: graph D9 (Figure G.14)

00:02:49.97 VISUAL: graph D10 (Figure G.15)

00:02:51.01 PAUSE

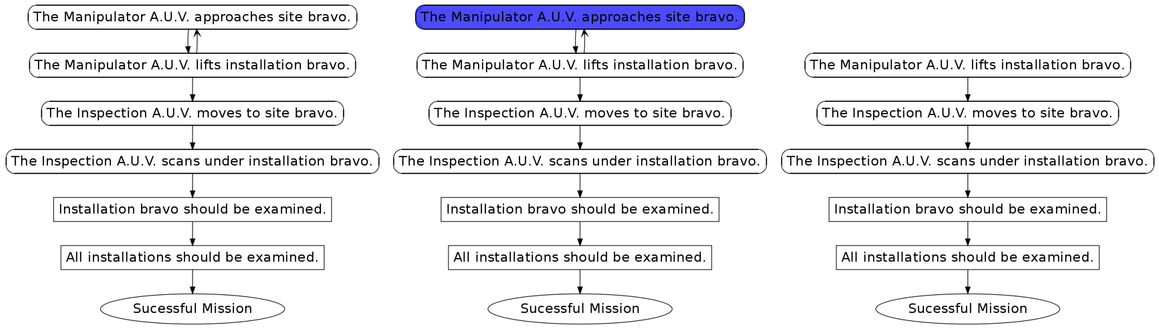
00:02:51.01 FOCUS: auv1

00:02:51.21 SPEECH: "In order that installation bravo should be examined, the Manipulator A.U.V. started to lower installation alpha."

00:02:51.71 VISUAL: graph D10 (Figure G.15)

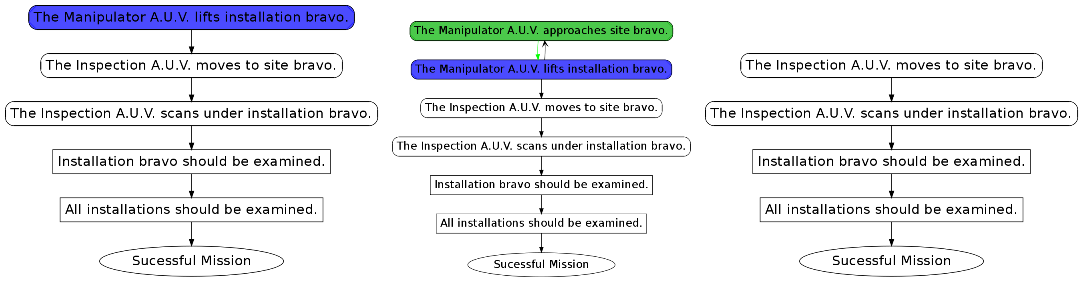
00:02:52.71 VISUAL: graph I16 (Figure G.13c)

00:02:58.44 PAUSE


The Manipulator A.U.V. approaches site bravo.
The Manipulator A.U.V. lifts installation bravo.
The Inspection A.U.V. moves to site bravo.
The Inspection A.U.V. scans under installation bravo.
Installation bravo should be examined.
All installations should be examined.
Sucessful Mission
The Manipulator A.U.V. approaches site bravo.
The Manipulator A.U.V. lifts installation bravo.
The Inspection A.U.V. moves to site bravo.
The Inspection A.U.V. scans under installation bravo.
Installation bravo should be examined.
All installations should be examined.
Sucessful Mission
The Manipulator A.U.V. lifts installation bravo.
The Inspection A.U.V. moves to site bravo.
The Inspection A.U.V. scans under installation bravo.
Installation bravo should be examined.
All installations should be examined.
Sucessful Mission

(a) (b) (c)

Figure G.16


The Manipulator A.U.V. lifts installation bravo.
The Inspection A.U.V. moves to site bravo.
The Inspection A.U.V. scans under installation bravo.
Installation bravo should be examined.
All installations should be examined.
Sucessful Mission
The Manipulator A.U.V. approaches site bravo.
The Manipulator A.U.V. lifts installation bravo.
The Inspection A.U.V. moves to site bravo.
The Inspection A.U.V. scans under installation bravo.
Installation bravo should be examined.
All installations should be examined.
Sucessful Mission
The Inspection A.U.V. moves to site bravo.
The Inspection A.U.V. scans under installation bravo.
Installation bravo should be examined.
All installations should be examined.
Sucessful Mission

(a) (b) (c)

Figure G.17

00:02:59.44 SPEECH: "Skipping forward by 2 minutes and 35 seconds"

00:03:04.16 SPEECH: "It finished doing this and put its lifting arm in passive mode."

00:03:04.66 VISUAL: graph I16 (Figure G.13c)

00:03:05.66 VISUAL: graph I17 (Figure 8.9)

00:03:07.84 SPEECH: "In passive mode the lifting arm will release the grabbed object."

00:03:11.64 PAUSE

00:03:11.84 SPEECH: "The Manipulator A.U.V. started to approach site bravo."

00:03:12.34 VISUAL: graph I17 (Figure 8.9)

00:03:13.34 VISUAL: graph I18 (Figure G.16b)

00:03:16.11 PAUSE

00:03:17.11 SPEECH: "Skipping forward by 1 minute"

00:03:20.28 SPEECH: "It finished doing this."

00:03:20.78 VISUAL: graph I18 (Figure G.16b)

00:03:21.78 VISUAL: graph I19 (Figure G.16c)

00:03:22.28 PAUSE

00:03:22.48 SPEECH: "It started to lift installation bravo and put its lifting arm in active mode."

00:03:22.98 VISUAL: graph I19 (Figure G.16c)

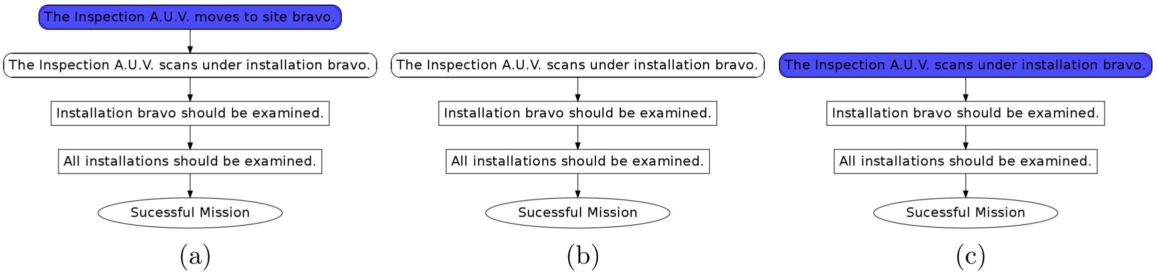

The Inspection A.U.V. moves to site bravo.
The Inspection A.U.V. scans under installation bravo.
Installation bravo should be examined.
All installations should be examined.
Sucessful Mission
(a)
The Inspection A.U.V. scans under installation bravo.
Installation bravo should be examined.
All installations should be examined.
Sucessful Mission
(b)
The Inspection A.U.V. scans under installation bravo.
Installation bravo should be examined.
All installations should be examined.
Sucessful Mission
(c)

Figure G.18

00:03:23.98 VISUAL: graph I20 (Figure G.17a)

00:03:27.08 SPEECH: "This was possible after the Manipulator A.U.V. approached site bravo."

00:03:27.08 VISUAL: graph I20 (Figure G.17a)

00:03:28.08 VISUAL: graph I21 (Figure G.17b)

00:03:30.59 VISUAL: graph I21 (Figure G.17b)

00:03:31.59 VISUAL: graph I20 (Figure G.17a)

00:03:32.10 PAUSE

00:03:33.10 SPEECH: "Skipping forward by 1 minute and 9 seconds"

00:03:37.21 SPEECH: "It finished lifting installation bravo."

00:03:37.71 VISUAL: graph I20 (Figure G.17a)

00:03:38.71 VISUAL: graph I22 (Figure G.17c)

00:03:40.30 UNFOCUS: auv1

00:03:40.30 PAUSE

00:03:40.30 FOCUS: auv2

00:03:40.50 SPEECH: "The Inspection A.U.V. started to move to site bravo."

00:03:41.00 VISUAL: graph I22 (Figure G.17c)

00:03:42.00 VISUAL: graph I23 (Figure G.18a)

00:03:44.62 PAUSE

00:03:45.62 SPEECH: "Skipping forward by 48 seconds"

00:03:49.56 SPEECH: "It finished doing this."

00:03:50.06 VISUAL: graph I23 (Figure G.18a)

00:03:51.06 VISUAL: graph I24 (Figure G.18b)

00:03:51.56 PAUSE

00:03:51.76 SPEECH: "It started to scan under installation bravo and put its installation scanner in active mode."

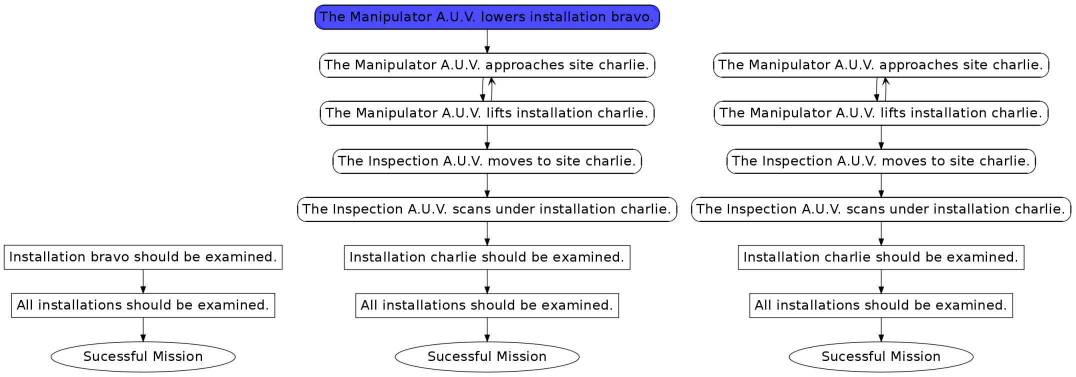
00:03:52.26 VISUAL: graph I24 (Figure G.18b)

00:03:53.26 VISUAL: graph I25 (Figure G.18c)

00:03:57.60 PAUSE

00:03:58.60 SPEECH: "Skipping forward by a quarter of a minute"

00:04:02.30 SPEECH: "It finished doing this and put its installation scanner in passive mode."


The Manipulator A.U.V. lowers installation bravo.
The Manipulator A.U.V. approaches site charlie.
The Manipulator A.U.V. lifts installation charlie.
The Inspection A.U.V. moves to site charlie.
The Inspection A.U.V. scans under installation charlie.
Installation bravo should be examined.
Installation charlie should be examined.
All installations should be examined.
Sucessful Mission

(a) (b) (c)

Figure G.19

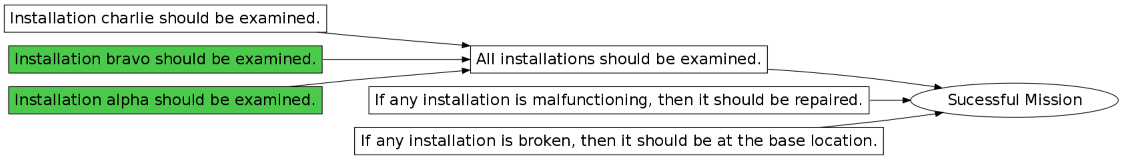

Installation charlie should be examined.
Installation bravo should be examined.
Installation alpha should be examined.
All installations should be examined.
If any installation is malfunctioning, then it should be repaired.
If any installation is broken, then it should be at the base location.
Sucessful Mission

Figure G.20

00:04:02.80 VISUAL: graph I25 (Figure G.18c)

00:04:03.80 VISUAL: graph I26 (Figure G.19a)

00:04:06.85 UNFOCUS: auv2

00:04:06.85 PAUSE

00:04:07.05 SPEECH: "The desire that installation bravo should be examined was satisfied."

00:04:07.55 VISUAL: graph I26 (Figure G.19a)

00:04:08.55 VISUAL: graph D11 (Figure G.15)

00:04:09.55 VISUAL: graph D11 (Figure G.15)

00:04:10.55 VISUAL: graph D12 (Figure G.20)

00:04:11.64 PAUSE

00:04:11.64 FOCUS: auv1

00:04:11.84 SPEECH: "In order that installation charlie should be examined, the Manipulator A.U.V. started to lower installation bravo."

00:04:12.34 VISUAL: graph D12 (Figure G.20)

00:04:13.34 VISUAL: graph I27 (Figure G.19b)

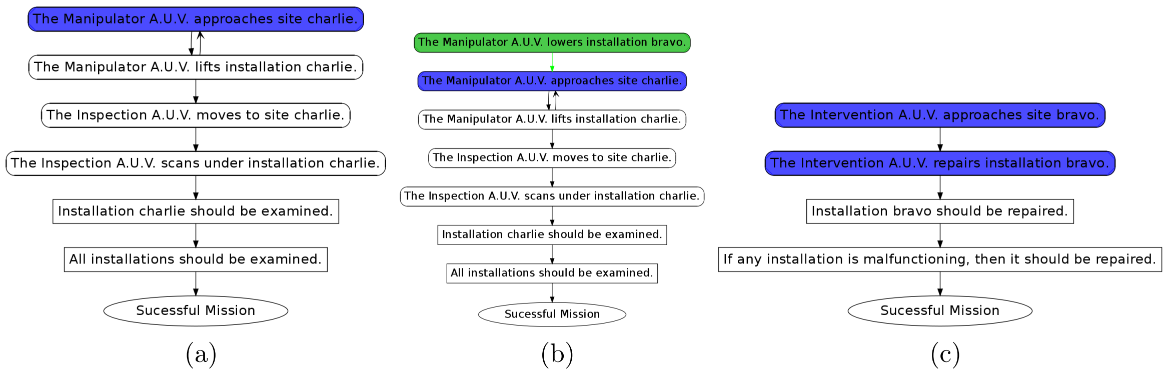
00:04:19.19 PAUSE

00:04:20.19 SPEECH: "Skipping forward by 2 minutes and 37 seconds"

00:04:24.96 SPEECH: "It finished doing this and put its lifting arm in passive mode."

00:04:25.46 VISUAL: graph I27 (Figure G.19b)

00:04:26.46 VISUAL: graph I28 (Figure G.19c)


The Manipulator A.U.V. approaches site charlie.
The Manipulator A.U.V. lifts installation charlie.
The Inspection A.U.V. moves to site charlie.
The Inspection A.U.V. scans under installation charlie.
Installation charlie should be examined.
All installations should be examined.
Sucessful Mission
The Manipulator A.U.V. lowers installation bravo.
The Manipulator A.U.V. approaches site charlie.
The Manipulator A.U.V. lifts installation charlie.
The Inspection A.U.V. moves to site charlie.
The Inspection A.U.V. scans under installation charlie.
Installation charlie should be examined.
All installations should be examined.
Sucessful Mission
The Intervention A.U.V. approaches site bravo.
The Intervention A.U.V. repairs installation bravo.
Installation bravo should be repaired.
If any installation is malfunctioning, then it should be repaired.
Sucessful Mission

(a) (b) (c)

Figure G.21

00:04:28.93 PAUSE

00:04:29.14 SPEECH: "It started to approach site charlie."

00:04:29.64 VISUAL: graph I28 (Figure G.19c)

00:04:30.64 VISUAL: graph I29 (Figure G.21a)

00:04:31.68 SPEECH: "This was possible after the Manipulator A.U.V. lowered installation bravo."

00:04:31.68 VISUAL: graph I29 (Figure G.21a)

00:04:32.68 VISUAL: graph I30 (Figure G.21b)

00:04:35.34 VISUAL: graph I30 (Figure G.21b)

00:04:36.34 VISUAL: graph I29 (Figure G.21a)

00:04:36.34 VISUAL: graph I29 (Figure G.21a)

00:04:36.84 VISUAL: BLANK

00:04:36.84 UNFOCUS: auv1

00:04:36.84 PAUSE

00:04:37.84 SPEECH: "Skipping backward by 3 minutes and 12 seconds"

00:04:41.98 FOCUS: auv2

00:04:42.18 SPEECH: "The Inspection A.U.V.'s installation scanner detected that installation bravo is malfunctioning."

00:04:48.26 UNFOCUS: auv2

00:04:48.26 PAUSE

00:04:48.26 VISUAL: BLANK

00:04:48.46 VISUAL: graph I29 (Figure G.21a)

00:04:48.46 SPEECH: "Installation bravo is at location north 20 east 20 depth 15."

00:04:48.46 FOCUS: o2

00:04:52.98 UNFOCUS: o2

00:04:53.18 SPEECH: "The desire that it should be repaired was introduced."

00:04:53.68 VISUAL: graph I29 (Figure G.21a)

00:04:54.68 VISUAL: graph D13 (Figure G.20)

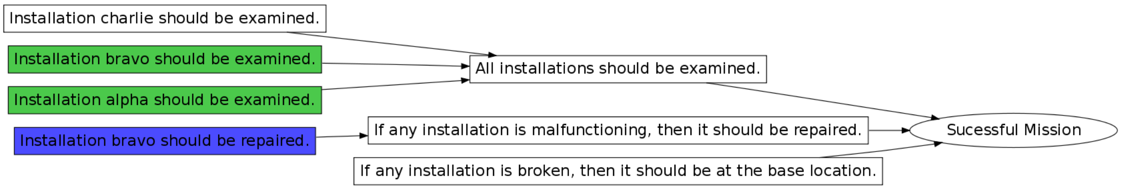
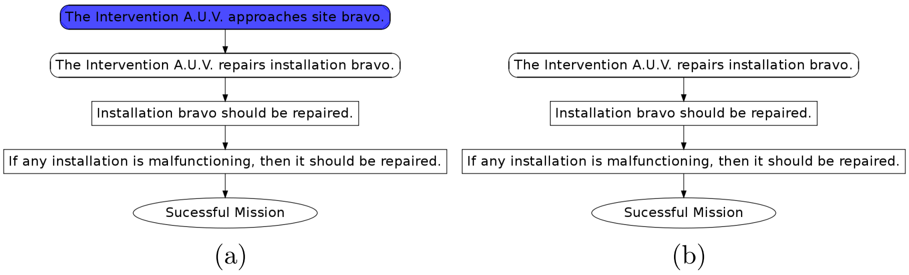

Installation charlie should be examined.
Installation bravo should be examined.
Installation alpha should be examined.
Installation bravo should be repaired.
All installations should be examined.
If any installation is malfunctioning, then it should be repaired.
If any installation is broken, then it should be at the base location.
Sucessful Mission

Figure G.22


The Intervention A.U.V. approaches site bravo.
The Intervention A.U.V. repairs installation bravo.
Installation bravo should be repaired.
If any installation is malfunctioning, then it should be repaired.
Sucessful Mission
The Intervention A.U.V. repairs installation bravo.
Installation bravo should be repaired.
If any installation is malfunctioning, then it should be repaired.
Sucessful Mission

(a)

(b)

Figure G.23

00:04:55.68 VISUAL: graph D13 (Figure G.20)

00:04:56.68 VISUAL: graph D14 (Figure G.22)

00:04:57.18 PAUSE

00:04:57.38 SPEECH: "A plan was created to satisfy this."

00:04:57.88 VISUAL: graph D14 (Figure G.22)

00:04:58.88 VISUAL: graph I31 (Figure G.21c)

00:05:00.41 PAUSE

00:05:00.41 FOCUS: auv3

00:05:00.61 SPEECH: "In order that installation bravo should be repaired, the Intervention A.U.V. started to approach site bravo."

00:05:01.11 VISUAL: graph I31 (Figure G.21c)

00:05:02.11 VISUAL: graph I32 (Figure G.23a)

00:05:07.52 PAUSE

00:05:08.52 SPEECH: "Skipping forward by 4 minutes and 48 seconds"

00:05:13.14 SPEECH: "It finished doing this."

00:05:13.64 VISUAL: graph I32 (Figure G.23a)

00:05:14.64 VISUAL: graph I33 (Figure G.23b)

00:05:15.14 PAUSE

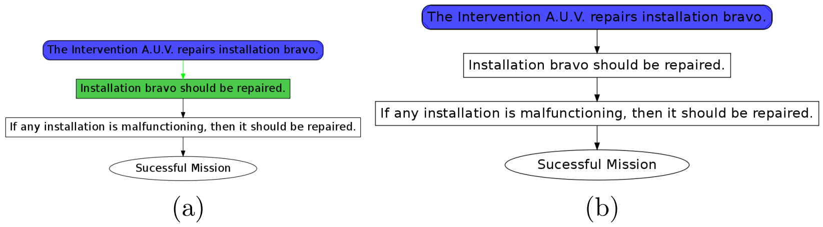
00:05:15.34 SPEECH: "It has the repair installations module and so it has an intervention arm and is able to repair installations."

00:05:21.45 SPEECH: "The Intervention A.U.V.'s repairing installation bravo means that the desire that installation bravo should be repaired will be satisfied."

00:05:21.45 VISUAL: graph I33 (Figure G.23b)

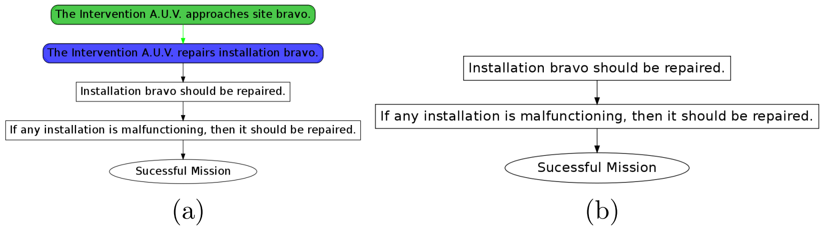
00:05:22.45 VISUAL: graph I34 (Figure G.24a)

00:05:26.35 VISUAL: graph I34 (Figure G.24a)


The Intervention A.U.V. repairs installation bravo.
Installation bravo should be repaired.
If any installation is malfunctioning, then it should be repaired.
Sucessful Mission
The Intervention A.U.V. repairs installation bravo.
Installation bravo should be repaired.
If any installation is malfunctioning, then it should be repaired.
Sucessful Mission

(a) (b)

Figure G.24


The Intervention A.U.V. approaches site bravo.
The Intervention A.U.V. repairs installation bravo.
Installation bravo should be repaired.
If any installation is malfunctioning, then it should be repaired.
Sucessful Mission
Installation bravo should be repaired.
If any installation is malfunctioning, then it should be repaired.
Sucessful Mission

(a) (b)

Figure G.25

00:05:27.35 VISUAL: graph I33 (Figure G.23b)

00:05:28.88 SPEECH: "It started to do this and put its intervention arm in forward mode."

00:05:29.38 VISUAL: graph I33 (Figure G.23b)

00:05:30.38 VISUAL: graph I35 (Figure G.24b)

00:05:32.72 SPEECH: "In forward mode the intervention arm will repair the installation in front of the A.U.V."

00:05:37.45 SPEECH: "This was possible after the Intervention A.U.V. approached site bravo."

00:05:37.45 VISUAL: graph I35 (Figure G.24b)

00:05:38.45 VISUAL: graph I36 (Figure G.25a)

00:05:40.86 VISUAL: graph I36 (Figure G.25a)

00:05:41.86 VISUAL: graph I35 (Figure G.24b)

00:05:42.36 PAUSE

00:05:42.56 SPEECH: "It finished repairing installation bravo and put its intervention arm in passive mode."

00:05:43.06 VISUAL: graph I35 (Figure G.24b)

00:05:44.06 VISUAL: graph I37 (Figure G.25b)

00:05:47.53 SPEECH: "In passive mode the intervention arm is deactivated."

00:05:51.30 UNFOCUS: auv3

00:05:51.30 PAUSE

00:05:51.50 SPEECH: "The desire that installation bravo should be repaired was satisfied."

00:05:52.00 VISUAL: graph I37 (Figure G.25b)

00:05:53.00 VISUAL: graph D15 (Figure G.26)

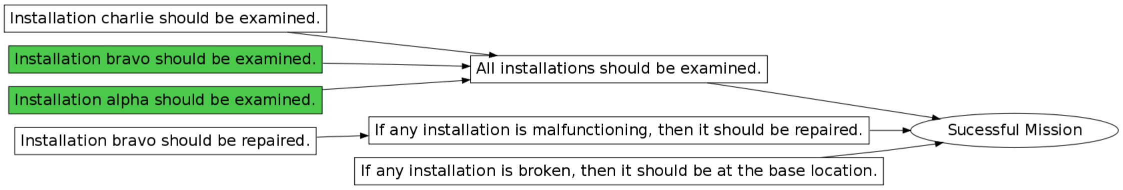

Installation charlie should be examined.
Installation bravo should be examined.
Installation alpha should be examined.
Installation bravo should be repaired.
All installations should be examined.
If any installation is malfunctioning, then it should be repaired.
If any installation is broken, then it should be at the base location.
Sucessful Mission

Figure G.26

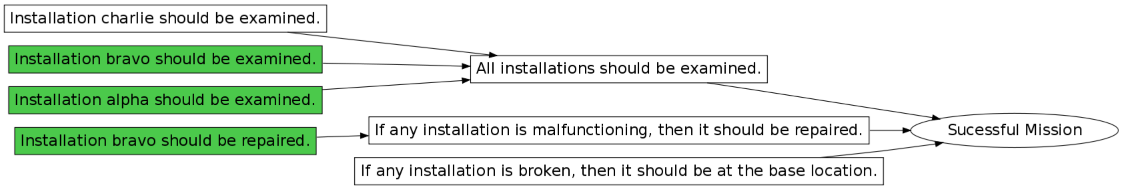

Installation charlie should be examined.
Installation bravo should be examined.
Installation alpha should be examined.
Installation bravo should be repaired.
All installations should be examined.
If any installation is malfunctioning, then it should be repaired.
If any installation is broken, then it should be at the base location.
Sucessful Mission

Figure G.27

00:05:54.00 VISUAL: graph D15 (Figure G.26)

00:05:55.00 VISUAL: graph D16 (Figure G.27)

00:05:55.89 PAUSE

00:05:56.09 SPEECH: "The desire that if any installation is malfunctioning, then it should be repaired was satisfied."

00:05:56.59 VISUAL: graph D16 (Figure G.27)

00:05:57.59 VISUAL: graph D17 (Figure G.28)

00:06:01.97 PAUSE

00:06:02.97 SPEECH: "Skipping backward by 1 minute"

00:06:05.93 FOCUS: auv1

00:06:06.13 SPEECH: "In order that installation charlie should be examined, the Manipulator A.U.V. finished approaching site charlie."

00:06:06.63 VISUAL: graph D17 (Figure G.28)

00:06:07.63 VISUAL: graph I38 (Figure G.29a)

00:06:08.63 VISUAL: graph I38 (Figure G.29a)

00:06:09.63 VISUAL: graph I39 (Figure G.29b)

00:06:13.30 PAUSE

00:06:13.50 SPEECH: "It started to lift installation charlie and put its lifting arm in active mode."

00:06:14.00 VISUAL: graph I39 (Figure G.29b)

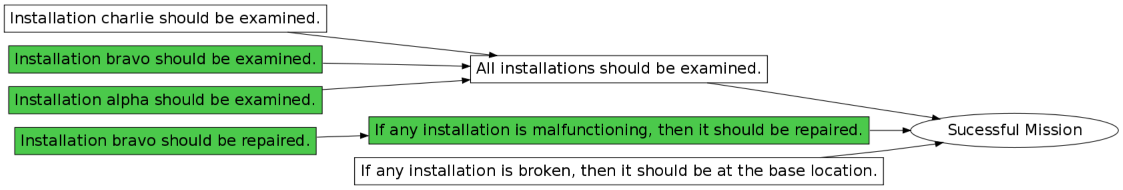

Installation charlie should be examined.
Installation bravo should be examined.
Installation alpha should be examined.
Installation bravo should be repaired.
All installations should be examined.
If any installation is malfunctioning, then it should be repaired.
If any installation is broken, then it should be at the base location.
Sucessful Mission

Figure G.28

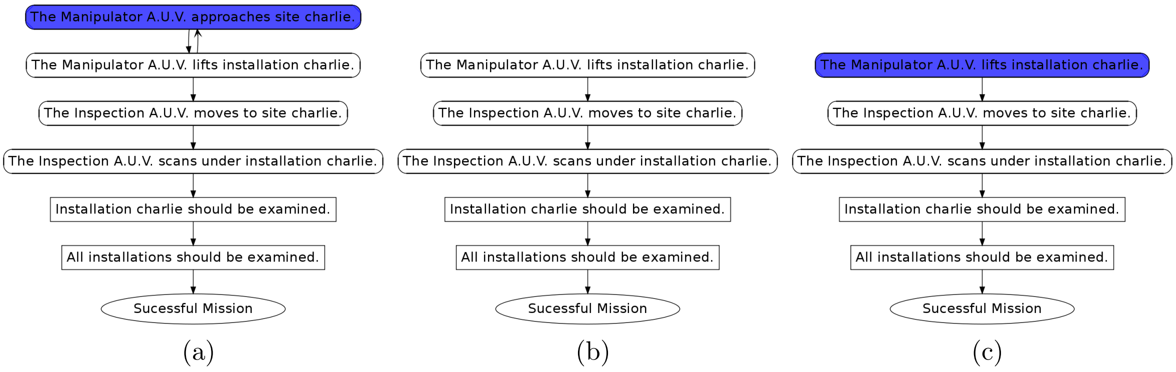

The Manipulator A.U.V. approaches site charlie.
The Manipulator A.U.V. lifts installation charlie.
The Inspection A.U.V. moves to site charlie.
The Inspection A.U.V. scans under installation charlie.
Installation charlie should be examined.
All installations should be examined.
Sucessful Mission
(a)
The Manipulator A.U.V. lifts installation charlie.
The Inspection A.U.V. moves to site charlie.
The Inspection A.U.V. scans under installation charlie.
Installation charlie should be examined.
All installations should be examined.
Sucessful Mission
(b)
The Manipulator A.U.V. lifts installation charlie.
The Inspection A.U.V. moves to site charlie.
The Inspection A.U.V. scans under installation charlie.
Installation charlie should be examined.
All installations should be examined.
Sucessful Mission
(c)

Figure G.29

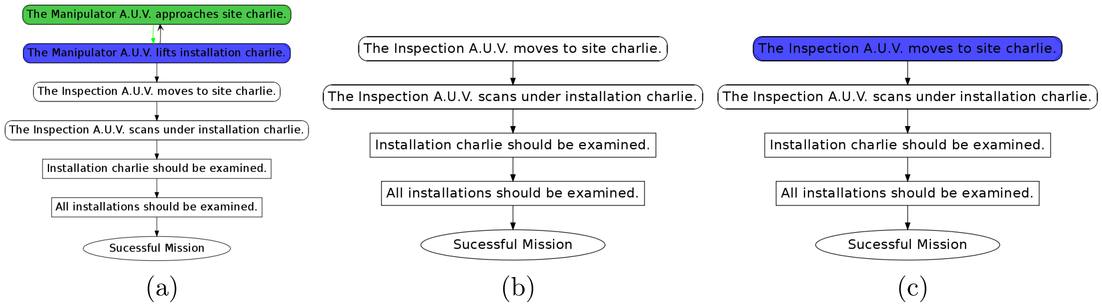

The Manipulator A.U.V. approaches site charlie.
The Manipulator A.U.V. lifts installation charlie.
The Inspection A.U.V. moves to site charlie.
The Inspection A.U.V. scans under installation charlie.
Installation charlie should be examined.
All installations should be examined.
Sucessful Mission
(a)
The Inspection A.U.V. moves to site charlie.
The Inspection A.U.V. scans under installation charlie.
Installation charlie should be examined.
All installations should be examined.
Sucessful Mission
(b)
The Inspection A.U.V. moves to site charlie.
The Inspection A.U.V. scans under installation charlie.
Installation charlie should be examined.
All installations should be examined.
Sucessful Mission
(c)

Figure G.30

00:06:15.00 VISUAL: graph I40 (Figure G.29c)

00:06:18.16 SPEECH: "It grabbed the nearest object"

00:06:20.36 SPEECH: "This was possible after the Manipulator A.U.V. approached site charlie."

00:06:20.36 VISUAL: graph I40 (Figure G.29c)

00:06:21.36 VISUAL: graph I41 (Figure G.30a)

00:06:23.93 VISUAL: graph I41 (Figure G.30a)

00:06:24.93 VISUAL: graph I40 (Figure G.29c)

00:06:25.43 PAUSE

00:06:26.43 SPEECH: "Skipping forward by 1 minute and 6 seconds"

00:06:30.63 SPEECH: "It finished lifting installation charlie."

00:06:31.13 VISUAL: graph I40 (Figure G.29c)

00:06:32.13 VISUAL: graph I42 (Figure G.30b)

00:06:33.80 UNFOCUS: auv1

00:06:33.80 PAUSE

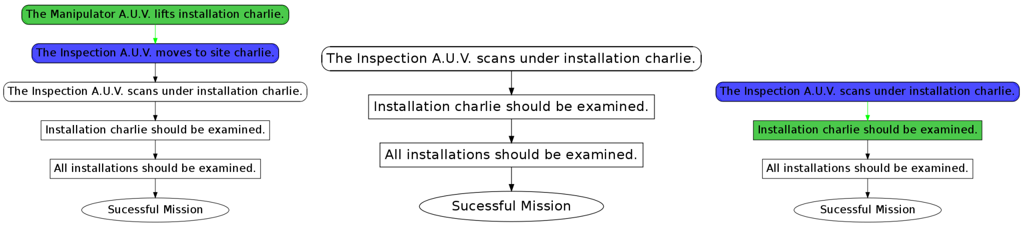
00:06:33.80 FOCUS: auv2

00:06:34.00 SPEECH: "The Inspection A.U.V. started to move to site charlie."

00:06:34.50 VISUAL: graph I42 (Figure G.30b)

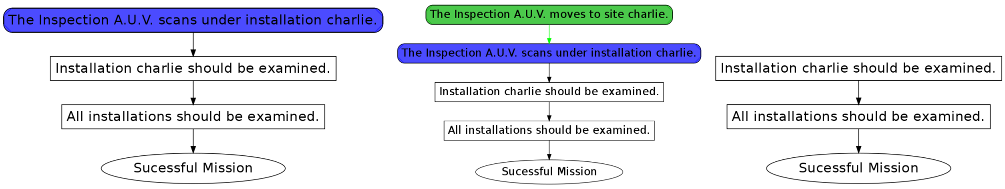
00:06:35.50 VISUAL: graph I43 (Figure G.30c)

00:06:37.88 SPEECH: "This was possible after the Manipulator A.U.V. lifted installation charlie."


The Manipulator A.U.V. lifts installation charlie.
The Inspection A.U.V. moves to site charlie.
The Inspection A.U.V. scans under installation charlie.
Installation charlie should be examined.
All installations should be examined.
Sucessful Mission
The Inspection A.U.V. scans under installation charlie.
Installation charlie should be examined.
All installations should be examined.
Sucessful Mission
The Inspection A.U.V. scans under installation charlie.
Installation charlie should be examined.
All installations should be examined.
Sucessful Mission

(a) (b) (c)

Figure G.31


The Inspection A.U.V. scans under installation charlie.
Installation charlie should be examined.
All installations should be examined.
Sucessful Mission
The Inspection A.U.V. moves to site charlie.
The Inspection A.U.V. scans under installation charlie.
Installation charlie should be examined.
All installations should be examined.
Sucessful Mission
Installation charlie should be examined.
All installations should be examined.
Sucessful Mission

(a) (b) (c)

Figure G.32

00:06:37.88 VISUAL: graph I43 (Figure G.30c)

00:06:38.88 VISUAL: graph I44 (Figure G.31a)

00:06:41.73 VISUAL: graph I44 (Figure G.31a)

00:06:42.73 VISUAL: graph I43 (Figure G.30c)

00:06:43.23 PAUSE

00:06:44.23 SPEECH: "Skipping forward by 43 seconds"

00:06:48.23 SPEECH: "It finished moving to site charlie."

00:06:48.73 VISUAL: graph I43 (Figure G.30c)

00:06:49.73 VISUAL: graph I45 (Figure G.31b)

00:06:50.98 PAUSE

00:06:51.18 SPEECH: "The Inspection A.U.V.'s scanning under installation charlie means that the desire that installation charlie should be examined will be satisfied."

00:06:51.18 VISUAL: graph I45 (Figure G.31b)

00:06:52.18 VISUAL: graph I46 (Figure G.31c)

00:06:57.78 VISUAL: graph I46 (Figure G.31c)

00:06:58.78 VISUAL: graph I45 (Figure G.31b)

00:06:58.98 SPEECH: "It started to do this and put its installation scanner in active mode."

00:06:59.48 VISUAL: graph I45 (Figure G.31b)

00:07:00.48 VISUAL: graph I47 (Figure G.32a)

00:07:03.13 SPEECH: "The status of the installation will be detected."

00:07:06.26 SPEECH: "This was possible after the Inspection A.U.V. moved to site charlie."

00:07:06.26 VISUAL: graph I47 (Figure G.32a)

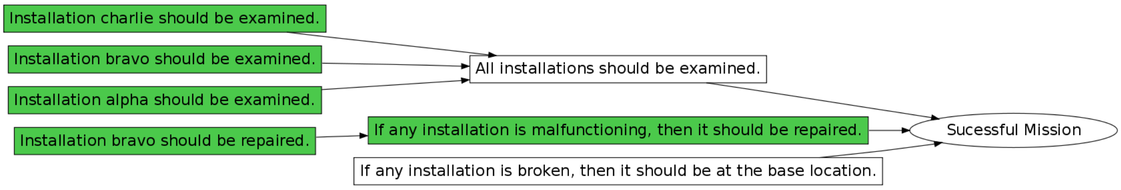

Installation charlie should be examined.
Installation bravo should be examined.
Installation alpha should be examined.
Installation bravo should be repaired.
All installations should be examined.
If any installation is malfunctioning, then it should be repaired.
If any installation is broken, then it should be at the base location.
Sucessful Mission

Figure G.33

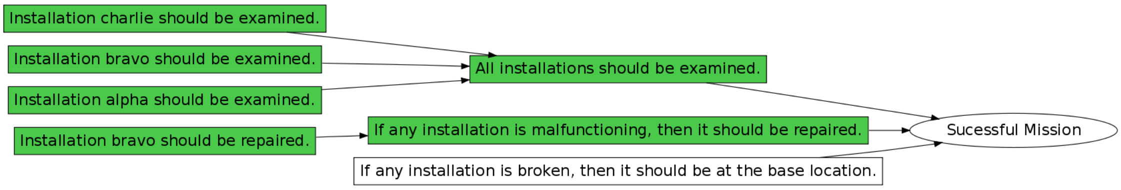

Installation charlie should be examined.
Installation bravo should be examined.
Installation alpha should be examined.
Installation bravo should be repaired.
All installations should be examined.
If any installation is malfunctioning, then it should be repaired.
If any installation is broken, then it should be at the base location.
Sucessful Mission

Figure G.34

00:07:07.26 VISUAL: graph I48 (Figure G.32b)

00:07:09.68 VISUAL: graph I48 (Figure G.32b)

00:07:10.68 VISUAL: graph I47 (Figure G.32a)

00:07:11.18 PAUSE

00:07:14.38 SPEECH: "It finished scanning under installation charlie and put its installation scanner in passive mode."

00:07:14.88 VISUAL: graph I47 (Figure G.32a)

00:07:15.88 VISUAL: graph I49 (Figure G.32c)

00:07:20.41 UNFOCUS: auv2

00:07:20.41 PAUSE

00:07:20.60 SPEECH: "The desire that installation charlie should be examined was satisfied."

00:07:21.10 VISUAL: graph I49 (Figure G.32c)

00:07:22.10 VISUAL: graph D18 (Figure G.28)

00:07:23.10 VISUAL: graph D18 (Figure G.28)

00:07:24.10 VISUAL: graph D19 (Figure G.33)

00:07:25.22 PAUSE

00:07:25.42 SPEECH: "The desire that all installations should be examined was satisfied."

00:07:25.92 VISUAL: graph D19 (Figure G.33)

00:07:26.92 VISUAL: graph D20 (Figure G.34)

00:07:29.36 VISUAL: graph D20 (Figure G.34)

00:07:29.86 VISUAL: BLANK

00:07:29.86 PAUSE

00:07:30.86 SPEECH: "Skipping backward by 25 seconds"

00:07:34.62 FOCUS: auv2

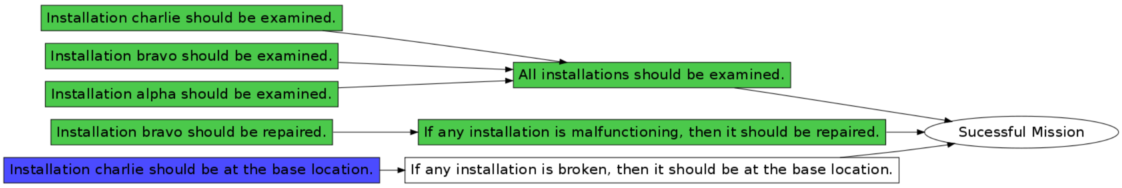

Installation charlie should be examined.
Installation bravo should be examined.
Installation alpha should be examined.
All installations should be examined.
Installation bravo should be repaired.
If any installation is malfunctioning, then it should be repaired.
Sucessful Mission
Installation charlie should be at the base location.
If any installation is broken, then it should be at the base location.

Figure G.35

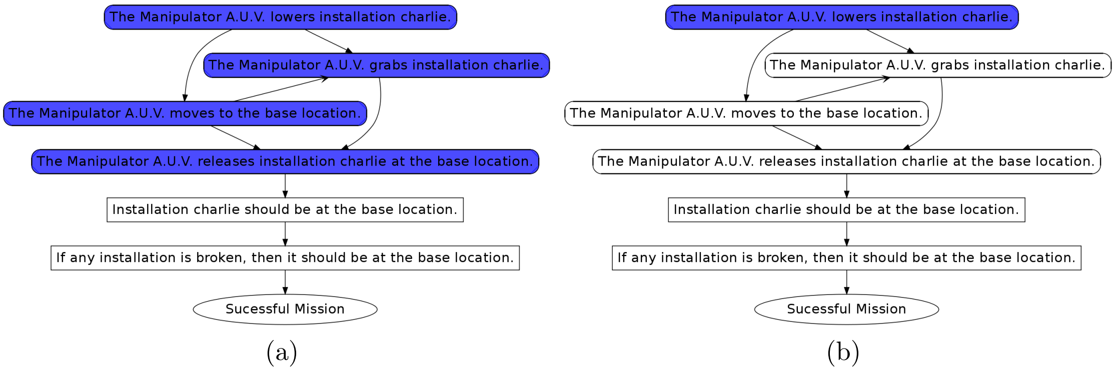

The Manipulator A.U.V. lowers installation charlie.
The Manipulator A.U.V. grabs installation charlie.
The Manipulator A.U.V. moves to the base location.
The Manipulator A.U.V. releases installation charlie at the base location.
Installation charlie should be at the base location.
If any installation is broken, then it should be at the base location.
Sucessful Mission
(a)
The Manipulator A.U.V. lowers installation charlie.
The Manipulator A.U.V. grabs installation charlie.
The Manipulator A.U.V. moves to the base location.
The Manipulator A.U.V. releases installation charlie at the base location.
Installation charlie should be at the base location.
If any installation is broken, then it should be at the base location.
Sucessful Mission
(b)

Figure G.36

00:07:34.82 SPEECH: "The Inspection A.U.V.'s installation scanner detected that installation charlie is broken."

00:07:40.58 UNFOCUS: auv2

00:07:40.58 PAUSE

00:07:40.58 VISUAL: BLANK

00:07:40.78 VISUAL: graph D20 (Figure G.34)

00:07:40.78 SPEECH: "Installation charlie is at location north 8 east 20 depth 15."

00:07:40.78 FOCUS: o3

00:07:45.10 UNFOCUS: o3

00:07:45.30 SPEECH: "The desire that it should be at the base location was introduced."

00:07:45.80 VISUAL: graph D20 (Figure G.34)

00:07:46.80 VISUAL: graph D21 (Figure G.35)

00:07:49.42 PAUSE

00:07:49.62 SPEECH: "A plan was created to satisfy this."

00:07:50.12 VISUAL: graph D21 (Figure G.35)

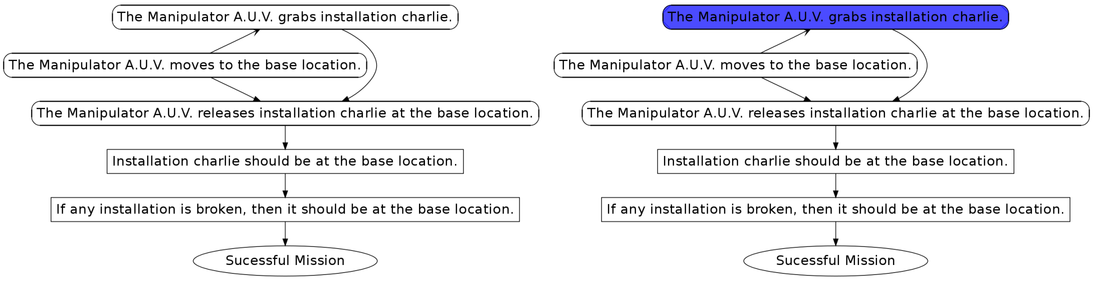
00:07:51.12 VISUAL: graph I50 (Figure G.36a)

00:07:52.65 PAUSE

00:07:53.65 SPEECH: "Skipping forward by 11 seconds"

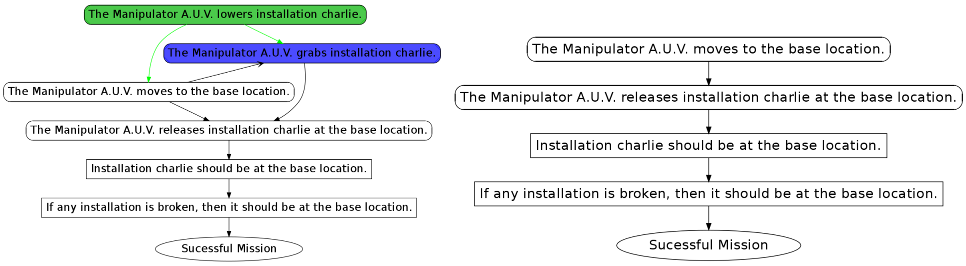
00:07:57.13 FOCUS: auv1

00:07:57.33 SPEECH: "In order that installation charlie should be at the base location, the Manipulator A.U.V. started to lower installation charlie."


The Manipulator A.U.V. grabs installation charlie.
The Manipulator A.U.V. moves to the base location.
The Manipulator A.U.V. releases installation charlie at the base location.
Installation charlie should be at the base location.
If any installation is broken, then it should be at the base location.
Sucessful Mission
The Manipulator A.U.V. grabs installation charlie.
The Manipulator A.U.V. moves to the base location.
The Manipulator A.U.V. releases installation charlie at the base location.
Installation charlie should be at the base location.
If any installation is broken, then it should be at the base location.
Sucessful Mission

(a) (b)

Figure G.37


The Manipulator A.U.V. lowers installation charlie.
The Manipulator A.U.V. grabs installation charlie.
The Manipulator A.U.V. moves to the base location.
The Manipulator A.U.V. releases installation charlie at the base location.
Installation charlie should be at the base location.
If any installation is broken, then it should be at the base location.
Sucessful Mission
The Manipulator A.U.V. moves to the base location.
The Manipulator A.U.V. releases installation charlie at the base location.
Installation charlie should be at the base location.
If any installation is broken, then it should be at the base location.
Sucessful Mission

(a) (b)

Figure G.38

00:07:57.83 VISUAL: graph I50 (Figure G.36a)

00:07:58.83 VISUAL: graph I51 (Figure G.36b)

00:08:05.36 PAUSE

00:08:06.36 SPEECH: "Skipping forward by 2 minutes and 37 seconds"

00:08:11.13 SPEECH: "It finished doing this and put its lifting arm in passive mode."

00:08:11.63 VISUAL: graph I51 (Figure G.36b)

00:08:12.63 VISUAL: graph I52 (Figure G.37a)

00:08:14.81 SPEECH: "It released the grabbed object."

00:08:17.32 PAUSE

00:08:17.51 SPEECH: "The Manipulator A.U.V. started to grab installation charlie and put its lifting arm in active mode."

00:08:18.01 VISUAL: graph I52 (Figure G.37a)

00:08:19.01 VISUAL: graph I53 (Figure G.37b)

00:08:23.40 SPEECH: "This was possible after the Manipulator A.U.V. lowered installation charlie."

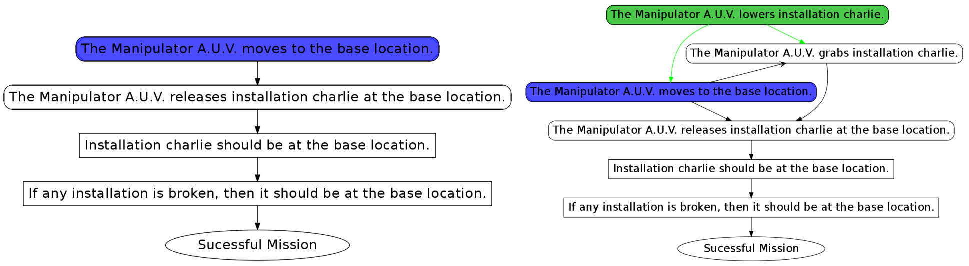
00:08:23.40 VISUAL: graph I53 (Figure G.37b)

00:08:24.40 VISUAL: graph I54 (Figure G.38a)

00:08:27.15 VISUAL: graph I54 (Figure G.38a)

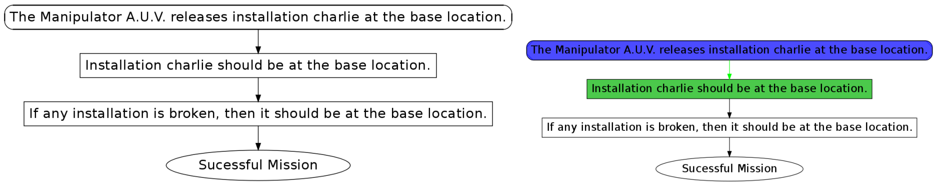
00:08:28.15 VISUAL: graph I53 (Figure G.37b)

00:08:28.65 PAUSE


The Manipulator A.U.V. moves to the base location.
The Manipulator A.U.V. releases installation charlie at the base location.
Installation charlie should be at the base location.
If any installation is broken, then it should be at the base location.
Sucessful Mission
The Manipulator A.U.V. lowers installation charlie.
The Manipulator A.U.V. grabs installation charlie.
The Manipulator A.U.V. moves to the base location.
The Manipulator A.U.V. releases installation charlie at the base location.
Installation charlie should be at the base location.
If any installation is broken, then it should be at the base location.
Sucessful Mission

(a) (b)

Figure G.39


The Manipulator A.U.V. releases installation charlie at the base location.
Installation charlie should be at the base location.
If any installation is broken, then it should be at the base location.
Sucessful Mission
The Manipulator A.U.V. releases installation charlie at the base location.
Installation charlie should be at the base location.
If any installation is broken, then it should be at the base location.
Sucessful Mission

(a) (b)

Figure G.40

00:08:28.85 SPEECH: "It finished grabbing installation charlie."

00:08:29.35 VISUAL: graph I53 (Figure G.37b)

00:08:30.35 VISUAL: graph I55 (Figure G.38b)

00:08:32.06 PAUSE

00:08:32.26 SPEECH: "It started to move to the base location."

00:08:32.76 VISUAL: graph I55 (Figure G.38b)

00:08:33.76 VISUAL: graph I56 (Figure G.39a)

00:08:34.99 SPEECH: "This was possible after the Manipulator A.U.V. lowered installation charlie."

00:08:34.99 VISUAL: graph I56 (Figure G.39a)

00:08:35.99 VISUAL: graph I57 (Figure G.39b)

00:08:38.74 VISUAL: graph I57 (Figure G.39b)

00:08:39.74 VISUAL: graph I56 (Figure G.39a)

00:08:40.24 PAUSE

00:08:41.24 SPEECH: "Skipping forward by 1 minute and 21 seconds"

00:08:45.64 SPEECH: "It finished moving to the base location."

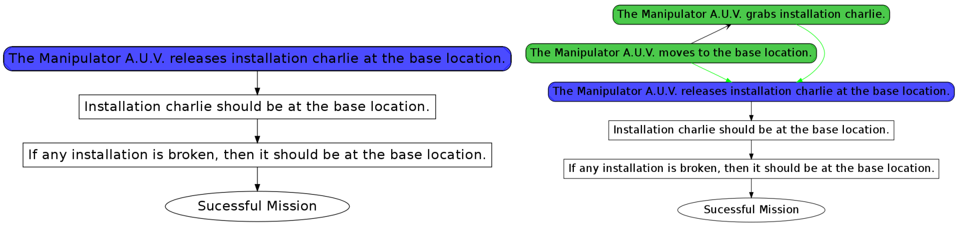
00:08:46.14 VISUAL: graph I56 (Figure G.39a)

00:08:47.14 VISUAL: graph I58 (Figure G.40a)

00:08:48.65 PAUSE

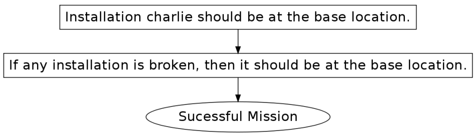
00:08:48.85 SPEECH: "The Manipulator A.U.V.'s releasing installation charlie at the base location means that the desire that installation charlie should be at the base location will be satisfied."

00:08:48.85 VISUAL: graph I58 (Figure G.39a)


The Manipulator A.U.V. releases installation charlie at the base location.
Installation charlie should be at the base location.
If any installation is broken, then it should be at the base location.
Sucessful Mission
The Manipulator A.U.V. grabs installation charlie.
The Manipulator A.U.V. moves to the base location.
The Manipulator A.U.V. releases installation charlie at the base location.
Installation charlie should be at the base location.
If any installation is broken, then it should be at the base location.
Sucessful Mission

(a) (b)

Figure G.41


Installation charlie should be at the base location.
If any installation is broken, then it should be at the base location.
Sucessful Mission

(a)

Figure G.42

00:08:49.85 VISUAL: graph I59 (Figure G.40b)

00:08:57.14 VISUAL: graph I59 (Figure G.40b)

00:08:58.14 VISUAL: graph I58 (Figure G.40a)

00:08:58.34 SPEECH: "It started to do this."

00:08:58.84 VISUAL: graph I58 (Figure G.40a)

00:08:59.84 VISUAL: graph I60 (Figure G.41a)

00:09:00.04 SPEECH: "This was possible after the Manipulator A.U.V. grabbed installation charlie and the Manipulator A.U.V. moved to the base location."

00:09:00.04 VISUAL: graph I60 (Figure G.41a)

00:09:01.04 VISUAL: graph I61 (Figure G.41b)

00:09:06.88 VISUAL: graph I61 (Figure G.41b)

00:09:07.88 VISUAL: graph I60 (Figure G.41a)

00:09:08.38 PAUSE

00:09:08.58 SPEECH: "It finished releasing installation charlie at the base location and put its lifting arm in passive mode."

00:09:09.08 VISUAL: graph I60 (Figure G.41a)

00:09:10.08 VISUAL: graph I62 (Figure G.42a)

00:09:14.82 UNFOCUS: auv1

00:09:14.82 PAUSE

00:09:15.02 SPEECH: "The desire that installation charlie should be at the base location was satisfied."

00:09:15.52 VISUAL: graph I62 (Figure G.42a)

00:09:16.52 VISUAL: graph D22 (Figure G.43)

00:09:17.52 VISUAL: graph D22 (Figure G.43)

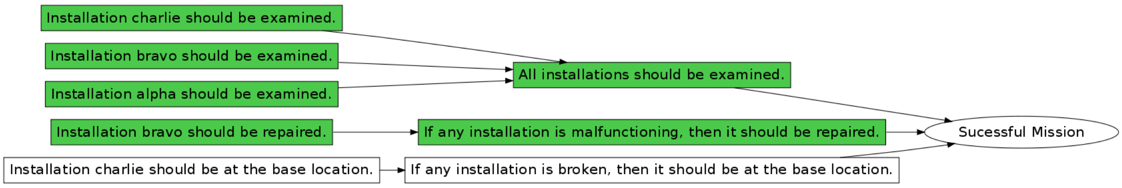

Installation charlie should be examined.
Installation bravo should be examined.
Installation alpha should be examined.
Installation bravo should be repaired.
Installation charlie should be at the base location.
All installations should be examined.
If any installation is malfunctioning, then it should be repaired.
If any installation is broken, then it should be at the base location.
Sucessful Mission

Figure G.43

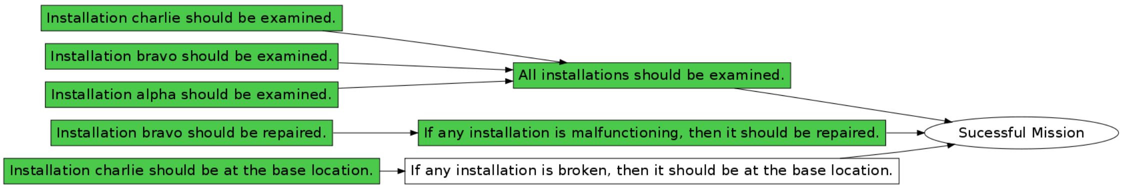

Installation charlie should be examined.
Installation bravo should be examined.
Installation alpha should be examined.
Installation bravo should be repaired.
Installation charlie should be at the base location.
All installations should be examined.
If any installation is malfunctioning, then it should be repaired.
If any installation is broken, then it should be at the base location.
Sucessful Mission

Figure G.44

00:09:18.52 VISUAL: graph D23 (Figure G.44)

00:09:20.22 PAUSE

00:09:20.42 SPEECH: "The desire that if any installation is broken, then it should be at the base location was satisfied."

00:09:20.92 VISUAL: graph D23 (Figure G.44)

00:09:21.92 VISUAL: graph D24 (Figure G.45)

00:09:26.77 PAUSE

00:09:26.97 SPEECH: "The mission succeeded."

00:09:29.47 VISUAL: graph D24 (Figure G.45)

00:09:30.47 VISUAL: graph D25 (Figure G.46)

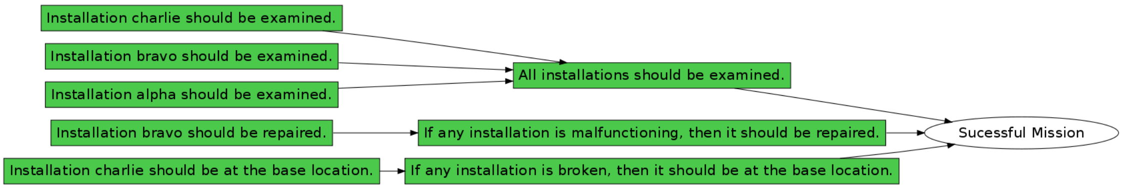

Installation charlie should be examined.
Installation bravo should be examined.
Installation alpha should be examined.
Installation bravo should be repaired.
Installation charlie should be at the base location.
All installations should be examined.
If any installation is malfunctioning, then it should be repaired.
If any installation is broken, then it should be at the base location.
Sucessful Mission

Figure G.45

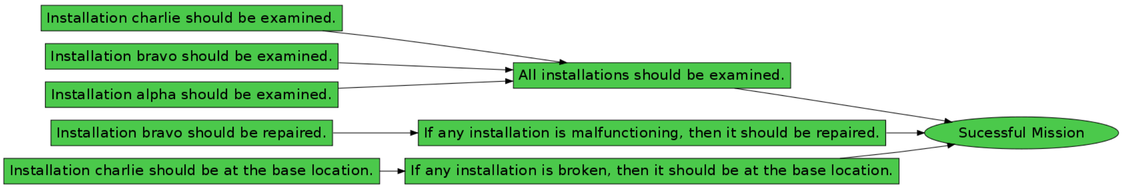
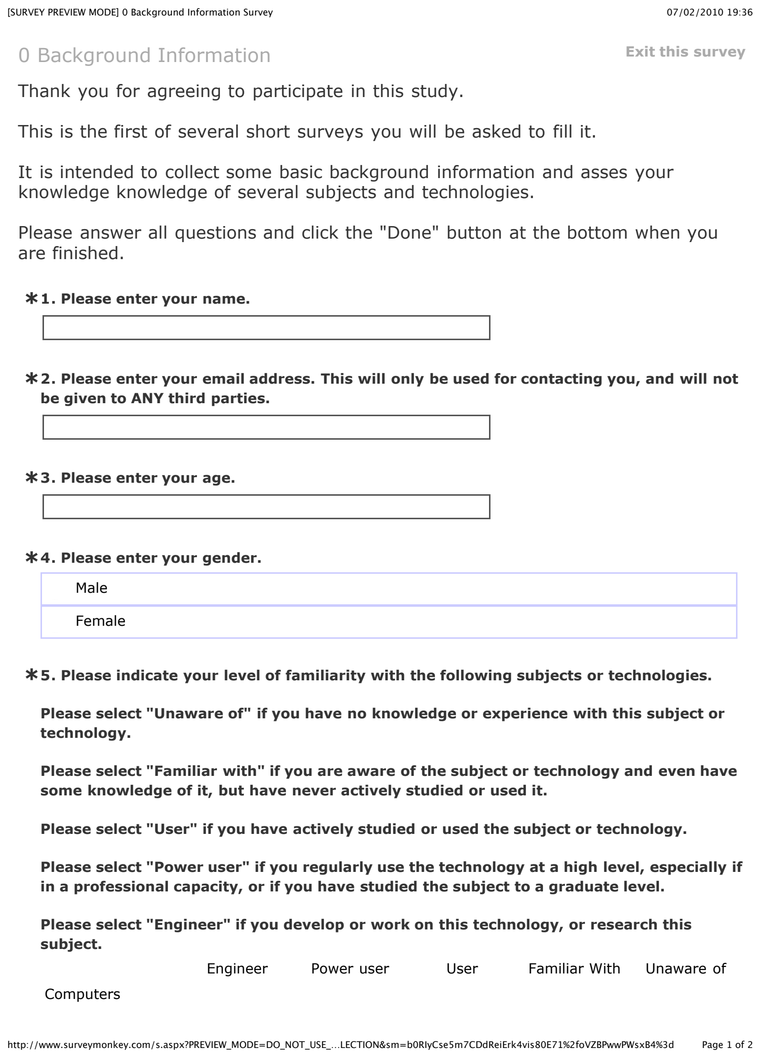

Installation charlie should be examined.
Installation bravo should be examined.
Installation alpha should be examined.
All installations should be examined.
Installation bravo should be repaired.
If any installation is malfunctioning, then it should be repaired.
Sucessful Mission
Installation charlie should be at the base location.
If any installation is broken, then it should be at the base location.

Figure G.46

# Appendix H

# Example Surveys and Instructions Given to Experiment participants

## H.1 Background information survey



[SURVEY PREVIEW MODE] 0 Background Information Survey 07/02/2010 19:36

0 Background Information

Thank you for agreeing to participate in this study.

This is the first of several short surveys you will be asked to fill it.

It is intended to collect some basic background information and asses your knowledge knowledge of several subjects and technologies.

Please answer all questions and click the "Done" button at the bottom when you are finished.

***1. Please enter your name.**

***2. Please enter your email address. This will only be used for contacting you, and will not be given to ANY third parties.**

***3. Please enter your age.**

***4. Please enter your gender.**

Male

Female

***5. Please indicate your level of familiarity with the following subjects or technologies.**

**Please select "Unaware of" if you have no knowledge or experience with this subject or technology.**

**Please select "Familiar with" if you are aware of the subject or technology and even have some knowledge of it, but have never actively studied or used it.**

**Please select "User" if you have actively studied or used the subject or technology.**

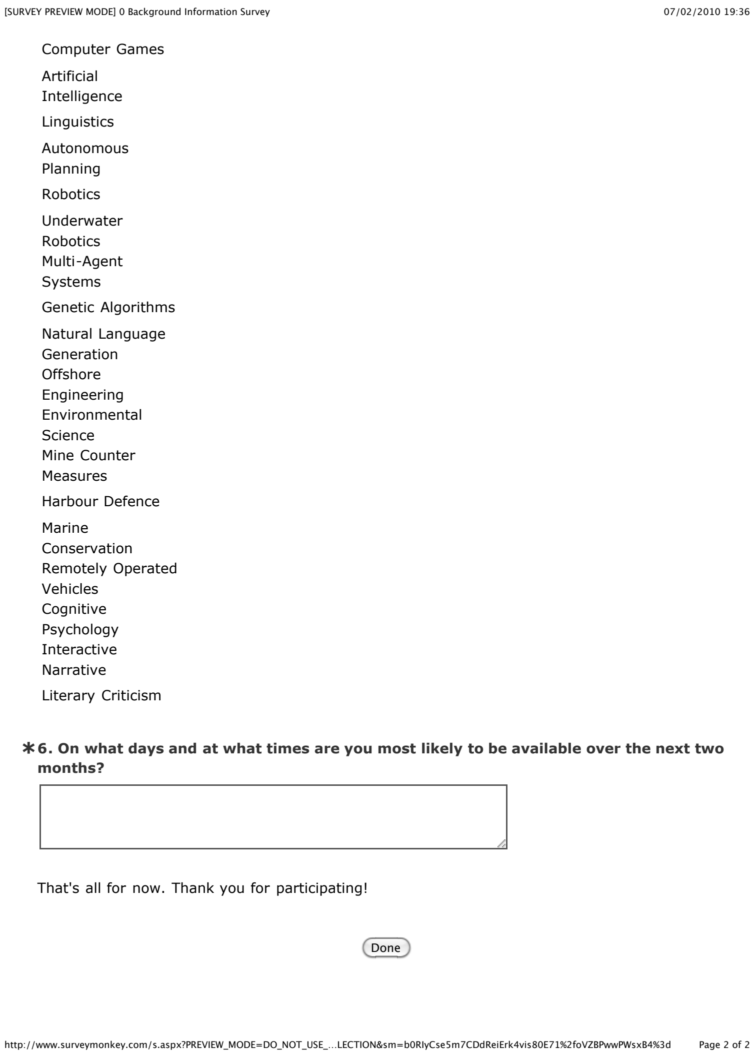
**Please select "Power user" if you regularly use the technology at a high level, especially if in a professional capacity, or if you have studied the subject to a graduate level.**

**Please select "Engineer" if you develop or work on this technology, or research this subject.**

| | Engineer | Power user | User | Familiar With | Unaware of |
|---|---|---|---|---|---|
| Computers | | | | | |

http://www.surveymonkey.com/s.aspx?PREVIEW_MODE=DO_NOT_USE_...LECTION&sm=b0RIyCse5m7CDdReiErk4vis80E71%2foVZBPwwPWsxB4%3d Page 1 of 2

Figure H.1: Background information survey, page 1.



[SURVEY PREVIEW MODE] 0 Background Information Survey 07/02/2010 19:36

Computer Games

Artificial Intelligence

Linguistics

Autonomous Planning

Robotics

Underwater Robotics

Multi-Agent Systems

Genetic Algorithms

Natural Language Generation

Offshore Engineering

Environmental Science

Mine Counter Measures

Harbour Defence

Marine Conservation

Remotely Operated Vehicles

Cognitive Psychology

Interactive Narrative

Literary Criticism

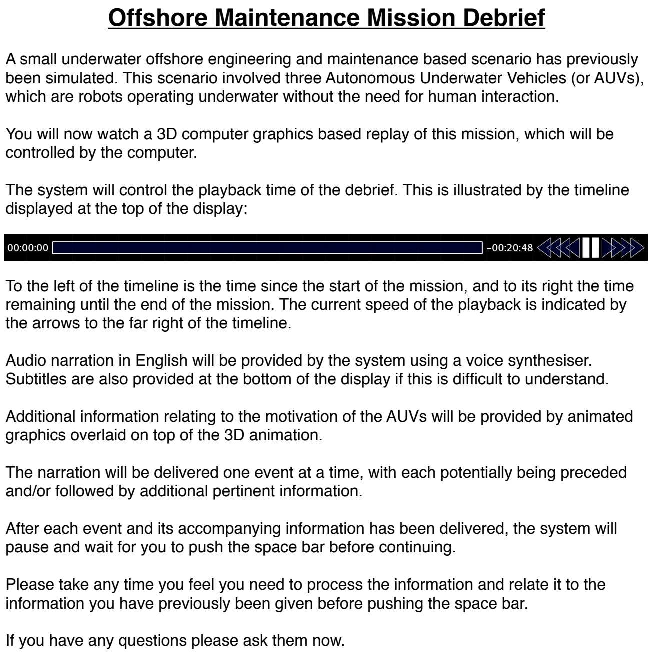
***6. On what days and at what times are you most likely to be available over the next two months?**

That's all for now. Thank you for participating!

http://www.surveymonkey.com/s.aspx?PREVIEW_MODE=DO_NOT_USE_...LECTION&sm=b0RlyCse5m7CDdReiErk4vis80E71%2foVZBPwwPWsxB4%3d Page 2 of 2

Figure H.2: Background information survey, page 2.

## H.2 Scripted Instructions



**Offshore Maintenance Mission Debrief**

A small underwater offshore engineering and maintenance based scenario has previously been simulated. This scenario involved three Autonomous Underwater Vehicles (or AUVs), which are robots operating underwater without the need for human interaction.

You will now watch a 3D computer graphics based replay of this mission, which will be controlled by the computer.

The system will control the playback time of the debrief. This is illustrated by the timeline displayed at the top of the display:

To the left of the timeline is the time since the start of the mission, and to its right the time remaining until the end of the mission. The current speed of the playback is indicated by the arrows to the far right of the timeline.

Audio narration in English will be provided by the system using a voice synthesiser. Subtitles are also provided at the bottom of the display if this is difficult to understand.

Additional information relating to the motivation of the AUVs will be provided by animated graphics overlaid on top of the 3D animation.

The narration will be delivered one event at a time, with each potentially being preceded and/or followed by additional pertinent information.

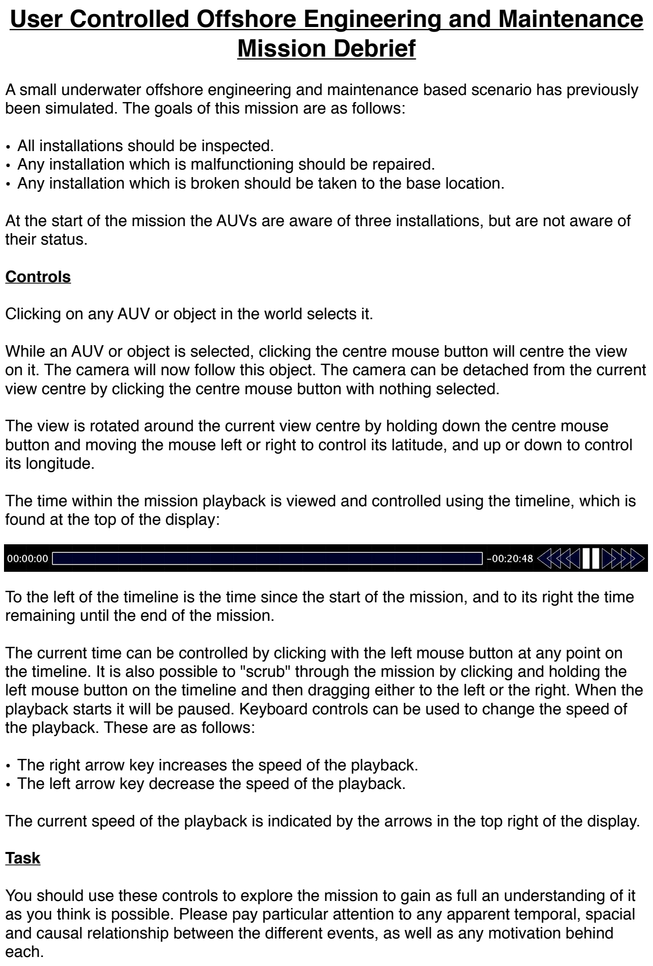
After each event and its accompanying information has been delivered, the system will pause and wait for you to push the space bar before continuing.

Please take any time you feel you need to process the information and relate it to the information you have previously been given before pushing the space bar.

If you have any questions please ask them now.

Figure H.3: Experimental case instructions for the installation repair scenario.

## H.3 User controlled Instructions



### User Controlled Offshore Engineering and Maintenance Mission Debrief

A small underwater offshore engineering and maintenance based scenario has previously been simulated. The goals of this mission are as follows:

- All installations should be inspected.
- Any installation which is malfunctioning should be repaired.
- Any installation which is broken should be taken to the base location.

At the start of the mission the AUVs are aware of three installations, but are not aware of their status.

**Controls**

Clicking on any AUV or object in the world selects it.

While an AUV or object is selected, clicking the centre mouse button will centre the view on it. The camera will now follow this object. The camera can be detached from the current view centre by clicking the centre mouse button with nothing selected.

The view is rotated around the current view centre by holding down the centre mouse button and moving the mouse left or right to control its latitude, and up or down to control its longitude.

The time within the mission playback is viewed and controlled using the timeline, which is found at the top of the display:

To the left of the timeline is the time since the start of the mission, and to its right the time remaining until the end of the mission.

The current time can be controlled by clicking with the left mouse button at any point on the timeline. It is also possible to "scrub" through the mission by clicking and holding the left mouse button on the timeline and then dragging either to the left or the right. When the playback starts it will be paused. Keyboard controls can be used to change the speed of the playback. These are as follows:

- The right arrow key increases the speed of the playback.
- The left arrow key decrease the speed of the playback.

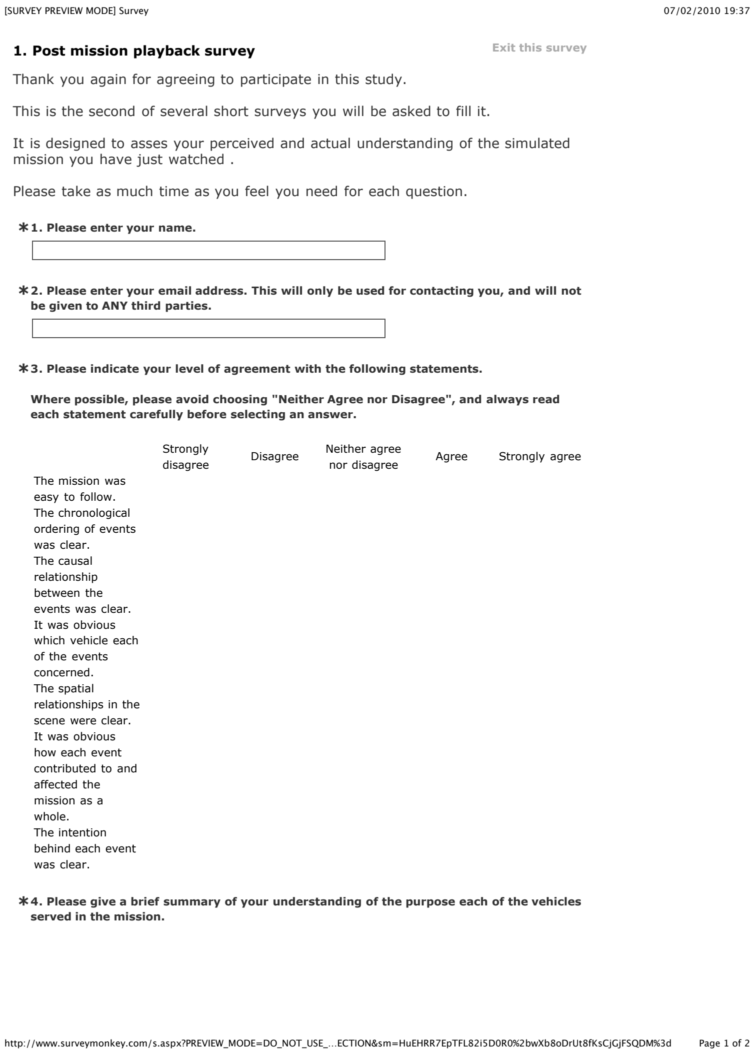
The current speed of the playback is indicated by the arrows in the top right of the display.

**Task**

You should use these controls to explore the mission to gain as full an understanding of it as you think is possible. Please pay particular attention to any apparent temporal, spacial and causal relationship between the different events, as well as any motivation behind each.

Figure H.4: User case instructions for the installation repair scenario.

## H.4 Post experiment Surveys



[SURVEY PREVIEW MODE] Survey 07/02/2010 19:37

**1. Post mission playback survey**

Exit this survey

Thank you again for agreeing to participate in this study.

This is the second of several short surveys you will be asked to fill it.

It is designed to asses your perceived and actual understanding of the simulated mission you have just watched .

Please take as much time as you feel you need for each question.

***1. Please enter your name.**

***2. Please enter your email address. This will only be used for contacting you, and will not be given to ANY third parties.**

***3. Please indicate your level of agreement with the following statements.**

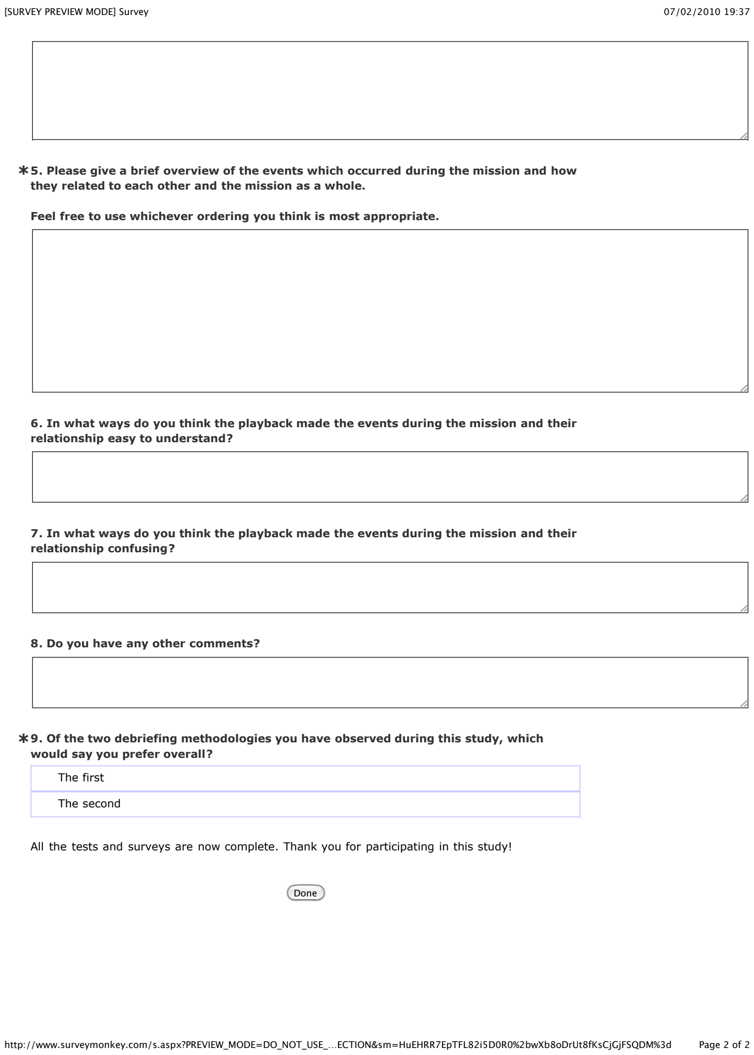
**Where possible, please avoid choosing "Neither Agree nor Disagree", and always read each statement carefully before selecting an answer.**

| | Strongly disagree | Disagree | Neither agree nor disagree | Agree | Strongly agree |
|---|---|---|---|---|---|
| The mission was easy to follow. | | | | | |
| The chronological ordering of events was clear. | | | | | |
| The causal relationship between the events was clear. | | | | | |
| It was obvious which vehicle each of the events concerned. | | | | | |
| The spatial relationships in the scene were clear. | | | | | |
| It was obvious how each event contributed to and affected the mission as a whole. | | | | | |
| The intention behind each event was clear. | | | | | |

***4. Please give a brief summary of your understanding of the purpose each of the vehicles served in the mission.**

http://www.surveymonkey.com/s.aspx?PREVIEW_MODE=DO_NOT_USE_...ECTION&sm=HuEHRR7EpTFL82i5D0R0%2bwXb8oDrUt8fKsCjGjFSQDM%3d Page 1 of 2

Figure H.5: Post experiment survey, page 1.



[SURVEY PREVIEW MODE] Survey 07/02/2010 19:37

*5. Please give a brief overview of the events which occurred during the mission and how they related to each other and the mission as a whole.

Feel free to use whichever ordering you think is most appropriate.

6. In what ways do you think the playback made the events during the mission and their relationship easy to understand?

7. In what ways do you think the playback made the events during the mission and their relationship confusing?

8. Do you have any other comments?

*9. Of the two debriefing methodologies you have observed during this study, which would say you prefer overall?

The first

The second

All the tests and surveys are now complete. Thank you for participating in this study!

http://www.surveymonkey.com/s.aspx?PREVIEW_MODE=DO_NOT_USE_...ECTION&sm=HuEHRR7EpTFL82i5D0R0%2bwXb8oDrUt8fKsCjGjFSQDM%3d Page 2 of 2

Figure H.6: Post experiment survey, page 2.

# Bibliography

[1] *Homeworld*, Computer Game, Sierra Entertainment Incorperated, 1999.

[2] R. van der Krogt, “Plan repair in single-agent and multi-agent systems,” Ph.D. dissertation, The Netherlands TRAIL Research School, 2005.

[3] S. Chatman, *Story and Discourse.* Cornell University Press, 1980.

[4] “The navy unmanned undersea vehicle (UUV) master plan,” Office of Naval Research, Tech. Rep., July 2004.

[5] D. Jones, “Operation Iraqi freedom: Mine countermeasures a success,” *Underwater Magazine*, July/August 2003.

[6] A. Cormack, D. M. Lane, and J. Wood, “Operational experiences for maritime homeland security operations,” in *IEEE International Conference Oceans 2010*, May 2010.

[7] J. Evans, P. Patron, B. Privat, N. Johnson, and C. Capus, “Autotracker: Autonomous inspection – capabilities and lessons learned in offshore operations,” in *Proceedings of the IEEE Oceans Conference 2009*, 2009.

[8] J. Jamieson and I. Tena, “Autonomous vehicle meets new challenges,” in *Proceedings of Deep Offshore Technology*, February 2009.

[9] M. Schmiing, P. Afonso, F. Tempera, and R. Santos, “Integrating recent and future marine technology in the design of marine protected areas - the azores as case study,” in *Proceedings of the IEEE Oceans Conference 2009 - Europe*, June 2009.

[10] C. Kunz, C. Murphy, R. Camilli, H. Singh, J. Bailey, R. Eustice, M. Jakuba, K. Nakamura, C. Roman, T. Sato, R. Sohn, and C. Willis, “Deep sea underwater robotic exploration in the ice-covered arctic ocean with auv,” in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, September 2008.

[11] "World record for autonomous pipeline tracking," Published on Hydro International Magazine website. http://www.hydro-international.com/news/id2482-World_Record_for_Autonomous_Pipeline_Tracking.html, last checked on 20/11/2009, June 2008.

[12] "Offshore pipelines inspected using gavia auv," Published on Hydro International Magazine website. http://www.hydro-international.com/news/id2748-Offshore_Pipelines_Inspected_using_GAVIA_AUV.html, last checked on 20/11/2009, October 2008.

[13] J. Vaganay, M. Elkins, D. Esposito, W. O'Halloran, F. Hover, and M. Kokko, "Ship hull inspection with the hauv: Us navy and nato demonstrations results," in *Proceedings of the IEEE Oceans Conference 2006*, 2006.

[14] R. P. Stokey, L. E. Freitag, and M. D. Grund, "A compact control language for auv acoustic communication," in *Proceedings of the IEEE Oceans Conference 2005*, 2005.

[15] C. Pêtrès, Y. Pailhas, P. Patrón, Y. Petillot, J. Evans, and D. Lane, "Path planning for autonomous underwater vehicles," *IEEE Transactions on Robotics*, vol. 23, no. 2, pp. 331–341, April 2007.

[16] C. Petres, Y. Pailhas, P. Patron, J. Evans, Y. Petillot, and D. Lane, *Underwater Vehicles.* IN-TECH, 2008, ch. Trajectory Planning for Autonomous Underwater Vehicles, pp. 399–416.

[17] SAUC-E Committee, "Student autonomous underwater competition - europe 2008," Web Page: http://www.dstl.gov.uk/news_events/competitions/sauce/08/index.php. Last Checked: 18/11/2008, 2008.

[18] "Student autonomous underwater challenge - europe (SAUC-E) 2009," Published on DSTL website, competions section: http://www.dstl.gov.uk/news_events/competitions/sauce/09/index.php, last checked on 15/11/2009, 2009.

[19] P. Rincon, "Teams vie for underwater robot prize," Published on BBC Online News, Science and Environment Section. http://news.bbc.co.uk/1/hi/sci/tech/8143541.stm, last checked on 15/11/2009, July 2009.

[20] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. N. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, M. Gittleman, S. Harbaugh, M. Hebert, T. M. Howard, S. Kolski, A. Kelly, M. Likhachev, M. McNaughton, N. Miller, K. Peterson, B. Pilnick, R. Rajkumar, P. Rybski, B. Salesky, Y.-W. Seo, S. Singh, J. Snider, A. Stentz, W. R. Whittaker, Z. Wolkowicki, J. Ziglar,

H. Bae, T. Brown, D. Demitrish, B. Litkouhi, J. Nickolaou, V. Sadekar, W. Zhang, J. Struble, M. Taylor, M. Darms, and D. Ferguson, “Autonomous driving in urban environments: Boss and the urban challenge,” *Journal of Field Robotics*, vol. 25, no. 8, pp. 425 – 466, July 2008.

[21] J. Sprinkle, J. M. Eklund, H. Gonzalez, E. I. Grøtli, B. Upcroft, A. Makarenko, W. Uther, M. Moser, R. Fitch, H. Durrant-Whyte, and S. S. Sastry, “Model-based design: a report from the trenches of the darpa urban challenge,” *Software and Systems Modeling*, vol. 8, no. 4, pp. 551–566, September 2009.

[22] S. D. Prior, “The MoD grand challenge 2008,” *Defence management journal*, no. 42, pp. 22–23, 2008.

[23] “Heriot-watt university triumph,” Published on Hydro Internation Magazine website. http://www.hydro-international.com/news/id2669-HeriotWatt_University_Triumph.html, last checked on 20/11/2009, September 2008.

[24] “SAUC-E Victory for Nessie IV,” Published on Hydro Internation Magazine website. http://www.hydrointernational.com/news/id3320-SAUCE_Victory_for_Nessie_IV.html, last checked on 20/11/2009, July 2009.

[25] J. Cartright, N. Johnson, B. Davis, Z. Qiang, T. L. Bravo, A. Enoch, G. Lemaitre, H. Roth, and Y. Petillot, “Nessie III autonomous underwater vehicle for SAUC-E 2008,” in *Proceeding of the Unmanned Underwater Vehicle Showcase 2008*, 2008.

[26] F. Maurelli, J. Cartwright, N. Johnson, and Y. Petillot, “Nessie IV autonomous underwater vehicle wins the SAUC-E competition,” in *Proceedings of Robotica 2010*, 2010.

[27] L. Erman, F. Hayes-Roth, V. R. Lesser, and R. Reddy, “The hearsay-II speech-understanding system: Integrating knowledge to resolve uncertainty,” *Computing Surveys*, vol. 12, no. 2, pp. 213–253, June 1980.

[28] V. Lesser, “Cooperative multi-agent systems: A personal view of the state of the art,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 11, no. 1, pp. 133–142, 1999.

[29] D. D. Corkill, “Collaborating software: Blackboard and multi-agent systems & the future,” in *Proceedings of the International Lisp Conference*, New York, New York, October 2003.

[30] R. C. Arkin and K. S. Ali, "Integration of reactive and telerobotic control in multi-agent robotic systems," in *Proceedings of the third international conference on Simulation of adaptive behavior : from animals to animats 3.* MIT Press, 1994, pp. 473 – 478.

[31] T. Taylor, "A genetic regulatory network-inspired real-time controller for a group of underwater robots," in *Intelligent Autonomous Systems 8*, 2004, pp. 403–412.

[32] C. Sotzing and D.M.Lane, "Improving the co-ordination efficiency of limited communication multi-AUV operations using a multi- agent architecture," *Journal of Field Robotics*, vol. 27, no. 4, pp. 412–429, July 2010.

[33] C. C. Sotzing, N. A. Johnson, and D. Lane, "Improving multi-AUV coordination with hierarchical blackboard-based plan representation," in *Proceedings of the 27th Workshop of the UK Planning and Scheduling Special Interest Group*, December 2008, (Accepted for publication).

[34] A. Aguiar, J. Almeida, M. Bayat, B. Cardeira, R. Cunha, A. Hausler, P. Maurya, A. Oliveira, A. Pascoal, A. Pereira, M. Rufino, L. Sebastiao, C. Silvestre, and F. Vanni, "Cooperative autonomous marine vehicle motion control in the scope of the EU GREX project: Theory and practice," in *Proceedings of the IEEE Oceans Conference 2009 - Europe*, June 2009.

[35] J. Kalwa, "The GREX-Project: Coordination and control of cooperating heterogeneous unmanned systems in uncertain environments," in *Proceedings of the IEEE Oceans Conference 2009 - Europe*, June 2009.

[36] R. Engel and J. Kalwa, "Relative positioning of multiple underwater vehicles in the GREX project," in *Proceedings of the IEEE Oceans Conference 2009 - Europe*, June 2009.

[37] J. Alves and M. Schneider, "A communication infrastructure for cooperative operation of a fleet of heterogeneous autonomous marine vehicles: Concepts and developments within the GREX project," in *Proceedings of the IEEE Oceans Conference 2009 - Europe*, June 2009.

[38] L. Brignone, J. Alves, and J. Opderbecke, "GREX sea trials: first experiences in multiple underwater vehicle coordination based on acoustic communication," in *Proceedings of the IEEE Oceans Conference 2009 - Europe*, June 2009.

[39] M. Ghallab, D. S. Nau, and P. Traverso, *Automated Planning: Theory and Practice.* Morgan Kaufmann, 2004.

[40] R. Fikes and N. J. Nilsson, "STRIPS: A new approach to the application of theorem proving to problem solving," *Artificial Intelligence*, vol. 2, pp. 189–208, 1971.

[41] A. Tate, "Generating project networks," in *Proceedings of IJCAI-77*, 1977, pp. 888–893.

[42] E. D. Sacerdoti, *A Structure for Plans and Behavior.* Elsevier, New York, 1977.

[43] K. Erol, J. Hendler, and D. S. Nau, "Semantics for hierarchical task-network planning," University of Maryland, Tech. Rep., March 1994.

[44] ——, "Complexity results for HTN planning," University of Maryland, Tech. Rep., 1994.

[45] K. Erol, "Hierarchical task network planning: Formalization, analysis, and implementation," Ph.D. dissertation, Department of Computer Science, The University of Maryland, 1995.

[46] M. Fox and D. Long, "PDDL 2.1: An extension to PDDL for expressing temperal planning domains," *Journal of Artificial Intelligence Research*, vol. 20, pp. 61–124, December 2003.

[47] A. Gerevini and D. Long, "Plan constraints and preferences in PDDL3," University of Brescia and University of Strathclyde, Tech. Rep., August 2005.

[48] M. E. desJardins, E. H. Durfee, C. L. O. Jr., and M. J. Wolverton, "A survey of research in distributed, continual planning," *AI Magazine*, vol. 20, no. 4, pp. 13–22, 2000.

[49] P. Patrón and Y. Petillot, "The underwater environment: a challenge for planning," in *Proceedings of the 27th Workshop of the UK Planning and Scheduling Special Interest Group*, 2008.

[50] N. A. Johnson and D. Lane, "Abstracting the planner down down: An architecture for planner based control of autonomous vehicles," in *Proceedings of the 27th Workshop of the UK Planning and Scheduling Special Interest Group*, 2008.

[51] P. Ridao, J. Yuh, J. Batlle, and K. Sugihar, "On AUV control architecture," in *Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000)*, vol. 2, 2000, pp. 855–860.

[52] P. Patrón, E. Miguelanez, Y. R. Petillot, D. M. Lane, and J. Salvi, "Adaptive mission plan diagnosis and repair for fault recovery in autonomous underwater vehicles," in *Proceedings of the IEEE Oceans Conference 2008*, 2008.

[53] P. Patrón, D. M. Lane, and Y. Petillot, "Interoperability of agent capabilities for autonomous knowledge acquisition and decision making in unmanned platforms," in *Proceedings of the IEEE Oceans Conference 2009 - Europe*, 2009.

[54] I. Woodrow, C. Purry, A. Mawby, and J. Goodwin, "Autonomous AUV mission planning and replanning - towards true autonomy," in *Proceedings of the 14th International Symposium on Unmanned Untethered Submersible Technolog*, August 2005.

[55] K. Collins, "Untethered AUV's can reduce costs for offshore inspection jobs," in *Proceedings of the 1993 Offshore Technology Conference*, May 1993.

[56] V. Chapman, S. Appleyard, and L. Thomas, "Commander trust in autonomous systems: the role of implicit instructions and system feedback," in *Proceedings of the 1st Annual SEAS DTC Technical Conference.* QinetiQ Centre for Human Sciences, July 2006.

[57] J. L. Hinton, M. Williams, and B. Kirby, "Human centred perspecivies on mission planning for autonomous systems," in *Proceedings of the 1st Annual SEAS DTC Technical Conference.* BAE Systems Advanced Technology Centre, July 2006.

[58] N. Johnson, P. Patron, and D. Lane, "The importance of trust between operator and AUV: Crossing the human/computer language barrier," in *Proceedings of the IEEE Oceans Conference 2007 - Europe*, June 2007.

[59] SeeByte Ltd., "Seetrack military," Software. Online brochure available at http://www.seebyte.com/Products/Military/index.html. Last checked on 26/11/2009.

[60] Hydroid Inc., "Remus vehicle interface program." Software. Online brochure available at http://www.hydroid.com/software.html. Last checked 26/11/2009.

[61] Hafmynd Efh., "Gavia contol center," Software. Online brochure available at: http://www.gavia.is/products/user_software.html. Last checked on 26/11/2009.

[62] Autonomous Solutions Inc., "Mobius command and control," Software. Online brochure available at http://autonomoussolutions.com/brochure/mobius.pdf. Last checked on 10/04/2010.

[63] B. C. Davis, P. Patrón, M. Arredondo, and D. M. Lane, "Augmented reality and data fusion techniques for enhanced situational awareness of the underwater domain," in *Proceedings of the IEEE Oceans Conference 2007 - Europe*, 2007.

[64] B. C. Davis, *Underwater Vehicles.* IN-TECH, 2008, ch. Enhanced Testing of Autonomous Underwater Vehicles using Augmented Reality & JavaBeans, pp. 77–96.

[65] S. Banks and C. Lizza, "Pilot's associate: a cooperative, knowledge-based system application," *IEEE Expert*, vol. 6, no. 3, pp. 18 – 29, June 1991.

[66] C. A. Miller and M. D. Hannen, "User acceptance of an intelligent user interface: a rotorcraft pilot's associate example," in *Proceedings of the 4th international conference on Intelligent user interfaces.* ACM Press, New York, NY, USA, 1998, pp. 109 – 116.

[67] C. A. Miller, "Bridging the information transfer gap: Measuring goodness of information fit," *Journal of Visual Languages & Computing*, vol. 10, no. 5, pp. 523–558, October 1999.

[68] C. C. Sotzing, N. Johnson, and D. M. Lane, "Triton: A system for enhanced operator awareness during autonomous underwater missions with minimal communication," Presentation given at the 2009 SEAS DTC Conference. Published online at: http://www.seasdtc.com/events/2009_conference/downloads/pdf/communications_and_control/CC008_presentation.pdf, 2009.

[69] H. Kroemer, "Quasi-electric fields and band offsets: Teaching electrons new tricks," Nobel Lecture. Retrieved online at: www.nobelprize.org/nobel_prizes/physics/laureates/2000/kroemer-lecture.pdf on 18/12/2009, November 2000.

[70] D. A. Norman, *The Design of Everyday Things*, 3rd ed. The MIT Press, October 1998.

[71] A. Dix, J. Finlay, G. D. Abowd, and R. Beale, *Human Computer Interaction*, 2nd ed. Prentice Hall, September 2003.

[72] C. A. Miller, "Rules of etiquette, or how a mannerly AUI should comport itself to gain social acceptance and be perceived as gracious and well-behaved in polite society." in *Working Notes of the AAAI-Spring Symposium on Adaptive User Interfaces*, March 2000, pp. 80–81.

[73] *Apple Human Interface Guidelines*, Apple Computer, Inc., 1 Infinite Loop, Cupertino, CA 95014, February 2006.

[74] Entertainment Software Association, "2009 sales, demographic and usage data: essential facts about the computer and video games industry," Retreived online at http://www.theesa.com/facts/pdfs/ESA_EF_2009.pdf on 19/12/2009, 2009.

[75] S. E. Siwek, "Video games in the 21st century: Economic contributions of the us entertainment software industry," Retreived online at http://www.theesa.com/facts/pdfs/VideoGames21stCentury.pdf on 19/12/2009, November 2007.

[76] Entertainment and Leisure Software Publishers Association, "Computer and video games: important facts at your fingertips," Retreived online at http://www.elspa.com/docs/Fact_Card_01.pdf on 19/12/2009, 2004.

[77] *Homeworld 2*, Computer Game, Sierra Entertainment Incorperated, 2003.

[78] R. Friedhoff and T. Kiely, "The eye of the beholder," *Computer Graphics World*, vol. 13, no. 8, August 1990.

[79] B. McCormick, T. DeFanti, and M. Brown, "Visualization in scientific computing," *Computer Graphics*, vol. 21, no. 6, November 1987.

[80] K. Kundu, C. Sessions, M. desJardins, and P. Rheingans, "Three-dimensional visualization of hierarchical task network plans," in *Proceedings of the Third International NASA Workshop on Planning and Scheduling for Space*, Houston, Texas, October 2002.

[81] D. E. Wilkins, *Practical Planning : Extending the Classical AI Planning Paradigm.* Morgan Kaufmann., 1998.

[82] M. de Weerdt, "Plan merging in multi-agent systems," Ph.D. dissertation, Delft University of Technology, 2003.

[83] R. van der Krogt and M. de Weerdt, "Self-interested planning agents using plan repair," in *Proceedings of the ICAPS 2005 Workshop on Multiagent Planning and Scheduling*, 2005, pp. 36–44.

[84] G. G. Robertson, J. D. Mackinlay, and S. K. Card, "Cone trees: animated 3D visualizations of hierarchical information," in *Proceedings of the SIGCHI conference on Human factors in computing systems: Reaching through technology*, 1991, pp. 189 – 194.

[85] K. Yee and D. Fisher, "Animated exploration of dynamic graphs with radial layout," in *Proceedings of the 2001 Symposium on IEEE Information Visualization*, 2001.

[86] D. Jurafsky and J. H. Martin, *Speech and Language Processing.* Prentice Hall, 2000.

[87] J. Weizenbaum, "Eliza—a computer program for the study of natural language communication between man and machine," *Communications of the ACM*, vol. 9, no. 1, pp. 36 – 45, January 1966.

[88] K. V. Deemter, E. Krahmer, and M. Theune, “Real versus template-based natural language generation: A false opposition?” *Computational Linguistics*, vol. 31, no. 1, pp. 15–24, March 2005.

[89] E. Reiter and R. Dale, *Building Natural-Language Generation Systems*. Cambridge University Press, 2000.

[90] E. Reiter, “SimpleNLG package,” Web Page : http://www.csd.abdn.ac.uk/ ereiter/simplenlg/. Last Checked: 1/12/2008.

[91] A. Gatt, F. Portet, E. Reiter, J. Hunter, S. Mahamood, W. Moncur, and S. Sripada, “From data to text in the neonatal intensive care unit: Using NLG technology for decision support and information management.” *AI Communications*, vol. 22, pp. 153–186, 2009.

[92] J. Hunter, Y. Freer, A. Gatt, R. Logie, N. McIntosh, M. van der Meulen, F. Portet, E. Reiter, S. Sripada, and C. Sykes, “Summarising complex ICU data in natural language.” in *AMIA 2008 Annual Symposium Proceedings*, 2008, pp. 323–327.

[93] J. Hunter, A. Gatt, F. Portet, E. Reiter, and S. Sripada, “Using natural language generation technology to improve information flows in intensive care units.” in *Proceedings of the 5th Conference on Prestigious Applications of Intelligent Systems*, 2008.

[94] E. Reiter, A. Gatt, F. Portet, and M. ven der Meulen, “The importance of narrative and other lessons from an evaluation of an NLG system that summarises clinical data.” in *Proceedings of the 5th International Conference on Natural Language Generation.*, 2008.

[95] S. Nowson, “Being john motson: Towards a computation model of football commentary,” Master’s thesis, University of Edinburgh, 2001.

[96] M. O’Donnell, C. Mellish, J. Oberlander, and A. Knott, “ILEX: an architecture for a dynamic hypertext generation system,” *Natural Language Engineering*, vol. 7, no. 3, pp. 225–250, September 2001.

[97] A. C. Graesser, K. Hauft-Smith, A. D. Cohen, and L. D. Pyles, “Advanced outlines, familiarity and text genre on retention of prose,” *Journal of Experimental Education*, vol. 48, 1980.

[98] N. Pennington and R. Hastie, “Evidence evaluation in complex decision making.” *Journal of Personality & Social Psychology*, vol. 51, no. 2, pp. 242–258, August 1986.

[99] ——, “Explanation-based decision making: Effects of memory structure on judgment.” *Journal of Experimental Psychology: Learning, Memory, & Cognition*, vol. 14, no. 3, pp. 521–533, July 1988.

[100] ——, “Explaining the evidence: Tests of the story model for juror decision making,” *Journal of Personality and Social Psychology*, vol. 62, no. 2, pp. 189–206, 1992.

[101] ——, “Reasoning in explanation-based decision making,” *Cognition*, vol. 49, pp. 123–163, 1993.

[102] R. M. Pirsig, *Zen and the Art of Motorcycle Maintenance: An Inquiry into Values.* William Morrow & Company, 1974.

[103] ——, *Lila: An Inquiry into Morals.* Bantam Books, 1991.

[104] N. Stephenson, *Cryptonomicon.* Avon, 1999.

[105] V. Propp, *Morphology of the Folk Tale.* University of Texas Press, 1968.

[106] A. C. Graesser, B. Olde, and B. Klettke, *Narrative Impact: Social and Cognitive Foundations*, 1st ed. Lawrence Erlbaum, July 2002, ch. How does the mind construct and represent stories?

[107] R. A. Zwaan and G. A. Radvansky, “Situation models in language comprehension and memory,” *Psychological Bulletin*, vol. 123, no. 2, pp. 162–185, 1998.

[108] K. Ehrlich and P. N. Johnson-Laird, “Spatial descriptions and referential continuity,” *Journal of Verbal Learning and Verbal Behavior*, vol. 21, pp. 296–306, 1982.

[109] M. Singer, M. Halldorson, J. C. Lear, and P. Andrusiak, “Validation of causal bridging inferences in discourse understanding,” *Journal of Memory and Language*, vol. 31, no. 4, pp. 507–524, August 1992.

[110] R. A. Zwaan, J. P. Magliano, and A. C. Graesser, “Dimensions of situation model construction in narrative comprehension.” *Journal of Experimental Psychology: Learning, Memory, and Cognition*, vol. 21, pp. 386–397, 1995.

[111] R. A. Zwaan, “Processing narrative time shifts,” *Journal of Experimental Psychology: Learning, Memory, and Cognition*, vol. 22, no. 5, pp. 1196–1207, September 1996.

[112] C. Crawford, *Chris Crawford on Interactive Storytelling.* New Riders, October 2004.

[113] "Game developers conference," Official Website: http://www.gdconf.com/ (Last check 15/12/2009).

[114] C. Crawford, "The journal of computer game design," Online archive: http://www.erasmatazz.com/Library.html last checked: 12/04/2010.

[115] M. Cavazza and D. Pizzi, "Narratology for interactive storytelling: A critical introduction," *Technologies for Interactive Digital Storytelling and Entertainment*, pp. 72–83, 2006.

[116] J. Niehaus and R. M. Young, "Towards improving recall and comprehension in automatically generated narratives," in *Working Notes of the Workshop on Narrative Learning Environments at the Twelfth International Conference on Artificial Ingelligence and Education*, July 2005.

[117] J. M. Thomas and R. M. Young, "Becoming scientists: Employing adaptive interactive narrative to guided discovery learning," in *Proceedings of the AIED 2007 Workshop on Narrative Learning Environments*, 2007.

[118] Y. Cheong and R. M. Young, "A framework for summarizing game experiences as narratives," in *Proceedings of the Second Conference on Artificial Intelligence and Interactive Digital Entertainment.* Liquid Narrative Group, 2006.

[119] Y. Cheong, A. Jhala, B. Bae, and R. M. Young, "Automatically generating summary visualizations from game logs," in *Proceedings of the Fourth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2008.

[120] Y. Chen, B. W. Wah, and C. Hsu, "Temporal planning using subgoal partitioning and resolution in SGPlan," *Journal of Artificial Intelligence Research*, vol. 26, pp. 323–369, August 2006.

[121] J. Hoffmann, "The Metric-FF planning system: Translating 'ignoring delete lists' to numeric state variables," *Journal of Artificial Intelligence Research*, vol. 20, pp. 291–341, December 2003.

[122] J. Hoffmann and B. Nebel, "The FF planning system: Fast plan generation through heuristic search," *Journal of Artificial Intelligence Research*, vol. 14, pp. 253–302, May 2001.

[123] C. C. Sotzing, J. Evans, and D. M. Lane, "A multi-agent architecture to increase coordination efficiency in multi-auv operations," in *Proceedings of the IEEE Oceans Conference 2007 - Europe*, June 2007.

[124] G2One, Inc., "Groovy dynamic scripting language for the java virtual machine," Web Page : http://groovy.codehaus.org/. Last Checked: 18/11/2008, 2008.

[125] K. Hamilton, D. Lane, K. E. Brown, J. Evans, and N. K. Taylor, "An integrated diagnostic architecture for autonomous underwater vehicles," *Journal of Field Robotics*, vol. 24, no. 6, pp. 497 – 526, June 2007.

[126] jME Development Team, "Java Monkey Engine," Web Page : http://www.jmonkeyengine.org/. Last Checked: 31/01/2011.

[127] L. D. Team, "The Lightweight Java Game Library," Web Page : http://lwjgl.org/. Last Checked: 1/12/2008.

[128] Khronos Group, Inc., "OpenGL - the industry standard for high performance graphics," Web Page : http://www.opengl.org/. Last Checked: 1/12/2008.

[129] FengGUI Development Team, "FengGUI: Java GUIs with OpenGL," Web Page: http://www.fenggui.org/doku.php. Last Checked: 1/12/2008.

[130] C. Haase, "Java Timing Framework," Web Page : https://timingframework.dev.java.net/. Last Checked: 1/12/2008.

[131] AT&T Labs, "GraphViz - Graph Visualization Software," Web Page : http://www.graphviz.org/. Last Checked: 1/12/2008.

[132] C. C. Sotzing and D. M. Lane, "Improving the coordination efficiency of multiple auv operations using prediction of intent," in *Proceedings of the 3rd SEAS DTC Technical Conference*, 2008.

[133] N. Biggs, E. LLoyd, and R. Wilson, *Graph Theory 1736-1936.* Clarendon Press, 1976.

[134] D. L. Applegate, R. E. Bixby, V. Chvátal, and W. J. Cook, *The Traveling Salesman Problem: A Computational Study.* Princeton University Press, 2006.

[135] A. H. Land and A. Doig, "An automatic method of solving discrete programming problems," *Econometrica*, vol. 28, no. 3, pp. 497–520, July 1960.

[136] M. Fischetti and P. Toth, "An additive bounding procedure for the asymmetric travelling salesman problem," *Mathematical Programming*, vol. 53, January 1992.

[137] G. Carpaneto, M. Dell'Amico, and P. Toth, "Exact solution of large-scale, asymmetric traveling salesman problems," *ACM Transactions on Mathematical Software*, vol. 21, no. 4, pp. 394 – 409, December 1995.

[138] M. Fischetti and P. Toth, "A polyhedral approach to the asymmetric traveling salesman problem," *Management Science*, vol. 43, no. 11, pp. 1520–1536, November 1997.

[139] N. Ascheuer, M. Jünger, and G. Reinelt, "A branch & cut algorithm for the asymmetric traveling salesman problem with precedence constraints," 1998.

[140] G. Dantzig, R. Fulkerson, and S. Johnson, "Solution of a large-scale traveling-salesman problem," *Journal of the Operations Research Society of America*, vol. 2, no. 4, pp. 393–410, November 1954.

[141] D. L. Applegate, R. E. Bixby, V. Chvátal, and W. J. Cook, "Implementing the dantzig-fulkerson-johnson algorithm for large traveling salesman problems," *Mathematical Programming*, vol. 97, pp. 91–153, 2003.

[142] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, second edition ed. MIT Press and McGraw-Hill, 2001, ch. 16 "Greedy Algorithms".

[143] G. Gutin, A. Yeo, and A. Zverovich, "Traveling salesman should not be greedy: domination analysis of greedy-type heuristics for the TSP." *Discrete Applied Mathematics*, vol. 117, pp. 81–86, 2002.

[144] J. Bang-Jensen, G. Gutin, and A. Yeo, "When the greedy algorithm fails," *Discrete Optimization*, vol. 1, pp. 121–127, 2004.

[145] G. Bendall and F. Margot, "Greedy type resistance of combinatorial problems," *Discrete Optimization*, vol. 3, pp. 288–298, 2006.

[146] M. Dorigo and L. M. Gambardella, "Ant colonies for the travelling salesman problem," *Biosystems*, vol. 43, no. 2, pp. 73–81, July 1997.

[147] L. Buriol, P. Franca, and P. Moscato, "A new memetic algorithm for the asymmetric traveling salesman problem," 1999.

[148] C. Moon, J. Kim, G. Choi, and Y. Seo, "An efficient genetic algorithm for the traveling salesman problem with precedence constraints," *European Journal of Operational Research*, vol. 140, no. 3, pp. 606–617, August 2001.

[149] I.-C. Choi, S.-I. Kim, and H.-S. Kim, "A genetic algorithm with a mixed region search for the asymmetric traveling salesman problem," *Computers & Operations Research*, vol. 30, no. 5, pp. 773–786, April 2003.

[150] L. V. Snyder and M. S. Daskin, "A random-key genetic algorithm for the generalized traveling salesman problem," Department of Industrial Engineering and Management Sciences, Northwestern University, Tech. Rep., 2004.

[151] M. Mitchell, *An Introduction to Genetic Algorithms*, ser. Complex Adaptive Systems. Cambridge, Massachusetts: The MIT Press, February 1998.

[152] M. Buckland, *AI Techniques for Game Programming*, 1st ed., ser. The Premier Press Game Development Series. Cincinnati, Ohio: Premier Press, October 2002.

[153] T. Ong and J. J. Leggett, "A genetic algorithm approach to interactive narrative generation," in *Proceedings of the fifteenth ACM conference on Hypertext and hypermedia*, 2004, pp. 181 – 182.

[154] C. Mellish, A. Knott, J. Oberlander, and M. O'Donnell, "Experiments using stochastic search for text planning," in *Proceedings of International Conference on Natural Language Generation*, 1998.

[155] W. M. Spears and K. A. DeJong, "On the virtues of parameterized uniform crossover," in *In Proceedings of the Fourth International Conference on Genetic Algorithms*, 1991, pp. 230–236.

[156] D. Levine, "Application of a hybrid genetic algorithm to airline crew scheduling," *Computers & Operations Research*, vol. 23, no. 6, p. 1996, 1996.

[157] M. A. Walker, "Redundancy in collaborative dialogue," in *Proceedings of the 14th conference on Computational linguistics*, vol. 1, 1992.

[158] P. C. Gordon, B. J. Grosz, and L. A. Gilliom, "Pronouns, names, and the centering of attention in discourse," *Cognitive Science*, vol. 17, no. 3, pp. 311–347, 1993.

[159] Sun Microsystems Laboratories Speech Integration Group, "FreeTTS 1.2 - a speech synthesizer written entirely in the java programming language," Published online at: http://freetts.sourceforge.net/docs/index.php Last checked 14/04/2010.

[160] The Internet Movie Database, "Requiem for a dream," Online: http://www.imdb.com/title/tt0180093/ (Last Checked 21/12/2009), 2000.

[161] B. Falk, "Review of requiem for a dream," Published online at BBC.co.uk: http://www.bbc.co.uk/films/2001/01/16/requiem_for_a_dream_2001 _review.shtml (Last Checked: 21/12/2009), August 2001.

[162] P. Tatara, "Just say no to 'requiem for a dream'," Published online at CNN.com: http://edition.cnn.com/2000/SHOWBIZ/Movies/10/09/ review.requiem.dream/index.html (Last Checked: 21/12/2009), October 2000.

[163] S. M. Rhodes, D. Murphy, and P. J. B. Hancock, “Developmental changes in the engagement of episodic retrieval processes and their relationship with working memory during the period of middle childhood,” *British Journal of Developmental Psychology*, 2011.

[164] N. Karamanis, “Entity coherence for descriptive text structuring,” Ph.D. dissertation, University of Edinburgh, 2004.

[165] N. Karamanis and H. M. Manurung, “Stochastic text structuring using the principle of continuity,” in *Proceedings of INLG*, 2002.

[166] N. Karamanis, C. Mellish, M. Poesio, and J. Oberlander, “Evaluating centering for information ordering using corpora,” in *Computational Linguistics*, vol. 35, March 2009, pp. 29–46.

[167] E. Althaus, N. Karamanis, and A. Koller, “Computing locally coherent discourses,” in *ACL '04 Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, 2004.

[168] M. Lapata, “Automatic evaluation of information ordering: Kendall's tau,” in *Computational Linguistics*, vol. 32, no. 4, 2006, pp. 471–282.

[169] Sun Microsystems, “Java,” Web Page: http://www.java.com/en/. Last Checked: 1/12/2008, December 2008.

[170] *Java Development Kit 5.0 Documentation*, 1st ed., Sun Microsystems.

[171] J. Gosling and H. McGilton, “The java language environment,” Sun Microsystems, White Paper, May 1996.

[172] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, and F. Yergeau, “Extensible markup language (XML) 1.0 (third edition),” World Wide Web Consortium (W3C), Tech. Rep., February 2004, retreived from: http://www.w3.org/TR/2004/RE_xml_20040204/.

[173] W3C Consortium, “Owl web ontology language overview,” Published online at : http://www.w3.org/TR/owl-features/ Last checked 14/04/2010.

[174] ——, “Resource description framework (RDF),” Published online at: http://www.w3.org/RDF/ Last checked 14/04/2010.

[175] ——, “SPARQL Query Language for RDF,” Published online at: http://www.w3.org/TR/rdf-sparql-query/ Last checked 14/04/2010.

[176] I. G. Tollis, G. D. Battista, P. Eades, and R. Tamassia, *Graph Drawing: Algorithms for the Visualization of Graphs.* Prentice Hall, August 1998.

[177] Stanford Center for Biomedical Informatics Research, “Protégé ontology editor and knowledge-base framework,” Published online at: http://protege.stanford.edu/ Last checked 14/04/2010.

[178] Eclipse Foundation, “Eclipse integrated development environment,” Published online at: http://www.eclipse.org/ Last checked 14/04/2010.

[179] Jena Development Community, “Jena - a semantic web framework for java,” Published online at: http://jena.sourceforge.net/ Last checked 14/04/2010.

[180] ——, “ARQ - A SPARQL Processor for Jena,” Published online at: http://jena.sourceforge.net/ARQ/ Last checked 14/04/2010.

[181] Clark & Parsia LLC, “Pellet: OWL 2 Reasoner for Java,” Published online at: http://clarkparsia.com/pellet Last checked 14/04/2010.

[182] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz, “Pellet: A practical OWL-DL reasoner,” *Journal of Web Semantics*, vol. 5, no. 2, 2007.