# Remote Access to a Prototyping Laboratory

## Thesis

# Remote Access to a Prototyping Laboratory

**Stephen Lathwell BSc (Hons), MSc (Optoelectronics)**

A thesis submitted for the degree of

**DOCTOR OF PHILOSOPHY**

of the Open University

**August 2007**

Department of Design and Innovation

The Open University

Walton Hall

Milton Keynes

EX12

RESEARCH SCHOOL

## Library Authorisation Form

Please return this form to the Research School with the two bound copies of your thesis to be deposited with the University Library. All candidates should complete parts one and two of the form. Part three only applies to PhD candidates.

### Part One: Candidates Details

Name: STEPHEN LATHWELL PI: M72.037.64

Degree: DOCTOR OF PHILOSOPHY

Thesis title: REMOTE ACCESS TO A PROTOTYPING LABORATORIES

### Part Two: Open University Library Authorisation

I confirm that I am willing for my thesis to be made available to readers by The Open University Library, and that it may be photocopied, subject to the discretion of the Librarian.

Signed: S.P. Lathwell Date: 5-XI-2k7

### Part Three: British Library Authorisation [PhD candidates only]

If you want a copy of your PhD thesis to be available on loan to the British Library Thesis Service as and when it is requested, you must sign a British Library Doctoral Thesis Agreement Form. Please return it to the Research School with this form. The British Library will publicise the details of your thesis and may request a copy on loan from the University Library. Information on the presentation of the thesis is given in the Agreement Form.

Please note the British Library have requested that theses should be printed on one side only to enable them to produce a clear microfilm. The Open University Library sends the fully bound copy of theses to the British Library.

The University has agreed that your participation in the British Library Thesis Service should be voluntary. Please tick either (a) or (b) to indicate your intentions.

(a) ☑ I am willing for The Open University to loan the British Library a copy of my thesis. A signed Agreement Form is attached

(b) ☐ I do not wish The Open University to loan the British Library a copy of my thesis.

Signed: S.P. Lathwell Date: 5-XI-2k7

# Abstract

There is a growing global demand for continuing adult higher education particularly in science and engineering subjects. New technologies are emerging which would enable the development of a Remote Access Laboratory for rapid prototyping of Artificial Intelligence, as a learning environment for mechatronic engineering, in which high precision electromechanical devices are designed to exhibit autonomous behaviour.

Secondary research investigated the learning theories for a Remote Access Laboratory, and the current practices for distance learning, involving groupware in shared activity 'collaboratories'. Having determined that the laboratory would need a multi-user interactive environment architecture, with the requirement for adaptability to rapid developments, a distributed software architecture was selected. The laboratory design was subsequently argued to be best served by Intelligent Agents in a Multi-Agent system.

The aims of the research were to establish the viability of a Remote Access Laboratory for mechatronic experimentation, and to evaluate the technologies required to implement such a laboratory environment for rapid prototyping. These were achieved by developing a novel user interface, based on a multi-functional screen layout, and a graphical specification facility to provide robotic navigation that is intuitive to use and does not require text-based programming.

The research investigated the prototyping of robotic behaviour, which used Programming by Demonstration as an innovative technique to prototype robot navigation. The method of designing behaviours met an anticipated need to allow the robot to interact with an environment, to achieve goals under conditions of uncertainty, while requiring a level of abstraction in the behaviour design. The interface structured a composite of the designed behaviours into prototype Artificial Intelligence using a hierarchical behaviour architecture, which complied with the principles of Object Orientated programming. This was subsequently a new and original programming method to facilitate rapid prototyping of Artificial Intelligence design and structuring.

Experimentation involved 20 participants attempting to accomplish a series of tasks which involved using the prototyped interface and an existing text-based robot programming system. The participants were profiled by their formal qualifications, knowledge and experience. The experimental data obtained were used to establish a comparative measure of the prototype interface success compared with an existing distance-learning, home experiment kit, in the form of a small controllable model vehicle. The data obtained provided strong evidence to support the hypothesis that a Programming by Demonstration based system for rapid prototyping is more flexible and easier to use than a previously existing distance learning text-based system. The Programming by Demonstration system showed great promise, being quicker for prototyping, and more intuitive. The learning interface design pioneered new techniques and technologies for rapid prototyping of Artificial Intelligence in a Mechatronics Remote Access Laboratory.

# Acknowledgements

Particular thanks are due for:

My supervisor Dr. Anthony Lucas-Smith, who taught me the difference between background knowledge, and a critical review of the literature, and fortunately had a sense of humour and great patience.

Professor George Rzevski, who gave me the initial idea for the PhD.

Claire ('Squeekie') who helped by listening, transcribing the experimental recordings, cooking the food, and never having any doubts.

Professor Stephen Potter, who kindly gave valuable advice and recommendations.

The 20 participants who all generously gave their valuable time to do the experimentation.

Richard Hearne who managed to combine 2 video streams of recorded data onto a CD in less than 24 hours, on 20 occasions.

George Bellis who supplied technical advice for the robots when needed.

The Lucas-Smith household who suffered the 10·30 - 11pm telephone calls, when I was seeking immediate, and probably not as important as it seemed at the time, advice.

Rodney Buckland who gave encouragement, and I still owe him a pint.

My parents who gave moral support during the hardships in my personal life, so I could continue with this, plus the holidays, just to make sure there were some.

My brother Nicholas, who made sure I got out occasionally, and tried to keep me informed about the world beyond my computer.

Claire's parents who kept Claire and me well fed.

# Contents


| | Page |
|---|---|
| Abstract | i |
| Acknowledgements | ii |
| Contents | iii |
| Figures | vii |
| Tables | ix |
| Glossary | xi |
| Chapter 1 **Introduction to Remote Access to Prototyping Laboratories** | 1 |
| 1.1 Context of the Research | 1 |
| 1.2 Research Origins | 2 |
| 1.3 Principle Research Issues | 5 |
| 1.3.1 Requirements of a Distance Learning Environment | 5 |
| 1.3.2 Adopting Technology in Distance Learning | 6 |
| 1.3.3 Defining the Problem | 7 |
| 1.4 Research Aims | 7 |
| 1.5 Research Questions | 8 |
| 1.6 Research Objectives | 8 |
| 1.7 New Knowledge | 9 |
| 1.8 Structure of the Thesis | 9 |
| Chapter 2 **State-of-the-Art for a Distance Learning Laboratory** | 12 |
| 2.1 Introduction | 12 |
| 2.2 Distance Learning Theories | 12 |
| 2.2.1 Introduction to Distance Learning Theories | 12 |
| 2.2.2 Principle Distance Learning Theories Considered | 15 |
| 2.2.2.1 The (Re)conceptualisation Cycle | 16 |
| 2.2.2.2 Experiential Learning | 16 |
| 2.2.2.3 Social-Constructivist Theory | 17 |
| 2.2.2.4 The Centre for Advanced Learning Technologies (CALT) | 18 |
| 2.2.3 Conclusions about Distance Learning Theories | 19 |
| 2.3 Distance Learning Engineering Laboratories | 20 |
| 2.3.1 The Expectations of Engineering Laboratories | 21 |
| 2.3.2 Engineering Laboratory Theory | 21 |
| 2.3.2.1 Objectives of an Engineering Instructional Laboratory | 21 |
| 2.3.2.2 Types of Engineering Laboratories | 22 |
| 2.3.2.3 Distance Learning Engineering Laboratory Design | 23 |
| 2.3.3 Access to Distance Learning Laboratories | 24 |
| 2.3.4 Laboratory Architectures | 24 |
| 2.3.5 Conclusions about Distance Learning Engineering Laboratories | 26 |
| 2.4 Collaboratories | 27 |
| 2.4.1 Principles of Collaboratories | 27 |
| 2.4.2 Examples of Collaboratories | 27 |
| 2.4.2.1 Virtual Collaborative Environment (VCE) | 27 |
| 2.4.2.2 Distributed Collaboratory Experimental Environment (DCEE) | 28 |
| 2.4.2.3 A Virtual Training Laboratory at Queen Mary College | 29 |

| | Page |
|---|---|
| 2.4.3 Summary of Collaboratories | 30 |
| 2.5 A Distance Learning Laboratory Architecture | 31 |
| 2.5.1 Role of Intelligent Agents | 31 |
| 2.5.2 Role of Blackboard Agents | 33 |
| 2.5.2.1 The Blackboard Agent Architecture Components | 34 |
| 2.5.2.2 Examples of Blackboard Agent Implementation | 35 |
| 2.5.2.3 Summary of Blackboard Agent Architectures | 37 |
| 2.5.3 Specific Agent Architectures | 38 |
| 2.5.3.1 An Agent Architecture for Tracking Other Agents | 38 |
| 2.5.3.2 A Knowledge Base Agent Architecture | 38 |
| 2.5.3.3 Multi Agent Architectures | 39 |
| 2.5.3.4 Summary of Specific Agent Architectures | 41 |
| 2.6 Conclusions about a Distance Learning Laboratory | 42 |
| Chapter 3 **State-of-the-Art for an Interface to a Distance Learning Laboratory** | 44 |
| 3.1 Introduction | 44 |
| 3.2 A Distance Learning Environment | 44 |
| 3.2.1 Intelligent Tutoring Systems (ITS) | 45 |
| 3.2.2 Design Considerations | 46 |
| 3.2.3 Summary of a Distance Learning Environment | 47 |
| 3.3 Human Computer Interfacing | 47 |
| 3.3.1 The Theories of Human Computer Interfacing | 48 |
| 3.3.1.1 Distributed Cognition | 49 |
| 3.3.1.2 Activity Theory | 50 |
| 3.3.1.3 External Cognition | 52 |
| 3.3.2 HCI Design in Practice | 53 |
| 3.3.3 Summary of Human Computing Interfacing | 55 |
| 3.4 Programming by Demonstration | 56 |
| 3.4.1 Programming Methods | 57 |
| 3.4.1.1 Direct Programming | 57 |
| 3.4.1.2 Indirect Programming | 58 |
| 3.4.1.3 Learning by Human Demonstration | 59 |
| 3.4.2 Programming by Demonstration Methods | 60 |
| 3.4.3 Programming by Demonstration Systems | 62 |
| 3.4.4 Programming using Programming by Demonstration | 64 |
| 3.4.4.1 Writing Programs | 64 |
| 3.4.4.2 Reading Programs | 65 |
| 3.4.4.3 Executing Programs | 65 |
| 3.4.5 The Limitations of Programming by Demonstration | 66 |
| 3.4.6 Programming Sub-Optimality | 66 |
| 3.4.7 Summary of Programming by Demonstration | 67 |
| 3.5 Conclusions for an Interface to a Remote Access Laboratory | 68 |
| Chapter 4 **A Proposed Architecture for a Distance Learning Laboratory** | 70 |
| 4.1 A Laboratory in Distance Learning | 70 |
| 4.2 A Proposed Laboratory Architecture | 71 |

| | Page |
|---|---|
| 4.2.1 Influences on the Laboratory Architecture | 71 |
| 4.2.1.1 The Prototype Artificial Intelligence | 71 |
| 4.2.1.2 Communicating Intentions | 73 |
| 4.2.1.3 Interpreting Intentions | 73 |
| 4.2.2 The Conceptual Design | 74 |
| 4.2.2.1 The Mechatronic Device Control Agent | 75 |
| 4.2.2.2 The Program Administrator | 75 |
| 4.2.2.3 The Mechatronic Device | 76 |
| 4.2.2.4 The Knowledge-base Agent | 76 |
| 4.2.3 The Physical Design | 76 |
| 4.2.3.1 The Mechatronic Device Operator Agent | 77 |
| 4.2.3.2 The Book-Keeping Agent | 79 |
| 4.2.3.3 The Interface between the Laboratory and its User | 79 |
| 4.2.3.4 Communication between the Agents | 80 |
| 4.3 Conclusions about a Proposed Architecture for a Distance Learning Laboratory | 83 |
| Chapter 5 **Development of a Prototype Interface** | 85 |
| 5.1 Introduction | 85 |
| 5.1.1 The Design Proposals | 85 |
| 5.1.1.1 Distributed Cognition | 86 |
| 5.1.1.2 Activity Theory | 86 |
| 5.1.2 Anticipated Problems | 87 |
| 5.1.3 Principles of Prototyping | 87 |
| 5.2 Programming Language Development | 89 |
| 5.2.1 Programming Language Definition | 90 |
| 5.2.2 Programming Language: 'Vocabulary' | 91 |
| 5.2.3 Program Control Structures | 92 |
| 5.2.4 Programming Language: 'Functions' | 93 |
| 5.2.5 Principles of Object Orientated Programming | 94 |
| 5.3 The Prototype Interface Design | 94 |
| 5.3.1 The Human Computer Interface Methodologies | 94 |
| 5.3.2 Programming by Demonstration (PbD) | 102 |
| 5.3.3 Object Orientation | 103 |
| 5.3.4 Goal-Based Behaviours | 104 |
| 5.3.5 Sensor-Activated Behaviours | 106 |
| 5.3.6 Goal-Activated Behaviours | 107 |
| 5.3.7 Editing | 107 |
| 5.4 Simulation Operations | 108 |
| 5.4.1 Goal-Based Operations | 108 |
| 5.4.2 Sensor-Activated Operations | 109 |
| 5.4.3 Goal-Activated Operations | 111 |
| 5.5 Conclusions | 113 |
| Chapter 6 **Design of the Experiment** | 115 |
| 6.1 The Definition of an Experiment | 115 |
| 6.2 Scientific Method | 115 |
| | 115 |

| | Page |
|---|---|
| 6.3 Design of the Experiment | 116 |
| 6.3.1 Premises for Experiment Design | 117 |
| 6.3.2 Optimising the Experiment's Design | 119 |
| 6.3.3 Variables | 119 |
| 6.4 Experiment Data Analysis | 121 |
| 6.5 Error Analysis | 122 |
| 6.5.1 Accuracy, Precision and Tolerance | 122 |
| 6.5.1.1 Accuracy | 122 |
| 6.5.1.2 Precision | 123 |
| 6.5.1.3 Tolerance | 123 |
| 6.5.2 Experimental Errors | 124 |
| 6.5.3 Systemic Errors | 124 |
| 6.5.4 Random Errors | 125 |
| 6.5.5 Blunders | 126 |
| 6.6 Experimental Ethics | 126 |
| 6.6.1 Consent | 126 |
| 6.6.2 Responsibility | 127 |
| 6.6.3 Experimental Preparation | 128 |
| 6.6.4 Confidentiality | 128 |
| 6.7 Conclusions to Design of the Experiment | 129 |
| Chapter 7 **Experimental Procedures** | 131 |
| 7.1 Introduction | 131 |
| 7.2 The Aim of the Experiment | 132 |
| 7.2.1 The Objective of the Experiment | 132 |
| 7.3 The Variables | 133 |
| 7.3.1 Required Degree of Certainty 'In all things that are uncertain at the start' | 134 |
| 7.3.2 The Participants | 135 |
| 7.3.3 Components of Variation | 135 |
| 7.3.4 Randomisation | 135 |
| 7.3.5 Blocking | 136 |
| 7.3.6 The Independent Variables | 136 |
| 7.3.7 The Dependent Variables | 139 |
| 7.3.8 Maximising the Data Obtained | 140 |
| 7.3.9 Minimizing the Required Number of Experiments | 140 |
| 7.4 Experiment Activities | 140 |
| 7.5 Experimental Equipment | 141 |
| 7.5.1 The Tasks | 143 |
| 7.6 "Buy-in" of Results | 144 |
| 7.7 Conclusions of Experimental Procedures | 144 |
| Chapter 8 **Data Obtained and Interpretation** | 146 |
| 8.1 Introduction to Data Obtained and Interpretation | 146 |
| 8.2 The Participants | 146 |

| | Page |
|---|---|
| 8.3 Raw Data Gathered | 150 |
| 8.3.1 Comparison of Participants' Timed Activity | 150 |
| 8.3.2 Measures of Activity Success | 154 |
| 8.3.3 The Participants' Dialogue | 156 |
| 8.3.4 The Usability Questionnaire Results | 158 |
| 8.3.4.1 The PbD System Questions | 158 |
| 8.3.4.2 The RBS System Questions | 163 |
| 8.4 Comparative Usability of the Two Interfaces | 166 |
| 8.4.1 Repetitions of the Experiment | 166 |
| 8.4.2 The Like/Dislike Ratio from the Experiment Dialogue | 167 |
| 8.5 General Conclusions about the Experimentation | 168 |
| Chapter 9 **Conclusions** | 170 |
| 9.1 Conclusions to the Research Questions | 170 |
| 9.2 Conclusions to the Aims and Hypothesis | 173 |
| 9.3 Conclusions for Remote Access to Prototyping Laboratories | 174 |
| 9.4 Further Research | 176 |
| 9.5 Future Open-Learning Access to Online Laboratories | 178 |
| 9.6 Reflective Practice | 180 |
| 9.7 Implications of the Research | 183 |
| **References** | 184 |
| **Appendices** | 200 |
| Appendix A The Interface Design | 200 |
| Appendix B An Explanation of the Experiment | 204 |
| Appendix C Usability Questionnaire | 217 |
| Appendix D Experimentation Timings | 223 |

# Figures


| | | Page |
|---|---|---|
| 2.1 | A model of the change process, Razmerita *et al.* [2004] | 19 |
| 2.2 | The basic blackboard agent architecture | 33 |
| 2.3 | A blackboard agent architecture for the Remote Access Laboratory | 37 |
| 3.1 | '*Mediational Model*' Mwanza [2001] | 51 |
| 3.2 | The basic activity structure, '*Activity Triangle Model*' Mwanza [2001] | 51 |
| 4.1 | The user and laboratory interaction | 71 |
| 4.2 | The intelligent agents within the laboratory | 72 |
| 4.3 | The Mechatronic Device Operator Agent based on the blackboard agent architecture | 78 |
| 4.4 | Communications between the interface and the laboratory | 80 |
| 4.5 | Hierarchical and peer-to-peer communications | 81 |
| 4.6 | Multi-agent communications | 82 |
| 4.7 | The complete laboratory architecture | 83 |
| 5.1 | The outline of the interface in 3 segments: user input; program output and program manipulation | 95 |
| 5.2 | The Operations Map and associated buttons | 96 |
| 5.3 | The Operations Map and editing function | 97 |
| 5.4 | The user input and text output | 98 |
| 5.5 | Relation of sensor output to user input | 99 |
| 5.6 | The angle of incidence between the robot and obstacle, and activated sensors | 99 |
| 5.7 | Creating a mechatronic behaviour function | 100 |
| 5.8 | Manipulating a developed program | 101 |
| 5.9 | The means of activating the developed mechatronic program | 102 |
| 5.10 | The advantages of Object Orientated Programming for the Programming by Demonstration interface | 104 |
| 5.11 | Locating goals on the Operations Map | 105 |
| 5.12 | Identifying sub-goals | 105 |
| 5.13 | Demonstrating a sensor-Activated Behaviour | 106 |
| 5.14 | Demonstrating a goal-Activated Behaviour with sensor activity | 107 |
| 5.15 | The simulation of a goal-to-goal based behaviour | 109 |
| 5.16 | The search for Sensor-Activated Behaviours | 110 |
| 5.17 | The operation of a Sensor-Activated Behaviour | 110 |
| 5.18 | The recursive operation of a Sensor-Activated Behaviour | 111 |
| 5.19 | The operation of a Goal-Activated Behaviour | 111 |
| 5.20 | Locating a Sensor-Activated Behaviour relevant to a Goal-Activated Behaviour | 112 |
| 5.21 | A Sensor-Activated Behaviour for continuing a Goal-Activated Behaviour | 112 |
| 5.22 | The completed simulation of the demonstrated behaviour | 113 |

| | | Page |
|---|---|---|
| 7.1 | The layout of the experimentation laboratory | 142 |
| 7.2 | The physical layout of the laboratory | 142 |
| 7.3 | A close-up of the robotic vehicle | 143 |
| 8.1 | The distribution of the sampled participants | 148 |
| 8.2 | The participants' experiment timings | 149 |
| 8.3 | The timings for testing the PbD System by the PbD-first sample group | 151 |
| 8.4 | The timings for testing the PbD System by the RBS-first sample group | 151 |
| 8.5 | The timings for testing the RBS System by the PbD-first sample group | 152 |
| 8.6 | The timings for testing the RBS System by the RBS-first sample group | 153 |
| 8.7 | The mechatronic vehicle orientation for design of behaviours | 155 |
| A.1 | Goal-Based Behaviour | 201 |
| A.2 | Sensor-Activated Behaviour | 201 |
| A.3 | Determining sub-goals | 202 |
| A.4 | Goal-Activated Behaviours | 202 |
| A.5 | Operating simulator functions | 203 |
| A.6 | Editing designed robot behaviour | 203 |

# Tables


| | | Page |
|---|---|---|
| 2.1 | An outline of current advocated learning theories | 15 |
| 2.2 | Comparison of agent theories | 32 |
| 3.1 | Comparison of HCI design theories | 49 |
| 5.1 | The search pattern for determining a best fit of sensor activations to a Sensor-Activated Behaviour | 109 |
| 8.1 | A description of the participants | 147 |
| 8.2 | The PbD-first group's attempts to achieve success | 154 |
| 8.3 | The RBS-first group's attempts to achieve success | 155 |
| 8.4 | Participant dialogue during the experiments | 157 |
| 8.5 | Usability questionnaire responses for the PbD system | 159 |
| 8.6 | The suggested names for buttons on the PbD interface | 162 |
| 8.7 | Usability questionnaire responses for the RBS system | 164 |
| 8.8 | Repetitions of the experiment | 166 |
| 8.9 | The ratio of likes/dislikes | 167 |
| 8.10 | The evidential and stated preference of the participants | 168 |
| D.1 | The test of the PbD system by the PbD first sample group | 224 |
| D.2 | The test of the RBS system by the PbD-first sample group | 224 |
| D.3 | The test of the PbD system by the RBS-first sample group | 224 |
| D.4 | The test of the RBS system by the RBS-first sample group | 224 |

# Glossary

| | |
|---|---|
| Agent Architecture | The internal design of an Intelligent Agent |
| ALQAAA | A Linear Quasi-Autonomous Agent Architecture: an agent tracking architecture |
| ARCHON | An Intelligent Agent design methodology with a common communication platform |
| Artificial Intelligence (AI) | A program which can operate under conditions of uncertainty |
| ATLA | Alternatives to Laboratory Animals |
| BB1 | B. Hayes-Roth Blackboard Agent Architecture |
| Blackboard Agent | An Intelligent Agent which has a blackboard for centralised control |
| CALT | Centre for Advance Learning Technologies: The European Union's research into distance learning |
| Collaboratory | A software application for multiple users to collaborate |
| Cognitive Dimensions | An abstraction for describing features of a design in Psychology's External Cognition theory |
| COUGAAR | Cognitive Agent Architecture: A Distributed Agent Architecture |
| CTSEF | Central Texas Science and Engineering Fair |
| Data Stream Management | The software control of Input and Ouput data |
| DARPA | (US) Defence Advanced Research Projects Agency |
| DCEE | Distributed Collaboratory Experiment Environment: an integrated environment of product data and engineering tools |
| Distance Learning | Learning from a location geographically separate from the education provider |
| Distributed Agent Architecture | A form of Multi-Agent System where agents' knowledge overlaps |
| Distributed Intelligent System | A large software system with emergent intelligent behaviour |
| e-learning | Learning using education resources on the internet |
| ESRC | Engineering and Science Research Council, UK funding body for Research. |
| GAIA | A methodology for agent-orientated analysis and design |
| GOMS | Goals Operations Methods and Selection Rules. A method of designing a computer software interface |
| HCI | Human Computer Interface |
| HPMEC | Open University Human Participants and Material Ethics Committee |
| Intelligent Agents (IA) | Software which tries to fulfil a set of goals in a complex, dynamic environment |
| Intelligent Training Systems (ITS) | A software architecture intended for user learning |
| LabVIEW | A laboratory equipment simulator software package |

| | |
|---|---|
| Knowledge Source (KS) | A blackboard agent's data |
| MASCOT | Multi-Agent Supply Chain Coordination Tool: agent based software architecture design tool |
| MATLAB/SIMULINK | A laboratory equipment simulator software package |
| Mechatronics | A fusion of mechanics, electronics and intelligent control systems |
| MHP | Model Human Processor: a method of designing a computer software interface |
| MOYRA | Mechatronic Operations by Related Actions: the PbD system designed during this research |
| MSC VisualNastran 4D | A Laboratory equipment simulator software package |
| Multi-Agent System (MAS) | Multi-Agent System: a system comprising more than one agent |
| Programming by Demonstration (PbD) | A programming method based on acquiring human knowledge from observing human performance |
| Remote Access Laboratory | A laboratory which is geographically remote from its user, and accessed via the Internet |
| UML | Unified Modelling Language: a methodology for modelling Object Oriented Software |
| VCE | Virtual Collaborative Environment: Sandia's National Laboratory application to remotely program and control mechatronic devices |
| Z | A program specification language |

# Chapter 1
# Introduction to Remote Access to Prototyping Laboratories

## 1.1 Context of the Research

As concerns are now growing across the world about the increasing demand for adult higher education, universities and governments are investigating the potential of distance learning methods to meet this demand. The context of this investigation was to address the use of an internet-accessed laboratory as a component of a distance-learning course in mechatronics. Students have in the past undertaken experimentation in distance learning using an experiment kit at home, which was expensive. Reductions in costs of education are sought by replacing the home experiment kit with a remote access laboratory, either simulated or real, which could be used for all the experimentation, and accessed by an appropriate interface.

New technologies are emerging which would enable the operation of an autonomous and robust remote access laboratory. Developments in Intelligent Agents (IA), and subsequent knowledge manipulation technologies could be used to develop an autonomous remote access laboratory. The integration of these new technologies is now identified as an achievable task.

Within this environment the question addressed was how to develop a Remote Access Laboratory used to develop and test prototyped Artificial Intelligence (AI) in a coherent, cost-effective manner. This research anticipated a growing trend of distance learning in the 21st century, supported by autonomous laboratories, accessed over the Internet.

It sought to provide evidence for the viability of a remote access laboratory in the specific context of a distance learning course in mechatronics, where the student is designing prototype Artificial Intelligence for robots.

## 1.2 Research Origins

The initial speculative brief was as following:

> *This particular research is concerned with an experimental system for remote access to the Open University Mechatronics Laboratory. It is proposed to design the following system.*
>
> *An existing autonomous intelligent vehicle, connected by an infrared communication system to a laboratory computer which, in turn, is connected to the Internet, operates in a designated area of the laboratory. A number of video cameras will monitor the movement of the vehicle connected to the same laboratory computer.*
>
> *Remotely located designers of the vehicle's artificial mind will download their prototype Artificial Intelligence to the laboratory computer, observing its behaviour, in real time, by video signals sent from the laboratory cameras and displayed on their machines.*
>
> *The task will include the design of a remote access system according to the above description, testing the system in operation and making suggestions for its further improvement.*

Mechatronics, as the fusion of electronics and mechanics in the design of devices, and programming, referred to above as the '*artificial mind*', is required to demonstrate Artificial Intelligence when controlling the devices. The '*artificial mind*' program for the remainder of this thesis is called **prototype Artificial Intelligence** (AI), where AI is defined as a system's ability to achieve a goal or sustain desired behaviour under conditions of uncertainty.

The conclusions drawn from the brief, determined the initial course of research:

- The laboratory was to be a mechatronic prototyping laboratory for analysing prototype Artificial Intelligence using a mechatronic device.

- A laboratory would be remotely accessed, using the Internet, such that:

  (i) The user designs and sends prototype Artificial Intelligence to the laboratory for evaluation.

  (ii) The laboratory transmits a video signal in real time to the user.

- The laboratory user experiments with the prototype Artificial Intelligence to develop desired behaviour.

The initial speculative brief provided an introduction to the project. However, preliminary research revealed evidence that there were wider issues involved. There was a significant growth in adult learning, which was not being catered for by traditional learning establishments.

Mechatronics, as an engineering subject, has both theory and practice as important components. The majority of courses involve students attending physical laboratories as apprentices. The alternative is to use simulators, referred to as virtual laboratories, although there is some research about the use of real equipment remotely located at a distance.

The theory for a form of interface to the laboratory is not established, but closest are the general theories for human-computing interfaces and for Intelligent Training Systems. These reflect emerging learning theories, providing a tutoring interaction with the student. However, the problem will be the need for flexibility to enable prototype Artificial Intelligence development, while also allowing for analysis of the completed prototype Artificial Intelligence.

A problem with current robotic programming is the use of computer programming language methodologies. A method was sought for developing prototype Artificial Intelligence, which both remains within the established science of computer language development and allows students the freedom of development without the use of a formal text-based computer language. To test prototype Artificial Intelligence quickly for correctness in behaviour, a simplified simulator is considered necessary.

The research focus was on **Programming by Demonstration** as a software development method, to design prototype Artificial Intelligence. The underlying reasoning was that successful implementation of a Remote Access Laboratory was dependent on the user successfully designing prototype Artificial Intelligence, and that existing methods were not adequate for the task. Programming by Demonstration is described by Kaiser *et al.* [1995] as:

> *Two basic aspects of the interaction between the robot and the user.....Firstly, the user wants to configure and instruct the robot. This requires translating the user's language into the robot's, i.e., to compile user intentions into actual robot programs. Secondly, to allow the user to efficiently control and maintain the robot, necessitating translating low-level numerical representations used by the robot into an understandable form, i.e., symbols have to be built from signals. What is desired is to enable the robot to perform these tasks partly autonomously, i.e., to learn semantically meaningful descriptions of its own perceptions, actions, and states, and to use these descriptions both to communicate the robot's knowledge to the user and to interpret the user's demonstrations, i.e., to acquire human knowledge from observing human performance.*

The Programming by Demonstration based interface was to be supported by an Intelligent Agent system. Intelligent agents are described in Maes [1994] as:

> *An agent is a system that tries to fulfil a set of goals in a complex, dynamic environment: it can sense the environment, using its sensors and act upon the environment through its actuators. An agent's goals can take many different forms: they can be "end goals", or particular states the agent tries to achieve; they can be a selective reinforcement or reward that the agent attempts to maximise; they can be internal needs or motivations that the agent has to keep within certain viability zones and so on.*

## 1.3 Principle Research Issues

The overall research problem was to identify the functional requirements of a remote access mechatronics laboratory, possessing the potential for both distance learning, and the ability to develop and test a device possessing prototype Artificial Intelligence. The results were intended to answer the research questions developed in Section 1.5.

### 1.3.1 Requirements of a Distance Learning Environment

> *The terms "distance education" or "distance learning" have been applied interchangeably by many different researchers to a great variety of programs, providers, audiences, and media. Its hallmarks are the separation of teacher and learner in space and/or time, the volitional control of communication between student and the distant instructor, and non-contiguous communication between student and teacher, mediated by print or some form of technology.* [Sherry, 1996]

To develop a distance learning laboratory, the first requirement to consider is what the premises of a higher educations operation are.

Massy and Zemsky [1995] refers to higher education's belief in *its own purpose and educational and intellectual values.*

- *Traditional academic values*: teaching methods; notions of productivity; faculty autonomy, and standardised student-teacher ratios and class-sizes.
- *Productivity*: Most faculties think in terms of scholarship, especially research, and teaching is usually viewed as scholarship related.
- *Research*: incentives for teaching are few and research is significant, creating a '*academic ratchet*' a movement towards research production and reduced class loads.

Schamber [1988] proposes that:

> *It is essential to consider their ages, needs, cultural and socio-economic backgrounds, interests and experiences, education levels and familiarity with distance education methods and delivery systems, of the distance learners.*

When considering the anticipated student, Fjuk [1995] reported that

> *... The primary target group for most distance and open learning situations is the adult work force of our society. The student - the adult worker, usually with an established life with family and friends - needs a flexible (further) educational situation free from place and often time, constraint.*

The above outlined the environment required for distance learning, explaining the current understanding of what distance learning means to both student and higher education establishments; with a discussion of higher education expectations and operational circumstances, with the needs of expected distance learning students. These are important determinants for any development of distance learning technology, with the expectation of an academic value by the higher education establishments and flexibility in the provision for learning by the student.

### 1.3.2 Adopting Technology in Distance Learning

Archer *et al.* [1999] provides an analysis of future distance learning development and growth based on the Christensen [1997] book, *The Innovator's Dilemma: When New Technologies Cause Great Firms to Fail.* The development of distance learning support technologies are argued to be a significant change affecting traditional universities, which can either adopt the technology or lose potential students. Myers *et al.* [2004] reviews the debate about adopting learning technologies, citing Carlson [2000] that distance learning technologies are being adopted by a need for technologically literate students, with continuous development of new distance learning technologies.

Massey and Zemsky [1995] reports that distance learning technology is adopted for both the expected gains from use, and a facility's ability to successfully teach in a distance learning environment:

- **Economies of Scale:** After an initial investment, the increase in cost per additional student is usually low.
- **Mass Customisation:** The technology must allow the faculty to accommodate individual differences in students goals, learning styles, and abilities, while providing improved convenience for both students and faculty of an "any time any place" operation.

### 1.3.3 Defining the Problem

Previous research on remote access laboratories, has had varying results, providing a development methodology, [Bourne *et al.* 2005; Johnson *et al.* 2003], and discussed in greater detail in 2.3 below. Research on online laboratory experimentation is still a developing and controversial subject. The problem of how a remote access laboratory can be designed for a mechatronics course comprises:

- how a student could design AI to control a mechatronic device,
- the design of the user interface,
- the design of the physical laboratory.

## 1.4 Research Aims

The aims of the research were:

1. To establish the viability of remote access facilities to augment distance learning.
2. To design and evaluate technology which can provide an environment for students to learn to rapidly develop prototype Artificial Intelligence for a mechatronic device.

3. To test the hypothesis:

**Programming by Demonstration could prove a more intuitive approach to the complexity of developing an emergent intelligent behaviour than text-based programming.**

## 1.5 Research Questions

The research, as expressed by the thesis title, was required to make a significant contribution to the evidence supporting the use of remote access laboratories within a distance learning based institution. The following research questions resulted:

1 What are the criteria for designing a remote access laboratory for prototyping Artificial Intelligence, as part of a distance learning organisation's available tools?

2 What fusion of technologies should be used to develop a Mechatronics prototyping laboratory?

3 What methodology and technologies could assist in rapid prototyping Artificial Intelligence in a distance learning mechatronics course?

4 What design of interface to such a laboratory would allow appropriate analysis and demonstration of prototype Artificial Intelligence?

## 1.6 Research Objectives

- Carry out the selection, design and application of technologies to establish a remote access laboratory, which can create and test prototype Artificial Intelligence programs.
- Design a method for a user to develop prototype Artificial Intelligence.
- Design and implement a prototype interface, which allowed a laboratory user ease of access to the laboratory's internal functions, without being made aware of the internal construction of the laboratory.

## 1.7 New Knowledge

The research was intended to lead to a significant contribution in scholarship within the subject of Mechatronics at the Open University, and in a wider context of technology application, by establishing a novel contribution to support the use of Programming by Demonstration in developing prototype Artificial Intelligence, this was to be tested in a Remote Access Laboratory, within a distance learning environment.

## 1.8 Structure of the Thesis

This thesis considers the circumstances of designing a remote access laboratory, and what technologies should be used, and comprises the following chapters:

Chapter 1 **Introduction to Remote Access to Prototyping Laboratories**

This chapter has introduced the context and origins of the research, with the principle research issues and requirements for a remote access laboratory. The objectives are identified as a need for developing and testing a prototype Artificial Intelligence in a distance learning mechatronics engineering laboratory environment.

Chapter 2 **State-of-the-Art for a Distance Learning Laboratory**

The current theories for distance learning are discussed with current expectations and how they relate to a remote access laboratory. Current approaches relating to an engineering laboratory are examined next, beginning with the purposes of an engineering laboratory before reviewing current remote access laboratory design.

Chapter 3 **State-of-the-Art for an Interface to a Distance Learning Laboratory**

The chapter establishes the design principles of Intelligent Tutoring Systems for distance learning. This outlines the use of internet-based interfaces and the problems associated with their design and application. The next research area considered is current Human-Computer Interface design theories. This firstly considers Cognitive Psychology approaches before Activity Theory based design. Finally, the research area of Programming by Demonstration and the related principles of communicating knowledge are reviewed, since the analysis of Programming by Demonstration is based on its suitability to develop prototype Artificial Intelligence, for testing in a Remote Access Laboratory.

Chapter 4 **A Proposed Architecture for a Distance Learning Laboratory**

This chapter proposes an Intelligent Agent based architecture for a remote access laboratory. Intelligent Agents with blackboard based architectures for receiving designed prototype Artificial Intelligent behaviour are described. The agent architectures are further developed with proposals for their communication architecture with the separation between their specific functions and their communication knowledge.

Chapter 5 **Development of a Prototype Interface**

The prototype Programming by Demonstration based Human-Computer Interface for the laboratory is defined, with its potential application in developing a prototype Artificial Intelligence for a remote access laboratory. A method of developing a prototype Artificial Intelligence is described together with the principles of programming which determine the success of any programming methodology.

Chapter 6 **Design of Experiments**

This chapter explains the scientific method of experimentation used in the research. The statistical analysis methods used are explained with an explanation of the errors associated with this experimentation. Finally the ethics involved in this experimentation is outlined.

Chapter 7 **Experimental Procedures**

This chapter explains the experimental procedures used, related to the objectives of the experiment and the variables involved in the experimentation. It also describes the experiment's activities and the equipment used.

Chapter 8 **Data Obtained and Interpretation**

This chapter provides the raw data obtained from the experimentation, with analysis, interpretation and summary. The implications of the results are discussed

Chapter 9 **Conclusions**

This chapter concludes the research findings, and discusses the validity of the hypothesis. It identifies specific areas for future research necessary to implement a remote access laboratory, and comments on the overall significance of this research. It includes a section on self-reflection, where the learning process resulting from undertaking a PhD is considered.

# Chapter 2
# State-of-the-Art for a Distance Learning Laboratory

## 2.1 Introduction

This chapter initially discusses the principles and concepts of distance learning and remote access technologies, then examines specific, interrelated areas of research to establish the current state-of-the-art concepts and techniques necessary to design a remote access laboratory for a distance-learning course. The discussion follows the progression:

- **Distance Learning** is a review of the current theories advocated for distance learning, and how they relate to a successful Remote Access Laboratory.
- **Distance Learning Laboratories** is a review of the expectations and theory of a distance learning laboratory. The theory explains the objectives, types and designs of distance learning laboratories.
- **Collaboratories** relate to software which allows multiple users to interact with each other and with tools in a laboratory setting, and an explanation of the theory and current designs.
- **Intelligent Agents** relates to the design and application of intelligent agents which are advocated for the internal design of a remote access laboratory.

## 2.2 Distance Learning Theories

### 2.2.1 Introduction to Distance Learning Theories

Johnson *et al.* [2003] defines learning as a:

> *... formal educational process in which the majority of the instruction occurs when student and instructor are not in the same place. Instruction may be synchronous or asynchronous. Distance education may employ correspondence study, or audio, video, or computer technologies.*

Morse and Truman [1996] argues that a learning institution's objective is to provide a means for students to learn, with the varying distance-learning philosophies' success depending on replicating face-to-face classroom interaction, unless the student characteristic reduces a need for classrooms.

Bourne *et al.* [2005] promotes '*five pillars of quality online learning*', intended to evaluate distance learning progress. However, without providing a benchmark these become subjective and relativistic measurements.

1. **Learning effectiveness**: Koper and Olivier [2004] citing Merrill [1994] states learning is effective when learners form new knowledge using existing knowledge to solve real problems. Menges and Austin [2001] were cited comparing studies of online and face-to-face instruction-based learning environments, stating the perceived wisdom that technology does not influence student results or satisfaction, citing Johnson *et al.* [2000]; enhance student learning, citing Moore and Kearsky [1996], Clark [1994]. Any student learning improvements are due to the teaching method built into the use of the technology [Setchi, 2007]. Massey and Zemsky [1995] argues that technology can overcome the limitations of time and space for traditional education activities, reasserting the importance of good communication between teachers and students, and arguing that extensive computer mediation is, however, not always compatible with fields of study concerning questions of meaning, values, culture and philosophy.

2. **Access**: The issues are discussed in Coventry [1995]. Since the development and widespread adoption of Broadband the problems of internet connectivity has been solved. Hashemi *et al.* [2006] reports that students can make mistakes and working at any time, increases the student's opportunities to improve his/her competence in a physical laboratory.

3. **Faculty satisfaction:** Bourne *et al.* [2005] reports that this includes '*support, rewards, and personal satisfaction*'. Chickering and Erhmann [1996] and Graham *et al.* [2000] argues that online learning environments benefit faculty and students with improved: interaction between students, and with faculty, learning methods, communication of expectation, and learning method diversity.
4. **Student satisfaction:** While argued as student satisfaction, Bourne *et al.* [2005] identifies more the need for eliciting student interaction or collaboration. By implementing and assessing laboratory formats, students are expected to collaborate in modelling and controlling dynamic engineering systems, and improving data capture for both conceptualization and theory use, [Kypuros and Connolly, 2005].
5. **Cost effectiveness:** Lifelong learning is becoming a competitive necessity in employment, with a shift from academic emphasis to competency attainment, and faculty roles becoming more specialized. Faculty are demanding reduced workloads implying automated work process support. A current issue is sustainable Open Educational Resources, which are: to support learning, teachers, and assure the quality of education (free) [Hylén, 2006, Johnstone, 2005]. Downes [2007] argues that they are sustainable, by adapting Wikipedia's model as an Open Resource.

Experimentation is non-deterministic, challenging students to research, problem solve, and inquire about their own answers, [Mizell, 1994]. Laboratory work requires cognitive skills for problem solving. Dimitracopoulou and Petrou [2003] argues the development of collaborative technologies is due to advances in two fields of research:

(a) **The development of learning theories:** The importance of a *social element* (interactivity between students) in learning, has led to new theories emerging with a social and cultural-based dimension to the learning process, resulting in further development of learning technologies.

(b) **Advances in information and communication technologies**: These have created new forms of communication, allowing networked cooperation and collaboration. The secondary research included reviewing collaboratory learning technologies, with various formats for a collaboratory in a learning context.

A laboratory operates as a learning process function, where the learner tests new knowledge gained, and its operational success depends on providing the required experimentation interaction. Various Distance Learning Theories are examined next.

## 2.2.2 Principle Distance Learning Theories Considered

Table 2.1 An outline of current advocated learning theories

| Author | Presented Theory | Theory Origins | Method of Teaching and Learning | | | | |
|---|---|---|---|---|---|---|---|
| | | | Taught theory | Practical Experience | Dialog | Iterative Process | Who to learn from |
| Coventry [1995] | (Re)conceptualisation Cycle | Kolb [1984] | Yes | Testing new knowledge | For deep learning | Yes | Tutor |
| Müller and Ferreira [2005] | Experiential Learning | Kolb [1984] | No | Starting point for learning | To share experience | Yes | Tutor |
| Koper & Olivier [2004] | Social-Constructivist Learning | Vygotsky [1978] | No | Authentic problems | Depends on learning | Yes | Depends on learning |
| Bonk & Cunningham [1998] | Social-Constructivist Learning | Vygotsky [1978] | No | Embedded learning in authentic tasks | Social influence on learning | No | Depends on learner |
| Siemens [2005] | Social-Constructivist Learning | Vygotsky [1978] | No | Best teacher of knowledge | Learning by opinion | Yes | A learning decision |
| Nabeth *et al.* [2005] | Model of Change Process | Rogers [1995] | No | Experience in context | Learning by discussing | Yes | Knowledge discovery! |

Key: For Against

The table establishes a diversity of theories on how students should learn a subject. The most important issue for any engineering and science-based subject is the tuition of a body of theory by an experienced tutor. The (Re)conceptualisation Cycle approves the teaching of theory, using a laboratory for testing newly learned knowledge. Experiential Learning advocates tuition from a tutor, but not the teaching of theory. Further, experimentation should be an examination of theory in realistic circumstances, which is supported by Social-Constructivist Learning and the (Re)Conceptualisation Cycle, but the experimentation-based learning process should be controlled, to maximise the student's learning experience, and prevent time-costly mistakes.

### 2.2.2.1 The (Re)conceptualisation Cycle

The theory of Coventry's [1995] research paper on the '*(Re)conceptualisation Cycle*' uses constructivist principles, as Sherry [1996] explains, the student constructs knowledge by developing and using an image and interacting with the material to be learned. Coventry's proposal to provide effective distance learning theory requires clear communication and effective tools operation by both user and laboratory, and summarised as comprising:

- **Conceptualisation** supporting the presentation of content which involves
  - Orientation - the outline of what to be learned
  - Exploration – independently exploring the subject being learnt
  - Experimentation - interacting with the learning environment
- **Construction** providing resources for the doing of learning tasks which involves
  - Selecting - picking out what is to be learned
  - Linking - combining old and new information
  - Classifying - comparing old and new information and linking the two
- **Dialogue** support through communication which involves
  - Discussion - tutorial and peer-to-peer contact is paramount
  - Reflection - fundamental provided that the topic has been discussed
  - Reification – consolidation of discussion and reflection

### 2.2.2.2 Experiential Learning

Müller and Ferreira [2005] reports on the Virtual Laboratory MARVEL (Mechatronics: Access to Remote and Virtual E-Learning) project, which

> *...is focused on supporting learning practice based on social constructivism, combined with experiential and collaborative learning.*

Experiential learning is learning both by '*concrete experience*', followed by '*reflective observation*'.

The Experiential Learning theory used by MARVEL advocates that knowledge is created by a cyclic iterative process transforming experiences, by '*reflection and conceptualisation*'. Müller and Ferreira's [2005] views of Experiential Learning is that:

> *...Hands-on learning in real-physical labs or workspaces provide reach opportunities for experiential learning, because the learner can 'experience' theory in a more familiar form, since the practical experiment enables the students to "observe and reflect on" the results of learning tasks and assignments. Each experiment or practical work task may therefore be seen as a starting point to understand its underlying theoretical principles.*

This is a contradiction of the purpose of an experiment, where it is used to test a hypothesis. Developing the theory after running the experiment requires prescience in running the experiment, otherwise how does the researcher identify the key theoretical interest variables?

### 2.2.2.3 Social-Constructivist Theory

Bonk and Cunningham [1998] explains Social-Constructivist learning as:

> *...Instruction* [which] *should provide opportunities for embedding learning in authentic tasks leading to participation in a community of practice.*

Siemens [2005] explains the '*community of practice*' principle as:

> *Decision making is itself a learning process. Choosing what to learn and the meaning of incoming information is seen through the lens of a shifting reality. While there is a right answer now, it may be wrong tomorrow due to alternations in the information climate affecting the decision.*

This theory places an emphasis on a '*community of practice*' to be both knowledgeable about a subject and a suitable source for tuition. Social-Constructivist theory is refuted for significant reasons:

1. Both science and engineering have mathematical principles, immutable definitions and laws/rules for understanding the subject.

2. The philosophical principle underpinning Social-Constructivist Learning was first promoted by Spinoza, about whom Bertrand Russell stated '*Intellectually, some others have surpassed him, but ethically he is supreme. As a natural consequence, he was considered a man of appalling wickedness*' [Russell, 1946]. Spinoza is cited by Russell as arguing: '*there is no right or wrong, for wrong consists in disobeying the law*'.
3. A good Engineering education is not learning the principles of engineering design by being prosecuted for ignoring health and safety rules and regulations, and having attendance at a Criminal Law Court as the learning experience.

Social-Constructivist Theory of Learning is believed unsuitable for teaching any subject with an associated theory.

### 2.2.2.4 The Centre for Advanced Learning Technologies (CALT)

The European Union's CALT research focuses on both learning and change/innovation at the individual level. The methods involve the use of Intelligent Agents, multimedia and virtual reality to acquire and adopt new knowledge, by motivating the individual and engendering interaction. Nabeth *et al.* [2005] states that:

> *...the success of e-Learning... ...has been at best disappointing, and is certainly very far from fulfilling the high expectations that the more forward-looking students, educators and institutions had of it. The reason for this limited success originates, in our belief, from too narrow and conservative vision of the learning processes to be supported. In most of the cases, e-learning systems still rely upon the same good old educational classroom-based instructor-led teaching method that has existed for years... ...and that is characterised by* (1) *A relatively passive and anonymous student considered as a recipient of learning materials that are delivered to him/her.* (2) *A body of knowledge to be offered that is dominantly of generic theoretical/conceptual nature...* ...[and tracking] *how this material is actually absorbed by them. Whilst this method that has been successfully applied for mass education can be considered as adequate to complement the training of inexperienced learners co-located in a same campus or school for acquiring the basic body of theoretical knowledge, it falls short of accommodating the needs of more demanding and experienced distributed knowledge workers....*

The premises of the Centre for Advanced Learning Technologies projects are:

> *...e-learning has to rely on a new vision that requires a fundamental shift from current content-oriented e-learning solutions towards a more user-centred interactive and collaborative model of learning.*
>
> *...the learner is no longer a simple passive receiver of data and information, but is seen as a participant that is actively engaged through a rich set of interactions (e.g. learning by doing, educational games, simulation environments, problem based learning, learning by discussing, knowledge discovery, etc).* [Razmerita *et al.*, 2004]

Razmerita *et al.* [2004] explains that a model of participants' learning comprises '*a model of change process*', figure 2.1 below, [Angehrn and Nabeth 1997; Manzoni and Angehrn 1998].

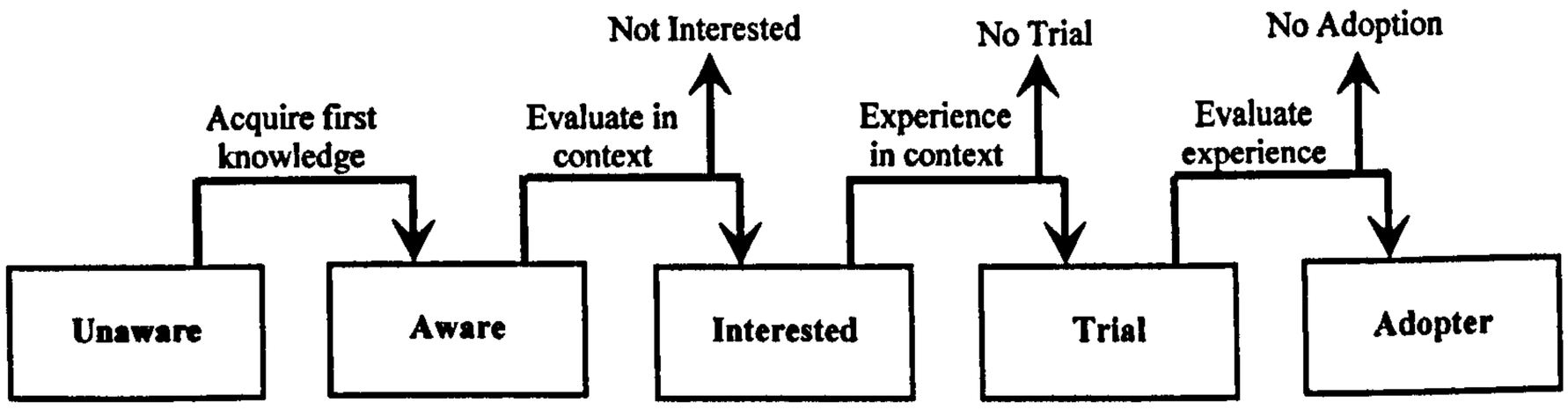



Figure 2.1 A model of the change process, [Razmerita *et al.*, 2004]

There is a difference from theory-based subjects, where student development depends on learning a presented body of theory; instead knowledge is subjectively provided, [Roda *et al.*, 2001].

## 2.2.3 Conclusions about Distance Learning Theories

The remote access laboratory is intended for a user to develop and test prototype Artificial Intelligence and it depends on operating as an instructional and development tool. The learning theories presented are now compared.

Morse and Truman [1996] premises distance learning theories on replicating campus-based learning, while Bourne *et al.* [2005] argues for '*five pillars of quality online learning*' which were subsequently analysed, with computer mediation considered not always effective for subjects requiring discernment of meaning, values, culture and philosophy.

Finally, for the current learning theories advocated the (Re)conceptualisation Cycle approves the teaching of theory, using a laboratory for testing newly-learned knowledge. Experiential Learning advocates tuition from a Tutor, but not the tuition of theory.

## 2.3 Distance Learning Engineering Laboratories

This thesis will investigate distance learning engineering laboratories, for distance learning students to experiment relevant to their studies. Forinash and Wisman [2005] objects to adopting remote labs for distance learning engineering degrees, arguing that:

> *No experiment can be performed with zero error, so one must determine with what degree of certainty the data support a particular hypothesis. Coming to terms with the inaccuracy and imprecision of results requires knowledge of the interplay between experimental design and data analysis. Some laboratory skills, such as statistical analysis of data, can be learned in the abstract, outside of the laboratory. Experimental design, however, can only be learned from using real equipment in real experiments, often through a certain amount of trial and error. It should be no great surprise that student practice of experimentation is needed to understand science, that educational abstractions alone are not enough.*

The remote access laboratory being researched is intended for testing prototype Artificial Intelligence. The user will be working directly within a programming environment, with the subsequent mechatronic device activity considered suitable to operate remotely.

### 2.3.1 The Expectations of Engineering Laboratories

Feisel and Rosa [2005] states:

> *Engineering is a practising profession... ...The overall goal of engineering education is to prepare students to practice engineering... ...Thus, from the earliest days of engineering education, instructional laboratories have been an essential part of undergraduate and, in some cases, graduate programs. Indeed, prior to the emphasis on engineering science, it could be said that most engineering instruction took place in the laboratory.*

The remote access laboratory being researched is aimed at being suitable for tuition, as the instructional work involves interacting with a programming environment. Having a mechatronic device located remotely will replace a real-world engineering laboratory, as the laboratory proposed is being used to establish the validity of a developed prototype Artificial Intelligence.

### 2.3.2 Engineering Laboratory Theory

#### 2.3.2.1 Objectives of an Engineering Instructional Laboratory

Feisel and Rosa [2005] describes an engineering instructional laboratory objectives in terms of the following characteristics and components:

1. **Instrumentation** ...using sensors, instruments, and/or software to obtain data.
2. **Models** ...evaluating a theory's ability to predict physical events, and test the relationship between measured data, theory and existing rules.
3. **Experimentation** ...learning to prepare an experiment, specifying equipment and procedures, implementing these procedures and interpreting the data to characterize an engineering material, component, or system.
4. **Data analysis** ...learning to collect, analyse, and interpret data, to use measurement systems and conversions, to judge magnitudes and to form and support conclusions.

5. **Design** …learning to design and assemble a product using equipment, materials or methodologies, to meet requirements and specifications, testing and debugging a prototype system or process to meet requirements.
6. **Learning from failure** …identifying the causes of failure and engineering effective solutions.
7. **Creativity** …demonstrating appropriate thinking for problem solving.
8. **Psychomotor skills** …demonstrating competence with the tools and resources.
9. **Safety** …responsibly demonstrating health, safety and environmental issues.
10. **Communication** …learning to communicate effectively about laboratory work to various audiences.
11. **Teamwork**. …working effectively together, assigning tasks and responsibilities to meet objectives, and reporting.
12. **Ethics in the laboratory**. …working ethically and with integrity, including reporting information objectively.
13. **Sensory Awareness**. …learning the limitations of human abilities.

The proposed remote access laboratory will support the objectives of an instructional laboratory. These objectives proposed above can be grouped into the ability to work in a laboratory setting, the ability to relate theory to practical work, the ability to problem solve and the ability to work with others. The element of teamwork cannot be deemed an absolute, as there is an expectation that an engineer may need to work independently.

### 2.3.2.2 Types of Engineering Laboratories

Feisel and Rosa [2005] argues that there are three types of engineering laboratory, possessing varying purposes:

- *Development laboratories* to obtain experimental data, for use in guiding design and development by answering specific questions, otherwise, to determine if a design performs as intended, by comparing results with specifications and to show compliance or where to make changes.
- *Research laboratories* used for determining general and systemic knowledge to increase current global knowledge.
- *Instructional laboratories*, to learn '*something*' that practising engineers know. The '*something*' needs defining if a laboratory's usage is to be beneficial.

As an instructional laboratory, the remote access laboratory will be used for the student to learn how to program a mechatronic device's prototype Artificial Intelligence.

#### 2.3.2.3 Distance Learning Engineering Laboratory Design

Bourne *et al.* [2005] argues there are two designs of distance learning online laboratories:

- **Web-based simulations:** often referred to as virtual labs; these are equivalent to physical labs for explaining and reinforcing concepts also supported by Forbus *et al.* [1999]. Simulations provide limited capability for experimentation and cannot always accurately apply theory or concepts to the physical world, while Page *et al.* [2000] argues that the poorly applied theory is the user's failings.
- **Remote laboratories:** allow manipulation and observation of real equipment located at a distance, also reported by Campbell *et al.* [2002], Tait and Chao [2003], Gröber *et al.* [2007].

Bourne *et al.* [2005] considers that remote laboratories may become increasingly common, though the widespread adoption of remote access laboratories will depend on students accepting them for studying on distance-learning engineering degrees.

### 2.3.3 Access to Distance Learning Laboratories

Johnson *et al.* [2003] investigates US Colleges exemplary distance learning courses. Three means of accessing a laboratory were identified for distance learning courses:

(a) **On-campus skill acquisition:** students attend a college for laboratory work only.

(b) **Internships or clinical experience:** students complete the laboratory course as either apprentices or student interns. Evidence of the laboratory work is submitted as work samples with verification from the supervisor.

(c) **Computer-based simulation:** this allows the development of skills in a controlled environment without the danger or cost of a "real life" situation.

The problem with on-campus skill acquisition identified above is that students still have to attend the college for the laboratory work. This negates Fjuk's [1995] assertion of '*...a flexible (further) educational situation free from place and often time, constraints*' referred to in 1.3.1 above. Internship and clinical experience entail a time constraint. The problem with a simulation which emulates perfect conditions is that, in reality, engineering experimentation is potentially less than perfect.

### 2.3.4 Laboratory Architectures

Kypuros and Connolly [2005] reports three laboratory architectures being tested in US Universities, using multiple visualization means, individual and collaborative exercises.

1. *Inter-university laboratory architecture* is where two campuses' students work in joint collaborative learning. The first campus students develop a computerised simulation of the experimental system's test conditions and parameters. The second campus students perform the experiments, acquire and process the data. This format is designed to consolidate resources that are not equally available at both campuses, and the roles of the two campuses can be reversed for a subsequent project.

2. *A remote-accessible laboratory architecture* is where students model a remotely-accessible system. The experiments are accessed using LabVIEW's Remote Panels, which allow parameter manipulation, data collection, and view real-time dynamic response via video feedback. The experiment can be a more complex real-world problem, animated in 3D developed using MSC VisualNastran 4D and be Internet accessible. This format develops the concepts of a remote access laboratory system, and is intended to help students expand the use of concepts to real-world engineering problems. The model allows students to change physical system parameters and to prototype controllers.
3. *A virtual laboratory architecture* is similar to the previous laboratory except it uses a 3D virtual system [Johansson and Astrorn, 1996; Johansson *et al.*, 1998] developed using MSC VisualNastran 4D. It provides animated output, and time- and/or frequency-domain plots. A Java applet allows remote access of the virtual system for simulation, data acquisition, and controller prototyping, without using any specialized software. Students develop models and simulations using MATLAB/SIMULINK and upload their controller designs to the virtual system, for testing, and viewing results.

All three laboratory formats use simulators, which suffer from only working within the limits of designed parameters. Any parameter not designed for will not be used.

Feisel and Rosa [2005] proposes using a simulated laboratory for several reasons:

- **Pre-lab experience**: this will help students prepare for experimentation in a physical lab, supported by Gleixner *et al.* [2002] and reduce experimentation time.

- **Experimental studies:** laboratory simulations are suitable if systems are too large [Rauwerda *et al.*, 2006], expensive or dangerous for students [Zary *et al.*, 2006], [Bardeen *et al.*, 2006]. Simulator laboratories are more realistic due to various innovations, for example, adding budget and time limits into the problem specifications, [Jayakumar *et al.*, 1995]. Use of random elements can make simulations more realistic, and simulators may emulate physical experiments more closely in the future.
- **Laboratory substitute:** students who use simulators and two physical laboratory experiments have a similar performance to using traditional laboratories, [Cambell *et al.*, 2002].

The use of a simulator is subsequently rejected for a physical remotely-accessed laboratory, as a simulator does not allow for unpredictability. The laboratory substitution argument mostly relies on the use of physical laboratory, requiring continued research to use a simulator to replace the physical laboratory.

### 2.3.5 Conclusions about Distance Learning Engineering Laboratories

A current research issue with a remote access laboratory is how to answer the question: can a remote access laboratory provide the same educational value as a physical laboratory? Forinash and Wisman's [2005] objections to distance learning laboratories, (see 2.3 above), argues against using simulators to replace physical laboratories. The use of a physical remotely accessed laboratory can adequately fulfil the expectations and objectives of an engineering instructional laboratory. This thesis provides a significant contribution to the validation of using a distance learning laboratory, through the use of both a simulator, and subsequent use of a physical laboratory.

## 2.4 Collaboratories

Dewan *et al.* [1994] defines a collaborative application as:

> *...a software application that (a) interacts with multiple users, that receives input from multiple users and displays output to multiple users, and (b) couples these users, that is, allows one user's input to influence the output displayed to another user.*

This definition is intended to cover all the possible concepts and designs of collaborative applications, because the connection between any two users is undefined. Dewan *et al.* [1994] supplies a detailed definition, but argues that the detail overqualifies an application, so is not general.

### 2.4.1 Principles of Collaboratories

Dewan *et al.* [1994] proposes some principles for a collabatory's functional design:

- **Specification**: It should be easy for users to specify how to collaborate.
- **Performance**: A collaboratory's response time must be acceptable.
- **Grouping**: Users should specify collaboration for a set of '*objects*' sharing a definition, instead of specifying collaboration for each individual '*object*'.
- **Automation**: It should be easy for programmers to collaborate.

### 2.4.2 Examples of Collaboratories

#### 2.4.2.1 Virtual Collaborative Environment (VCE)

Davies *et al.* [1994] describes Sandia National Laboratory's Virtual Collaborative Environment (VCE), remote programming and control of mechatronic devices, which allow '*expensive capital equipment*' sharing. The VCE described has a high volume data network requiring high-speed network transmission.

The system requires the laboratory's users to have two computer workstations, one to handle the video conferencing and video interfacing, the other to operate the mechatronic device's interface, using a graphics control system. The current system components are:

- *A graphical model* corresponding to the robot and its environment.
- *A graphics workstation* simulating and displaying the robot interaction functions.
- *Simulation software* to display and preview real-time robot motion for user validation before actuation, with automatic collision detection to verify safe paths of operation.

Tasks are selected and defined by the user with an automated planning and programming system to fufill the instructions and the user accepts or rejects the plans.

In contrast, the proposed remote access laboratory interface only uses one computer. The interface includes a simulator to model the mechatronic device and environment, which previews and validates a mechatronic device's behaviours.

### 2.4.2.2 Distributed Collaboratory Experimental Environment (DCEE)

Fernando and Dew [1998] reports on the DCEE architecture's primary function to integrate product data and engineering tools in a distributed environment. The DCEE allows geographically dispersed personnel to share and manipulate product data in a 3D environment, while discussing complex and detailed issues. The system requirements are to:

- Provide synchronous data distribution amongst users with immediate changes propagation,
- allow users to vary tools and information by having a user's perspective,

- support many development phases and be upgraded,
- adhere to product data standards in data structures and data representation,
- provide a virtual environment to represent data structures,
- enable networking independence to utilise effectively a wide range of diverse networks.

Maintaining a collaboratory's software architecture is considered essential in the design control software, as the equipment is changed regularly, requiring the experiment and interface control programs to reflect the changes. The system is expected to be updated easily, using programs as building blocks and not complete control systems.

The underlying collaboratory infrastructure is a common interoperability framework, connecting various components with a common interface. Tools are '*plug and play*' via a [logical] resource manager, requesting resources as necessary. Data and result files are available to all collaborators, [Agarwal *et al.*, 1998].

The proposed laboratory internal software architecture will need to be flexible, allowing mechatronic device alterations to reflect technology advances. This is considered important, as the purpose of the laboratory is to allow students to learn how to program the technologies they will expect to meet in their subsequent career.

#### 2.4.2.3 A Virtual Training Laboratory at Queen Mary College

Queen Mary College proposes a virtual laboratory controlled by intelligent agents, [Norman and Jennings, 2002], intended to improve postgraduate telecommunications students' training, allowing guidance and exchange of ideas with more experienced colleagues; access to research papers and books, and experimental tools for evaluating new ideas and hypotheses.

A multi-agent system allows the decomposition of a Remote Access Laboratory software control architecture from a single potentially cumbersome software entity, to agents managing scarce resources, and integrating any existing systems. The agents are anticipated to benefit users by introducing them to each other and providing relevant research material which they may not be previously aware of.

### 2.4.3 Summary of Collaboratories

The developed collaboratories corroborate the theory of Collaborative Applications proposed by Dewan *et al.* [1994], whose contribution to the subject is proposing multiuser-edited '*whiteboards*' suitable for users' interaction in collaborative applications. Dewan and Shen [1998] reports that the interaction structures between users require '*...access specifications to be associated with persistent objects*'. Golab and Özsu [2003] explains that this is a Data Stream Management problem requiring multiplexing and demultiplexing of data, or mixing both stream data with static data, and argues:

> *designing an effective data stream management system requires extensive modifications of nearly every part of a traditional database, creating many interesting problems such as adding time, order, and windowing to data models and query languages.*

This problem is of current research interest in database management systems, because there is a growth in applications which have '*long-running, continuous, standing, and persistent queries*', [Golab and Özsu, 2003], and in collaboratories the issue is connecting data to an object being examined and manipulating the data. This is important as a collaboratory's success depends on the methods used to store and retrieve experimental data, and introduce new experiments.

If a collaboratory is difficult to use, then internal and remote users will not be encouraged to use it, ultimately precipitating its demise. If the system is too complicated to add new experiments and technology, inflexibility would result in a loss of operational ability, making the collaboratory unattractive for future users.

## 2.5 A Distance Learning Laboratory Architecture

The proposed Remote Access Laboratory is expected to have an internal software architecture which allows a user's prototype Artificial Intelligence to operate a mechatronic device. The proposed laboratory has to be sufficiently flexible to allow mechatronic device alterations to reflect technology advances. This is considered important, as the purpose of the laboratory is for students to learn how to program the technologies they expect to meet in their subsequent careers. The laboratory's success depends on the flexibility to introduce new experiments, which could potentially be achieved by using the concept of Intelligent Agents to fragment and distribute control.

> *Intelligent computer agents are both the original goal and the ultimate goal of artificial intelligence research. In striving toward that goal, our community has followed a practical research strategy of "divide and conquer," with different sub-communities attacking important component functions of intelligence, such as planning, search, knowledge representation, vision, and natural language.* [Hayes-Roth *et al.*, 1995]

### 2.5.1 Role of Intelligent Agents

Intelligent agent architectures developed from implementing Distributed Intelligence Systems, or Multi-Agent Systems, with two competing architecture design principles:

- Agents based on deliberation, generally planning, termed *Deliberative Agents*.
- Agents, governed by situation reaction rules, termed *Reactive Agents*.

Either one of these intelligent agent architecture's weakness is the other's strength [Davidsson, 1996].

Bryson [2000] considers Autonomous Agent Architectures as design methodologies, a combination of design knowledge and strategies. The architecture design knowledge is argued as obtained by both reasoning and experience. Reasoning is argued as the early papers for an agent's architecture, which provide the hypothesis and early architectures as evidence. Experiential knowledge is argued as explicit reports and/or '*unpublished record of failed projects or missed deadlines*'. The Intelligent Agents theories being examined are compared in table 2.2 below.

Table 2.2 Comparison of agent theories

| Author | Theory Provided | Agent Architecture | Multi Agent System | Multi Agent Communication |
|---|---|---|---|---|
| Hayes-Roth *et al.* [1995] Hayes-Roth [1995] | Program based behaviours | BB1 (blackboard agent) | No | Not considered |
| Sadeh *et al.* [2001] | Knowledge encapsulated behaviours | MASCOT (blackboard agent) | No | Not considered |
| Corkill [2005] | Principled agent architecture | High Level Data Fusion Agent (blackboard agent) | No | Not considered |
| Davidsson [1996] | Agents which track other agents | A Linear Quasi-Autonomous Agent | Yes | Co-operation |
| Jennings *et al.* [1995] | Agent design framework | ARCHON | Yes | Co-operation |
| Chavez *et al.* [1997] | Decentralised multi agent framework | Challenger | Yes | Negotiation |
| Wooldridge *et al.* [2000] | Design & analysis agents | Gaia | Yes | Negotiation |
| Helsinger & Wright [2005] | Distributed blackboard agents | COUGAAR (blackboard agent) | Yes | Negotiation |

The agents reported show an incremental development of Agent architecture theories, with a significant change in the theory from single agent based architectures with Hayes-Roth BB1 to multi-agent based architectures with Helsinger and Wright's COUGAAR. The shift in research focus subsequently shows a change in the Multi-Agent architectures from co-operative to negotiated interaction. The research in the blackboard agent architecture has focused on the operation of a Blackboard control systems, which has proven problems, with a parallel growth in research of multi-agent systems comprising blackboard agents.

### 2.5.2 Role of Blackboard Agents

A blackboard agent is an agent architecture which utilises a memory space for all the agents' operation knowledge. The importance of this architecture is the requirement of a mechatronics Remote Access Laboratory to have a means to receive and test the user's prototype Artificial Intelligence.

The blackboard intelligent agent model was based on Feigenbaum [1977] and Shortliffe [1976]. Carver and Lesser [1992] outlines the basic blackboard agent architecture shown in figure 2.2 below, comprising a blackboard, knowledge sources and control mechanism.

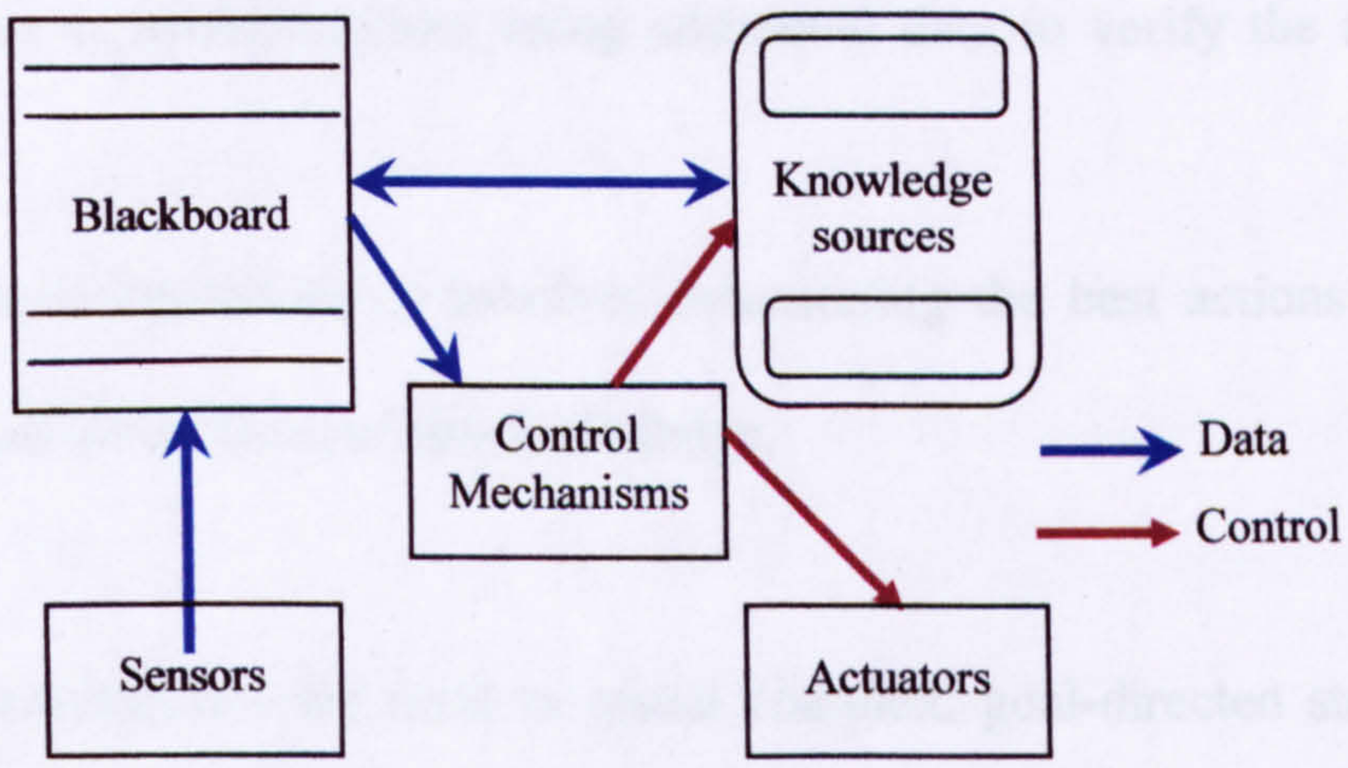



Figure 2.2 The basic blackboard agent architecture

Corkill [2003] argues that a blackboard agent's collaborative power is the Knowledge Sources which provide limited data, since providing all the data is considered too distractive. With partial knowledge-sharing using the blackboard, knowledge sources are triggered by partial activation states, with a search to find additional data needed to activate their behaviours. The result is that when information is shared, locating information has to be fast, complete and highly relevant, [Lesser and Erman, 1980]. Corkill [2005] explains that previous Blackboard Systems have been criticised in recent years due to their *ad-hoc* evaluation of belief values, to determine control decisions.

### 2.5.2.1 The Blackboard Agent Architecture Components

The blackboard is the agent's global database containing data and hypotheses, structured into areas, for possible efficient retrieval of associated hypotheses. A problem with a single blackboard is how to integrate goal-directed factors with data-directed, agenda-based control.

The knowledge sources contain the agent's current state and are used to create new hypotheses or modify existing ones. Ideally knowledge sources are independent, with interactions incrementally and opportunistically modifying hypotheses on a blackboard.

- Incremental hypothesising (*evidence aggregation*) involves using incomplete data to hypothesise a partial solution, using additional data to verify the solution, [Rich, 1991].
- Opportunistic hypothesising involves determining the best actions to achieve the agent's goal given the available knowledge.

The control mechanisms are used to create complex, goal-directed strategies. Carver and Lesser [1992] explains the problem with blackboard architecture control, that a system needs to deal with multiple sensors, generating large amounts of data, or hard, real-time deadlines. Multiple passive sensors require control mechanisms to prevent blackboard data overloading, while real-time deadlines require deterministic control with the facility to assess possible action duration, and dynamically alter the Knowledge Source's activation conditions. Self-activating independent knowledge sources may not need control mechanism, but have two potential drawbacks:

1. Knowledge sources require sequential execution, using multi-threaded programming.

2. Blackboard agents are typically used for combinational data problems, but only if a solution does not require executing all the possible solution knowledge sources.

### 2.5.2.2 Examples of Blackboard Agent Implementation

**The BB1 Agent Architecture**

Hayes-Roth *et al.* [1995] research involved '*multi-agents in unpredictable environments*' using the BB1 agent architecture '*for spontaneous goal generation and selection*'. The BB1 is a multi-layered blackboard agent architecture, with the blackboards having a component library and reusable domain expertise for knowledge sources; and an application configuration method to select and configure architecture components as a circumstantial control mechanism.

A high level task specific language is developed to control both decision and domain-action specification, allowing control sharing strategies among similar task applications [Hayes-Roth, 1995]. The physical level behaviours are environment perceptions and actions, while the cognitive level behaviours provide more abstract reasoning. The examples given are '*...situation assessment, planning, problem solving'*. The control software is event-driven, with sequenced planned, stepped data comprising: start condition(s); intended activity and stop conditions. The software can be modified and developed dynamically, with competing behaviours providing adaptability through the criteria of a behaviour best performing a task within the current parameters, and matching the current constraints closest to the behaviour being executed, [Hayes-Roth, 1995].

**MASCOT (Multi-Agent Supply Chain Coordination Tool)**

Sadeh *et al.* [2001] reports on the MASCOT architecture, which is based on previous architectures using modular encapsulation of problem-solving Knowledge Sources. These preceding architectures successfully integrated multiple knowledge sources to develop solutions using a shared data structure for a diversity of applications, [Erman *et al.*, 1980; Corkill, 1991; Carver and Lesser, 1992].

**COUGAAR (Cognitive Agent Architecture)**

Helsinger and Wright [2005] describes the COUGAAR as a distributed agent architecture, comprising a COUGAAR Node, which may have multiple blackboard architecture agents sharing computer resources. The Blackboard is described as an agent's local memory store supporting, controlling and providing access for communications transactions. The advantage proposed is that it allows the agent's '*developers to concentrate on the domain-specific issues of their application*'. The agents are homogenous, with specific operations determined by a '*plugin*'. The Blackboard contents are tasks, assets and '*PlanElements*'.

**Principled Blackboard Intelligent Agent Architecture**

Corkill [2005] explains that 34 participants at a DARPA-based workshop held in December 2001 recommended '*the technical foundations for an advanced high-level data fusion approach*' for an Intelligent Agent Architecture, uniting the Blackboard control methods advanced by Erman *et al.* [1980], Corkill [1991], Carver and Lesser [1992], '*with a more "principled" Blackboard representation*'. The advantage argued for Blackboard Intelligent Agent Architectures is control flexibility, as the agent operations can be either data-directed or model-directed as appropriate.

### 2.5.2.3 Summary of Blackboard Agent Architectures

The Blackboard agent architecture can be adapted for testing prototype Artificial Intelligence, as the architecture allows the design of an external '*intelligence*' program for use as its Knowledge Sources. The BB1 architecture provides significant functional evidence for using a programming language to create the Knowledge Sources, together with the COUGAAR architecture which is developed to allow external Knowledge Source design The transmission of a prototype Artificial Intelligence to a receiving intelligent agent is illustrated as a block diagram in Figure 2.3.

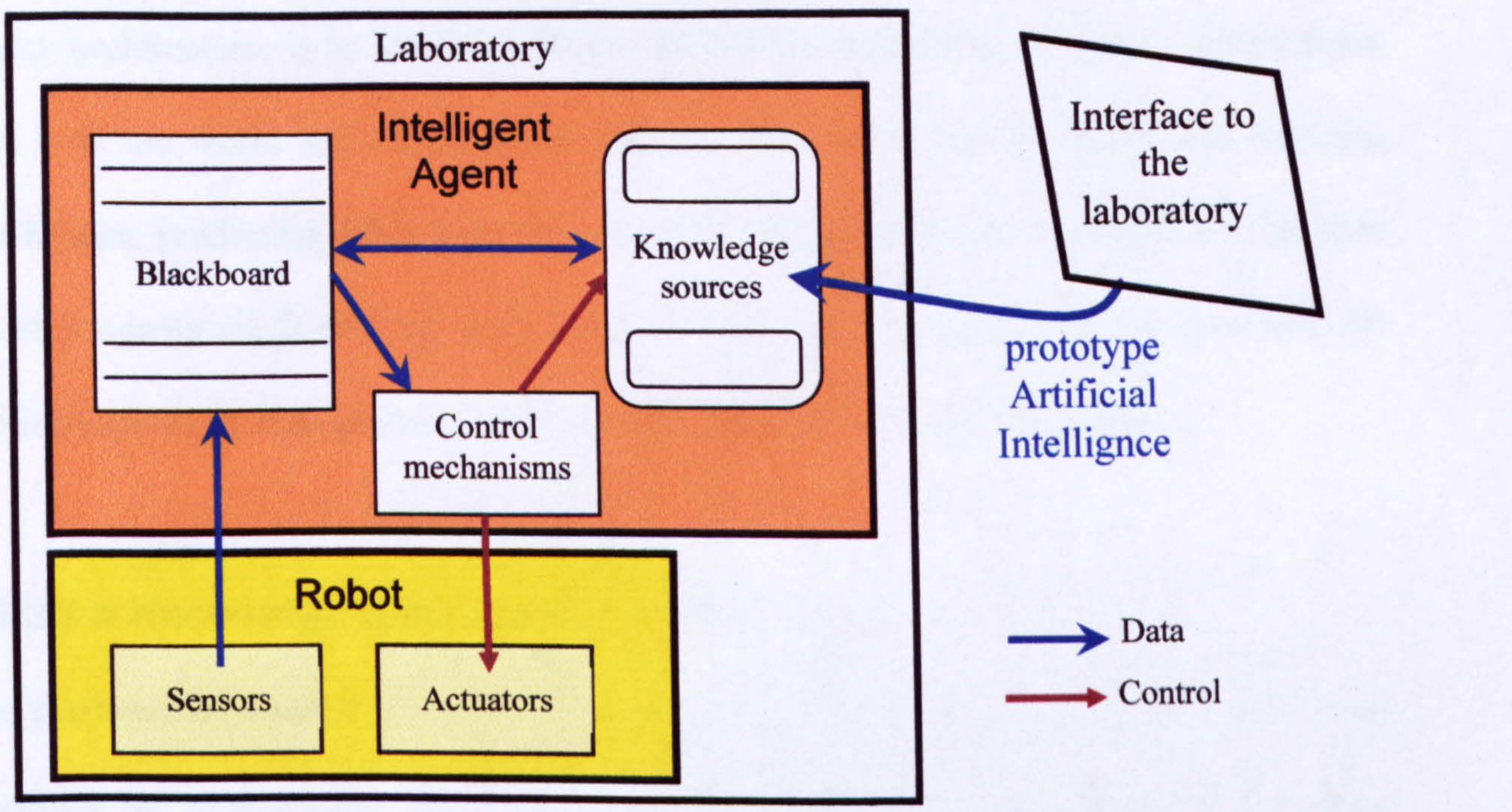



Figure 2.3 A blackboard agent architecture for the Remote Access Laboratory

Sadeh *et al.* [2001] arguing the success of blackboard architectures to integrate multiple knowledge sources to support a diversity of applications, establishes the concept of a Blackboard Agent which can use a range of mechatronic devices. Corkill [2005] continues the argument with the principle of using either data-directed or model-directed blackboard architectures. This establishes an architecture format compatible with the expectations of the prototype Artificial Intelligence format being proposed for the user to develop. The prototype Artificial Intelligence is based on modelled actions, switching to data-driven actions when interacting with the environment.

## 2.5.3 Specific Agent Architectures

### 2.5.3.1 An Agent Architecture for Tracking Other Agents

The researched laboratory is intended to operate multiple mechatronic devices requiring agent tracking for intelligent interaction between the devices. This involves observing other agents' actions, and inferring the high-level goals, plans and behaviours, [Tambe and Rosenbloom, 1996].

The solution proposed by Davidsson's [1996] Linearly Quasi-Anticipatory Autonomous Agent Architecture, is to include a World Model for predicting an agent's interactions with both the world and other agents. Davidsson reported that the agent had tracking capabilities, predicting other agent's actions by evaluating what the reactive behaviour of other agents are if nothing unexpected occurs, and preventing interference with its own objectives by manipulating its own formulated action plan accordingly.

### 2.5.3.2 A Knowledge Base Agent Architecture

The proposed laboratory is expected to operate more than one mechatronic device, each operated by a prototype Artificial Intelligence, without co-operating. A proposed knowledge-base agent architecture design, would combine current and existing data to store, refer and retrieve current knowledge, and access; manipulate and modify existing databases' data, collaborating and cooperatively processing with planning agents. McKay *et al.* [1996] argues for an optimal architecture to comprise the following capabilities: accepting an agent's query; generalising the query to access, retrieve and compute both the sought and related knowledge, before returning the knowledge to the enquiring agent.

**Queen Mary Virtual Laboratory**

Norman and Jennings [2002] describes the Queen Mary Virtual Laboratory used Knowledge-base agents as Mediator agents, which develop and maintain an abstraction of the system knowledge or resources. Mediator agents occupy a distinct, active layer between user-orientated information processing and resource management.

### 2.5.3.3 Multi-Agent Architectures

The structure of the agents and configuration that might be used is now considered. Since the early 1990s research has increasingly focused on the design of Multi-Agent Systems with their inter-agent interactions. The design of a multi-agent system includes the agents' ability to communicate with each other. Doran *et al.* [1997] argues that multi-agent system communication is based on three concepts:

- **Deliberation**: The agents plan actions to collaborate.
- **Negotiation**: The agents plan actions in competition with each other.
- **Co-operation**: defined as '*...a property of the actions of the agents involved*', arguing that agents are cooperating when a goal is achieved which no agent can accomplish alone, and achieve both their own, and other agents' goals.

Further, co-operation can be classified by the agent's motives:

- Self-interested co-operation: an agent co-operates to achieve its own goals.
- Altruistic co-operation: an agent acts for a group's interests, without furthering its own goals.

There are two methods of designing multi-agent systems from theory, using either specification languages such as Z, [Luck and d'Inverno, 1995], UML, [Odell *et al.*, 2000]; Logic Programming, [Thielscher, 2005], or software engineering methods, as described below.

**ARCHON**

Jennings *et al.* [1995] describes ARCHON as a decentralised software engineering methodology, used as an agent's design framework. The agents localise the system's objectives and are the smallest possible coherent autonomous entity, determined by overall efficiency, with a system having a large number of agents. The agents' goals are often interrelated, requiring interaction (controlled by the agent's ARCHON multi-agent interaction Layer) to meet global constraints and provide services and information.

Each agent's design either reuses existing software control systems, or uses control systems specifically designed for ARCHON functionality. The methodology consistently integrates multiple knowledge and data types, and produces partial results during component failure, due to overlapping functionalities.

**Gaia Architecture**

Wooldridge *et al.* [2000] reports that the Gaia architecture is intended to analyse and design agent-based systems, with explicit assumptions that the agents have access to computer resources equivalent to a UNIX process, and that the agents collaborate to achieve a global goal, without conflict(s). The Gaia architecture comprises a diversity of fewer than 100 separate agents' architectures including both a static multi-agent architecture and a static runtime operation.

The multi-agent interactions are function-orientated, epitomised by their communication. The inter-agent communication model is described as a directed graph, and includes an acquaintance model to define the communication links, and identify any potential communication bottlenecks.

**Challenger**

Chavez *et al.* [1997] reports '*Challenger consists of Intelligent Agents which manage local resources individually and communicate with one another to share their resources...*' The agents have relatively simple behaviours and use local information by inter-agent communications to achieve desired objectives and share available resources.

The design maximises performance by two properties:

- **Robustness** as it is argued that a multi-agent system with a centralised blackboard fails if the blackboard fails, the Challenger multi-agent system is decentralised.
- **Adaptability** the Challenger agents quickly adapt in a dynamic environment, providing minimal performance in the worst-case scenario.

Challenger analyses network delays, calculating the average response time, and develops a world model of inter-agent interaction as form of agent tracking. This world model of interaction development allows predictions of future interactions and determines the time taken for another agent to finish a job.

### 2.5.3.4 Summary of Specific Agent Architectures

The agent tracking and the knowledge-base agents provide examples of specific agent architecture design, utilised in multi-agent systems. The ARCHON and GAIA architectures both use a variety of agent architectures, ARCHON re-using existing software structures and GAIA allowing analysis of agents' designs for development.

Challenger as a multi-agent system was used to negotiate agent cooperation and collaboration, with the intent of maximising performance.

## 2.6 Conclusions about a Distance Learning Laboratory

The research problem posed concerns the feasibility of a remote access mechatronic laboratory to test a prototype Artificial Intelligence for intelligent behaviour. The following is concluded from the research reported above.

- A remote access laboratory is encapsulated by distance learning theory, serving the objectives of a distance learning institution.
- The learning theory which best fits an engineering remote access laboratory in distance learning requirements is the (Re)conceptualisation cycle, advocated by Coventry [1995]. This approves the teaching of theory, by using a laboratory for testing newly learned knowledge.
- For instructional laboratories, the experimentation should be representative of the real world for applying the associated theory.
- Advances are being made in the theory of design and use of laboratory formats which support the concept of a remote access laboratory.
- The current principles for distance learning laboratories are either using simulator technologies, or physical attendance to a laboratory. Attending a physical laboratory is unsatisfactory for distance learning, and research in simulation established its lacking in imitating the real world, as the simulator only operates within the limits of its programmed parameters.
- Collaboratory technology is providing evidence for the potential of remotely accessible prototyping laboratories, but its purpose is not conducive to collaborative interaction.
- The blackboard agent is identified as the premise for architectural development of a remote access laboratory, and is considered in Chapter 4. Research on how to access a prototyping laboratory by having an appropriate Human Computer Interface is considered in Chapter 3.

This research aimed to establish the viability for a physical remote access laboratory, which would not require attendance at its location, re-asserting the any-time, any-place principle of Distance Learning. The internal architecture of such a Remote Access Laboratory would include a multi-agent system. This is argued to have a blackboard agent to facilitate testing prototype Artificial Intelligence, using the prototype Artificial Intelligence as the agent's knowledge sources. A laboratory operating two or more mechatronic devices, needs agents able to track other agents' operations and negotiate when necessary. As a part of a multi-agent system there is a need for Knowledge-base agents, used for storing the laboratory's knowledge, including the intended students' successful experiments for analysis (and tutor's assessment). The prototype Artificial Intelligence is designed and developed by use of Programming by Demonstration, discussed next in Chapter 3.

# Chapter 3

# State-of-the-Art for an Interface to a Distance Learning Laboratory

## 3.1 Introduction

This chapter initially discusses the principles and concepts of Intelligent Training Systems within a distance learning environment; before examining specific, interrelated areas of research to establish the current state-of-the-art concepts and techniques necessary to design the interface for a distance learning Remote Access Laboratory. The discussion follows the progression:

- **A Distance Learning Environment**: a review of the current theories advocated for distance learning, and how they relate to a successful Remote Access Laboratory interface.
- **Human Computer Interfacing:** a review of the expectations and current theory for a Remote Access Laboratory interface. This critiques the current theories advocated as relevant for Human-Computer Interfaces.
- **Programming by Demonstration:** relates to the design of software which allows a user to interact with tools for creating a program without the use of a programming language, and includes an explanation of the theory.

## 3.2 A Distance Learning Environment

> *Engineering education is science and mathematics based subjects that are traditionally the hardest to teach online because of the need for laboratories and equation manipulation.* [Bourne *et al.*, 2005]

A concern already expressed arises from Chapter 2.2, with distance learning theories abrogating the importance of teaching a body of theory. Mechatronics as an engineering subject has theory as a manifestly important component.

### 3.2.1 Intelligent Tutoring Systems (ITS)

Intelligent Training Systems reflect emerging learning theories, with the teaching process divided into four separate functions: the planning of teaching actions; the monitoring of these actions with students; diagnosing any discrepancies between a student's behaviour and the expected outcome, and determining and correcting an error, [Siemer and Angelides, 1998; Asami *et al.*, 1998; Sison *et al.*, 2000].

Research in ITS has been led by a desire to identify and rectify student errors. Whilst there is no standard for ITS, a consensus is that ITS should comprise: a domain model with the knowledge about the domain to be taught; a student model with the representation of the emerging knowledge and skills, and a tutoring model to design and regulate instructional interactions with students, [Siemer, 1998].

While each model includes processes necessary for tutoring interaction, the systems shortcomings are: lack of complete domain knowledge, which was a deciding factor for Asami *et al.* [1998] to limit the system's scope. Further incomplete domain knowledge can lead to behavioural errors, and there can be many correct approaches to a solution, [Siemer 1998; Asami *et al.*, 1998; Sison, 2000]. Sison [2000] proposes a solution of unsupervised learning, where knowledge level errors are not known beforehand, with multiple classifications of a single object and variability allowed in the programming. Intention-based diagnosis should be made. Baker [2007] investigates the issue of off-task behaviour, and argues the student may be using the system as a game.

Ramadhan [1997] considers that Intelligent Tutoring Systems can be categorised by their primary means of solution analysis:

- Systems that can diagnose partial solutions, either using passive analysis which does not trace the user's intention or the design decisions, or active analysis, subdividing the system into smaller steps, and predicting if the user is following a correct design path.
- Systems that require entire solution code, which can be further subdivided according to its error handling: specification-based analysis, [Crowley and Medvedeva, 2005]; trace-based analysis, [Trella *et al.*, 2005]; I/O based analysis, [Butz *et al.*, 2006], and model answer based analysis, [Moritz *et al.*, 2005].

### 3.2.2 Design Considerations

Sherry [1996] provided a peer review on design considerations, arguing there are 5 factors affecting successful distance learning provision:

- **Systematic design and development** the laboratory flexibility has to allow for advances and developments, citing Willis [1992].
- **Interactivity** between the laboratory and users and amongst the users, citing Garrison [1990], NcNabb [1994].
- **Active learning** the users' involvement in their own learning, including understanding the material presented, citing Saettler [1990].
- **Visual Imagery** instructional images without '*oversimplification*' or '*superficiality*' citing White, [1987] or becoming entertainment, citing Ravitch [1987].
- **Effective Communication** so the user perceives things as intended by using appropriate objects with relevant attributes, citing Horton [1994].

### 3.2.3 Summary of a Distance Learning Environment

The system design to allow improvement requires structural flexibility, for continued future use. A rigid inflexible system would require significant alteration with any tutored course changes. Morse and Truman [1996] reports that while distance education and computer technologies augment the educational process, computer technology does not in itself improve education, but is only an enabling tool to be used in innovative and effective ways.

## 3.3 Human Computer Interfacing

> *One of the main impediments to an expanding role of robotics in society is the current difficult and unnatural programming interfaces available... [an] approach to robot programming is Programming by Demonstration (PbD)... ...Such a programming interface is very natural for a human to use, it does not require specialist knowledge, and can potentially program very complex tasks.* [Chen, 2005]

Rogers [2004] states that during the 1980s and '90s Human Computer Interface (HCI) designers referred to '*memory, attention, perception, learning, mental models and decision-making*' cognitive models to understand computer users' performance. Familiar cognitive models helped designers with design characteristics, and the cognitive theories helped with design decisions, consequently developing icons to improve user interaction. The problems are the fragmented and slogan-based adoption of psychology findings, and a partiality for citing singular research findings extensively and ignoring the original research context, [Green *et al.*, 1996].

### 3.3.1 The Theories of Human Computer Interfacing

Wright *et al.* [2000] explains most '*models of interaction are task-based*' and a task is '*the way in which a goal is attained taking into account factors such as competence, knowledge and constraints*', citing Card *et al.* [1983], Johnson [1992] and Green *et al.* [1988]. Further Wright *et al.* [2000] refers to Suchman [1987] to premise their proposed '*distributed information resources model*', where an interface includes abstract information structures to identify action resources, and the reference knowledge is distributed between the interface and its user. Wright *et al.* [2000] specifically proposes that the model would apply to single user interaction with an interface.

Rogers [2004] explains how Cognitive Theory failed as an HCI model in the 1980s, resulting in two new models to conceptualise and understand the assumed interactions taking place between a user and a computer: the Model Human Processor (MHP) and GOMS (Goals, Operators, Methods and Selection rules). Card *et al.* [1983] argues that MHP was a premise for predicting computer interface-user performance, and assessing the HCI's suitability for supporting various tasks. MHP was developed further with a set of predictive models, collectively referred to as GOMS. The various models and significant references are compared in Table 3.1.

There are four theories represented in table 3.1. GOMS is dismissed as a theory due to its limitation of user interaction to data entry tasks, and not modelling flexible interaction. Distributed Cognition and External Cognition have a problem of not modelling user input, and Distributed Cognition ignores the activity involved. Activity Theory, while providing solutions, requires experimentation with the proposed interface design to determine effectiveness.

Table 3.1 Comparison of HCI design theories

| Theory | Author | Interface design | Activity modelled | User input | HCI design | User interaction |
|---|---|---|---|---|---|---|
| GOMS | Olson and Olson [1991] | Assists in new product effectiveness | Limited to data entry tasks | Predictable behaviour | ------ | Does not model flexible interaction |
| Distributed Cognition | Wright *et al.* [2000] | Present knowledge vital to achieve task | ------ | ------ | What the user needs to know | Does not take account of action. |
| | Zhang and Norman [1994], Zhang [1996] | Capture rules in design | Design does not consider tasks | ------ | ------ | Assimilate rules but ignores actions |
| External Cognition | Green *et al.* [1996] | '*Cognitive Dimensions*' | Observe behaviour | ------ | Abstract dimension types | Observed cognitive behaviour |
| | Rogers [2004] | '*fundamental properties and design dimensions*' | Understood '*cognitive effort*' | ------ | Optimum '*interactive content*' | Can guide users' decisions |
| Activity Theory | Kuutti [1996] | HCI operating at several levels | Cognitive mediation | Depends on work practice | Consider HCI at several levels | Information system research |
| | Mwanza [2001] | Based on '*Activity Triangle Model*' | Analysed work practices and tools | Modelling work practice | Interpreted findings | Modelled activity system |
| | Béguin and Rabardel [2001] | Initial HCI design imprecision | Activity process of designing. | '*Catacreses*' | '*Instrumental genesis*' | Observe the user construct '*the tool*' |

### 3.3.1.1 Distributed Cognition

Roberts [1964] heralds the concept of socially distributed cognition, with Wright *et al.* [2000] citing Norman [1988] to propose knowledge as a function of both the world and the person's cognitive ability. The information an interface presents is argued as important to achieving a task, as the user's knowledge of the interface, implying that an interface's design involves considering the knowledge a user needs to know, and recall.

Suchman [1987] considers plans as representing possible action courses, arguing that they are subject to the consciousness, so can be manipulated and evaluated. Young *et al.* [1990] argues that a novice HCI user interprets choices to select which is appropriate, with Zhang and Norman [1994] presenting experimental evidence using Towers of Hanoi to demonstrate with disks, that the rule of not placing a larger disk on a smaller disk is held in the subject's memory, but a Russian Doll version captures the rule in its design. Scaife and Rogers [1996] advises considering how an interface can influence thinking and reasoning, while Zhang [1996] did not consider the role of tasks (actions) for designing displays.

Nardi [1996] [2002] advocates Activity Theory and criticises Distributed Cognition's usefulness to HCI, focusing on the extensive fieldwork required to obtain any conclusions or design decisions. The theory is considered significantly harder to apply than activity theory, as there are neither identifiable explicit data characteristics nor readily usable analytical methods. Rogers [2004] states that Distributed Cognition is not a quick-fix prescriptive method, but instead requires the interface designer to be accomplished in data analysis and uniting both detailed and abstracted investigation levels, instead demanding considerable time, effort and skill. For the HCI being prototyped, while allowing for various actions, the plans for actions are constrained. This limits the user action plans by defining the rules of the system within the system's state.

#### 3.3.1.2 Activity Theory

Kaptelinin [1996] explains that Activity Theory originated from Soviet philosophy of analysing tool use by a subject to achieve an object/objective, arguing that the object(ive) motivates the activity (tool use), and specifies the activity's direction. Kuutti [1996] argues that mediation involves a form of cognition.

Mwanza [2001] explains how Activity Theory methodologies were developed to analyse both organisations work practices and the supporting computer system design, originating with Vygotsky [1978] '*Mediational Model*', Figure 3.1. Further, Engeström [1987] augments Vygotsky's concept with a hierarchical model of human activity, establishing the expanded '*Mediational Model*', figure 3.2, to imitate human social activity.

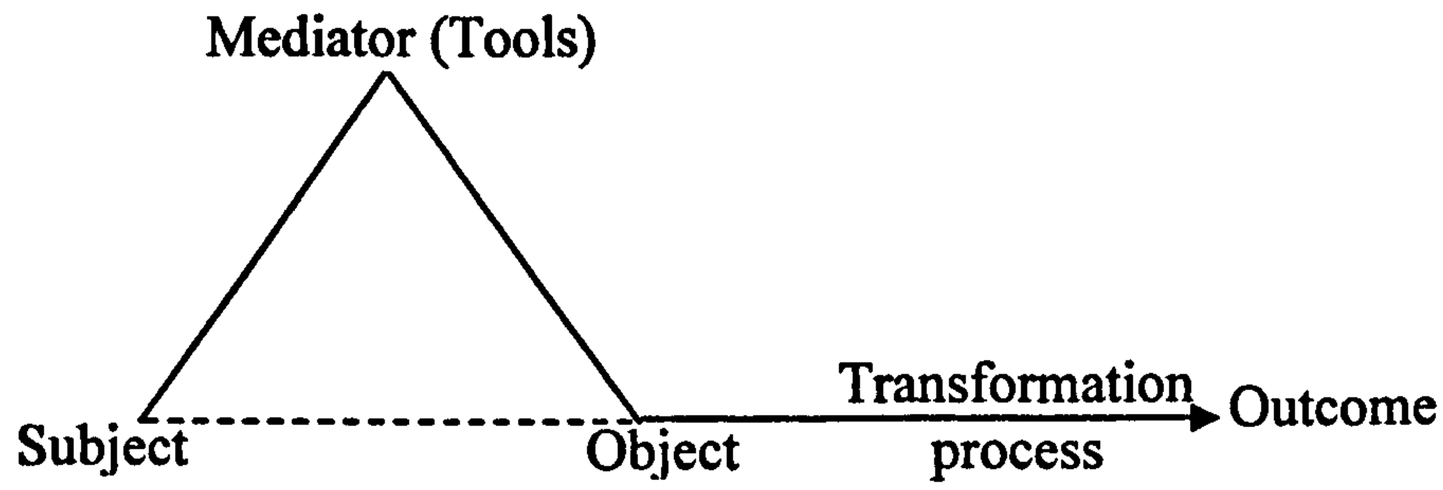



Figure 3.1 '*Mediational Model*' Mwanza [2001]

Mwanza [2001] argues that the Activity Triangle Model is a heuristic model capturing and unifying relevant concepts, providing a basis for interpreting and applying the theory, and placing an activity in a social and cultural framework. Nardi [1996] explains that the theory's advantage is its easily understood vocabulary, but there is no standard method for implementing the theory. Kaptelinin [1996] argues that the inapplicability of Activity Theory is due to its multiple basic principles for analysis, with an evolving theory framework, creating a variation in interpretation and application.

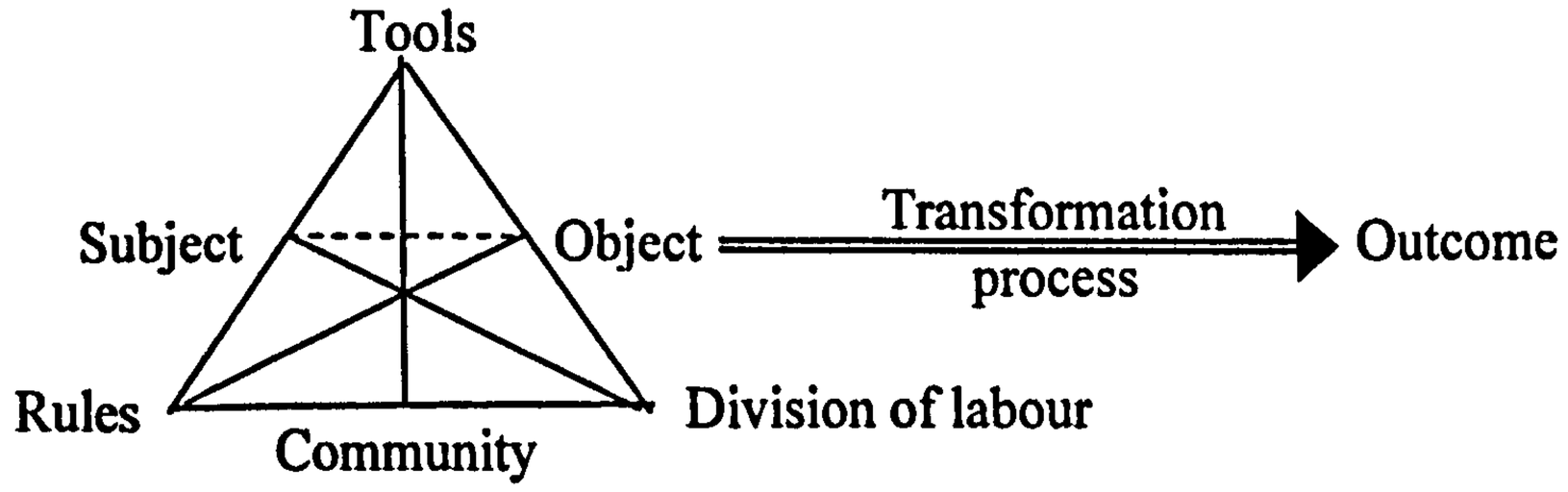



Figure 3.2 The basic activity structure, '*Activity Triangle Model*' Mwanza [2001]

Mwanza explains that the process of implementing the Activity Theory using the Activity Triangle Model involves: modelling the situation being examined to create the situation's Activity System; decomposing the situation's Activity System; generating research questions and conducting a detailed investigation, finally interpreting the findings. Further implementing the acquired model involves method validation and evidence of transferred and implemented theory. This complicated implementation method leads to a depiction of applying Activity Theory without explaining how the theory was applied.

**Catacreses**

'*Catacreses*' are explained by Béguin and Rabardel [2001] as potential development and evolution of an Activity's '*tool*' from its original design. Catacreses are proposed for adoption in Activity Theory, since an activity should include obtaining, or constructing, an appropriate tool to achieve the attempted object(ive). The tool's development is considered as '*instrumental genesis*', including spontaneous or planned systematic adaptation of a tool's function or structure.

Béguin and Rabardel [2001] analyses the design process using a computer, with a designer creating something unknown and, for design flexibility, with persisting uncertainty. During early computerised design, imprecision is '*impossible*' and argued incompatible with the designer's need and hindering the computer's mediating role, due to imposed design constraints through accurate data entry requirements. The result is a data entry constraint that initially compromises the design, before the design is modified.

### 3.3.1.3 External Cognition

Nakakoji and Yamamoto [2003] explain that external cognition is the external representation of an entity, and in HCI the emphasis is on a graphical (iconographic) external representation of a process.

Green *et al.* [1996] claims that '*cognitive dimensions*' suitably abstract different dimension types across applications, with solutions applicable to comparable problems. Designers and researchers consider '*cognitive dimensions*' easier to understand and learn than Activity Theory, as they encourage considering design solution trade-offs, and observed cognitive behaviour.

Rogers [2004] stated that her approach to '*fundamental properties and design dimensions*' can guide users' decisions, informing and confirming the external representation for the activity being designed, with Zhang [1997] proving that presentation is critical to the problem solving process.

Rogers argues that her methodology can establish the optimum '*interactive content*' structure and presentation with reference to understood '*cognitive effort*', and '*computational offloading*' which is the degree various icons affect the cognitive effort required to carry out different activities.

While external cognition has a proven track record in HCI with such icons as   and  used to indicate print, the software package used for the interface's design and prototyping did not allow for the adoption of iconographic buttons.

### 3.3.2 HCI Design in Practice

Kuutti [1996] considers that HCI has garnered the eminence of a research subject, citing Carroll [1987] for perpetuating the assumption HCI is '*grounded in the framework of cognitive science*'. Kuutti's rebuttal comprises '*Research is not ahead of practice*', instead current research is identifying why HCI solutions perform, and refutes the '*framework of cognitive science*' as the related theory is fragmented, disconnected and incoherent. Further Kuutti considers that there is a divide between HCI research and design, citing Bellotti [1988] survey of designs were not based on HCI research.

Kuutti [1996] cites Grudin [1990] reporting that HCI design has evolved from hardware '*outwards*', and Friedman [1989], with Human-Computer problems being relative to each other, with decreasing importance. Both Grudin and Friedman agree that problems are not solved, but when contrasted with new larger problems, recede in importance. Kuutti criticises Grudin for '*computer-centrism*', while accepting the advantage of considering HCI operating at several levels to solve conceptual problems and confusions, but Kuutti argues the theories did not help relate the concepts together.

Rogers [2004] argues for clarity in the HCI theories' intentions, and cites the Shneiderman [2002] five required theory formats in HCI: descriptive, for '*providing concepts*'; explanatory, for '*relationships and processes*'; predictive for '*user performance*'; prescriptive for design guidance, and generative to enable discoveries. The argued problem is that HCI designers have an unmet demand in a multitude of purposes from the theory, and while there is a need for both theory and application methods, theory has difficulty in providing the application methods.

Rogers proposes that designers be researchers, and that theory-based HCI designs can contribute to the adoption of new techniques with the development of a design language for future research and design. Rogers believes a common language is increasingly essential with HCI expanding in designers, products and users, requiring replacement of the theoretical jargon for the non-theorist and allow a greater knowledge transfer between designers and theorists. The problem is designers need advice at design time but researchers confirm design correctness after implementation. An answer is iterative design, involving users in the design process, but identifying real-life circumstances becomes a problem. The current alternative is to allow a user to personalise the HCI.

### 3.3.3 Summary of Human Computer Interfacing

Rogers [2004] explains there is a re-emphasis for a theoretical component to parallel the procedural and theoretical developments in HCI interfaces interactive design. Barnard *et al.* [2000], Hollan *et al.* [2000], Kaptelinin [1996], and Sutcliffe [2000] argues for a theoretical foundation to HCI design, while Castel [2002] argues that there is a lack to current HCI design.

Rogers argues that while the early application of cognitive human memory theories to HCI optimised icon design and command names, cognitive theories are based on experimental conditions not a workplace. Both Hollan *et al.* [2000] and Bourguin *et al.* [2001] agree that psychology based HCI premised on the '*Human Information Processor*' is limited, and new HCI developments can be supported by psychology's theoretical and experimental structure, but disagree with each other about the theory. Hollan *et al.* [2000], advocates Distributed Cognition, Bourguin *et al.* [2001] Activity Theory. The differences are the subject(s) of study is/are both the user and HCI for Distributed Cognition and just the HCI as a tool for Activity Theory.

Barab and Plucker [2002] presents Distributed Cognition philosophy as external objects changing the system during activity, and affecting the user's knowledge citing Cole and Engeström [1993]; Perkins [1993]; Salomon [1993]. Distributed Cognition requires abstracting knowledge and action together, and by formalising this knowledge the ability and talent are distributed across a system and not embodied only in a person.

Activity Theorists are rebuked for focusing on activity to transform an object and the desire to appropriate an entire system's activity, without concern for isolated activity. Activity Theorists consider activity is distributed across subjects and tools from a subject's community relative perspective, but ignore both the person's state of mind, and the environment. The problem with modelling activity in an organization is the "Hawthorn Effect": people respond the way they think that the researchers want them to respond. The problem with any analysis of how people work is the reasons people go to work, identified in Human Resource Management as a mix of tangible goals, money and promotion, and intangible goals, satisfaction and self-esteem. Maslow's Hierarchy of Needs expresses this as the issue of Safety Needs with job security, and Social Needs with meeting people. When workers are studied, the need for job security is enhanced and their social needs are suppressed.

## 3.4 Programming by Demonstration

Ehrenmann *et al.* [2001] argues that as robots are adopted by a consumer market, the consumer will reject modern robotic user interfaces, and programming techniques. Biggs and MacDonald [2003] considers that most people have minimal technical skills requiring easier, flexible programming systems

This research determined that Programming by Demonstration (PbD) Systems cannot be classified into mutually exclusive types. Subsequently to reflect the current dynamism of research in PbD, this report's taxonomy of PbD systems is based on knowledge: how it is obtained, and used.

- **How knowledge is obtained by the system:**
    - **Direct Programming**: Text Based Systems, Iconic Programming

- **Indirect Programming:** Programming by Observation, Automatic Programming, Programming by Demonstration

- **How the knowledge controls the robot:**
  - Reactive Systems
  - Deliberative Systems
  - Hybrid Systems
  - Behaviour Based Systems
  - Hierarchical Architectures

As can be identified, this form of classification does not exclusively describe each PbD System. All PbD have some form of knowledge: input; format; storage; and operation.

## 3.4.1 Programming Methods

### 3.4.1.1 Direct Programming

Wright and Cockburn [2005] explains that direct programming is when a robot's knowledge/behaviours are programmed using graphical or text-based systems The methods include adjustable preferences and defaults; macros based applications, and scripting languages. Lau and Weld [1998] explains that each technology resolves a programming issue, but has an associated limitation. Adjustable preferences and defaults interfaces are simple to use, but limited to operations considered and implemented during its design. Macros allow action sequence creation, but limited when variations are required during task repetition. Text-based programming is based on traditional programming languages. Scripting Languages create sophisticated control sequences, but require programming experience, with, knowledge of the scripting language and application interface.

This research is expected to overcome the '*traditional*' programming language problems allowing a user to develop skills to robotic behaviours design.

### 3.4.1.2 Indirect Programming

Wright and Cockburn [2005] explains that a robot's knowledge is programmed using Learning Systems: Programming by Demonstration, and Instructive Systems. Learning from a tutor providing, and explaining, case examples, is called: Programming by Demonstration (PbD), [Lieberman, 1993, 1994; Friedrich *et al.*, 1995; Schaude and Dillmann, 1995]. Chen [2005] explains that this form of programming is argued '*natural for a human to use, it does not require specialist knowledge, and can potentially program very complex tasks*'. Biggs and MacDonald [2003] explains that these programming systems, do not allow direct control of a robot. The system generates the robot's control code from the information entered into the system.

Previous research in learning techniques divided tasks between development, and utilisation. During development is world model creation, the internal knowledge of the world, and initial program operations. For utilisation is action and knowledge refinement. Kaiser *et al.* [1995] reports two causes for the human-machine interaction, first a task the system cannot perform. Second, a specified object is unknown.

In programming by observation the system learns by detecting and disseminating a teacher's demonstration. Voyles and Khosla [2001] explains a new task is demonstrated in real-time, without special behaviour requirements, or additional time. The only requirement is wearing tactile sensors: glove or fingertip coverings. The cause for errors is argued due to a robot's unstable grasps.

### 3.4.1.3 Learning by Human Demonstration

Voyles and Khosla [2001] cite Patrick [1992] and argues

> *Programming by Demonstration is the most natural paradigm for human programmers, because, training by demonstration and practice is the most often used method between humans.*

The assumption is when demonstrating a task robots can identify and parameterize the skills required by a task. An identified problem is: a robot dependency on both a teacher's ability and experience in providing knowledge or instructions, and the format of system instruction, either at skill or task level. Ehrenmann *et al.* [2001] argues that the PbD aims are to generalise and abstract the demonstration, to reflect a user's intention, and optimally model the problem solution. The best demonstration is a generalised problem with parameters for distinguishing both the specific problem being solved and a solution being demonstrated.

Nicolescu [2003] argues that humans learn by complex interaction and instruction methods, comprising demonstration; instruction, and directive cues or gestures. The learner relies on task demonstration; supervised practice trials with rectifying feedback, and additional demonstrations to learn the generalisation. Complexity results in both effective teaching and learning. In contrast, during Learning by Demonstration, most of this complex interaction is overlooked, with instead the use of only 1 or few interactions. Nicolescu reasons that additional teaching

> *significantly improves the learning process by conveying more information about the task, while ...allowing for a very flexible robot teaching approach.*

Callinon and Billard, [2007] addresses this issue by adding a social component to the teaching process and user interface.

This research developed a PbD system which is argued to have improved usability when developing a program. Edwards [2005] cites Norman [1988], for identifying usability problems, referred to as the '*Gulfs of Execution* and *Evaluation*'. Explaining the '*Gulf of Execution is the difficulty of translating a desired goal into an action to be executed*', and the '*Gulf of Evaluation is the difficulty of determining whether an observable state meets the desired goals*'. These two gulfs are argued as a direct result of text based programming. The Gulf of Evaluation is not understanding a program's text, a task argued only a computer can do reliably. The Gulf of Execution is a small change to the program text, renders the program invalid.

Circumventing these problems has led to partitioning large programs into smaller program components: Modules; Functions and Procedures. Edwards summarised the two gulfs problem with

> *...a major reason that programming is so hard is that text strings are a poor representation for programs*. [Edwards, 2005]

### 3.4.2 Programming by Demonstration Methods

Lau and Weld [1998] states that traditional PbD systems comprised: a Trace Generaliser to construct a program from a demonstration, including recognising conditional constructs, and an Interaction Manager, which describes a resulting program to the user, and obtains program execution authorisation. Biggs and MacDonald [2003] argue that these systems deficiency are they imitate single demonstrations, without allowing for changes or errors. A researched solution has been to introduce intelligence, resulting in both more sensor and actuator information from the demonstration, and flexible task execution.

Kaiser *et al.* [1995] argues a robotic control system's learning ability depends on how knowledge is contained and accessed for control. The efficiency of a learnt task is based on the PbD system's skill-base; cognitive and reasoning abilities, and its interface. The Human Computer Interface is detailed more fully below.

A system's cognitive ability is its knowledge of objects, and the methods to detect and/or identify the objects. For this research, the system has limited cognitive ability, being unable to identify unique objects, but instead simple sensor activations, allowing a student to learn the principles of robotic programming, without the complexity of object identification, early in the learning process.

A system's skill base is argued the basic robotic abilities without requiring a real-world model, and is defined as: '*the learned power of doing a thing competently*' by Kaiser *et al.* [1996] and '*A pattern of activity which describes an aptitude or ability that achieves or maintains a particular goal*' by Nicolescu [2003]. Kaiser *et al.* [1995] define skill learning as '*perception action transformations involving no model knowledge, that represent basic capabilities of the robot*', defining tasks as '*sequences of actions that accomplish a complete goal directed behaviour*'. Voyles and Khosla [2001] argues that as a skill is difficult to quantify, it is difficult to program. As the skill involved increases, the less a system is suited to using traditional program languages.

Nicolescu [2003] argues that for any system to learn tasks directly, it is practical to supply a basic skill base. An approach to PbD learning tasks without having a pre-existing skill base has two problems: new skill learning may not use previously learnt skills, and complex tasks learning required learning both a skills set and sequencing.

Learning task complexity are argued due to: reactive policies map sensors directly to actuators, [Hayes and Demiris, 1994]; the progression of skills or tasks requiring explicit step sequences, [Kuniyoshi *et al.*, 1994]; the environment allows implicit sequence representation, [Brooks *et al.*, 1988]; Task complexity increases with higher level components and constraints, [Nicolescu, 2003]. The researched system skill-base comprised only the abilities to quantify sensor data, and determine unique motor outputs.

A system's reasoning ability is the mechanism allowing a program generated to perform a specified task, combining a system's skill-base with available object knowledge. Nicolescu [2003] argues that: if the environment does not influence behaviour, then the system should learn to reproduce demonstrated trajectories, a strategy where a robot will fail to achieve its goal in a dynamic environment. If the environment influences the behaviours of the robot, the system should learn task representations. This research system's reasoning ability is intended to transform the system user's intentions into motor actions, display received sensor data to the user, and determine what demonstrated user intentions are for the circumstances presented to the system/robot.

### 3.4.3 Programming by Demonstration Systems

There are several types of PbD systems described in this section, Reactive and Deliberative, Hybrid, Behaviour and Hierarchical systems.

Reactive Systems connect a robot's sensors to its effectors without using complex reasoning. Brooks [1986] argues that the results are rapid responses to unpredictable environments, and provide robustness. However, a reactive system does not maintain state or internal representation of the world and learning is limited to reactive policies.

Deliberative Systems use both sensory information and the stored knowledge of a world model, to determine next actions. The world model is either pre-programmed or developed from sensor information. Possible paths are planned using the world model to reach a given goal. The knowledge in the world model needs to be complete or highly detailed.

Hybrid Systems comprise both the deliberative element of reasoning possible paths to the goal, and the reactive element of immediate actions. Any conflicts between reactive actions and deliberative planned actions are resolved by a middle (arbitration) layer, [Gat, 1998]. The arbitration layer is considered the hybrid systems design challenge.

Matarić [1997] argues that Behaviour Based Control design requires a centralised world model with behaviours using '*fast, realtime responses, and similar representations and execution time*', unlike a deliberative system which uses behaviours that operate on different time scales. Molnar *et al.* [2004] uses an embedded Behaviour Based Control system for a submersible. While a Hierarchical Partial Order Execution Architecture uses a task structure, which is: '*dynamically expanded at execution time*' [Pearson *et al.*, 1993; Simmons, 1994; Tambe and Rosenbloom, 1995]; '*completely provided a priori*' [Nicolescu and Matarić, 2002].

A Hierarchical Abstract Behaviour Based Architectures uses two components: **perceptions, and actions,** to build the architecture. Nicolescu and Matarić [2003] refers to perceptions as '*abstract behaviour*', containing its pre-conditions, goals and '*primitive behaviours*': the actions. Activation of a perception depends on specific pre-conditions, and the previous abstract behaviour post-conditions.

A '*behaviour network*' is built by connecting sequential abstract behaviours, and networking the behaviour sequences. The tasks connect into flexible abstractions with increasing complexity, allowing behaviour reusability: creation of complex behaviour sequences, and flexibility to learn new tasks, and the behaviour network develops increasing abstraction.

The PbD system to be prototyped had a hierarchical behaviour control system, with reactive behaviours, allowing the prototype Artificial Intelligence to both operate with long term goals and resolve immediate problems.

## 3.4.4 Programming using Programming by Demonstration

Wright and Cockburn [2003] considers programming as three fundamental activities: Writing Programs; Reading Programs and Executing Programs.

### 3.4.4.1 Writing Programs

The knowledge transfer from designer to system using a representation the robot can store. Text-based programming is still the most common method of writing programs. Wright and Cockburn [2005] argues that the problems with text programming are difficulties in determining errors in the text, and correct programs can have execution errors, where an error can mask one or more other errors, or, accumulate to create an error which may be hard to diagnose and trace to sources. Ehrenmann *et al.* [2002] reports a robot programming method comprising demonstrating the actions performed between grasps, and during the grasps. Chen and McCarragher [1998], [2000], Chen and Zelinsky [2001] argues for using multiple demonstrations, as single demonstrations are rarely the optimal solution, and creates a flexibility of maximising speed or accuracy.

This research is based on providing a user with the ability to create intelligent behaviour comprising goal activated and, sensor activated action plans. The behaviours are demonstrated using diagrams. The Goal-Based Behaviours are designed by determining locations a robot is intended to move from and to. Goal-Activated Behaviours are activated at pre-determined goals, with Sensor-Activated Behaviours activated by sensor states.

### 3.4.4.2 Reading Programs

Wright and Cockburn [2005] argues that this is the understanding of stored knowledge. Onda *et al.* [2002] uses a virtual environment to perform demonstrations, allowing sensor information retrieval, and creating specialised behaviours. Zollner *et al.* [2002] reports using fingertip sensors to detect fine manipulation of objects. Kaiser *et al.* [1996] reports graphically viewing the complete demonstration results comprising viewing learnt demonstrated behaviours, and editing, rearranging, or using various segments separate of the learnt demonstration as reusable code.

This research allows a user to view the robot's behaviours, as designed, with a presentation of sensor values. The angle and distance text values can be edited providing precision as graphic design can be imprecise.

### 3.4.4.3 Executing Programs

Wright and Cockburn [2005] argues that this is observing either a simulation or a robot's performance of its knowledge. A simulator is often provided with a programming language, but use of the simulator requires staff training. This research provides a simulation facility, allowing analysis of behaviours with a detail a laboratory may not physically provided.

### 3.4.5 The Limitations of Programming by Demonstration

Witten *et al.* [1996] reports that a weakness in PbD is '*an inability to take advantage of domain knowledge or user hints*'. Nevill-Manning and Witten [1995] [1997] and Witten *et al.* [1996] reports a method '*for detecting hierarchical structure in sequences'* by determining patterns in the demonstration. While the methodology is argued as simple, it is not reported how to apply it to obtain appropriate knowledge.

The system being researched needs to provide a hierarchical architecture allowing both deliberative and reactive behaviours, and as such this allows planning based on sensor operations, and specific action plans. Further, the sensor based reactive behaviours are both context specific and generalised sufficiently to be recursive.

### 3.4.6 Programming Sub-Optimality

Friedrich and Kaiser [1995] identify sub-optimality causes as: demonstrations including '*unnecessary, incorrect, or unmotivated actions*'; or ambiguity about when the action(s) operate. The demonstration may not accomplish the intended task if '*the user does not know enough about the task*'. Further, unintentional sub-optimality is when unnecessary actions are included in the task, or an action is not included in the task.

Nicolescu, [2003] suggests that to prevent sub-optimality either include a user's intentions as data with various aspects of the demonstration, improving learning of the demonstration; or viewing task performance, to identify and resolve sub-optimality with feedback. Friedrich and Dillmann [1995] identifies as an issue that additional information is: '*burdensome for the teacher as he or she needs to provide (at each step) information on what goals she/he had in mind, and what actions/used objects were relevant*'.

Nicolescu [2003] argues that the immediate advantages of correcting a robot's observed task performance are: no knowledge is required about either the PbD system architecture, or, how the demonstrated task is coded.

Chen [2005] argues that any demonstration can anticipate inconsistencies at both control and task level, with a resulting sub-optimality for the demonstration. Further it is an important part of a PbD system to identify a sub-optimality as noise, and remove it at both task and control levels. This is due to robotic performance not being enhanced by optimised task control details without optimising the actions.

### 3.4.7 Summary of Programming by Demonstration

Programming by Demonstration is based on resolving the problem of programming without using a text-based programming language. The researched system being prototyped allows development of robotic behaviour as a hierarchy of tasks with action sequences, a method proposed and used by di Iorio *et al.* [2007]. The instruction system design needed to resolve the problem of understanding human demonstration, and resolve two separate issues. Firstly, the PbD system has to resolve the two problems of the Gulf of Execution and the Gulf of Evaluation. These are the ability of a programmer designer to understand the program designed, and recognise if a program has achieved expectations. Secondly, the PbD system has to resolve the issues of sub-optimal programming and noise. This is where actions are specified for a task but do not optimise the task's operations.

The systems ability to cognate, identify its surroundings is based on its sensors. For the purpose of tuition, this system does not uniquely identify macro objects, but allows the user to interact with sensor inputs, and relate these to reactive behaviours.

The user can both demonstrate expected behaviours and determine when behaviours should be acted on, in a hierarchical format. This though means that the user is presented with the design of the Hierarchy of Behaviours. The Sensor-Based Behaviours are recursive: the research has not provided evidence of other Hierarchical Behaviour Architectures with recursive behaviours. The prototyped architecture also includes Deliberative Behaviours, allowing the designer to create robot action plans. Further the researched system allows the design of actions based on sensor operations.

## 3.5 Conclusions for an Interface to a Remote Access Laboratory

The research problem posed concerns the environment for rapid development of a prototype Artificial Intelligence for intelligent behaviour. The following is concluded from the research reported above.

- Accepting the interface to the laboratory is only an enabling tool suggests Morse and Truman [1996]. For developing a prototype Artificial Intelligence, the design of the interface as a tool can be determined by Activity Theory.
- The problem with Activity Theory is the analysis of an interface's design by experimentation is subject to the "Hawthorn Effect".
- As a tool the Interface may be adapted by its users, and has to be allowed for, as this is catacreses, and can be analysed for '*instrumental genesis*'.
- There is a disparity between the theoretical approaches and application methods for HCI design. While Bourguin *et al.* [2001] and Hollan *et al.* [2000] both agree that HCI should have a psychological theory and experimental structure, but they disagree on the theory, arguing for Distributed Cognition and Activity Theory respectively.

- A problem with robot programming is the lack of wide-spread programming skills. A solution to this is not to '*program*' the robot but '*demonstrate*' the expected robotic behaviours, and create a program from the demonstration. There is a variety of demonstration methods, direct using graphical or text-based systems, and indirect which includes Programming by Demonstration (PbD). PbD is a method of programming which allows the programmer to demonstrate the intended behaviours for the robot to operate.
- The programming usability problems are the '*Gulf of Evaluation*' and '*Gulf of Execution*' Norman [1988], and PbD is limited further by its cognitive and reasoning ability.
- The proposed PbD system adds to the current body of knowledge by establishing a system where a robot is programmed without the use of a programming language. The current body of knowledge comprises the use of demonstration, which is either translated by video camera in Programming by Observation, or by physical manipulation of the robot, which is translated by means of a Trace Generaliser. Instead the programming is achieved by graphical demonstration.
- The PbD programming method proposed is anticipated to overcome the gulfs of evaluation and execution and prevent programming sub-optimality.

# Chapter 4

# A Proposed Architecture for a Distance Learning Laboratory

This chapter develops an architecture for a remote access laboratory, based on the state-of-the-art discussed in chapter 2, and for testing prototype Artificial Intelligence, the design of which was discussed in Chapter 3.

## 4.1 The Laboratory in Distance Learning

Chapter 2 identified the benefits and drawbacks of a distance learning laboratory education, assuming the experiment for education is established, the equipment works, and that the experimentation is non-deterministic and requires problem solving solutions. A difficulty arises when a distance learning laboratory has to replicate physical interaction, including different approaches to a problem, each to achieve a working solution. Can a distance learning laboratory deal with multiple solutions to the same problem or different approaches to problem solving? The laboratories discussed were either purely software simulations or emulations of the laboratory experience.

With the development of collaboratories, there is evidence that laboratories accessible over the Internet are feasible [Kies, 1997]. However these are under human control and supervision, and the question is whether humans can be replaced by intelligent agents, for supervising a laboratory's internal operations. Replacing humans is problematic for the physical maintainance of the laboratory for such actions as repairing a shorted cable, restarting '*crashed*' machines, actions that still lie in the realm of human operation. Intelligent Agents only operate within their host machine using its accessible actuators, sensors and communications.

The proposed laboratory is activated by intelligent agents, operating as control software, testing the prototype Artificial Intelligence, which can be modified as the experiments are continued.

## 4.2 A Proposed Laboratory Architecture

### 4.2.1 Influences on the Laboratory Architecture

The initial assumption was that the operations of the laboratory comprise three activities: testing the prototype Artificial Intelligence, amending it, and communicating with the laboratory user.

#### 4.2.1.1 The Prototype Artificial Intelligence

The prototype Artificial Intelligence is expected to operate available mechatronic devices, activating appropriate actuators, and determining valid and useful data from sensors, and relating sensor data to actuator commands, to achieve goals.

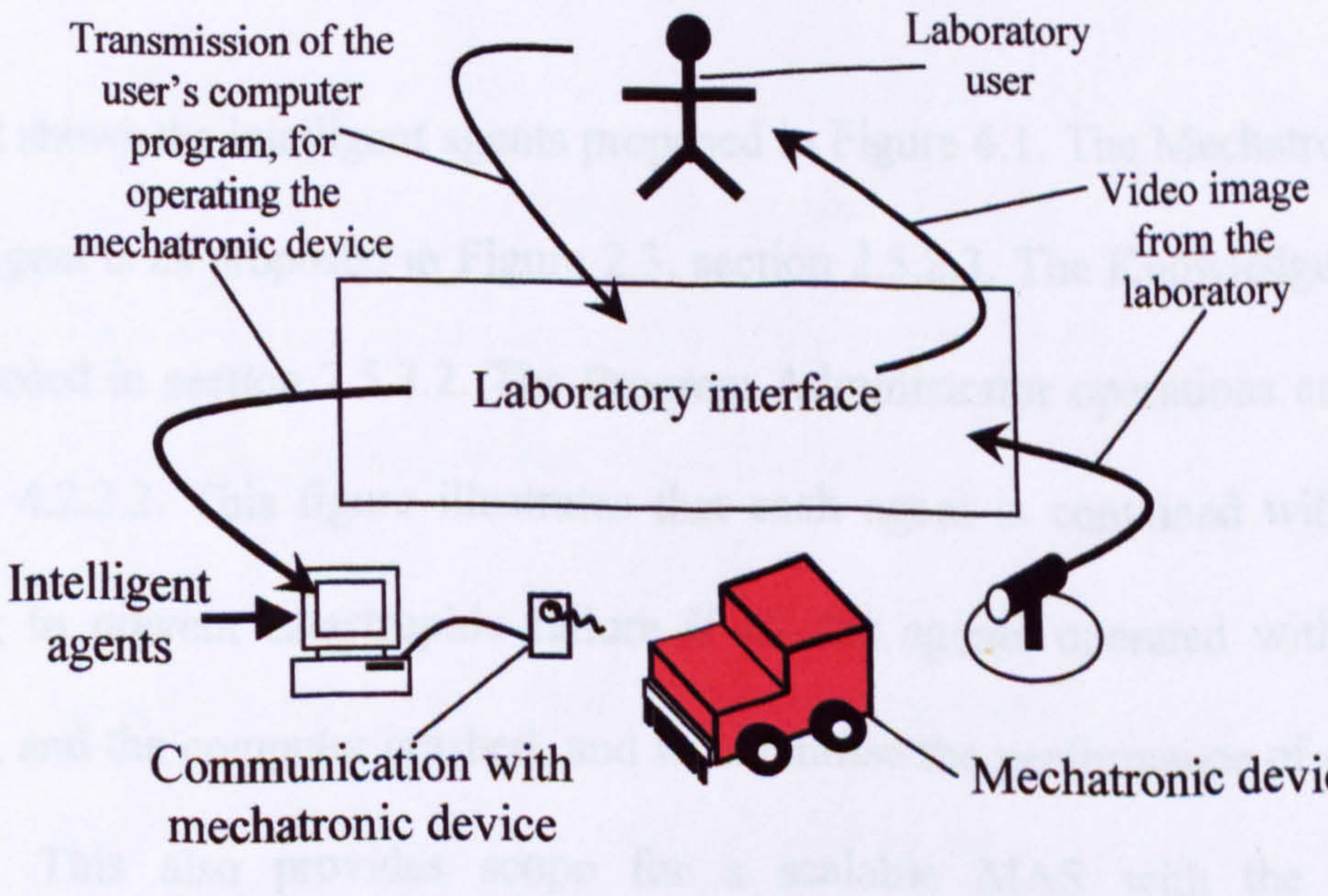



Figure 4.1 The user and laboratory interaction

Figure 4.1 illustrates the interaction between the laboratory and its user, enabled by the laboratory interface, which facilitates the transmission of the user's prototype Artificial Intelligence to the Laboratory, and a video signal from the laboratory to the user. Further, figure 4.1 illustrates the communication of the prototype Artificial Intelligence with the Intelligent Agents contained in the computer, and exemplifies a means of communication with the mechatronic device.

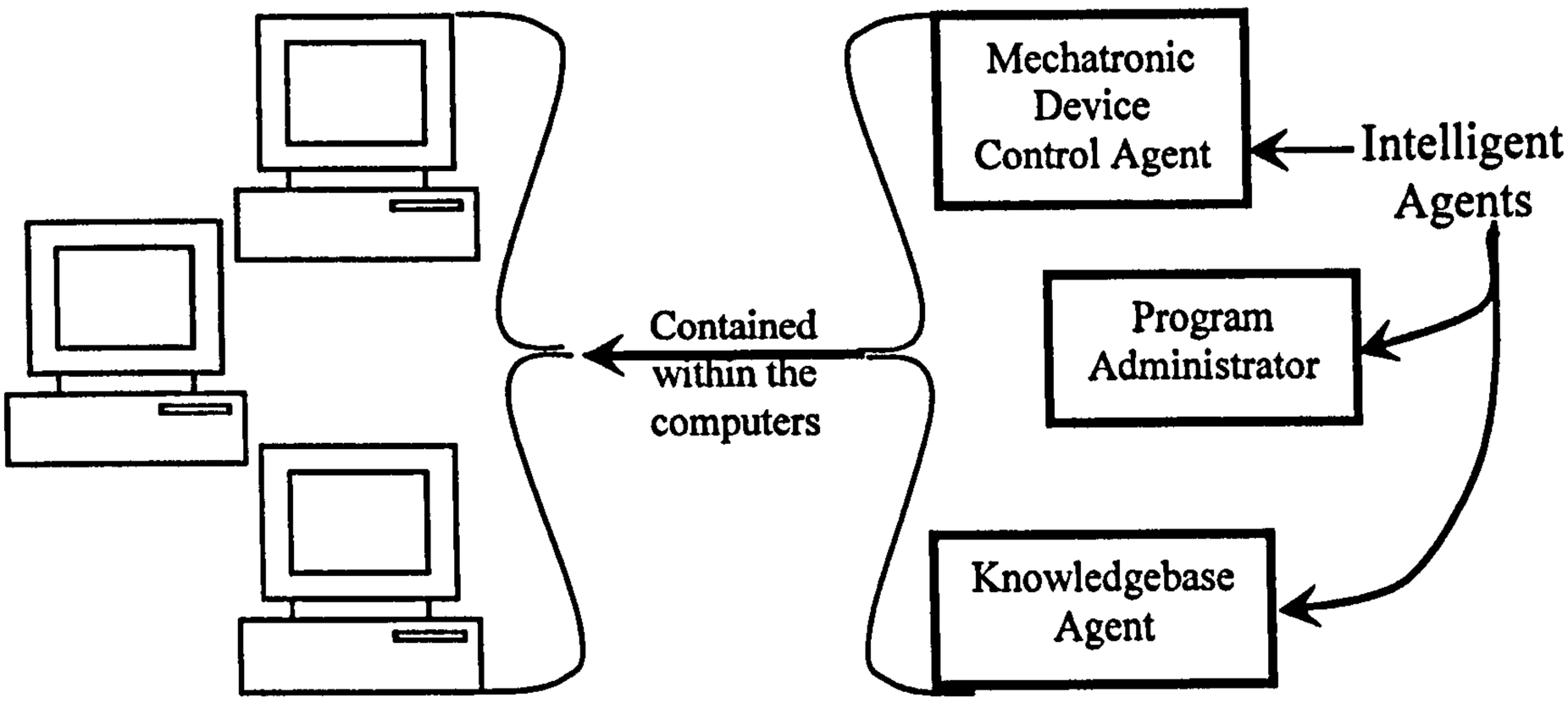



Figure 4.2 The intelligent agents within the laboratory

Figure 4.2 shows the intelligent agents proposed in Figure 4.1. The Mechatronic Device Control Agent is as proposed in Figure 2.3, section 2.5.2.3. The Knowledgebase Agent is as proposed in section 2.5.3.2. The Program Administrator operations are discussed further in 4.2.2.2. This figure illustrates that each agent is contained within its own computer; to prevent catastrophic failure if all the agents operated within a single computer, and the computer crashed, and to maximise the performance of each agent's operation. This also provides scope for a scalable MAS with the addition of mechatronic devices.

#### 4.2.1.2 Communicating Intentions

The prototype Artificial Intelligence is expected to react to all situations, so its intentions need to be easily understood and precise, as misunderstanding can lead to undesirable results. The only restrictions are the capabilities of the mechatronic device to perform an operation and any limits set for the operation of the mechatronic device. For communication of intentions there is a need to communicate with the Mechatronic Device Operating Agent, without which any intended work would not be achieved.

#### 4.2.1.3 Interpreting Intentions

Testing a prototype Artificial Intelligence comprises the use of rules: functions/procedures, and, if...then....else... computer language syntax, a complex issue.

The Mechatronic Vehicle Operations Agents subsequently need the following programming knowledge:

- the programming syntax rules to code the prototype Artificial Intelligence,
- the rules for function and scope of variables, and data creation,
- a programming command set of available functions,
- a means of defining the prototype Artificial Intelligence goals,
- a means of detecting invalid or unachievable goals.

To achieve this there is need for the following types of knowledge.

- *Syntax knowledge*: comprising a program command set, variable and function scope, data rules and the rules to create, store, access and amend program knowledge. Syntax knowledge is maintained for testing programs.

- *Decision knowledge*: comprising decision-making logic rules, to understand the prototype Artificial Intelligence program, requiring sensitivity to changes in conditions. Decision Knowledge determines how a program's knowledge is used to achieve goals.
- *Operation knowledge*: required for data interaction with the mechatronic device, comprising valid commands to the device and receiving sensor data. Without operation knowledge the decision knowledge and syntax knowledge are '*passive*' programs possessing no reference to the mechatronic device.

### 4.2.2 The Conceptual Design

There are three types of Intelligent Agents proposed for the laboratory design: a Mechatronic Device Operating agent to test the prototype Artificial Intelligence, a Book-keeping agent to store the laboratory knowledge, and an interface to communicate between the agents and the user(s). The assumptions are that each intelligent agent:

- operates autonomously within the context for which it was designed,
- can collaborate or negotiate with other intelligent agents to achieve its goals,
- can learn from past experience.

For the proposed Mechatronic Device Operating agents, the Blackboard Agent Architecture is advocated to operate user-supplied control '*program*' tasks. The agent would use the prototype behaviours to operate a mechatronic device within the laboratory, without prior knowledge of what the supplied goals will be. The Blackboard Agent Architecture (BB1) promoted by Hayes-Roth *et al.* [1995], Hayes-Roth [1995], is conceptually ideal for the laboratory, because it allows diagnosis of a user's software, and can support a multitude of separated goals. <split paragraph here>

The Blackboard agent can accept and operate a prototype Artificial Intelligence as part of the laboratory architecture, which would act as the agent's control plans. Hayes-Roth used an interpreter, as the control plans were a script of uncompiled program operations.

#### 4.2.2.1 The Mechatronic Device Control Agent

The user is expected to transmit a prototype Artificial Intelligence to the laboratory, for the laboratory's Mechatronic Device Control Agent to test.

The Mechatronic Device Control Agent subsequently contains programming rules for testing the prototype Artificial Intelligence relative to the mechatronic device's circumstances within the laboratory. This requires:

- selecting an appropriate mechatronic device if more than one are available,
- specifying the goals to be achieved,
- identifying the user's intentions, activating appropriate actuators relative to sensor input.

To achieve effective operation of the prototype Artificial Intelligence, the Mechatronic Device Agent has a Program Administrator.

#### 4.2.2.2 The Program Administrator

The Program Administrator comprises a program interpreter, a mechatronic device operator, and a command set operator.

- *Program Interpreter*: operating the prototype Artificial Intelligence akin to the BB1 meta-controller, allowing access to sensor data, and to operate the actuators.

- *Mechatronic Device Operator*: operating as an intermediary between the prototype Artificial Intelligence and the mechatronic device, necessitating:
  - Rules for transmitting data to the mechatronic device
  - Rules for receiving results from the mechatronic device
  - Knowledge expressing the mechatronic device's capabilities.
- *Command Set Operator*: the set of fixed commands that operate the mechatronic device. It is assumed that the prototype Artificial Intelligence will not be written in the '*base code*' of the mechatronic device, instead being translated into these commands.

#### 4.2.2.3 The Mechatronic Device

A mechatronic device is anticipated to generate information from its sensors, for which the prototype Artificial Intelligence is expected to transmit consequent actuator activity.

#### 4.2.2.4 The Knowledge-base Agent

This will store all the multi-agent systems knowledge, comprising:

- a means to store and access the system's knowledge,
- rules for negotiating with other agents,
- a knowledge store of previously successful programs for use by laboratory users,
- rules for learning-based communication with the laboratory user.

### 4.2.3 The Physical Design

With the functional design of the laboratory established, the physical design of the laboratory is now discussed.

### 4.2.3.1 The Mechatronic Device Operator Agent

The Mechatronic Device Operator Agent architecture will contain a Program Administrator, comprising a Program Interpreter, a Mechatronic Device Operator and a Command Set Operator.

- *The Program Interpreter*: the rules and data for testing the prototype Artificial Intelligence. The rules determine the prototype Artificial Intelligence operation, using sensor data for sensitivity to changing operating circumstances. The data obtained is stored as variable knowledge.
- *The Mechatronic Device Operator*: required for the prototype Artificial Intelligence to send and receive data to and from a mechatronic device.
- *The Command Set Operator*: comprising the functions to operate the mechatronic device.

This agent tests the prototype Artificial Intelligence, which will comprise:

- a Blackboard to receive the prototype Artificial Intelligence,
- rules to operate a mechatronic device's actuators,
- rules to receive a mechatronic device's sensors,
- a program administrator operating the prototype Artificial Intelligence with rules for:
    - communicating to and receiving sensor data from a mechatronic device,
    - the prototype Artificial Intelligence to operate the mechatronic device.

This is shown in figure 4.3, below.

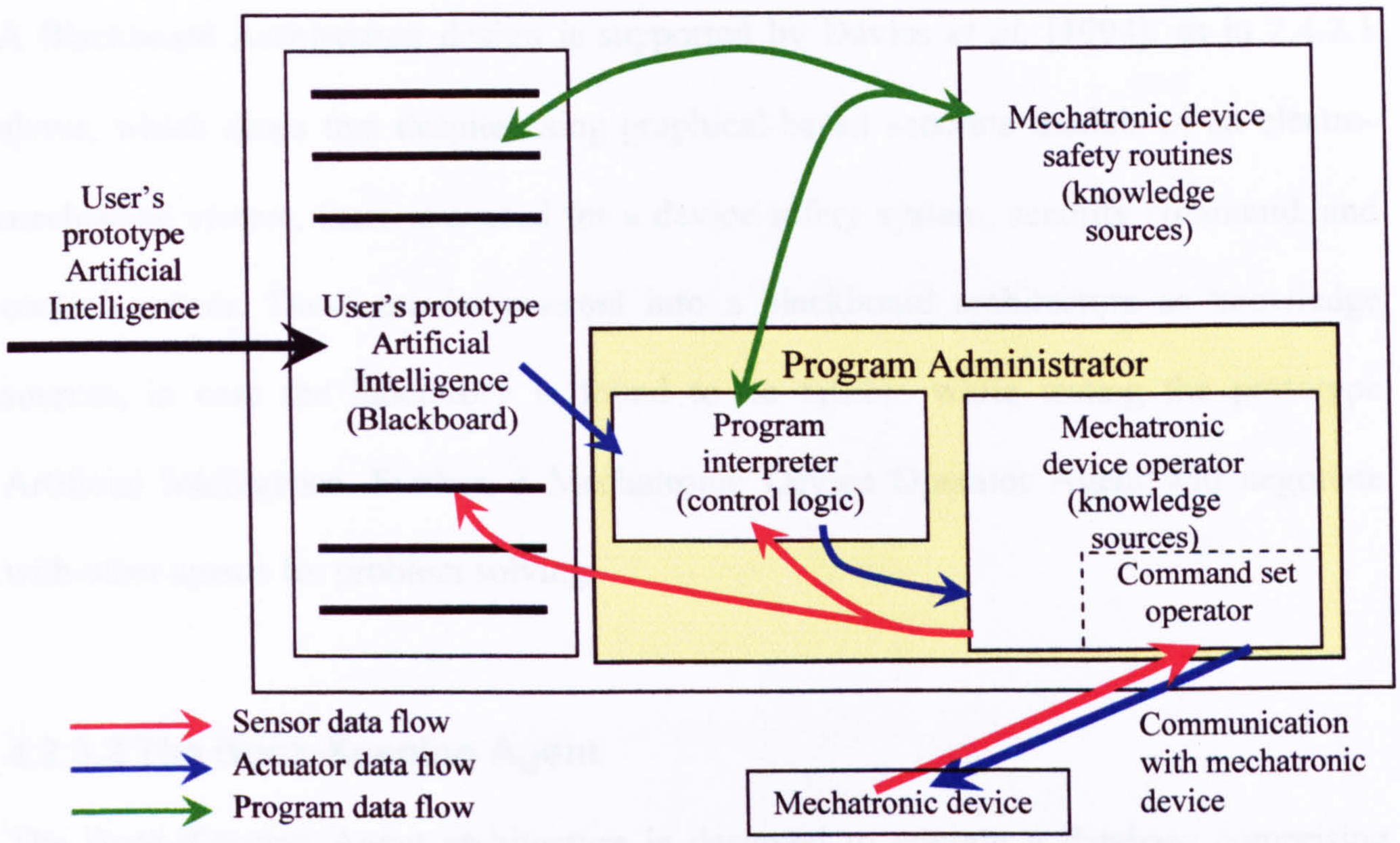



Figure 4.3 The Mechatronic Device Operator Agent based on the blackboard agent architecture

Figure 4.3 illustrates development of the Blackboard Agent architecture shown in figure 2.3, section 2.5.2.3. The data flow within a Mechatronic Device Operator Agent, together with its integration of the Program Administrator with the Mechatronic Device Control Agent, is shown in Figure 4.2 above as a separate agent. The data flow represents the use of the blackboard to store the prototype Artificial Intelligence which is then used to operate the Mechatronic Device while being moderated by Safety Routines.

When considering a suitable Intelligent Agent architecture design, the Blackboard Architecture was judged most appropriate. The architecture allows the insertion of prototype Artificial Intelligence into the agent, for testing by analysis a mechatronic device's operation. Blackboard architectures allow the insertion of the prototype Artificial Intelligence for use as knowledge sources. <split paragraph here>

A Blackboard Architecture design is supported by Davies *et al.* [1994], as in 2.4.2.1 above, which states that despite using graphical-based accurate models of an electro-mechanical system, there is a need for a device safety system, security command, and control system. These can be inserted into a blackboard architecture as knowledge sources, in case the laboratory is found to be fallible while testing the prototype Artificial Intelligence. Further, a Mechatronic Device Operator Agent will negotiate with other agents for problem solving.

#### 4.2.3.2 The Book-Keeping Agent

The Book-Keeping Agent architecture is designed to contain a database comprising system knowledge, negotiation rules, previously successful programs and rules for communicating with the user.

#### 4.2.3.3 The Interface between the Laboratory and its User

The laboratory requires a presence on the Internet, provided by designing an hypermedia communications platform to facilitate:

- a means to transmit the prototype Artificial Intelligence to the Mechatronic Device Control Agent,
- a live video transmission of the mechatronic device being operated,
- transmission of the mechatronic device's sensor data to the user,
- a stop signal for an emergency while testing the prototype Artificial Intelligence.

This is shown in Figure 4.4.

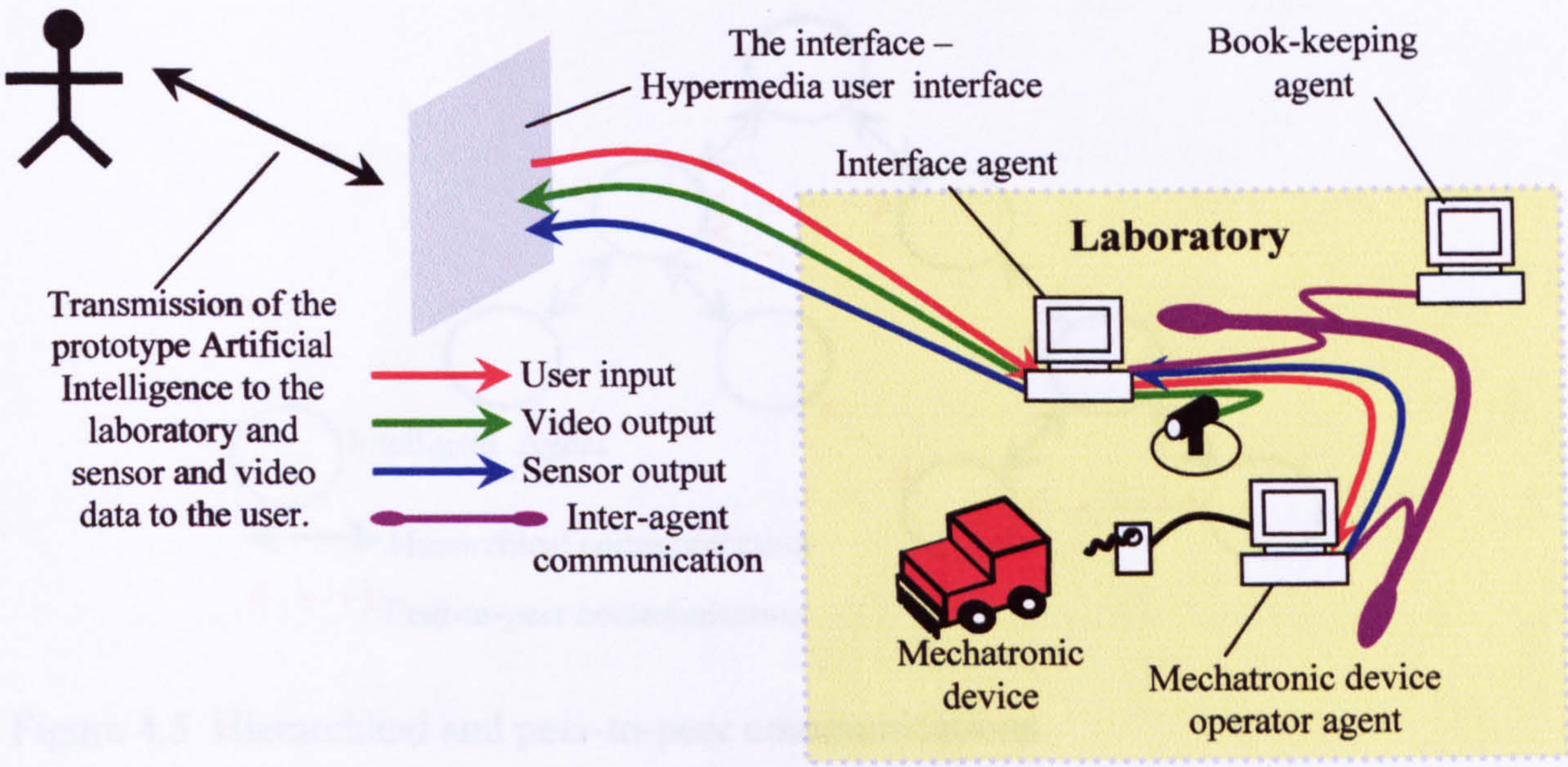



Figure 4.4 Communications between the interface and the laboratory

Figure 4.4 shows the communication flows between the laboratory, its user(s), and the intelligent agents within it. The laboratory interface is illustrated as an internet presence connecting the user and the laboratory. The data flow to the user comprises the received Mechatronic device sensor values and the video output. The data transmitted from the user is the prototype Artificial Intelligence. The data transmitted within the laboratory is between the agents, and actuator data from the Mechatronic Device Control Agent to the mechatronic device, and sensor data from the Mechatronic Device.

#### 4.2.3.4 Communication between the Agents

The assumption for the proposed multi-agent architecture is that the interface can communicate with all the Mechatronic Device Operator agents and the Book-Keeping agent, and the Book-Keeping Agent can communicate with all the other agents. A peer-to-peer and hierarchical multi-agent structuring is considered, shown in figure 4.5, below.

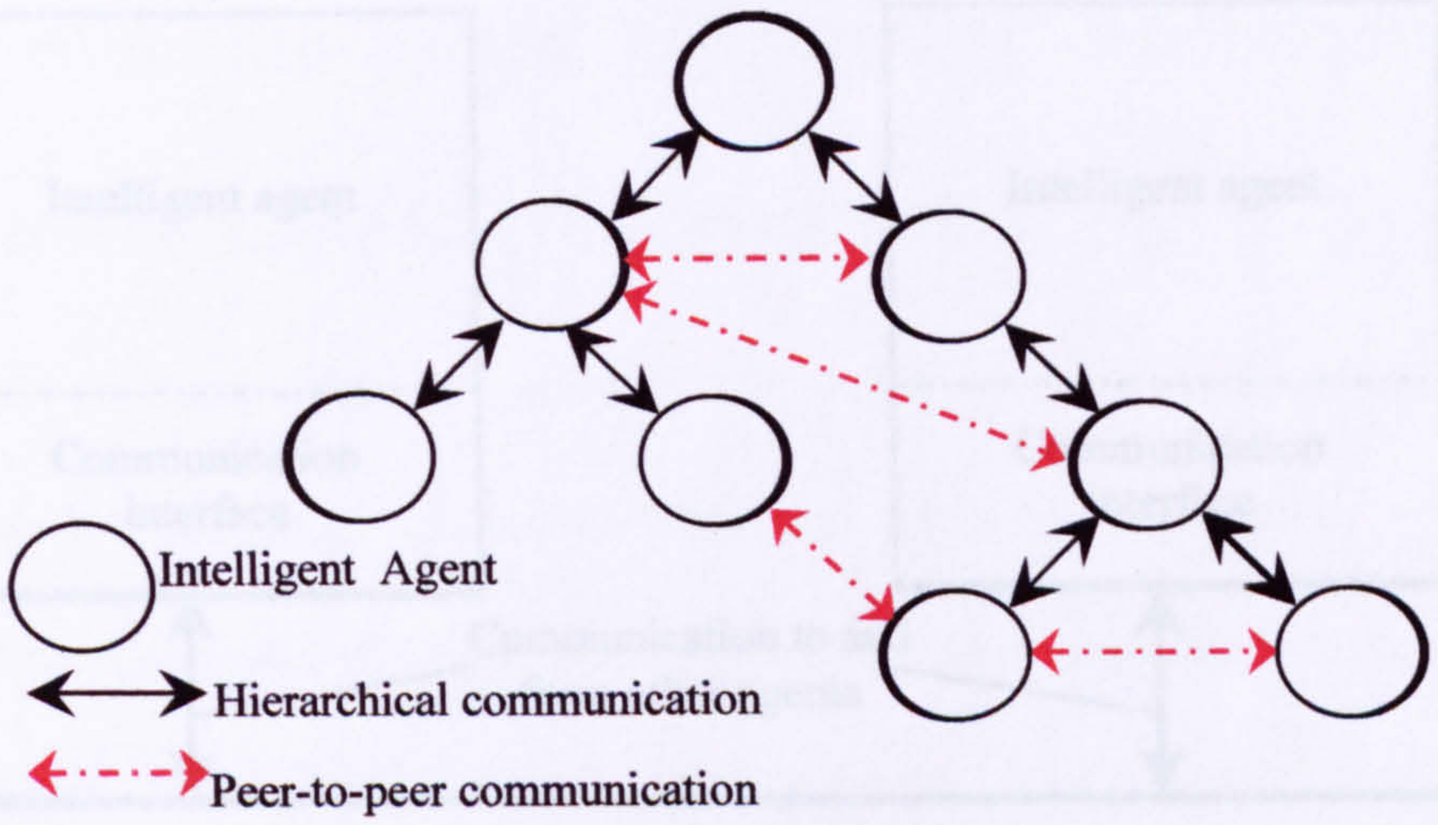



Figure 4.5 Hierarchical and peer-to-peer communications

The Mechatronic Device Operating agent needs to communicate with the interface to receive the prototype Artificial Intelligence for testing. Likewise the Book-keeping agent is expected to communicate with the other agents. A common communication interface for all the agents, similar to ARCHON, is considered appropriate, as this would allow the agents to communicate, as shown in figure 4.6, below.

The Challenger agents were conceptually different from the laboratory agents proposed. The proposed laboratory will have a '*Centralised Knowledge Agent*' with the Book-keeping agent, with knowledge fundamental to operations, unlike Challenger, with its agent operations paramount. Challenger's importance is that agents uniquely track each others' performance. The laboratory's agents can adopt this unique tracking method for obtaining knowledge.

The multi-agent systems papers, cited in 2.5.3.3 above, involve a communication package common to all the agents, to facilitate their inter-communication. Multi-agent interaction involves transmitting knowledge and intentions for negotiation, collaboration and co-operation, using a pre-determined format.

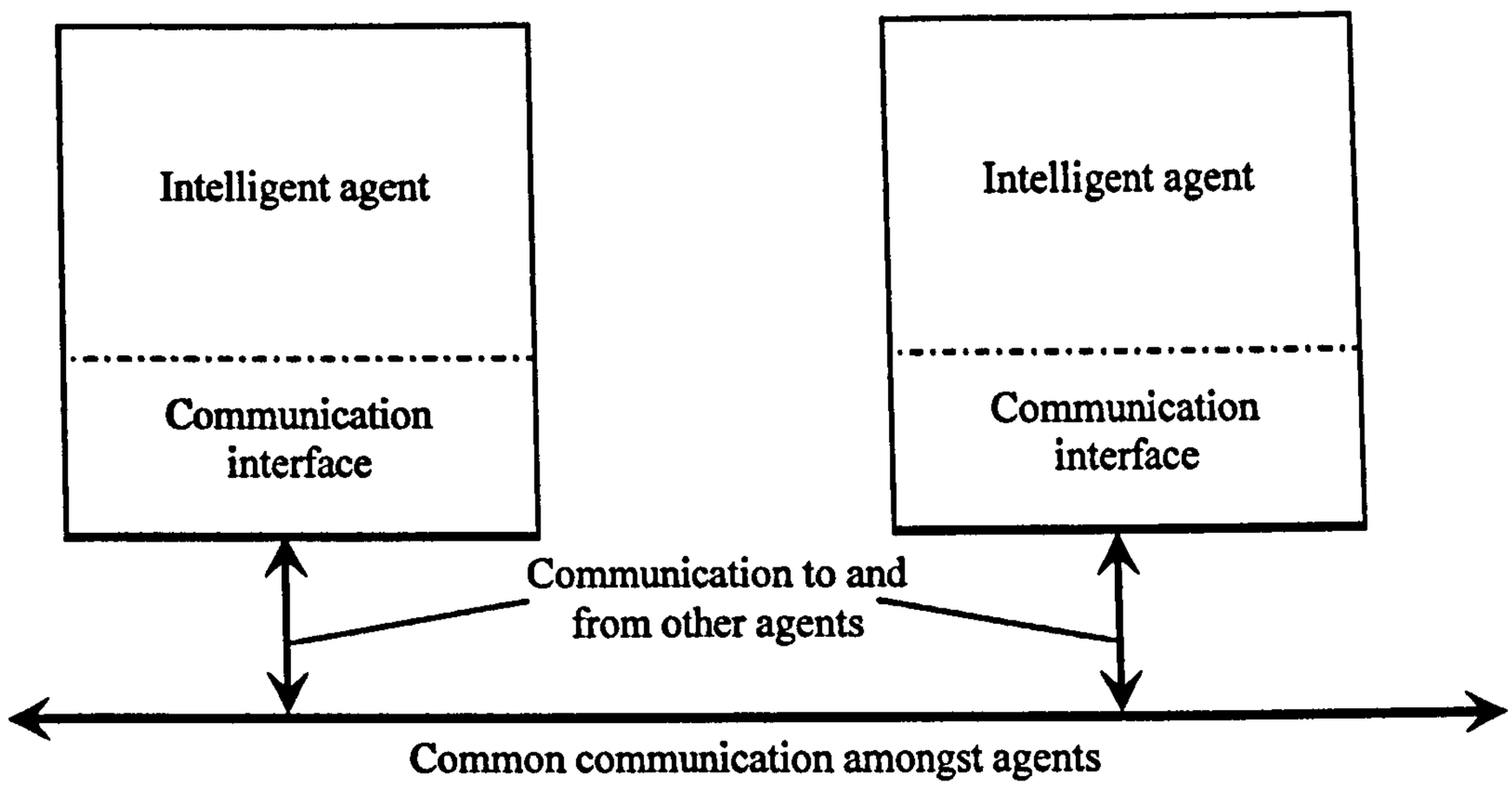



Figure 4.6 Multi-agent communications

If mechatronic devices can interact and obstruct each other's ability to attain goals, then there is a cause for conflict. When it arises, negotiations are needed, but peer-to-peer negotiation is increasingly difficult to achieve with increasing agent numbers. The use of a negotiation-based communication interface layer within the agent is a simpler solution. The principle for negotiation is that each agent operates within a strictly pre-defined function, with no restrictions on communications between the agents, and the agents are only accessible through the interface for the laboratory user. Any Intelligent Agent has the capacity to contact any other for requesting information. The outline structure of the intelligent agent architecture is shown in figure 4.5 above, with each agent having a communications engine and interface.

## 4.3 Conclusions about a Proposed Architecture for a Distance Learning Laboratory

Aim 1 of the research was to establish the viability of remote access facilities to augment distance learning. This has been the function of chapter 4. The internal laboratory functions are divided amongst specialised agents: Mechatronic Device Operating agents, and a Book-Keeping agent accessed by the user through an interface. Each agent contains a communication layer, to negotiate with other agents to resolve disputes and conflicts, as shown in figure 4.7

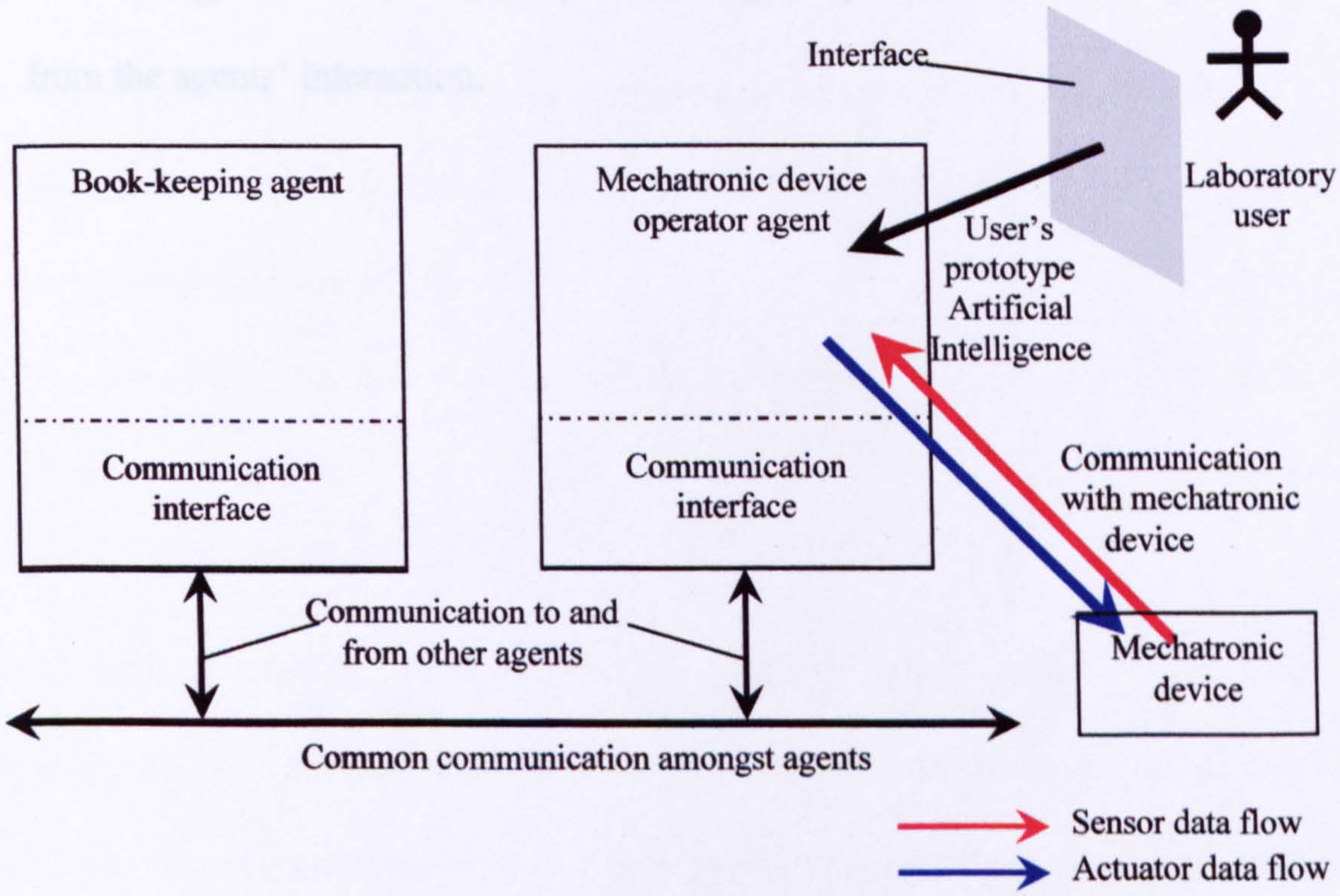



Figure 4.7 The complete laboratory architecture

A number of benefits are anticipated:

- Each agent is distinctly designed to promote the separation of operations, and reduce the possibility of conflicts, while making modification easier.
- The decentralised nature of a multi-agent system allows knowledge to be decentralised and relevant to the agent where it is stored.

- The distribution of tasks amongst separate agents allows rapid assimilation of new mechatronic devices into the laboratory.
- Modifications are limited to the modified agent.
- Adding agents is anticipated to not affect other agents' operability.
- A Book-Keeping agent potentially allows the agents to improve their operation, using others' experience.
- A decentralised multi-agent laboratory will be cheaper to design, program and maintain.
- The intelligence of the laboratory's multi-agent system will be emergent behaviour from the agents' interaction.

# Chapter 5

# Development of a Prototype Interface

## 5.1 Introduction

To evaluate and support research of a Remote Access Laboratory, there follows an examination of a prototype interface. The interface will substantiate the viability of a remote access laboratory to support a distance learning course, through the determination of its suitability to develop prototype Artificial Intelligence.

When designing the interface, the most important design considerations were:

1. The interface is to operate as a tool, allowing a user to design a robot's prototype Artificial Intelligence.
2. For education purposes, the interface has to be easy to understand, and invoke some form of enthusiasm.

The prototype interface as stated in 3.2 needs to prove a flexible tool by using Programming by Demonstration. The PbD system was considered suitable through its utilisation of programming language principles, and was subsequently used to design MOYRA, **M**echatronic **O**perations by **R**elated **A**ctions.

### 5.1.1 The Design Proposals

The design principles are primarily based on Activity Theory and Distributed Cognition, as discussed in 3.3 above.

#### 5.1.1.1 Distributed Cognition

Distributed Cognition is based on a user identifying the interface's available processes. The principles relating to HCI is that HCI should reduce the necessity for a user's recall about both its operations and the tasks available to the user.

To deter the user from actions which could lead to frustration about the interface, developing doubt and considering further activity as a waste of time, the design is segmented into associated features. Buttons with related functions are placed in proximity to their target activity.

#### 5.1.1.2 Activity Theory

Activity Theory analyses a tool's mediation between a subject and an object. As such, the interface is the tool between the mechatronic device and the mechatronic designer. The argument followed is that the object motivates activity, and specifies its direction. To this end, the tool has to efficiently mediate between the subject and the object.

The problems of implementing Activity Theory are: while the Activity Triangle Model provides a reference, its use requires imitation of the circumstances being examined, creating an Activity System Model. The modelled Activity System is split into the constituent interactions, before research questions are generated and a detailed investigation carried out, to obtain findings for interpretation. The final problem is how to interpret the findings.

### 5.1.2 Anticipated Problems

The interface's Programming by Demonstration function has to circumvent the related problems of the Gulfs of Execution and Evaluation, and the problems of sub-optimal programming and noise. The cause of the problems is the expectation for a mechatronic device's behaviour designer to conceptualise and model a behaviour in his/her head, before duplicating the modelled behaviour and writing it in computer textual language, while contending with the computer language's structure. The PbD system allows a purity of design for a robot's expected behaviours, with the activating conditions for each behaviour.

As stated in 3.3.1.1 above, when a computer is used for design, precision is required, at a cost of inflexibility, which can be incompatible with a designer's potential need for ambiguity to change the design during development. The PbD system requires a designer to conceptualise a mechatronic device's required behaviour. The advantage of PbD is the elimination of the textual programming limitations, allowing a designer to replicate conceptualised behaviour.

### 5.1.3 Principles of Prototyping

A prototype is a generally accepted experimental tool, but is not a full product implementation. It enables future modification, with the intended user having an opportunity to comment on a system's current functionality.

The laboratory was prototyped, selecting from a range of methods, [Gordon and Bieman 1995]:

- **Throw away prototype**: This is a working model with all the necessary features, but patched together. Users can interact with the system, getting accustomed to: the interface, the available features and output types. The prototype is intended for designers to realise the critical design considerations, which is important when the user is uncertain about the system's functionality.
- **A first implementation**: The prototype is completely operational, anticipating a final product with identical features. This type is useful when planning multiple installations of the same system. A full-scale working model allows realistic interaction with the system, while minimizing development costs.
- **Evolutionary prototype**: This is a model which includes some final system features, but not all. It develops a system incrementally in modules, so that features can be evaluated and incorporated into a final system without significant work to assemble the components. Prototypes of this type are generally part of the finished system.

A prototype allows intended users of the finished system to interact, experimentally showing any unexpected interaction, both providing reactions to the prototype, and suggesting additions or deletions for the available features. The advantages of prototyping are that it provides:

- the potential to change a system early in its development,
- an opportunity to stop developing a system that is not working,
- the ability to develop a system which works closely to the users' needs.

The reason for developing the prototype was to allow comprehensive examination of user expectations of the remote access laboratory. The most important part of the prototype was the user interface, since the principle activity was to elicit users' feedback.

## 5.2 Programming Language Development

Wirth [1974] argues that a programming language should be both easy to learn and use, safe from misinterpretation and misuse, while extensible without changing existing features, and capable of withstanding logical scrutiny. The language should have a machine-independent definition, efficiently using computer resources with a fast and compact compiler efficiently coding and economising storage, without complex and rarely-used optimisation routines. The language definition should be self-contained and complete, while implementation provides ready access to other facilities such as program libraries and subprograms written in different languages. The language should be hardware-independent, with compilers adapted for various processors and chipsets, while minimising compiler development time and cost.

The most important property of a program is whether it accomplishes the intentions of its user. Hoare [1973] states that a problem of program design is: *'deciding what a program is intended to do, and formulating this as a clear, precise, and acceptable specification'*. Further, Hoare considers implementation is fraught with difficulties: the division of complex tasks into simpler subtasks, defining both the subtask rationale, with comprehensible, well-organized methods for subtasks to interact. The argument is that a well-designed programming language conveys both how a program operates and what the program is expected to achieve.

Hoare [1973] argues further, for a new language to supersede existing languages, its design has to be extremely simplistic, for programmers to both rapidly learn its features, and identify which elements will solve a problem, allowing a programmer to concentrate on solving problems.

### 5.2.1 Programming Language Definition

The problem with a programming language definition is: which definition?

Programming language definitions are intended to serve as a specification of correctness for:

1. the language implementation,
2. the user to validate if a program performs its intended task.

Wegner [1976] states that future programming languages should each have a tractable formal language definition to determine program correctness. In the 1970s one programming language design objective was '*simplicity*', increasingly defined as the ease of developing a formal definition. Now a program is considered a '*what*' specification, for designated tasks or functions, plus the associated '*how*' specification. A '*correctness demonstration*' demonstrates that the program '*how*' specification has implemented the independent '*what*' specification.

Currently the method which a programmer uses to establish a program correctness, is to test particular cases and modify the program if the results are unexpected, [Hoare, 1969].

To establish correctness of the prototype Artificial Intelligence used for experimentation, it was tested continuously throughout an evolutionary prototyping development of the PbD system. The use of pure first and second level logic for the PbD system test was beyond this research's remit. The computer language developed is still a prototype; experimentation was to determine usability and continue improvement.

### 5.2.2 Programming Language: 'Vocabulary'

The study of computer languages is concerned with defining the finite number of structures which allows languages an infinite number of sentences, [Wegner, 1976].

Wirth [1974] cited Van Wijngaarden [1963] with the principle of '*Simplicity*' stating that

> '*... his point was languages are not only too complex, but due to this very complexity also too restrictive. 'In order that a language be powerful and elegant it should not contain many concepts and it should not be defined with many words*''.

This is interpreted as advocating analysing of a language and providing its fundamental principles unobstructed by the boundaries of applicability. However the counter-argument is that using languages without defined syntax rules, it can be difficult or impossible to identify programmed logic flaws. A solution is to design a programming language features in a form intuitive and memorable for both use and identifying logical consequences, to prevent ambiguous programming.

A language's crux is its variables, the named location in memory used by the program to store a value that may be modified. Hoare [1973] considers variables valuable, but potentially problematic as they can change register contents; store locations; contain peripheral conditions; or either change its own values or other programmed instructions. For high level languages, Hoare considers that this problem could have been avoided, but instead was worsened by pointers indirectly assigning variable values and, if accidentally misused, can cause disastrous data damage. A variable is normally declared before use with a declaration form of:

type *variable_list;*

The PbD system uses sets of distinct variables to create a prototype Artificial Intelligence, associated with the goals to be achieved, and both the sensor inputs and outputs. These variables are not specifically declared, thereby preventing the designer from misusing them.

### 5.2.3 Program Control Structures

These apply to when a program includes conditional logic for program flow, and are achieved by modular decomposition of programs, suitable both for bottom-up and top-down development. Structured programming has placed greater emphasis on the *if-then-else* and *while-do* constructs.

The underlying design principles are:

- *while <sensors not activated> do <continue current active behaviour>*
- *if <sensors activated> then <Sensor-Activated Behaviour> else <stop>*
- *if <goal> then <Goal-Activated Behaviour> else <next goal>*

### 5.2.4 Programming Language: 'Functions'

The building blocks of modern High Level Programming Language are functions. If a function uses variables, they are either declared as formal parameters or as local variables, with the subsequent block operating on the variables. A high order programming language function general format is:

*<return variable type> Function-Name(formal parameter(s));*

*local variable declaration*

*operations on variables*

The prototype Artificial Intelligence comprises behaviours which are specific to circumstances, the design of which were:

*< mechatronic device return state> Behaviour-Name<mechatronic device begin state>*

*while <sensors not activated>*

*if <sensors activated>*

*then <Sensor-Activated Behaviour>*

*else <stop>*

*do <behaviour>*

*return <mechatronic device state>*

A function allows code compaction, with the function performing itself, known as recursion. The disadvantage of recursion is the memory overhead of stored variables. Hoare [1973] considers that functions should make variables operations '*clearly manifest from its syntactic form*' and '*simple to understand and resistant to error*'. Hoare states that the function interface is the boundary between programs parts, and argues that the suitability of a function's use should be subject to the most rigorous compile-time check.

The use of recursion was considered important for specifying how to design prototype Artificial Intelligence. This allowed the prototype Artificial Intelligence to be constructed from a set of generic behaviours, each equivalent to a function.

### 5.2.5 Principles of Object Orientated Programming

Object Orientated Programming involves fragmenting a program into subdivisions of self-contained units called objects, comprising variables and data with local reasoning about the variables and data behaviour. The objects are organised into a hierarchical structure within a container called a class. Characteristics of Object Orientated Programming are: encapsulation, polymorphism and inheritance. These concepts were used to organise a designed prototype Artificial Intelligence using the PbD system.

**Encapsulation** This allows the private containment of behaviours.

**Polymorphism** Polymorphism allows the same behaviour activation state to be declared for different circumstances.

**Inheritance** This enables a behaviour to encapsulate another hierarchy of behaviours it depends on.

This is discussed and illustrated further in section 5.3.

## 5.3 The Prototype Interface Design

### 5.3.1 The Human Computer Interface Methodologies

The interface design was deliberately divided into three basic associated regions: user input, program output, and program manipulation, as shown below in Figure 5.1.

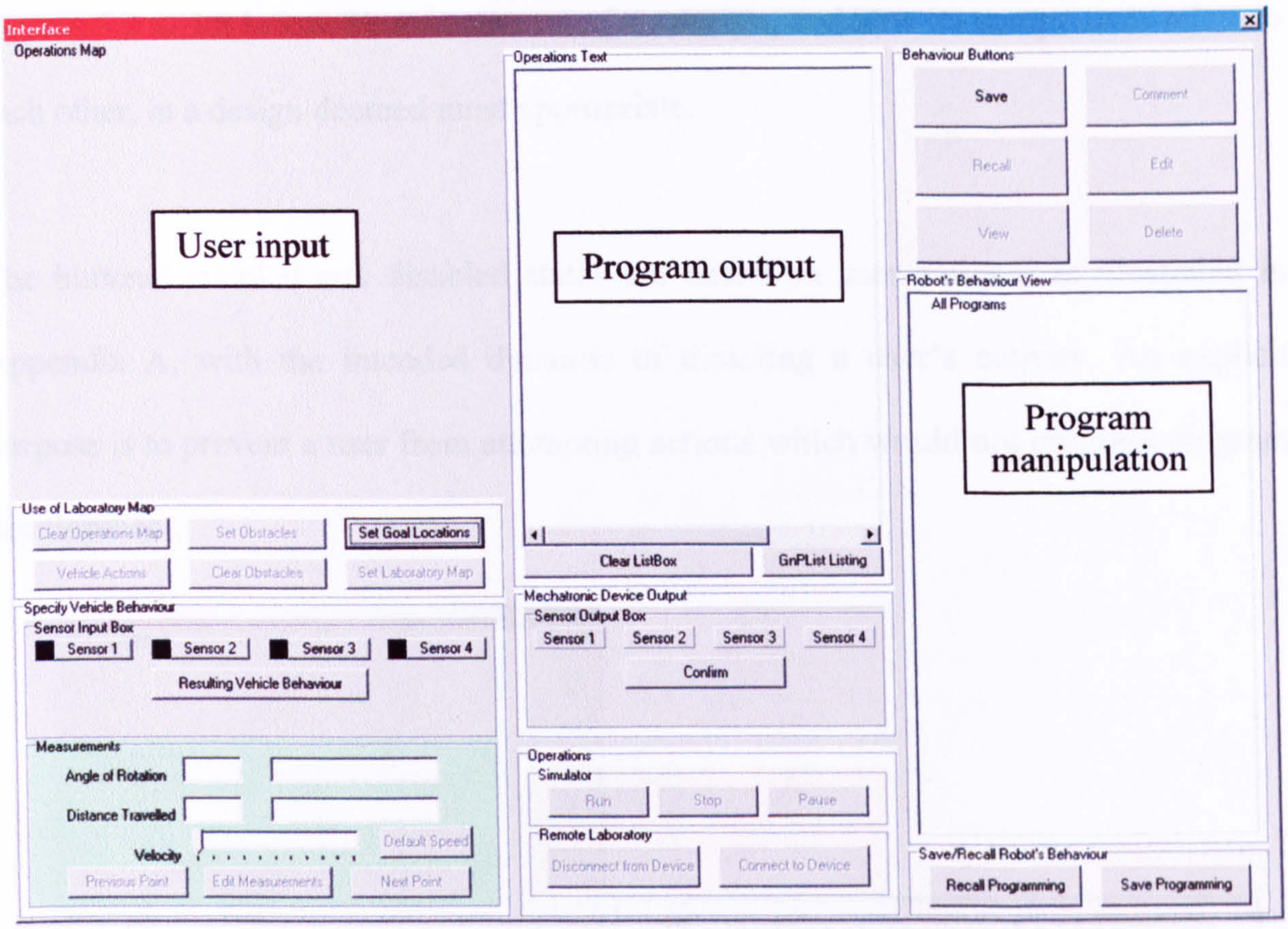



Figure 5.1 The outline of the interface in 3 segments: user input, program output and program manipulation

The problem with applying Distributed Cognition was identified by Activity Theorists as the need to experiment to identify correct and adverse design decisions affecting the interface's utility. The problem is compounded by the lack of technical standards; there are no analytical tools and no data characteristics to identify. However, Hollan *et al.* [2000] describes two principles:

1. A system can actively orchestrate subsystems to achieve different tasks.
2. Cognition is distributed across both the HCI and user.

Based on the Towers of Hanoi [Zhang and Norman, 1994] example in section 3.3.1.1, the HCI design is expected to capture the interface's operation knowledge, in responding to users' choices. The result is a dynamic interface responding to the users' actions when presenting available options. This methodology further constrains the user to action plans within the limitations of the interface.

Figures 5.2 to 5.8 below illustrate the interface design, and how its components relate to each other, in a design deemed most appropriate.

The buttons' enabled and disabled states are based on user actions, as illustrated in Appendix A, with the intended dynamic of directing a user's activity. An explicit purpose is to prevent a user from attempting actions which would not progress program development.

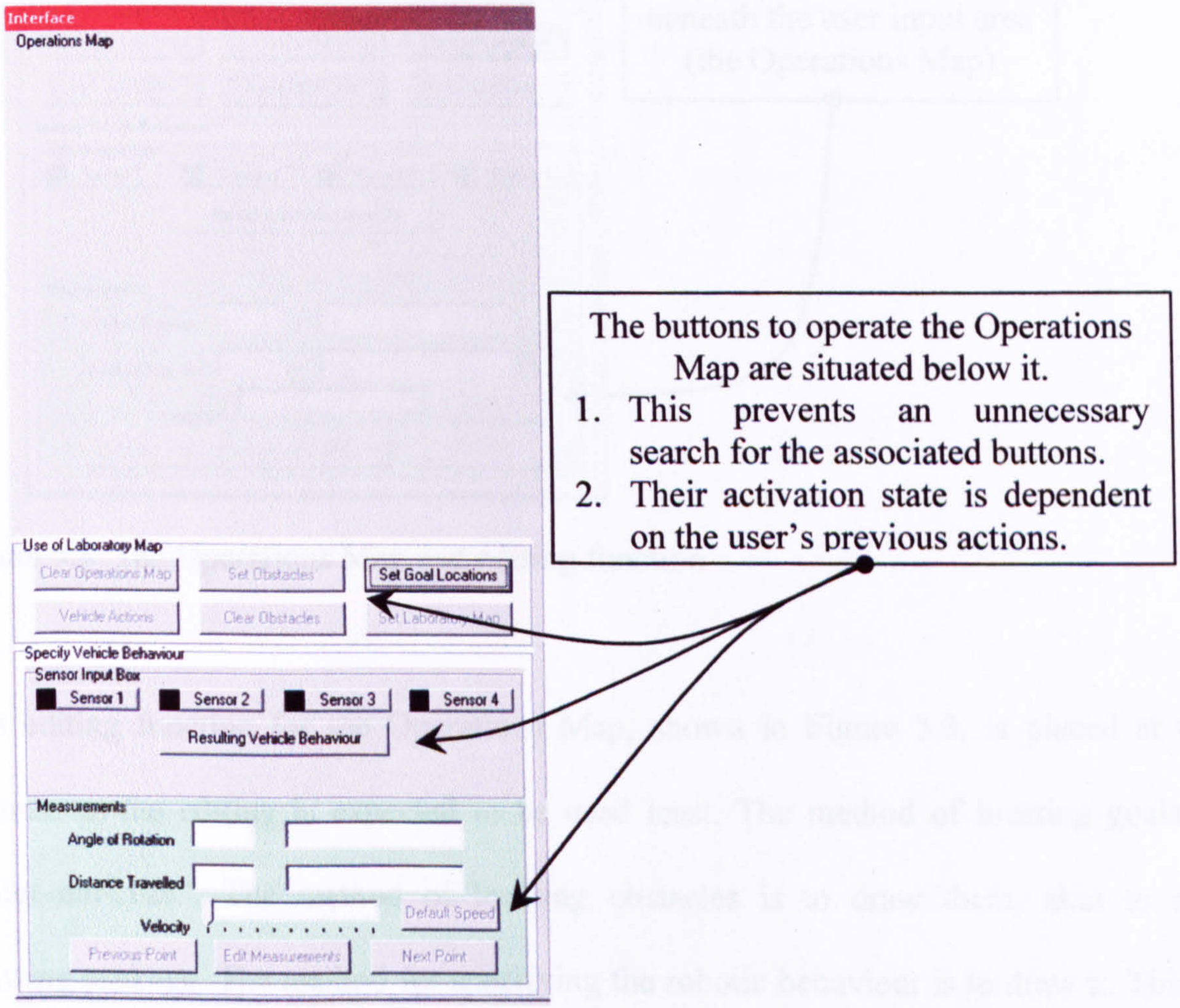



Figure 5.2 The Operations Map and associated buttons

The Operations Map performs several functions, which are dependent on the user activity. For setting goal locations, locating obstacles and running the simulator, the Operations Map is a direct mapping onto the laboratory workspace. For the mechatronic device behaviours, in drawing a path to be navigated, it is orientated to the mechatronic device. When linked to the laboratory, a video image is presented.

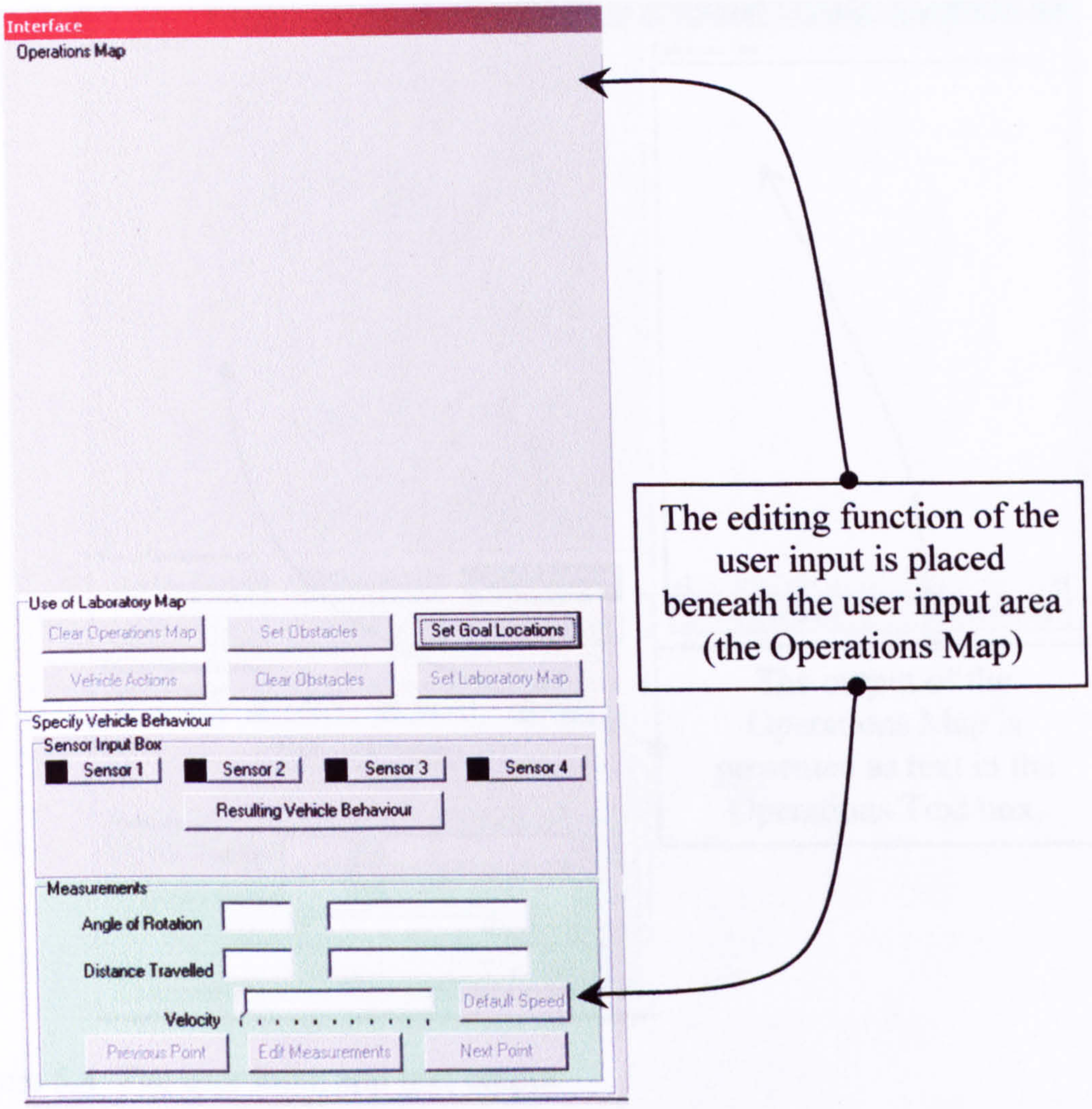



Figure 5.3 The Operations Map and editing function

The editing function for the Operations Map, shown in Figure 5.3, is placed at the bottom, as the editing is expected to be used least. The method of locating goals is 'point-and-click'. The method of locating obstacles is to draw them, akin to any drawing package. The method for specifying the robotic behaviour is to draw it. This is intended to be easy for rapid development of prototype Artificial Intelligence. The Measurements panel is used to specify the angle of rotation and any distances a robot is being programmed (and expected) to travel.

For editing the angles and distances, they are displayed in the Measurements Panel and can be textually edited.

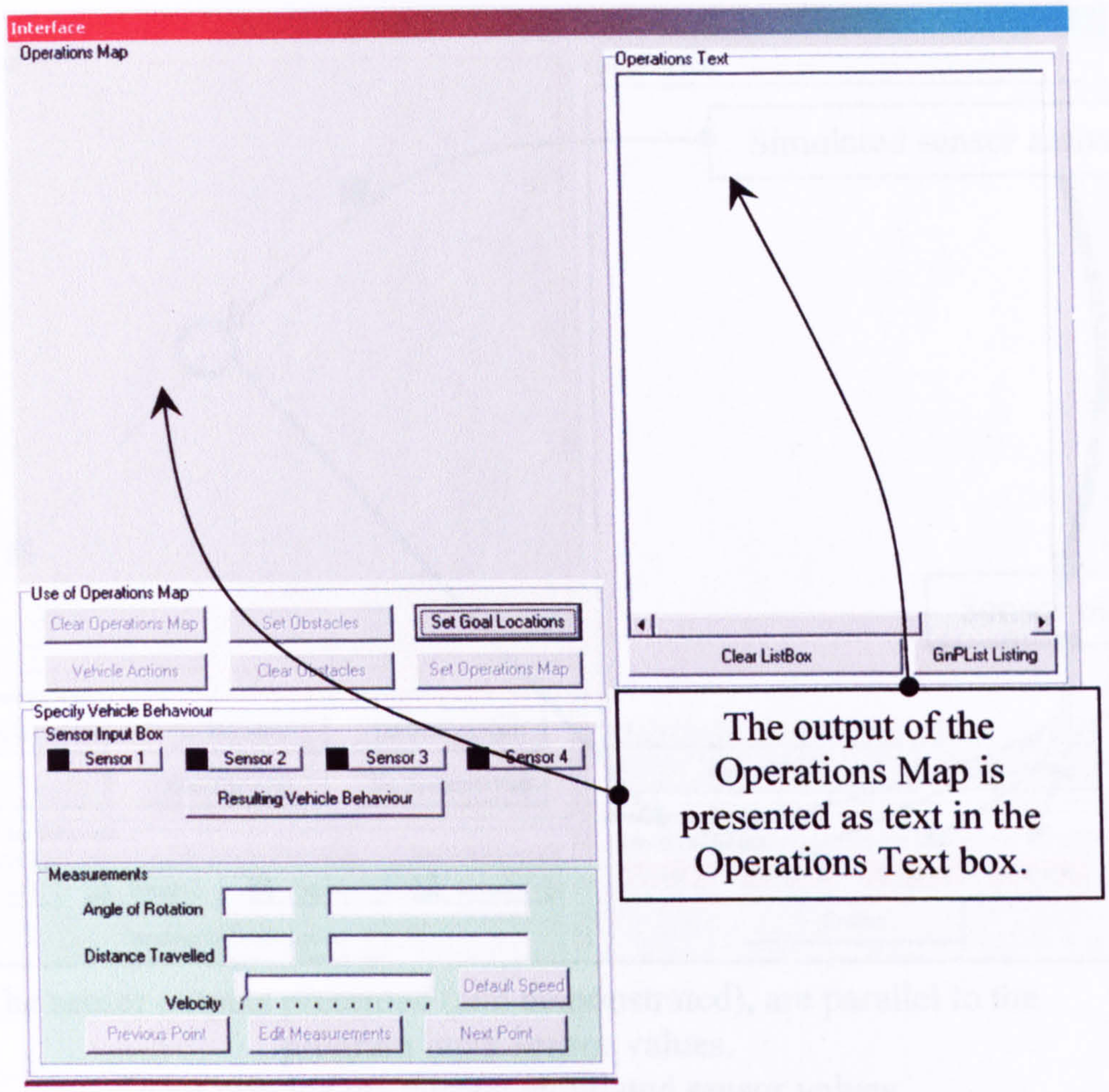



Figure 5.4 The user input and text output

The text box to the left of the Operations Map shown in Figure 5.4, can be adopted to facilitate a help function. The text box provides a duplicate description of the robotic operation being designed, with a function to allow the PbD user to view the underlying data describing the prototype Artificial Intelligence. The raw data is used to create the prototype Artificial Intelligence. The text box was considered important for future development. It was not developed further for the prototype, as it was considered an implementation issue.

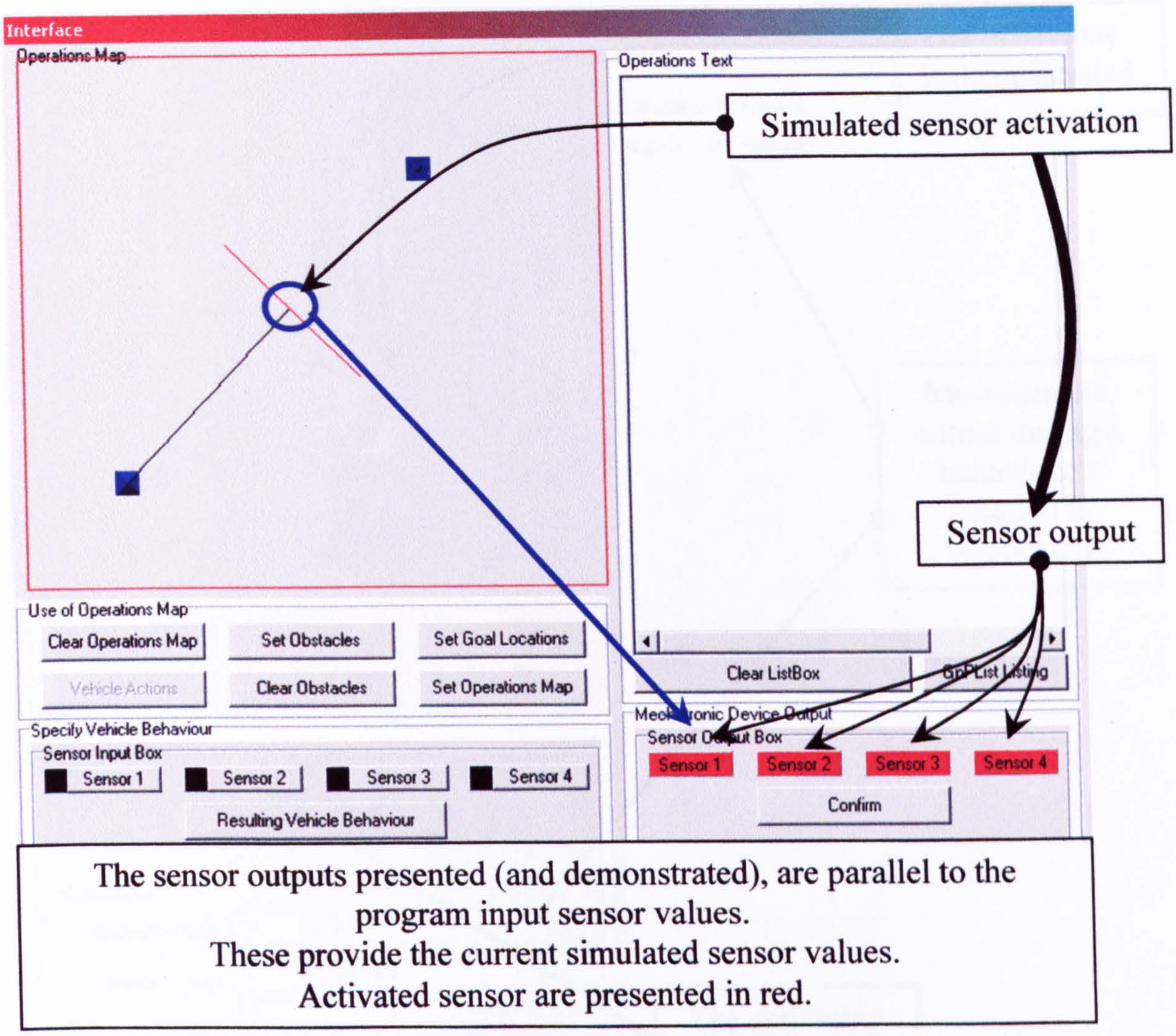



Figure 5.5 Relation of sensor output to user input

Figure 5.5 illustrates the relationship between the PbD interface's Sensor Output Box and the Operations Map during the design of robotic behaviour. The principle of using sensor activation for robotic behaviour allows the prototype Artificial Intelligence designer to explicitly translate sensor activations to actuator actions. The PbD interface calculates the sensor outputs by determining the angle of contact between the robot and the obstacle, as shown in figure 5.6 below.

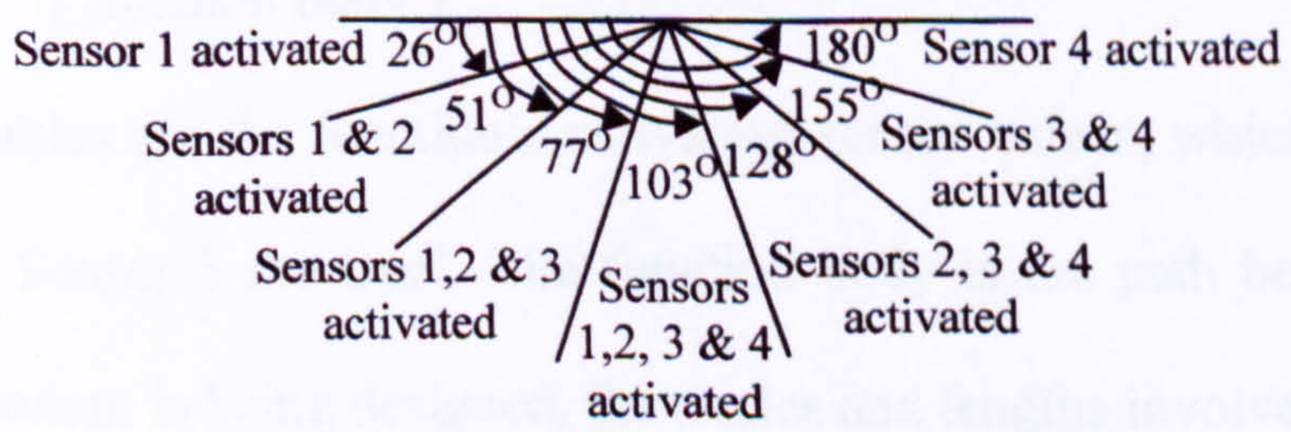



Figure 5.6 The angle of incidence between the robot and obstacle, and activated sensors

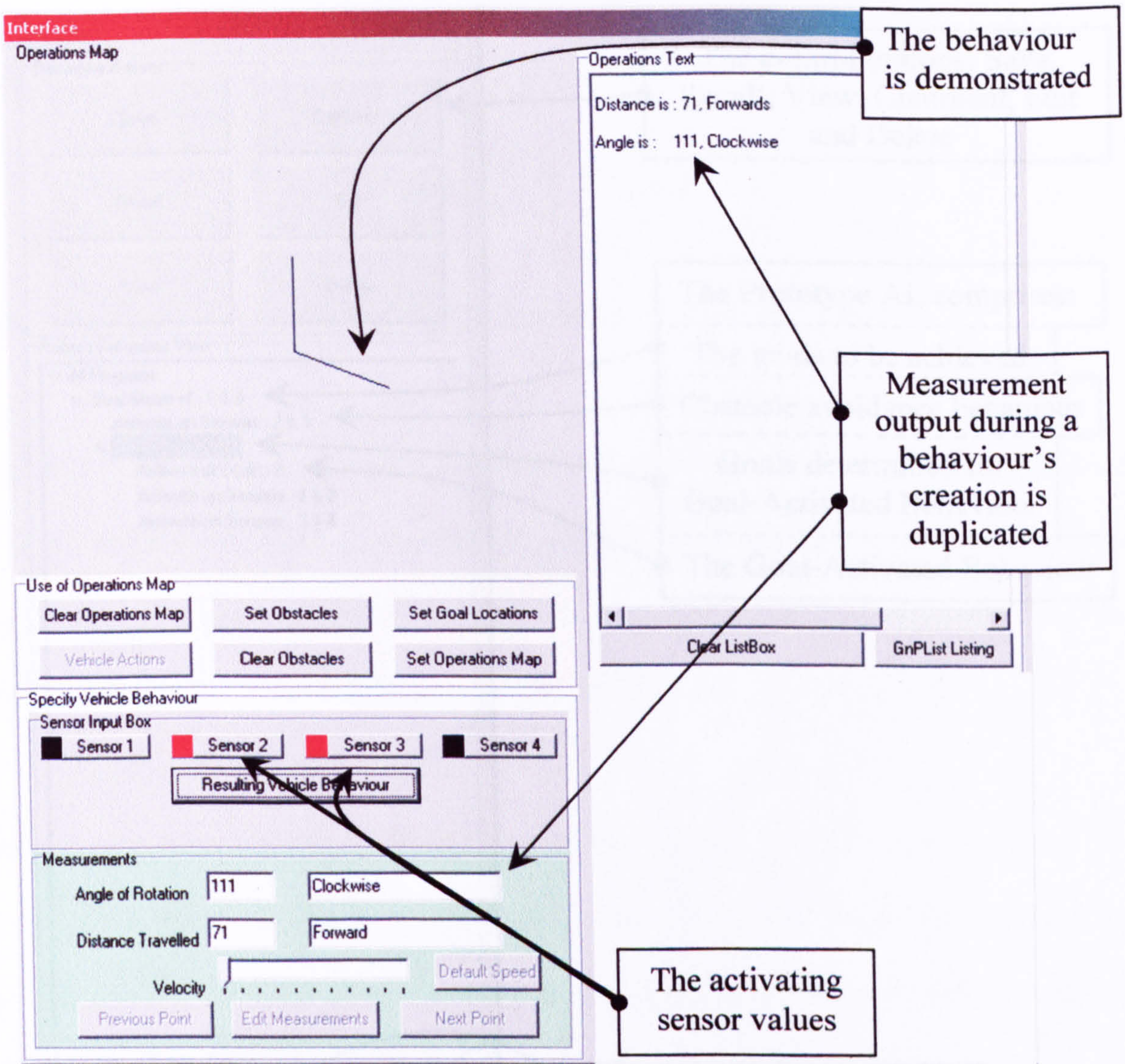



Figure 5.7 Creating a mechatronic behaviour function

Figure 5.7 above illustrates the method for creating mechatronic behaviour. This allows the designer to explicitly plot the expected path for the device to follow. The function follows a standard for programming design of:

Function (input variables)

{ function body }

The input variables are the function's activating sensor values, which in figure 5.7 are 'Sensor 2 and Sensor 3 are true'. The function body is the path being demonstrated. While the behaviour is being designed, the angles and lengths involved in the behaviour are displayed.

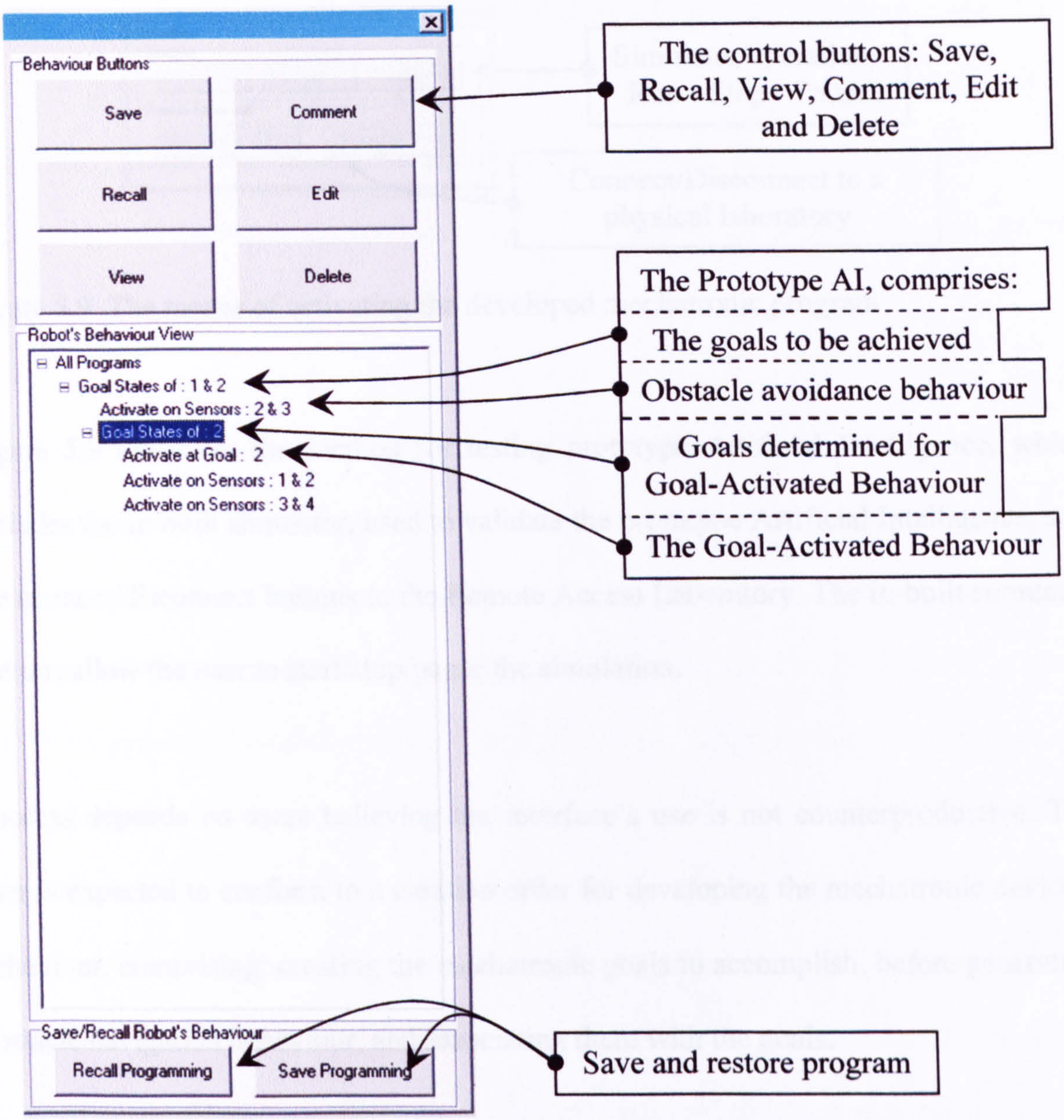



Figure 5.8 Manipulating a developed program

Figure 5.8 shows the operations in detail of the Program Manipulation part of the PbD interface. At the top are the prototype Artificial Intelligence construction control buttons, used to manipulate the design of the prototype Artificial Intelligence as a hierarchical set of behaviours, which can be observed in 'Robot's Behaviour View' above. At the bottom are the Save/Recall Programming buttons, used to save the hierarchy of robotic behaviours designed, or recall previously designed behaviours.

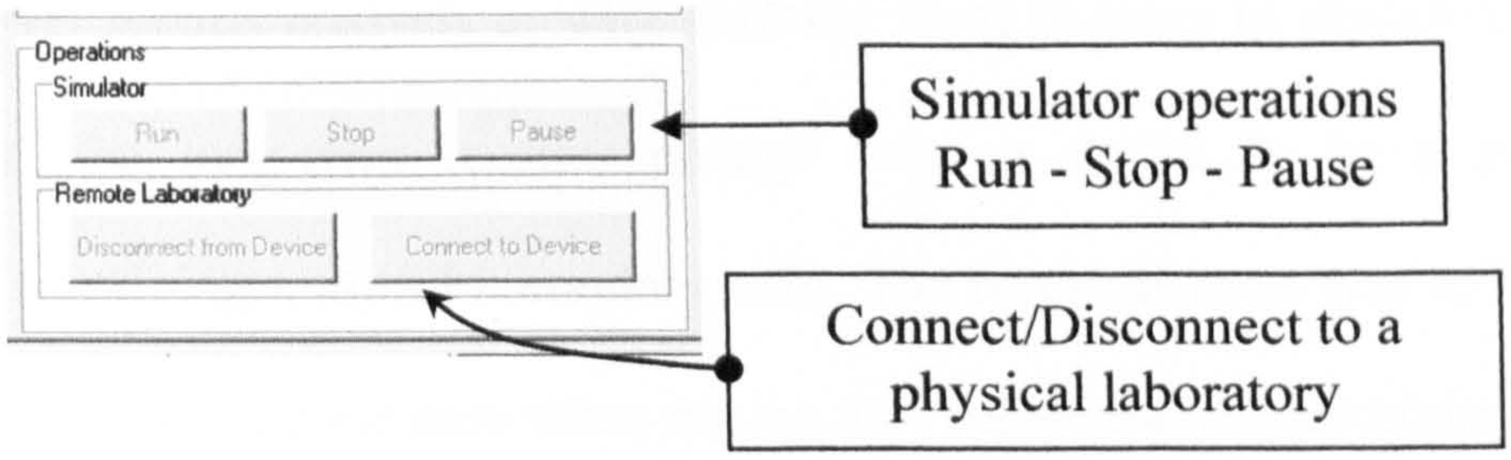



Figure 5.9 The means of activating the developed mechatronic program

Figure 5.9 illustrates the method for testing prototype Artificial Intelligence, which includes the in-built simulator, used to validate the prototype Artificial Intelligence, and the connect/disconnect buttons to the Remote Access Laboratory. The in-built simulator buttons allow the user to start/stop/pause the simulation.

Success depends on users believing the interface's use is not counterproductive. The user is expected to conform to a creation order for developing the mechatronic device's behaviour, comprising: creating the mechatronic goals to accomplish, before generating obstacle navigation behaviour, and associating them with the goals.

### 5.3.2 Programming by Demonstration (PbD)

The purpose of the interface design is to facilitate the development of a prototype Artificial Intelligence to operate as a mechatronic device. The development of a Programming by Demonstration based interface with the underlying data structures was intended to conform to programming language principles. PbD was proposed for programming a prototype Artificial Intelligence, as current research provides evidence that students have problems with learning programming languages, causing high failure and drop-out rates, [Bergin and Reilly, 2005].

Mamone [1992] reports analysis of why students want to learn to program, providing evidence that the long-term reasons change from a desire to be a professional programmer to gaining a useful secondary skill. The research issue was to identify the students' comfort-level; their ease when asking and answering programming questions, and their perception of programming achievement.

Nevalainen and Sajaniemi [2005] argues that programming is difficult for many students to learn, as many programming skills require understanding abstract concepts loops, pointers, and array-based methods. A proposed solution is to use methods and techniques that assist and enhance learning about abstract entities. Petre and Blackwell [1999] proposes visualisations for learning expert programming reasoning, with Hundhausen *et al.* [2002] and Mulholland [1998] advancing visualisation to learn both program concepts and the program language. Nevalainen and Sajaniemi [2005] argues for a solution of needing a programming language which is easy to learn and use; safe from misinterpretation and misuses; and capable of withstanding logical scrutiny, possessing visualised variables, as these are essential for a computer program's operation and subsequent understanding. Programs comprise variables, and operations on the variables: for loop control, functions and procedures in functional and procedural based languages, to classes and structures in Object Orientated Languages.

### 5.3.3 Object Orientation

The PbD system uses program encapsulation as an implied part of programming design. It is based on the principle that a specific design of mechatronic device is used for each program design, and that encapsulation allows specific behaviours to be designed without the problem of external reference. This shown in figure 5.10, below.

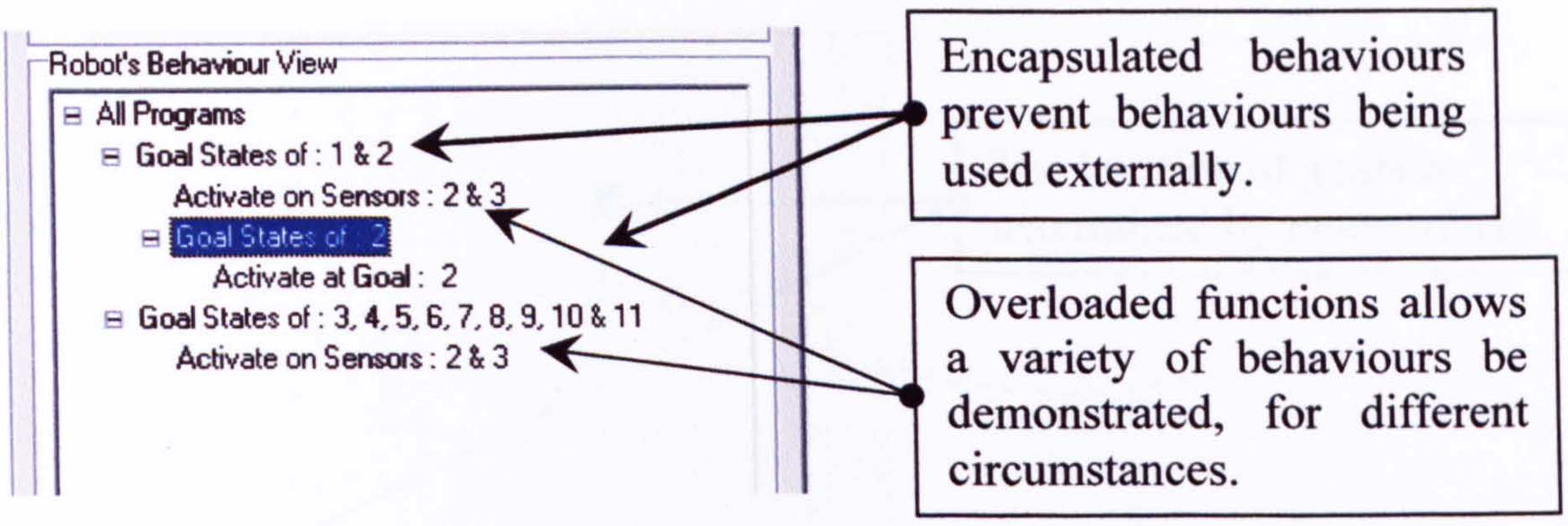



Figure 5.10 The advantages of Object Orientated Programming for the Programming by Demonstration interface

The program comprises functions defined by either sensor activation or by circumstance. The function names as demonstrated in Figure 5.10 above are:

- Sensor activation: '*Activate on Sensors: ...*'
- Circumstance activated: '*Activate at Goal: ...*'

Polymorphism allows overloading of Sensor-Based Behaviours. Further to encapsulation, is inheritance, which is used for the development of complex programs, where the overarching goal-based behaviour inherits and encapsulates a multitude of situation-based behaviour(s).

### 5.3.4 Goal-Based Behaviours

The procedure for implementing Goal-Based Behaviours and determining goals for them is shown in Appendix A. Goal locations are required in establishing the basic behaviour for a mechatronic device. Goal-Based Behaviours describe how a mechatronic device is expected to move from a starting location (goal) to an end location (goal). The placement of goals for Goal-Based Behaviours is illustrated in Figure 5.11 below.

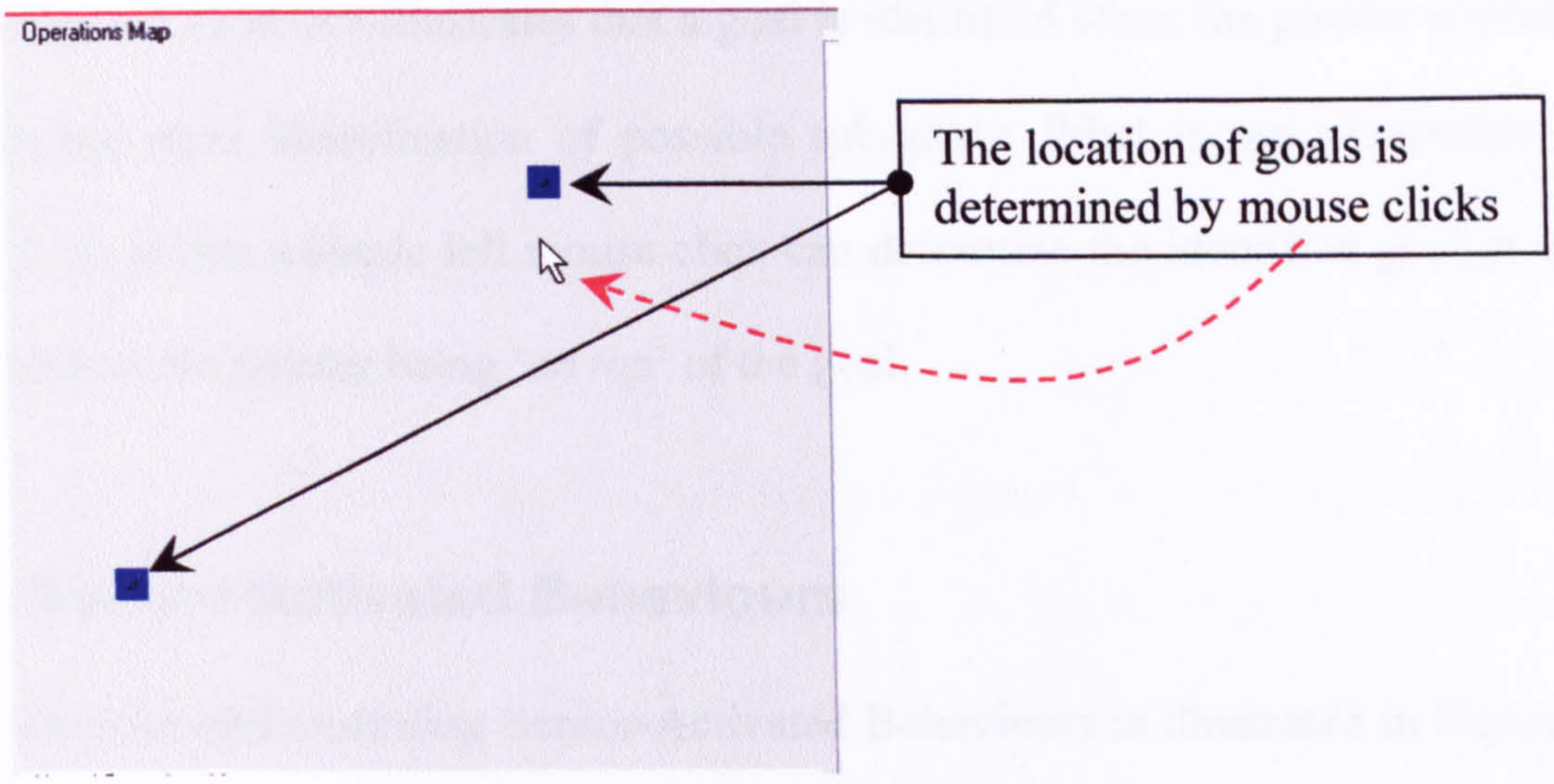



Figure 5.11 Locating goals on the Operations Map

A program designer can divide goals into sub-goals for designing a specific behaviour, as shown in Figure 5.12 below and these sub-goals can then be re-grouped for any subsequent specific behaviour. The establishment of a hierarchy of behaviours allows for both a complex composite behaviour and flexibility in developing behaviours.

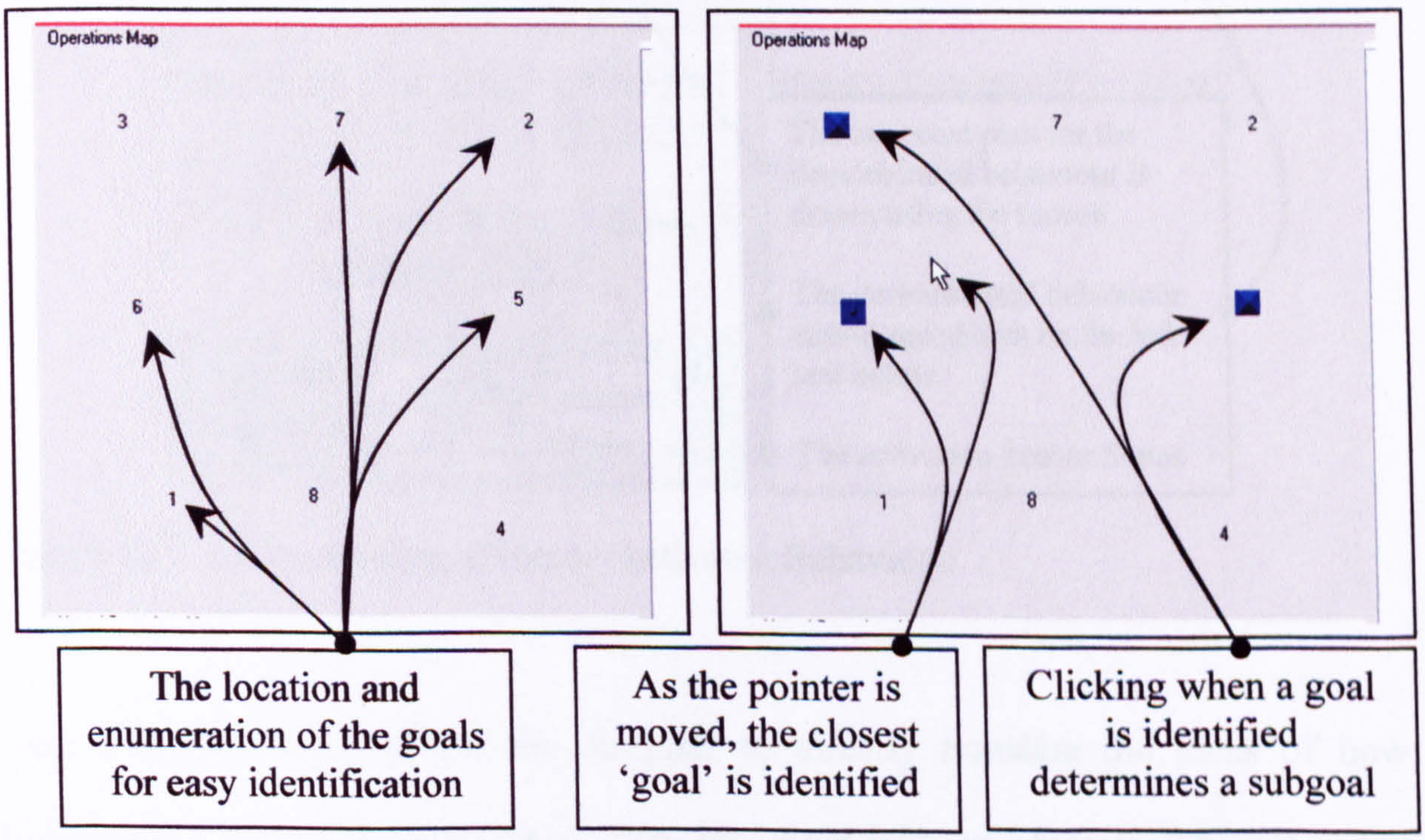



Figure 5.12 Identifying sub-goals

While figure 5.12 above illustrates that a goal is identified when the pointer approaches it, allowing rapid identification of possible sub-goals. What is not observable from figure 5.12, is that a single left mouse click can determine the identified goal as a sub-goal, without the pointer being '*on top*' of the goal.

### 5.3.5 Sensor-Activated Behaviours

The means for demonstrating Sensor-Activated Behaviours is illustrated in Figure 5.13 below.

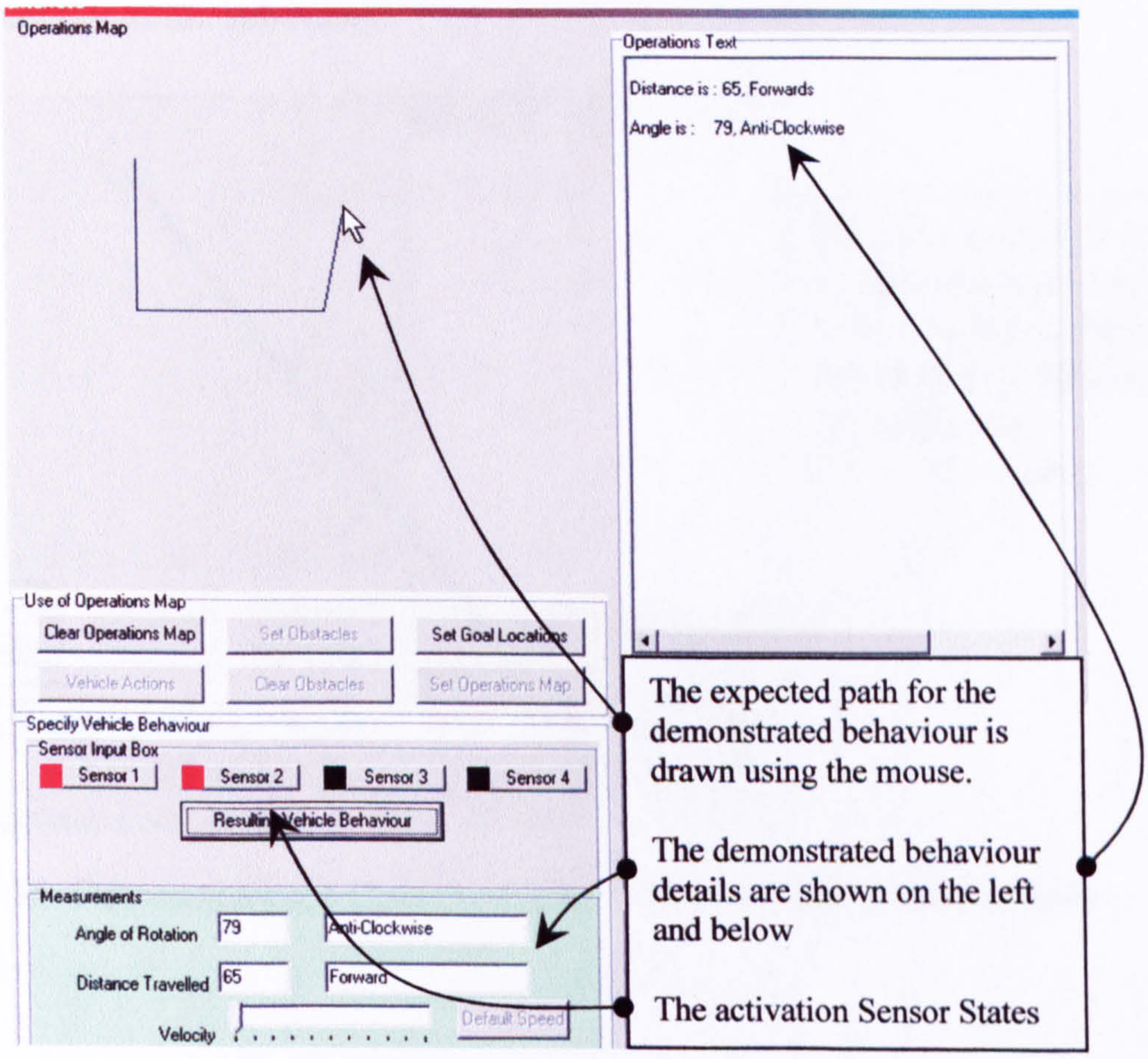



Figure 5.13 Demonstrating a Sensor-Activated Behaviour

The screen interface allows the designer to directly translate the ideas of how a mechatronic device should behave when presented with an obstacle. The behaviour is initiated by the sensor values determined by the designer.

### 5.3.6 Goal-Activated Behaviours

Goal-Activated Behaviours are more complicated than Sensor-Activated Behaviours. When the mechatronic device arrives at a specified goal this behaviour is identified and activated. The behaviour is an activity based on sensor interaction with the environment. The method of demonstrating this behaviour is illustrated in 5.14 below, with sensor activity demonstrated. The behaviour can include the use of sensors for interaction with the environment. In figure 5.14 the use of touch sensors is illustrated. If sensor states are activated, the resulting sensor values require confirmation before the designer continues the behaviour.

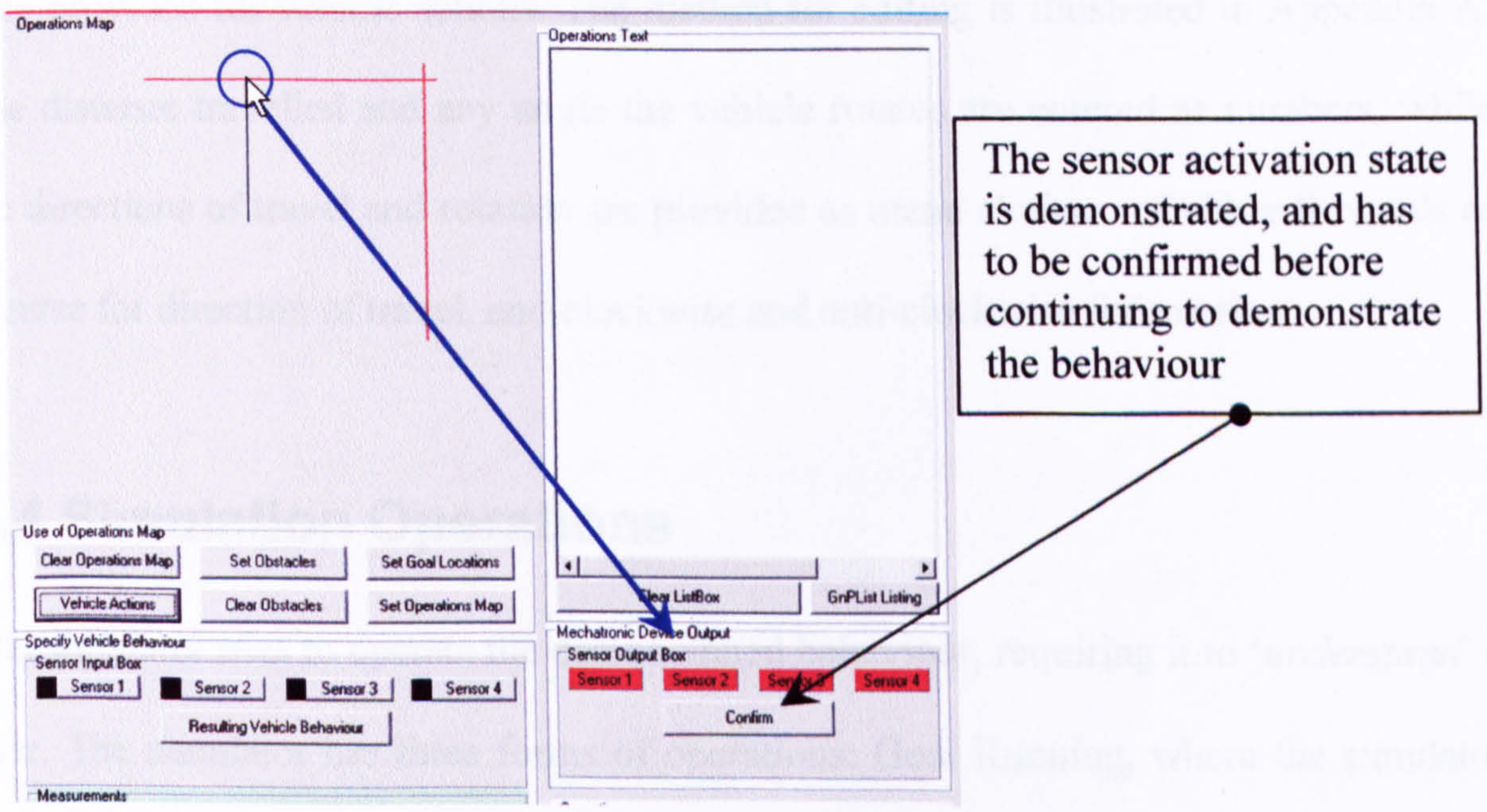



Figure 5.14 Demonstrating a Goal-Activated Behaviour with sensor activity

## 5.3.7 Editing

Hoare [1973] argues

> *program debugging can often be the most tiresome, expensive, and unpredictable phase of program development, particularly at the stage of assembling subprograms written by many programmers over a long period.*

The best solution is to not write '*buggy*' program code, and comprehensively comment on the algorithm(s).

Wirth [1974] argues that a programmer should choose suitable language features for a programming task, predicting all the issues of combining the language features, and a good language design will prevent programmer mistakes. With regard to both Hoare's and Wirth's assertions, the advocated Programming by Demonstration Language allows visual confirmation of the programmed actions, and editing is different from programming element, being undertaken by alphanumeric input. The editing feature is only provided for vehicle actions. The method for editing is illustrated in Appendix A. The distance travelled and any angle the vehicle rotates are entered as numbers, while the directions of travel and rotation are provided as menu choices: of either forwards or reverse for direction of travel, and clockwise and anti-clockwise for rotation.

## 5.4 Simulation Operations

The simulator tries to operate the demonstrated behaviour, requiring it to '*understand*' a user. The simulator has three forms of operations: Goal Running, where the simulator emulates a mechatronic device going between two user-specified goals. The second is sensor-activated operations, which are emulated when the simulated mechatronic device meets an obstacle. Finally, Goal-Activated Behaviour is emulated when the mechatronic device '*arrives*' at a user-specified goal for the behaviour's activation.

### 5.4.1 Goal-Based Operations

The simulator starts by determining if there is a goal to go to, then '*blindly*' traces the shortest path a mechatronic device would undertake, as shown in Figure 5.15 below.

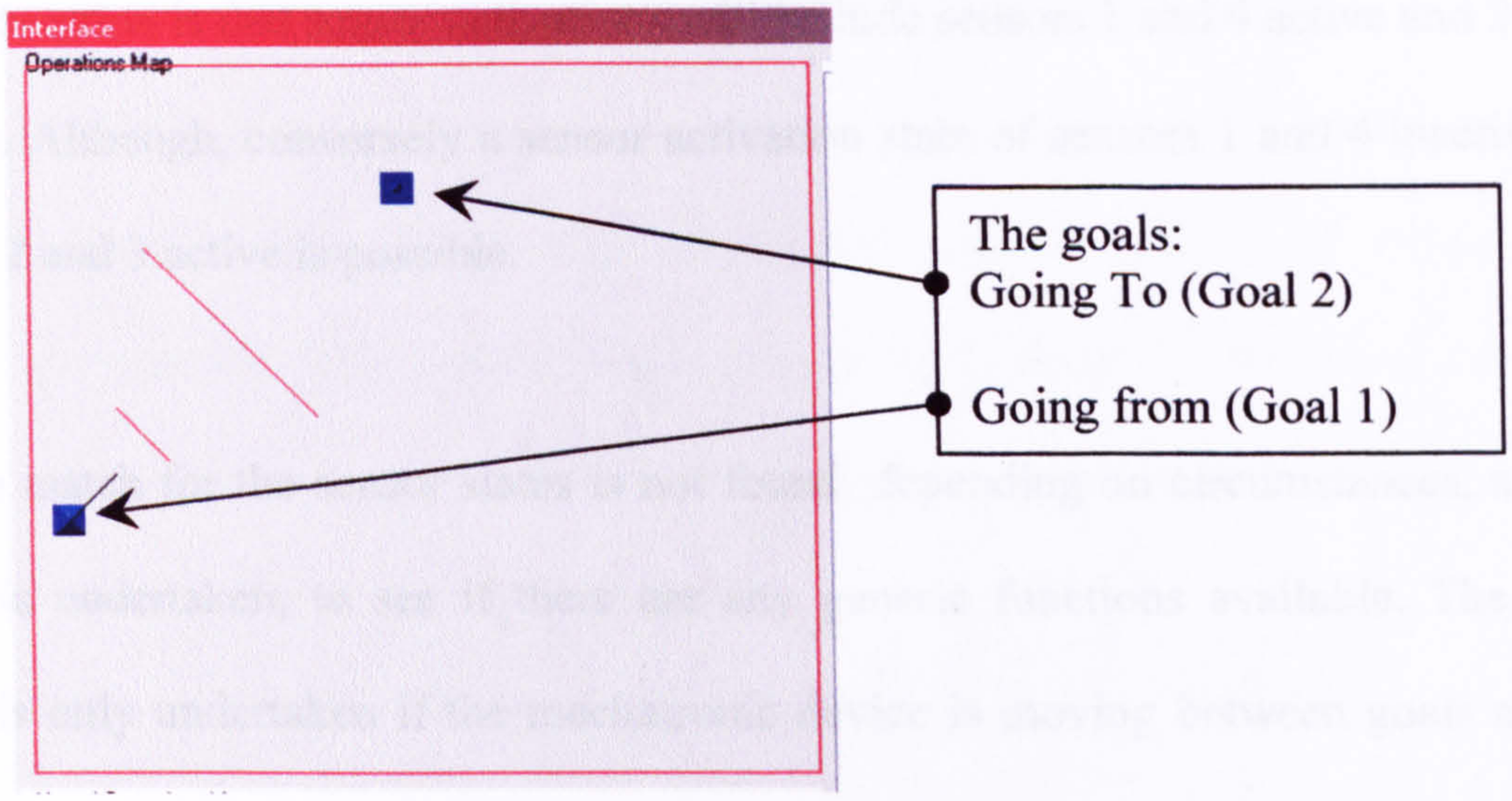



Figure 5.15 The simulation of a goal-to-goal based behaviour

If the path being traced meets a simulated obstacle, the simulator stops tracing the path further and starts the process of obstacle avoidance using Sensor-Activated Behaviours.

## 5.4.2 Sensor-Activated Operations

Sensor-Activated Behaviours are expected to be simulated for obstacle avoidance, but can also be used during the simulation of Goal-Activated Behaviours. When the behaviour was created, the demonstrated activating sensor values were based on AND logic. To determine which Sensor-Activated Behaviour to use when the simulator meets an obstacle, a combination of AND and OR logic is used, as shown in Table 5.1 below. This is a simplified search pattern which provides an efficient means for obtaining a best fit of available functions to the current sensor activation.

| Sensor truth value | Search pattern | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Sensor 1 = True | True | True | False | True | False | False | True | **Fail** |
| Sensor 2 = True | True | True | True | True | False | True | False | **Fail** |
| Sensor 3 = True | True | True | True | False | True | False | False | **Fail** |
| Sensor 4 = False | False | True | False | False | False | False | False | **Fail** |

Table 5.1 The search pattern for determining a best fit of sensor activations to a Sensor-Activated Behaviour

The assumption is that sensor activations will include sensors 1 and 4 active and 2 and 3 inactive. Although, conversely a sensor activation state of sensors 1 and 4 inactive and sensors 2 and 3 active is possible.

When a match for the sensor states is not found, depending on circumstances, a wider search is undertaken, to see if there are any generic functions available. The wider search is only undertaken if the mechatronic device is moving between goals and not undertaking a Goal-Activated Behaviour. The search is shown in Figure 5.16 below.

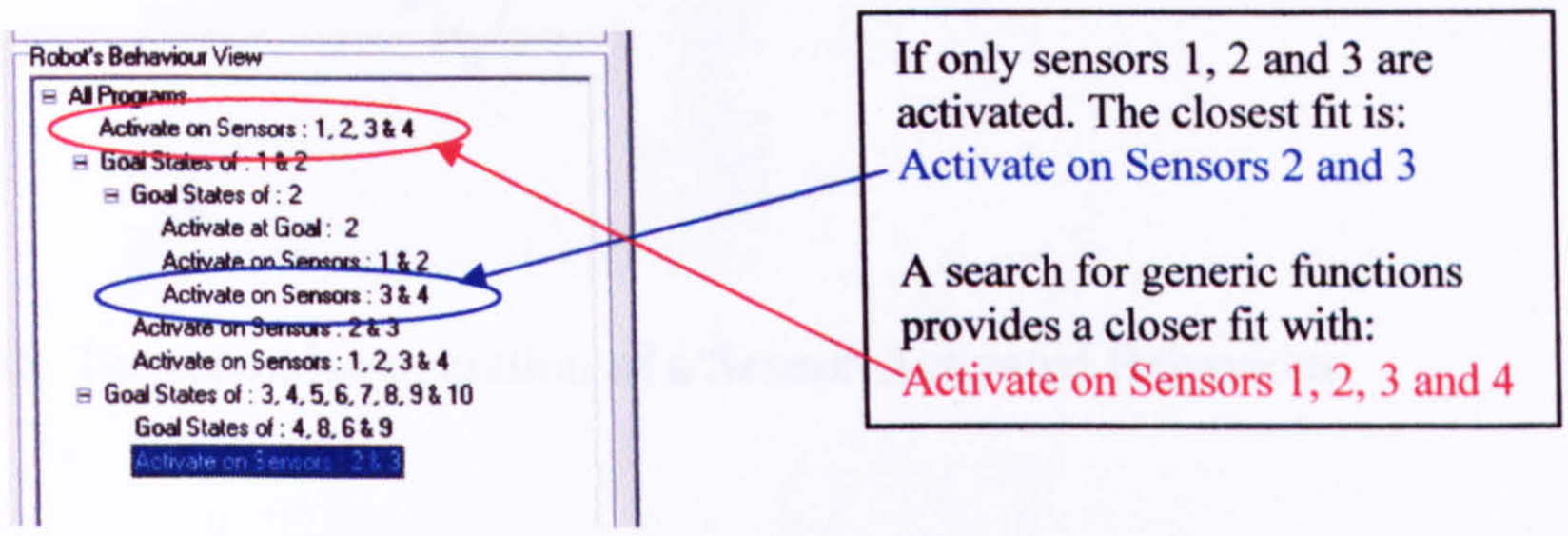



Figure 5.16 The search for Sensor-Activated Behaviours

For the demonstrated simulation, the most suitable Sensor-Activated Behaviour available is '*Activate on Sensors : 2 and 3*'. The simulator then displays the mechatronic device's predicted path in white, before tracing the path in black, as illustrated in Figure 5.17 below.

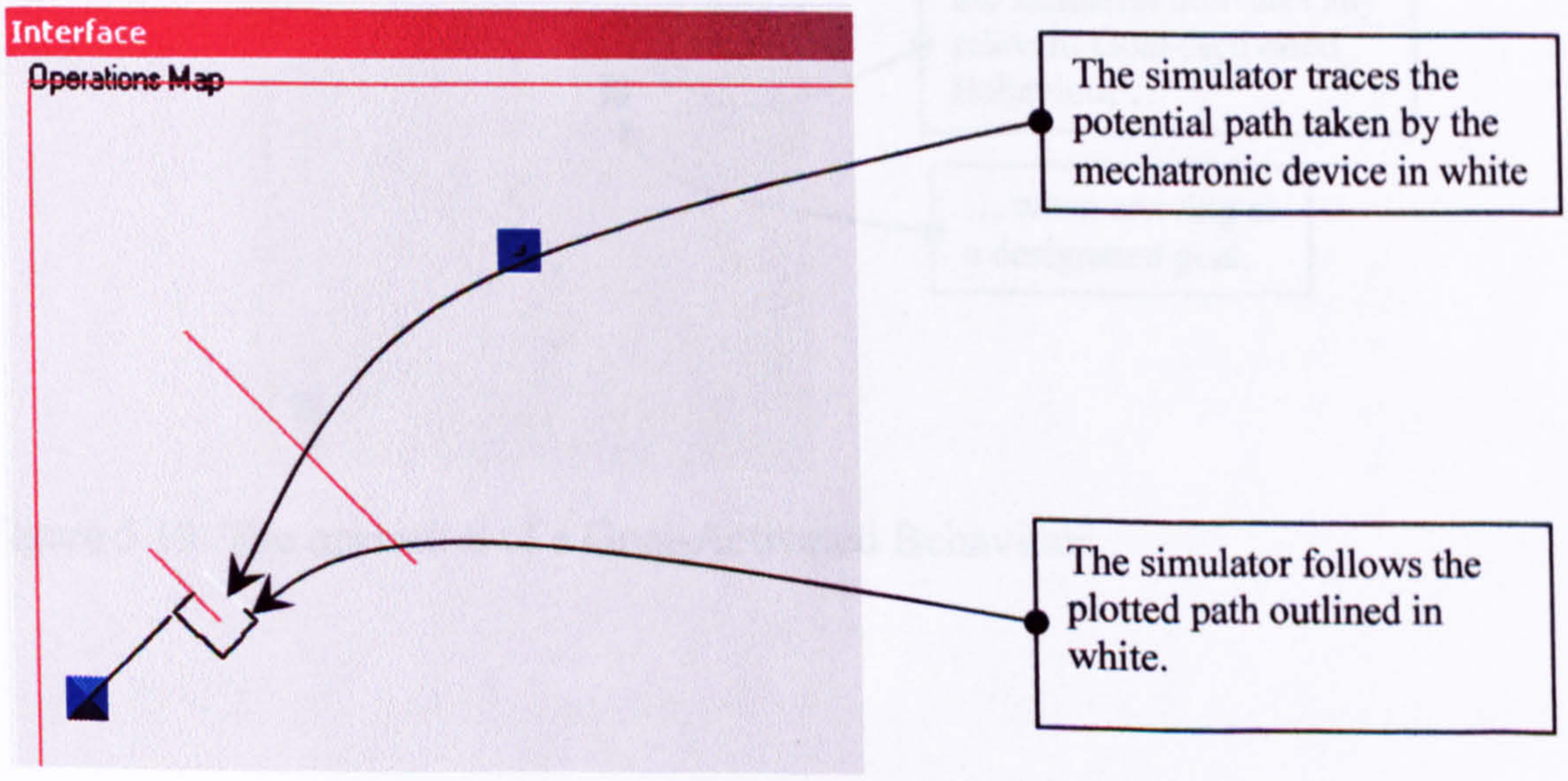



Figure 5.17 The operation of a Sensor-Activated Behaviour

When the Sensor-Activated Behaviour is finished, the Simulator resumes its previous behaviour. If the simulator meets an obstacle when running a Sensor-Activated Behaviour, the simulator searches for an appropriate Sensor-Activated Behaviour to navigate the obstacle. This can be seen in Figure 5.18, where the simulator detects an obstacle repeatedly, and recursively calls the same Sensor-Activated Behaviour.

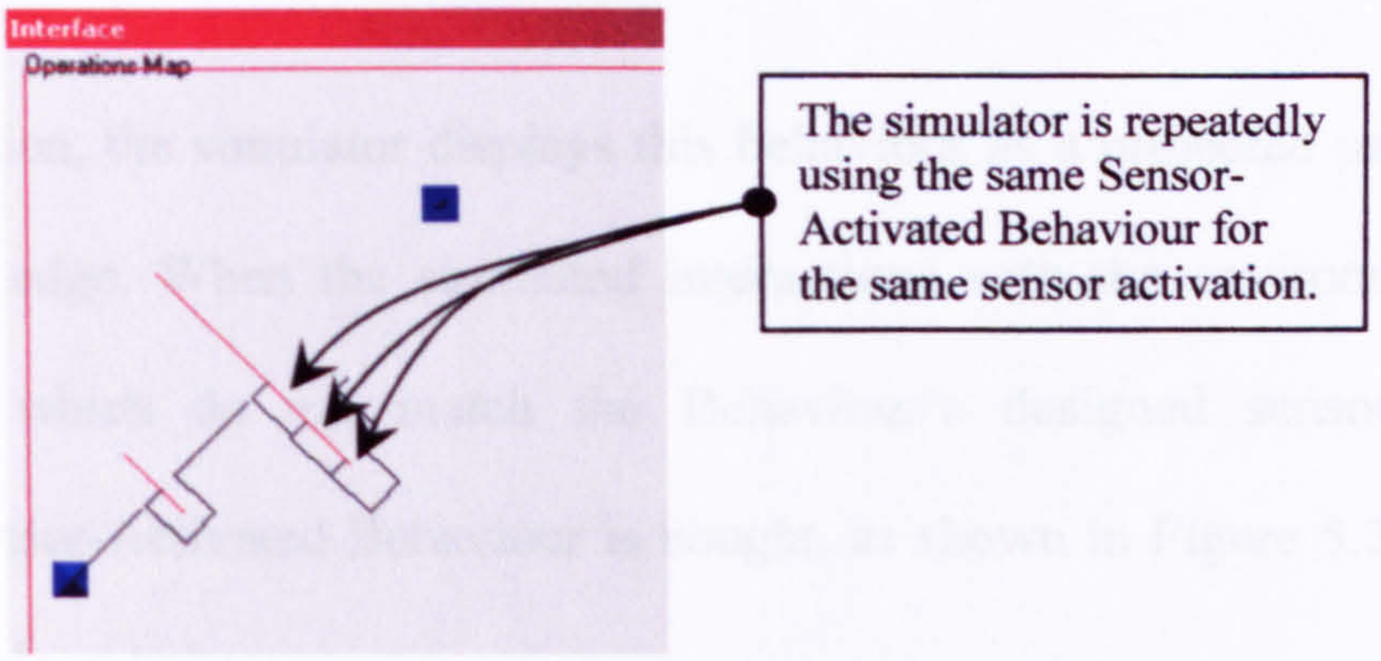



Figure 5.18 The recursive operation of a Sensor-Activated Behaviour

### 5.4.3 Goal-Activated Operations

When the simulated mechatronic device achieves a goal, the simulator determines if there are any Goal-Activated Behaviours, and simulates their behaviour, as illustrated in Figure 5.19 below.

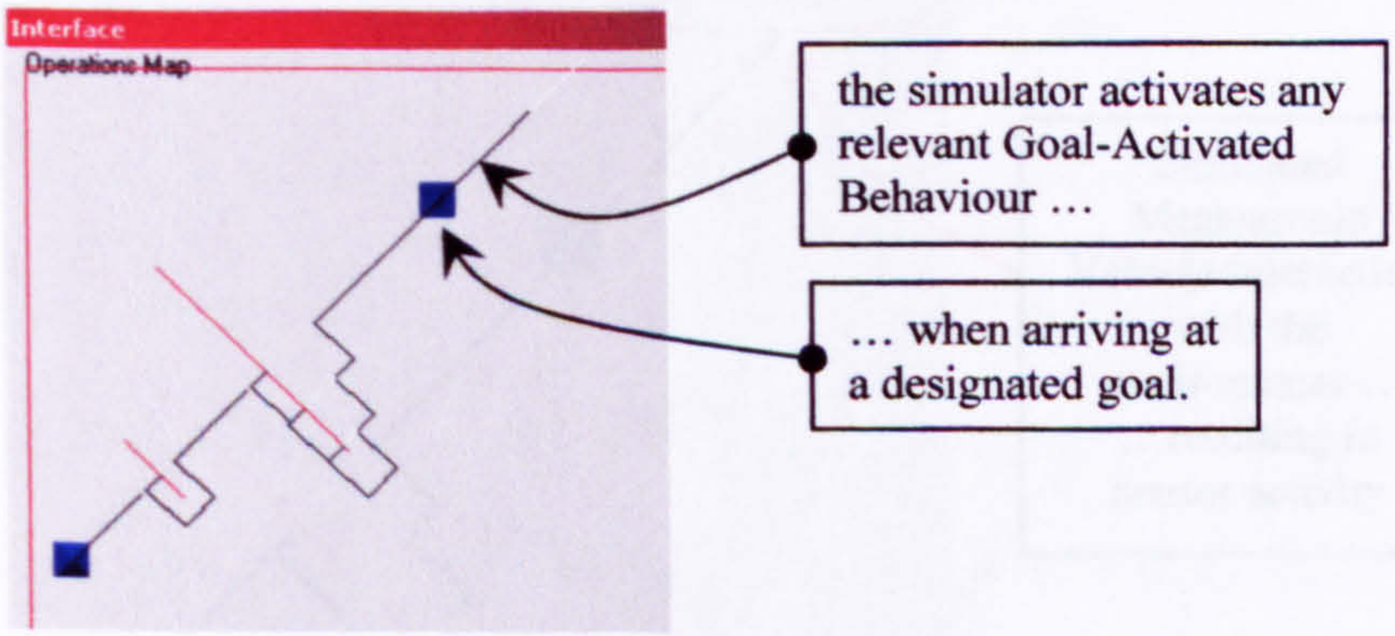



Figure 5.19 The operation of a Goal-Activated Behaviour

A Goal-Activated Behaviour comprises two forms of mechatronic device operations. The first is a specifically demonstrated path, where a device is expected to travel a specific distance. The second uses sensor activations to determine activities, the sensors values being used to demonstrate activity interactions with the environment. The demonstrated mechatronic vehicle activity continues until the sensors are activated.

During simulation, the simulator displays this behaviour as a projected path to beyond the laboratory edge. When the simulated interactions with the environment provide sensor inputs which do not match the Behaviour's designed sensor values, an appropriate Sensor-Activated Behaviour is sought, as shown in Figure 5.20, and if not found, the simulator halts.

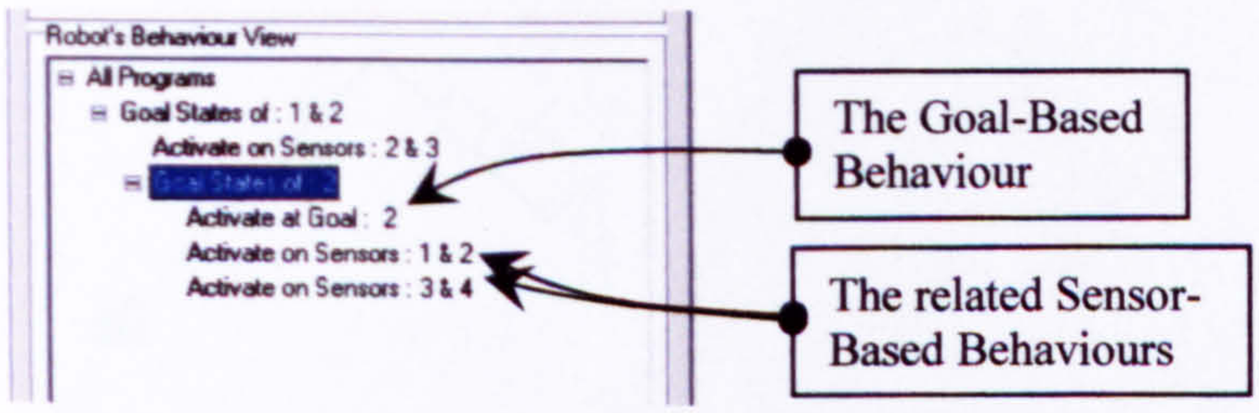



Figure 5.20 Locating a Sensor-Activated Behaviour relevant to a Goal-Activated Behaviour

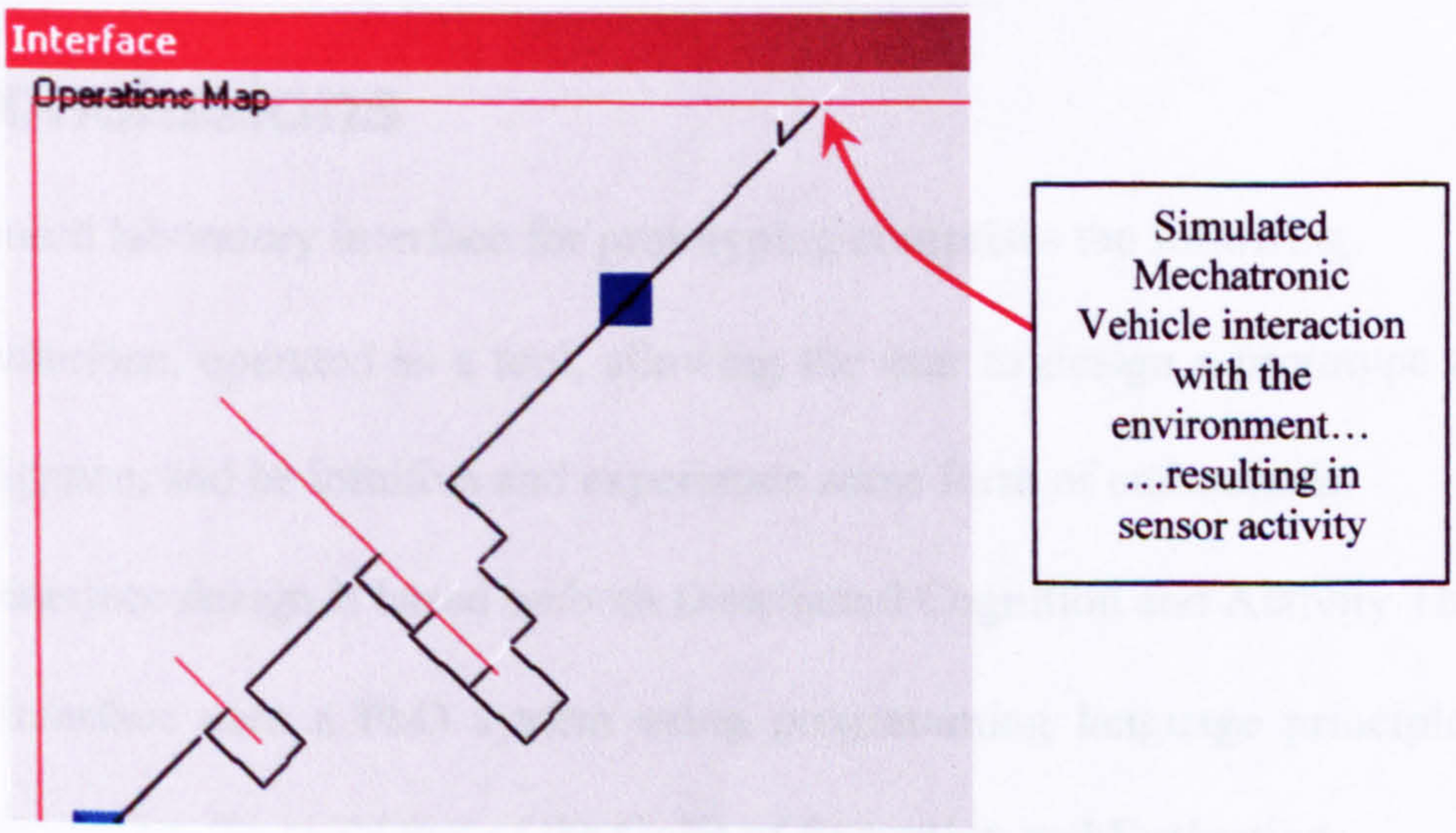



Figure 5.21 A Sensor-Activated Behaviour for continuing a Goal-Activated Behaviour

Figure 5.21 above illustrates the use of a Sensor-Activated Behaviour to continue a Goal-Activated Behaviour. The Sensor-Activated Behaviour allows the '*re-alignment*' of the simulated mechatronic device, for continuing the Goal-Activated Behaviour.

Completion of the demonstrated behaviour is shown in Figure 5.22, below. This illustrates the expected behaviour designed during the experiment explained in Chapter 7.

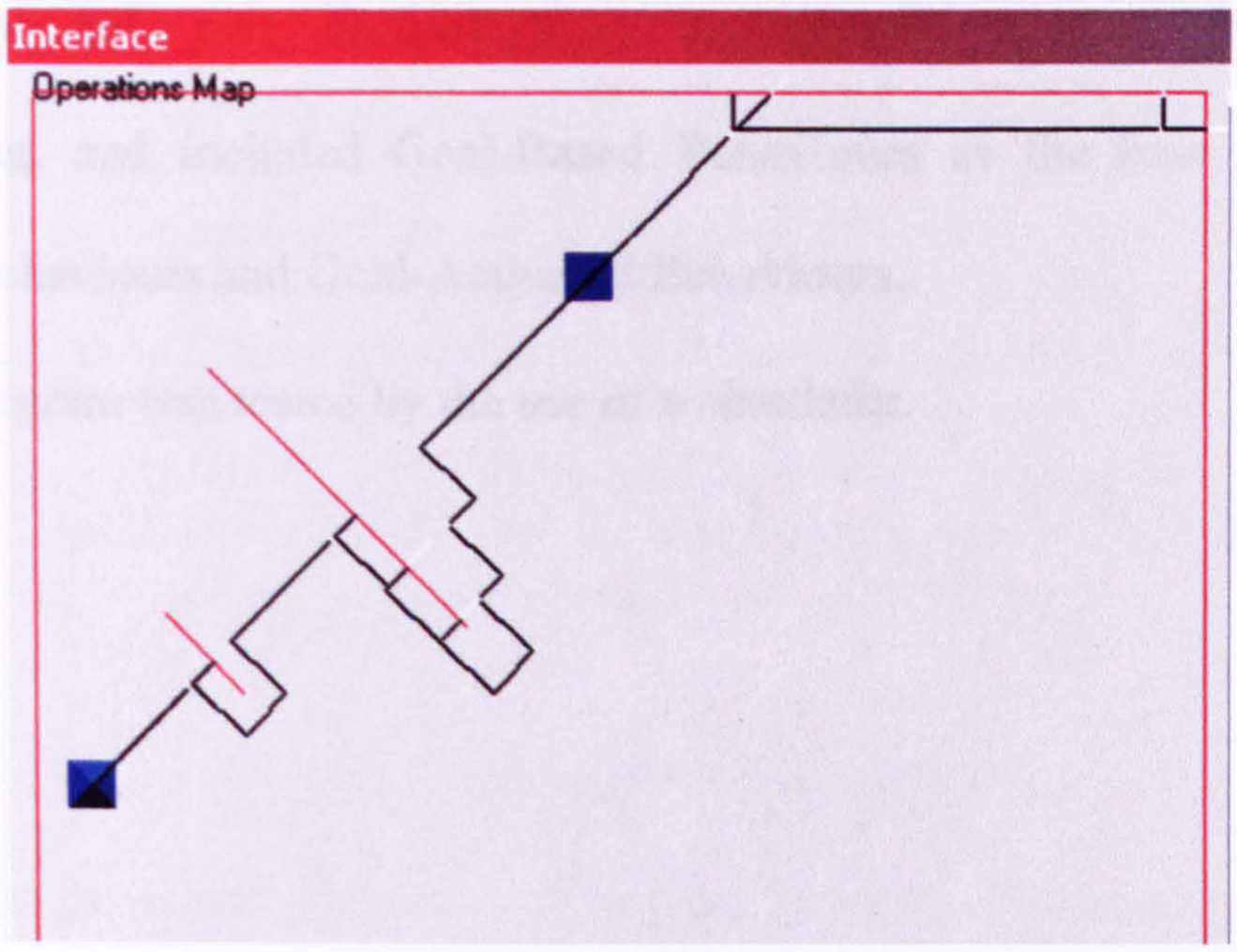



Figure 5.22 The completed simulation of the demonstrated behaviour

## 5.5 Conclusions

The proposed laboratory interface for prototyping comprises the following.

- The interface, operated as a tool, allowing the user to design a prototype Artificial Intelligence, and be intuitive and experience some form of enthusiasm.
- The interface design is based on both Distributed Cognition and Activity Theory.
- The interface uses a PbD system using programming language principles, while circumventing the problems of the Gulfs of Execution and Evaluation.

- The PbD system allows the user to design behaviours graphically, with the designer directly translating the behaviour from thought to a series of actions. This form of robotic programming has not previously been designed.
- The interface was prototyped, to allow interaction with intended user(s), providing reactions and suggested changes, with the advantage of accepting changes to the interface design early in its development.
- The PbD program design involved encapsulated program elements. The hierarchical program structure was derived from the principles of Object Orientated Programming, and included Goal-Based Behaviours as the base class, Sensor-Activated Behaviours and Goal-Activated Behaviours.
- The PbD program was tested by the use of a simulator.

# Chapter 6

# Design of the Experiment

## 6.1 The Definition of an Experiment

Festing [2001] provides a formal definition for an experiment, in his paper concerning ATLA (Alternatives to Laboratory Animals):

> *An experiment is a procedure for collecting scientific data in a systematic way in order to maximize the chance of answering an hypothesis correctly (confirmatory research) or to provide material for the generation of new hypotheses (exploratory research). Sometimes, an experiment is replicated in different laboratories or at different times, but provided that all replications involve the same scientific objective, and the data are suitably combined in the statistical analysis, it is considered a single experiment. Confirmatory research will normally involve formal significance testing, whereas exploratory research will normally involve looking for patterns in the data, and may not involve formal significance testing. However, there may be some overlap between these two types of experiment.*

This is an appropriate definition relevant to the experiment run to support the hypothesis. The experiment described in Chapters 7 and 8 collected data to support the hypothesis posed, as a confirmatory research experiment. It searched for patterns in data and there was a minimal mathematical component to the experiment.

## 6.2 Scientific Method

Central Texas Science and Engineering Fair, [2007] explains Scientific Method as:

> *The scientific method requires an initial proposal to explain initial observations. This is called the hypothesis. Experimentation is performed to reject, support or modify the initial hypothesis.*

An experiment is the study of cause and effect, with the Scientific Method considered the most productive to undertake for any experiment, involving the recognition of both control variables, and the deliberate manipulation other variables.

Wudka [1998] considers scientific method criticism that it often does not accept anything unproven, and facts previously thought impossible are now accepted. When investigating new events, existing theories are used and can be superseded by a new hypothesis to explain new experimental evidence, to develop a new theory. The feature of 'Experimental Method' in eliciting scientific knowledge is the deliberate manipulation of variables. Further, the precision involved in the manipulation and control of variables allows replication of the experiment, in all details. Kim and Kalb [1996] argues a well designed and performed experiment provides accurate data from significantly fewer experimental runs.

This experiment investigated the adoption of a new and novel form of programming for learning mechatronics. The precision in the control and manipulation of variables needed to be taken into account, as this experiment used human participants who can be unpredictable.

## 6.3 Design of the Experiment

> *The experiment might be conducted in a lab or in the "field". You might use direct observation to measure the [Direct Variables], or testing of some sort, or even a self-report, as in the imagery example. Try to be as objective and accurate as possible in recording your observations. Use a video or audio recording if necessary and possible, or prepare a checklist of target behaviours ahead of time, and keep accurate records. To control for experimenter bias, you might try to have an independent observer record and code the observations as noted above, ideally, one who doesn't know the purpose of the experiment.* [Waters, 2005]

### 6.3.1 Premises For Experiment Design

Kicinger and Wiegard [2003] proposes that good experiment design includes the following:

- **Exploration** The secondary research described in chapter 3 provided knowledge about the problem or system under experimental examination, in particular the environment for distance learning and the resulting human-computer interface.
- **Comparison** To define the controlled and measured variables during the experiment, with their levels or ranges of variation. This is described in sections 7.3.7 and 7.3.8. Analysis of the controlled and measured variables should determine variables fluctuations and outputs and prevent input variables worst-case conditions.
- **Explanation** An experiment creates a situation intended to examine any causal influence between two sets of identified variables. If the causal effect is identified during the experimental analysis, it is potential evidence that a causal link exists. This was undertaken in Chapter 8, in particular for the examination of participants' knowledge and experience, and the times taken to complete the tasks.
- **Demonstration** Experimental results need to be confirmed by a final experiment identifying the optimum results. This reveals any factors not tested which affect the results, with confirmation provided by unexpected results. However as this experiment tested human participants, the optimum results cannot be reproduced to order.
- **Theory Validation** In a proposed inferential experiment, the results support or contradict a formal causal relationship statement called the hypothesis. This is discussed in chapter 9, where suggestions for further research are made.

The premise for good experiment design has to consider the definitions above. The Alberta Government [2002] advises:

> *Two key considerations in designing an experiment are*
>
> *(i) Simplicity: By simplicity, we mean that the simplest experimental design be chosen among many possible candidates to achieve the same proposed objective(s).*
> *(ii) Efficiency: By efficiency, we mean that the investigation should be conducted as efficiently as possible; that is, every effort should be made to save time, money, personnel and experimental materials...*
>
> *Fortunately, most simple designs are also efficient (both statistically and economically).*

For this experiment, simplicity is provided by using just two types of system. Efficiency is provided by having only 10 tasks, which attempt to achieve a level of similarity between the two systems of: move between two points, detect obstacle, navigate obstacle, and park in a corner.

The Alberta Government [2002] considers the result of poor effective experimental design as

> *...the data collected can potentially be of little or no value to the attempted solution of the problem being investigated, due to little or no prior consideration given to the Design of the Experiment. The Design of an Experiment is the complete sequence of steps taken ahead of time to ensure that the appropriate data will be obtained in a way that permits an objective analysis leading to valid inferences with respect to the stated problem.*

Appropriate data was obtained by considering the two systems differences, and creating a series of tasks that finish with an identical task: parking in a corner, see Appendix B.

### 6.3.2 Optimising the Experiment's Design

In the Alberta Government [2002] paper, concerning how to undertake a research project, it proposes three basic principles to optimise experiment design:

- **Replication** the repetition of an experiment's treatments for two reasons:

  (i) '*Experimental error occurs when two or more identically treated experimental units fail to yield identical results*'. Repetition of treatments provides an experimental error estimate.

  (ii) Replicating the precision in the estimation of a factor's effect.

- **Randomization** applies the law of chance to experimental data, ensuring that experimental data be free from any systematic error, by making experimental errors independent and providing unbiased estimates of them.

- **Local control** experimental samples are grouped for homogeneity in each sample.

For the experiment conducted, there is a problem with replication, due to the use of human participants. Replication could only (in theory) be possibly achieved by the use of twins. The element of randomisation is achieved by the variety of participants, no two of whom can be considered identical or having any element of co-variance. For local control, there was an attempt to created homogeneity in the participants, which can be observed in section 8.2.

### 6.3.3 Variables

Variables are properties or characteristics of events, objects or people which can vary. They can be categorised into various groupings:

- **Dependent and independent variables** An independent variable is manipulated during an experiment, while a dependent variable is affected by an experiment's independent variable(s).

- **Qualitative and quantitative variables** Qualitative variables describe their attributes without numeric ordering, or are evaluated by categories. Quantitative variables are measured numerically.
- **Discrete and continuous variables** Discrete variables have discrete points on a scale. Continuous variables have a continuous scale.

The variables for the experiment are discussed in chapter 7, Experimental Procedures, section 7.3. However, any variable not recognised and considered may prove to be a confounding (uncontrolled) variable.

Graham [2006] considers the following as essential for the use of variables in an experiment:

1. ***Population size*** Too small a population can cause a significant difference between sets of data when none exists, or can indicate that no difference exists when there is one. Choice of sample number is explained in sections 7.3.3 and 8.2.
2. ***Components of variation*** The ability to impose control on the independent variables. The recorded components of variation for the experiment are explained in sections 7.3.4 and 8.2.
3. ***Randomisation*** A random experiment or trial provides results or observations, which are unpredictable or uncertain. It was intended to prevent any bias within the experimental data obtained. There was considered an element of randomisation when using human participants. Experience, knowledge, motivations and perspective are not identical for any two participants.
4. ***Blocking*** Blocking is when known or suspected biases are removed, and achieved by carefully ordering the tests to be performed to remove bias. This was achieved by using human participants, divided into two groups.

When dealing with the statistical analysis of experimental data, the population size has to be large enough to minimize the chance of either *Type I* or *Type II errors* occurring with the statistical analysis.

- **Type I error** The rejection of the hypothesis $H_0$ when in fact it is true. No observation is impossible and the probability of this error is the same as the level of significance.
- **Type II error** The acceptance of the hypothesis $H_0$ when in fact it is false. Unlike a Type I error which as a constant is only dependent on the level of significance, this error is dependent on which alternative hypothesis is true.

There was every attempt to prevent Type I and Type II errors. This was difficult due to the experiment not collecting statistical evidence, but using human experience and human opinions to gain sufficient data to support the hypothesis. The only solution was to use a sufficiently diverse sample from the parent population, and provide more than one measure to support or reject the hypothesis.

## 6.4 Experiment Data Analysis

Experimental data is based on a sample from some parent population, and a sample group statistic (parameter) is calculated as an estimate of the parameter's value in the entire population. Analysis involves calculating the following

- **Average or mean** These two terms are interchangeable. The sample group mean values are often compared for variables of interest. The mean for each sample group experimental timings was calculated and is shown in section 8.2.

- **Regression coefficient** This statistic gives the average change in variable Y for each one-unit increase in variable X. For linear regression, the coefficient is the straight line slope. A regression equation can be used to predict dependent variable value for an independent variable value. A regression co-efficient is calculated on the means as determined in section 8.2.

## 6.5 Error Analysis

> *Every experimental result is subject to error. One can attempt to minimize errors but cannot eliminate them completely.* [de Paula, 2001]

The consideration of experimental error is important for the successful completion of the experiment. These are the causes which lead to Type I and Type II errors, discussed in section 6.4.4 above.

### 6.5.1 Accuracy, Precision and Tolerance

Any physical measurement is subject to some degree of uncertainty due to the limitations of both accuracy and precision. All instruments have designed tolerances. Simanek [1996] argues

> *A measurement with relatively small indeterminate error is said to have high precision. A measurement with small indeterminate error and small determinate error is said to have high accuracy. Precision does not necessarily imply accuracy. A precise measurement may be inaccurate if it has a determinate error.*

#### 6.5.1.1 Accuracy

Accuracy refers to how close a measurement is to the correct value. It may be expressed in terms of absolute or relative error.

- **Absolute error** the difference between an observed (measured) value and the accepted value of a physical quantity, often referred to as experimental error.

- **Relative error** a ratio of absolute error to the accepted value, expressed as a percentage.

The accuracy of an experiment measurement can only be determined if, the measured quantity accepted correct value is already known.

The experiment conducted was considered to be accurate, reflecting the parent population's anticipated reaction to the prototyped interface compared to a previously developed text-based system. However the correct value is not known.

#### 6.5.1.2 Precision

Precision is the limitation of any measurements, determined by the scatter, or dispersal of obtained results, and a repeatedly measured quantity variance or standard deviation is considered an expedient guide to a methods precision. High precision is reflected by small variance. Precision is associated with random errors and can be improved by increasing the sample size.

The experiment was run with no expected values. Any scatter was assumed to be a human variation which would not be 'uniform'.

#### 6.5.1.3 Tolerance

Tolerance is the maximum allowable error for a measuring device. Instruments of high quality provide large measurement detail (good precision) and are designed for small error tolerance (good accuracy).

The measuring device was a video camera, and a 'computer screen' videoing station. These both give a timing resolution of 1/25$^{th}$ of a second. However, not every experiment was recorded with a time-frame reference. This was problematic for analysis.

### 6.5.2 Experimental Errors

Experimental error, when two identical experiments fail to give identical measurements, is due to:

- Mistakes made in implementing the experimental design, including mistakes in measuring experiment responses.
- All the uncontrolled variables combined effects which can influence the experiment's results whether identified by the investigator or not, but were assumed to be controlled through randomisation (randomised variables).

An anticipated problem was that a number of variables were based on experience.

**Errors:** Errors are normally classified in three categories:

1. Systematic Errors
2. Random Errors
3. Blunders

### 6.5.3 Systematic Errors

Systematic Errors come from identifiable causes and should be detected and eliminated.

This type of errors results in measurements which are consistently either too high or too low. These errors are a failure in accuracy. They may be of four kinds:

1. Instrumental. When a poorly calibrated instrument provides an error, it is a repetitive error. This error is consistent when the circumstances are repeated, and forms a bias in the statistics collated.

2. Observational. This is where the experimenter has made an identical and persistent mistake in all the statistics taken.
3. Environmental. This is where factors used as part of the model of the system provide a continuous erroneous output, such as low power outputs.
4. Theoretical. These are due to simplification of the model system or the use of approximations in the equations describing it.

The potential issues related to these errors in the experiment were:

1. The video camera could not be calibrated.
2. Experimenter error, not corrected by the experimenter identifying it.
3. The tasks undertaken. All had to be achievable by at least one participant.
4. The experiment tested a novel programming system, and did not have associated mathematical equations. The experiment did not involve a simulation to simplify the system.

### 6.5.4 Random Errors

Random errors are positive and negative fluctuations and the sources of such errors cannot always be identified. Possible sources are as follows:

1. **Observational** Errors in an observer's judgment when recording a measuring device (timer/potentiometer) scale
2. **Environmental** The unpredictable fluctuations in equipment performance.

Random errors, unlike systematic errors, can often be quantified by statistical analysis. Therefore, the effects of random errors on the quantity or physical law under investigation can often be identified.

Observational errors were resolved by using a video camera, for the analysis. The environmental factors could not be accounted for, but the assumption was that the video camera could operate within a fluctuating temperature range which is comfortable for the human participant, without significant change in performance. The random errors were considered to be negligible. However, there were no physical law or quantity to be measured.

### 6.5.5 Blunders

A final source of error, called a blunder, is an outright mistake. This is the recording of a wrong value, a misread scale, a forgotten digit when reading a scale or recording a measurement. A blunder would be evident if there were multiple measurements or if one person checks the work of another. There were no unanticipated blunders in the analysis of the experiment. Any blunders possible were the misidentification of the task being performed, or not identifying when there was an attempt to operate the RBS robot or PbD simulator.

## 6.6 Experimental Ethics

Scientific experimentation is about more than just the statistical tools available; it includes the scientist's decisions to obtain and interpret data. This includes the need for ethical behaviour in the performance of an experiment for future reproduction of results. The modern Ethical Experimentation code results from the Nuremberg War Crimes Trials.

### 6.6.1 Consent

The requirement is for informed and voluntary consent to participate.

The requirement is that the person involved is legally capable of consent, and can exercise free power of choice, the ESRC (the Engineering and Science Research Council) [2007] guidelines expect '*Research participants must participate in a voluntary way, free from any coercion*'.

The ESRC states further that the '*...subjects must be informed fully about the purpose, methods and intended possible uses of the research, what their participation in the research entails and what risks, if any are involved*'. Consent should be obtained in a consistent manner, as specified in the Research Ethics Framework, normally by use of a signed consent form, and sufficient time should be allowed from supplying the sheet to gaining consent, to prevent deception.

> *In all cases of research, researchers should inform subjects of their right to refuse to participate or withdraw from the investigation whenever and for whatever reason they wish. There should be no coercion of research subjects to participate in the research.* [ESRC, 2007]

This is summarised as '*the right to withdraw consent at any time*'. Further, any data provided is destroyed if requested, without adverse consequences.

Every participant was informed at the start of the experimental proceedings that the participant had the right to stop the experiments or leave the laboratory.

### 6.6.2 Responsibility

> *The responsibility for conduct of the research in line with relevant principles rests with the principal investigator.* [ESRC, 2007]

The Open University requires that approval for experimentation be obtained from the Open University Human Participants and Material Ethics Committee (HPMEC) before experimentation and data collection commences. Consent was obtained by identifying any possible risks to the participants and explaining how the experimental conduct would prevent any risk from occurring.

### 6.6.3 Experimental Preparation

> *Proper preparations should be made and adequate facilities provided to protect the experimental subject against even remote possibilities of injury, disability or death.* Nuremberg War Crime Trials [1949]

The Open University requires that there is a risk analysis before experimentation, with a risk management and harm alleviation protocol if necessary. The need is to make every effort to minimise the risk of harm, physical or psychological, from any researcher, institution, funding body or other persons.

This was undertaken for the issue of the robot falling off the laboratory work surface. To prevent this from happening, the work surface had a raised edge as a barrier.

### 6.6.4 Confidentiality

The Open University states that except through explicit written consent, researchers should respect and preserve participants' confidentiality. The experiment started with the participant signing a consent form which stated that no participant would be identified in relation to or by this research. Confidentiality of the participants has been maintained.

## 6.7 Conclusions to Design of the Experiment

The key issues addressed in the experiment were:

- The experiment was run by the principles of Scientific Method. This is a formalised method of designing and running an experiment, the principles of which were to determine the variables in the experiment, both dependent and independent, because any unrecognised variable would be a confounding variable. There was a sufficiently large sample to prevent a false result.
- The experiment design explored the environment for distance learning, and compared the controlled and measured variables, before examining for any causal links which needed to be explained, while it was kept simple by using just two types of systems for analysis, and efficient by having similar tasks for the two systems.
- The experiment was designed and conducted in such a way that the results could be replicated, preventing any systematic error. The problem was that human participants are not homogenous, and the experimental results were derived from human experience; the optimum results may not be reproduced to order, but were intended to support or reject the hypothesis.
- A problem to prevent with data analysis is Type I and Type II errors, which reject of the hypothesis when it is true, or support the hypothesis when it is false. The means to prevent these errors while using human participant knowledge and opinions, was to determine the results using multiple data sources for testing the hypothesis.
- This leads to the issue of preventing errors obtained during experimentation. Errors include incorrect recording of data and lack of precision subject to the tolerance in the measurement equipment. This can be due to mistakes in experiment design, comprising systematic errors, which could be due to the video camera, the experimenter, the tasks or the programming systems. There could be random errors, due to the experimenter or the equipment, or blunders.

- The use of human participants required ethical experimentation, with the participants both legally capable of consent and providing informed consent to participate, and with the right to withdraw consent at any time. Confidentiality must be maintained during reporting the results of the experiments.
- Preparation for the experiment has to include risk management and prevention of harm.

# Chapter 7

# Experimental Procedures

## 7.1 Introduction

A checklist is provided in synopsis on how to perform an experiment in the National Institute of Standards and Technology e-Handbook of Statistical Methods [NIST/SEMATECH, 2006].

*Checklist for successful DOE*

- *Check performance of measurement devices first.*
- *Keep the experiment as simple as possible.*
- *Check that all planned runs are feasible.*
- *Watch out for process drifts and shifts during the run.*
- *Avoid unplanned changes.*
- *Allow some time (and back-up) for unexpected events.*
- *Maintain effective ownership of each step in the experimental plan.*
- *Preserve all the raw data–do not keep only summary averages!*
- *Record everything that happens.*
- *Reset equipment to its original state after the experiment.*

The experiment was to evaluate the differences between an existing text-based programming system and a prototype multi-agent programming by demonstration system, based on participant performance. The experiment design was guided by StatSoft, Inc, [StatSoft, 2003].

The participants were required to use the two systems to accomplish a series of increasingly difficult tasks. Instructional information was provided with the tasks, for the participants to accomplish them. For example the rules based system included exampled instructions on how to program the rules into the system, see Appendix B.

## 7.2 The Aim of the Experiment

The aim of the experiment was to test the hypothesis, which was:

> **Programming by Demonstration would prove a more intuitive approach to the complexity of developing an emergent intelligent behaviour than text-based programming.**

To evaluate the hypothesis fully required analysis of the following:

- **The intuitiveness of the interface for design of programs**
- **The complexity of the designed robotic behaviour(s)**

### 7.2.1 The Objective of the Experiment

The experimental objective was to compare a Programming by Demonstration interface based system with an existing Mechatronics course (T395) interface system, providing evidence that the Programming by Demonstration system was:

- easier for participants who do not specialise in science or engineering subjects,
- easier for developing complex tasks,
- more flexible for programming the mechatronic device.

By directly comparing the two systems the experiments provided results for analysing the proposed programming by demonstration system strengths and weaknesses. The experiment revealed limitations, which could guide future research for distance learning systems, significantly improving the prototyped system.

To obtain desired results, each participant was expected to report on:

- **Ease of use** – the simplicity of methods to program tasks for a mechatronic device.
- **Rapid prototyping development** – which system allowed faster development of a mechatronic device autonomous behaviours.
- **Reduced mistakes and corrections** – using a Programming by Demonstration interface allowed a user to rapidly prototype behaviours and the mechatronic device to enact these behaviours.
- **Reduced need to learn new skills** – an issue of programming a mechatronic device was learning a programming language. A participant was hopefully expected to prefer the Programming by Demonstration interface compared with having to learn a computer language first, however simple the computer language.

To evaluate the hypothesis involved comparing a Programming by Demonstration interface's effectiveness with the existing mechatronics distance learning system.

## 7.3 The Variables

The dependent variables comprised analysing the characteristics of the participants. The independent Variables involved analysing the participants, who (as above) were categorised by:

1. The current education level. This was detailed as existing degree level qualification, from under-graduate to doctorate.
2. Previous experience background.

The independent variable was tested for by the participants completing a questionnaire before undertaking any actions on either of the two Interfaces.

### 7.3.1 Required Degree of Certainty 'In all things that are uncertain at the start'

This related the two different systems' operability being compared for their ease of use to achieve an expected task, and developing a program suite allowing the mechatronic device to operate autonomously.

Because the experiment analysed two different systems, utilising tasks of similar complexity for each, there was still uncertainty with the participants deciding which system was preferred, which was subjective for each participant. Reducing each participant's decision arbitrariness was achieved by directing their critique to a questionnaire, see Appendix C. The difficulty with a participant's critique was an overlap with pedagogy in the systems analysis. The experiments were intended to examine the practice of using a Programming by Demonstration System for the specific purpose of mechatronic device programming.

The principles of experiment design required records of relevant material properties and robot specifications, to allow any differences between the two systems to be identified and to prevent any skewing of results due to differences in system details.

### 7.3.2 The Participants

The participants were randomly selected, were given no prior notice of the details about the experiment to be undertaken. They were divided into two groups. One group undertook one system first, the second group undertook the other system first. The participants were in 2 groups of 10, which were considered to be sufficiently large to provide a statistically justifiable result, to reflect the parent population, and to support or reject the hypothesis.

### 7.3.3 Components of Variation

The categories by which the participants were assessed are:

- **Robotic specialists** Participants who were expected to be familiar with the principles of robotics, or specifically the T395 course system used as the alternative (text-based) system, from experience.
- **Graphic programmers** These were participants who were assumed to be familiar with the principles of graphic drawing, giving a potentially unique advantage in the use of the PbD system.
- **Computer programmers** These were assumed to be familiar with programming.
- **Computer users** These were assumed to be unfamiliar with the principles of programming, robotics or graphic programming.

### 7.3.4 Randomisation

Unpredictability could only be achieved by not knowing who the various participants were going to be, and the participants were unaware of the experiments in advance.

### 7.3.5 Blocking

For these experiments, the blocking was undertaken to avoid any prior experimental knowledge with one set of tasks, which could influence the results with the other system. From the four categories of participants defined in section 7.3.4, there were two groupings of each.

- One group undertook the T395 interface system tasks first, and then the Programming by Demonstration interface tasks.
- The second group undertook the Programming by Demonstration interface tasks first and then the set of T395 interface system tasks.

This prevented bias from only one experimentation format, causing a statistical skew.

### 7.3.6 The Independent Variables

The participants comprised Open University faculty members, students and other acquaintances, who were considered potential students for a Mechatronics course.

The constituency was expected to be sufficiently diffuse to allow a sub-ordering of results for analysis by:

- **Qualification by education level** Was the participant a member of faculty, post-graduate student or undergraduate currently studying for a degree? This allowed categorisation of participant ability to assimilate, understand and use new information.
- **Qualification by knowledge area** This categorised by familiarity with programming concepts. Too narrow a remit for knowledge would have skewed the results.

- **Qualification by age** was considered for 3 major reasons:
  - The humanities (psychology and education-based studies) argue that youthfulness allows greater versatility of thought processes, through greater ease or rapid assimilation of new unfamiliar concepts.
  - Computers are a recent development. Tuition of computer technology has proliferated to school level GCSE since 1988.
  - Age could be determined as a confounding variable if not accounted as a possible variable. As an uncontrolled confounding variable, any results provided could not be stated with any certainty.

  The alternative to a possible confounding variable (a variable that can affect the results but was not being measured), was to eliminate the variable. This would have been achieved by stating that each participant must be within a particular age range.
- **Qualification by computer literacy** The development of computers is relatively recent. In 1980 to 1990 home computing started to proliferate, comprising ZX 80/81/Spectrum, BBC A/B and Electron, Commodore Pet/64/128, Atari 5200. Mass manufactured Apple Mac and IBM/IBM compatible computers entered the general home ownership circa 1990. The internet had not become a popular tool until ~1992/5. Computer literacy as a pre-requisite through ownership and use of a computer was not a given universal.

The problems associated with these categories though had to be recognised.

- **The participant's education level** The hypothesis being tested included whether a PbD interface could allow the participant to program without relevant previous knowledge. While qualification level can be sub-divided into '*Had the participant got a Bachelor Degree? A Masters Degree or a PhD?*' the first degrees could be categorised further was a thorny issue.

- **The participant's specific knowledge** could likewise have been problematic. A participant may not have a single '*core*' of academic learning; there could be a knowledge spread, for example: my BSc(Hons) comprises both Business (Commerce) and Computing (Engineering) with significant (but not distinctly qualified in) Operations Research (Science/Maths), followed by an MSc in both Opto and Digital Electronics (Engineering). My external interests include music, (arts). Any categorisation of knowledge had to reveal programming-related knowledge.
- **Categorising the participants by age** could potentially yield an interesting skew on the results. Neither the participant's age nor how long a participant had used a computer could reliably measure either their ability or agility with computers or computer interfaces. However both length of time using computers and age could be more revealing. The younger a participant was, the more likely the participant had received a formal education in computer use, (GCSE/'A' Level). A more senior participant's computer literacy would be increasingly experience-based.
- **Categorising participants by computer literacy** There are two types of computer knowledge learning: formal and experiential. The categorisation could include:
  - the ability to navigate the operating system and use applications,
  - formal programming knowledge.

  If a participant whose computer literacy was based significantly on experience, navigating an operating system (GUI based, for example Windows or X-Windows) and application use, a question was whether the applications require any programming?

### 7.3.7 The Dependent Variables

Three variables could be analysed for the two systems:

- Total time taken to accomplish the tasks
- Number of errors
- Subjective satisfaction

The total time taken was measured in seconds from the beginning of the task until it was accomplished. The number of errors was the number of times the participant attempted to complete any of the tasks without success. Subjective satisfaction was assessed by using a questionnaire, Appendix C.

The experiment would obtain data to analyse the programming involved with both systems, for a mechatronic device to achieve increasingly complex tasks autonomously. The criteria for comparison were:

- **Time taken to complete tasks** to provide evidence of intuitiveness
- **Number of mistakes, repeats, retries and attempts** This was expected to be an indication of
  - **the intuitiveness of the interface,**
  - **the complexity of the task to be undertaken,**
  - **enthusiasm.**
- **Completion of tasks** A task would be unfinished for a combination of reasons. They require the ability to:
  - **understand the problem**
  - **consider a solution**
  - **program the solution.**
- **Test for efficiency:** This was the extent to which the system required minimal time to successfully complete tasks.

### 7.3.8 Maximising the Data Obtained

Maximising experimental data collection was achieved by using a videotape. The advantage was the ability to record the chronology of events, allowing a greater depth of analysis of the participants' behaviour, with recorded evidence of what each was experiencing when testing the two interfaces, through them undertaking a talk-through.

### 7.3.9 Minimizing the Required Number of Experiments

This was achieved by using videotape during experimentation, so there was a reduced need to repeat it. Further experimentation would have been to explore and analyse in greater depth any unexpected results.

## 7.4 Experiment Activities

The experiment was conducted at the Open University in a dedicated private laboratory. Participants entered the laboratory and were seated in front of a computer. They were asked to sign a consent form, and complete a questionnaire, comprising categorical questions about age, subjects studied and to what qualification level, and their background in computer education and experience.

Next, each participant was provided with an explanation of the experiment and instructions appropriate to the mechatronic programming system and interface being examined. After completing the second experiment system, participants were requested to complete a concluding questionnaire to examine their satisfaction with the two interfaces.

As the hypothesis sought to determine if a Programming by Demonstration System was an intuitive tool requiring no previous familiarity, there was no formal induction to either system used by the participant, other than a simple explanation that the systems were intended to program a robot to achieve a set of tasks within the experiment instructions. The tasks are attached in Appendix B.

## 7.5 Experimental Equipment

The resources acquired were:

A room with sufficient space for a work surface; a video-camera (fixed on a tripod), 2 chairs, one each for the participant, and the observer/experiment supervisor:

1. **Work surface** This needed to be large enough to support
    - **A personal computer** to operate both the programming systems,
    - **The work space** sufficient for a person to occupy
    - **A mechatronic device** The space required was estimated to be about 1metre square with the edges raised sufficiently to prevent the mechatronic device from falling off the workspace, and potentially injuring the participant.
2. **A personal computer** to operate both programming systems. The computer had an infra-red link for communication with the mechatronic device.
3. **A video camera** to record the participants' behaviour during the experiment. The video camera's positioning was intended to observe both the participant, and the activity on the computer screen.
4. **A computer screen recorder** to record the activity on the screen, which would be combined with the video camera view for the final document.
5. **Chairs** The participant and observer were both seated.

The layout of the experimentation room is shown in Figure 7.1 below

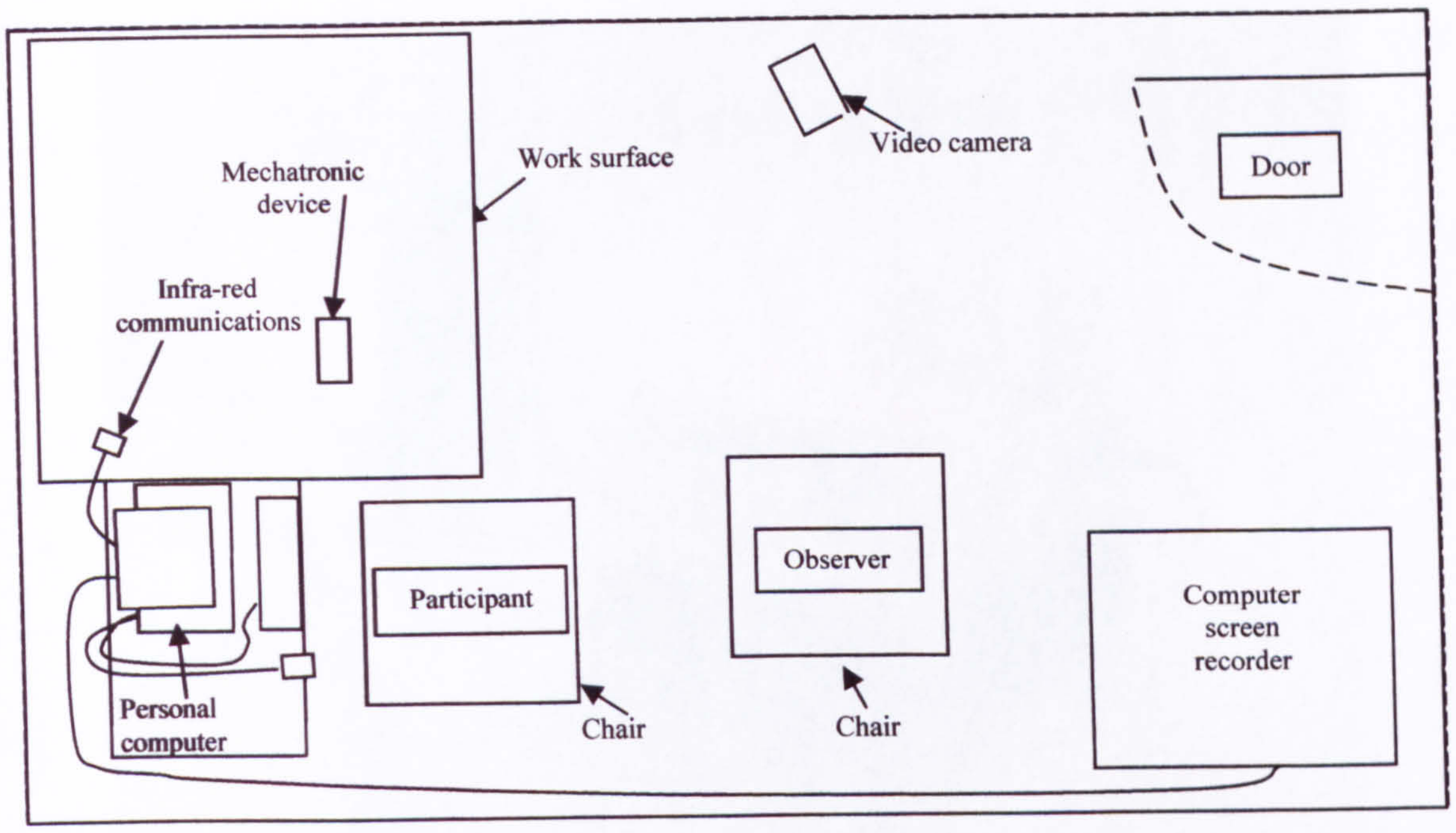



Figure 7.1 The layout of the experimentation laboratory

Figure 7.2 below shows the physical layout of the laboratory.

Figure 7.2 The physical layout of the laboratory

Figure 7.3 shows a close-up of the robotic vehicle used for the demonstration of the text-based system. The computer was run in stand-alone mode, and had copy of both systems on its hard-drive.

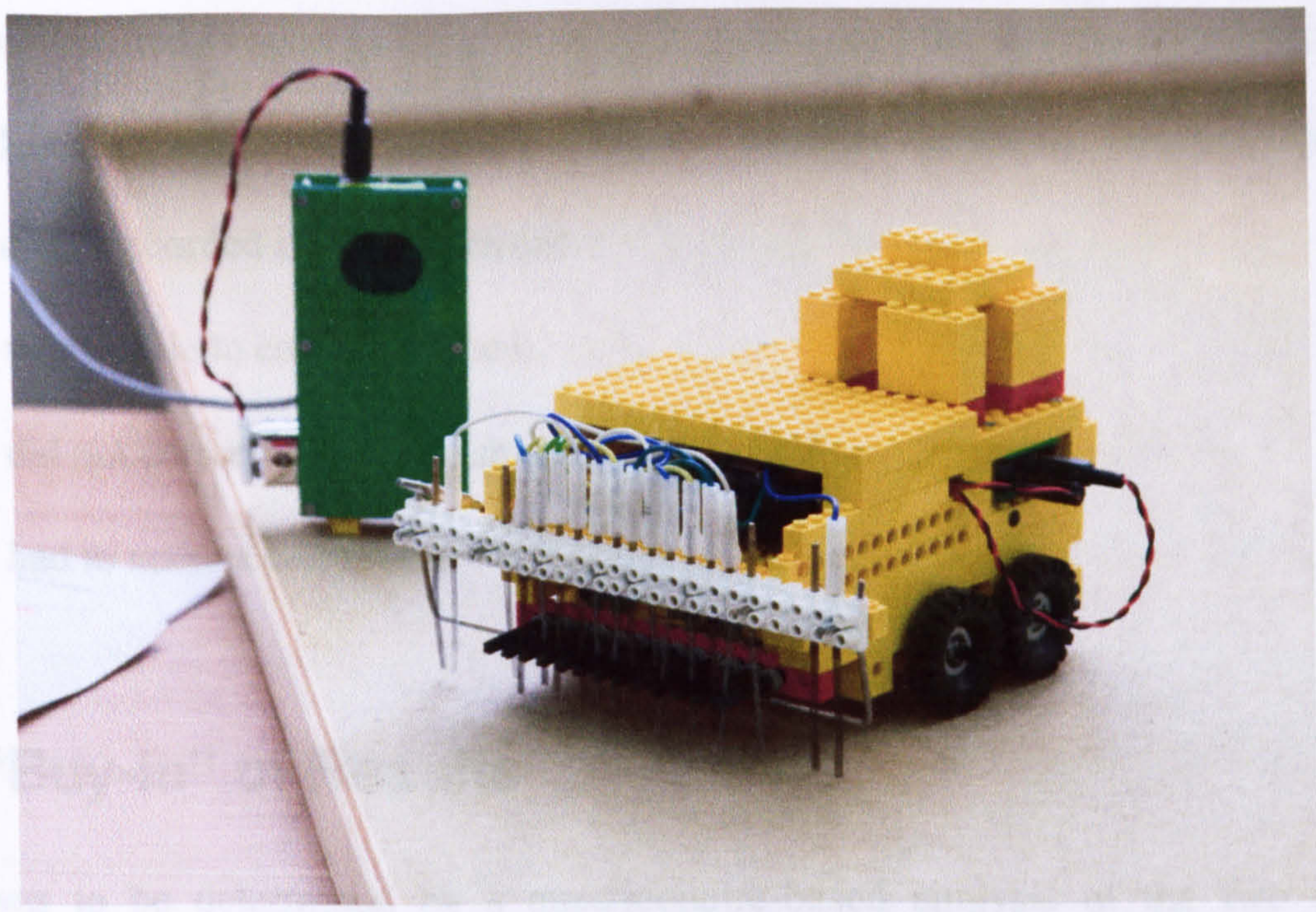


Figure 7.3 A close-up of the robotic vehicle

### 7.5.1 The Tasks

To test the effectiveness of a Program by Demonstration system, the tasks provided to both sets of participants were similar, see Appendix B.

Each participant was presented with an experiment packet containing a consent form, a preliminary questionnaire, and an exit questionnaire, reproduced in Appendix C. There were initial familiarising instruction of the two programming systems with formal instructions of what was expected for each task, see Appendix B.

The timed programme the participants were required to perform were

- a set of sub-tasks - these were for the robot to '*travel*' from one point to another in a controlled manner,
- to associate a range of inputs from sensors to resulting actions,
- to '*park the robot in a corner*'. This was a more abstract problem, requiring the participants to fully utilise the programming functionality.

The timing started as soon as the participant began to read the tasks to be accomplished.

The videotape recorded the participant from as soon as he/she entered the laboratory.

An error was recorded if the participant

- was unable to complete a task,
- did not correctly accomplish any part of a task,
- had to make a correction.

## 7.6 "Buy-in" of Results

This was to be determined by a questionnaire-based analysis of the two different systems, and the objective satisfaction of the participants. To obtain the objective satisfaction of the participants a questionnaire was utilised, comprising the questions:

- Which system did the participant consider was easier to use?
- Which system did the participant consider was more intuitive to use?
- Which system did the participant consider was more flexible to use?

## 7.7 Conclusions of Experimental Procedures

The following were concluded from Chapter 7

- The experiment was to compare an existing text-based system with the PbD interface.
- The experiment tested the hypothesis, evaluating the intuitiveness and complexity of designed robotic behaviours for the two systems.
- The independent variables being recorded were the current education of the participant and their previous background. The participants were randomly selected and analysed by education level, specific knowledge and, to prevent a confounding variable, age.
- The experiment tasks were of similar complexity in programmed behaviour.

- The dependent variables being measured were the total time taken to accomplish the tasks, the number of errors and the participant's subjective satisfaction with the system. This would comprise time taken to complete tasks, the number of mistakes, repeats, retries and attempts, the completion of tasks, and testing for efficiency.
- The experiments were video-taped to minimise the experiment runs and maximise data obtained. A questionnaire was presented at the end of the experiment.
- The hypothesised results were that the participants reported ease of use, rapid prototyping development, reduced mistakes and corrections and a reduced need to learn new skills, for the PbD system.
- The participants answered a questionnaire to determine which system they considered to be easier, more intuitive and more flexible to use.

# Chapter 8
# Data Obtained and Interpretation

## 8.1 Introduction to Data Obtained and Interpretation

This chapter describes the data obtained from the experimentation, and discusses the data's significance for supporting or rejecting the hypothesis. Throughout the chapter there are references to the two systems used for the experiment: the Rule Based System, previously successfully used for the Open University T395 home experimentation kit, and the Simulator System, which employs the Programming by Demonstration described in chapter 5. Throughout the chapter the T395 home experimentation kit is referred to as the RBS system, and the Programming by Demonstration/Simulator system as the PbD system.

## 8.2 The Participants

The 20 participants involved in the experimentation are profiled below in Table 8.1, which includes their formal qualifications, knowledge and experience. The participants were divided into 2 groups; the first tested the PbD system before testing the RBS, subsequently referred to as the PbD-first sample group; the second tested the RBS before the PbD system and are referred to as the RBS-first sample group. Both sample group participants were randomly selected.

Table 8.1 A description of the participants

| Participant | Sex | Age | Subjects Studied | | | Knowledge and Experience | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | First Degree | Second Degree | PhD | Computer Familiarity | Programming | Robotics | Graphic Programming |
| 1 | ♂ | 35-40 | BEng | No | Digital Imaging in progress | 22 years | ~20 years | None | Expert |
| 2 | ♂ | 60-65 | BSc | MSc | AI | 39 years | ~10years | Familiar | Familiar |
| 3 | ♀ | 26-30 | BA/BSc | No | No | 13 years | Principles | None | None |
| 4 | ♂ | 50-55 | BSc | No | Yes | 34 years | Sporadic | Familiar | Familiar |
| 5 | ♂ | 40-45 | BSc | No | In progress | 13 years | Principles | Expert | Expert |
| 6 | ♀ | 50-55 | BA | No | No | ~20 years | None | None | Expert |
| 7 | ♂ | 30-35 | BSc | No | In progress | 25 years | ~20 years | T395 | Expert |
| 8 | ♂ | 50-55 | BA | No | Digital Imaging | ~20 years | ~20 years | None | Expert |
| 9 | ♂ | 50-55 | BSc | MSc | Yes | 25 years | None | None | Familiar |
| 10 | ♀ | 35-40 | BSc | MSc | Yes | ~25years | ~10years | Familiar | None |
| 11 | ♂ | 25-30 | BSc | No | No | 15 years | 4 years | None | Familiar |
| 12 | ♂ | 60-65 | BSc | No | No | 40 years | 40 years | T395 | None |
| 13 | ♂ | 30-35 | BSc | PGCE | No | 26 years | 22 years | Familiar | Expert |
| 14 | ♂ | 60-65 | BSc | No | No | 25 years | ~100 hours | Expert | None |
| 15 | ♂ | 45-50 | BSc | No | Yes | 20 years | ~20 years | Expert | Expert |
| 16 | ♂ | 30-35 | BSc | No | In progress | 15-17 years | Principles | None | Expert |
| 17 | ♀ | 50-55 | BA | No | Psych in progress | 35+ years | 13 min | None | Familiar |
| 18 | ♀ | 36-40 | BA | MA | Psych in progress | 27 years | ~20 years | None | Familiar |
| 19 | ♀ | 46-50 | BA | MSc/MA | Business/ Sociology in progress | 20 years | None | None | None |
| 20 | ♂ | 35-40 | BSc | No | Yes | 25years | ~25years | Yes | None |

The distribution of the two sample groups in terms of their relevant experience is illustrated in the 'Star Diagram' in Figure 8.1. The 'Star Diagram' illustrates the 4 dimensions of the participants' experience. These were: the ability to program, experience in robotics, computer familiarity and graphic programming. Computer familiarity and programming experience were considered as distributed over time, although, the experience for either is not a linear gain. There is a significant gain early in the learning period, to obtain competence, with further gain being in specific areas of the subject.

Robotics and graphic programming were considered to be specialist subjects, and involve immersion in the subject. The diagram shows that there was a good spread of participants from the target parent population.

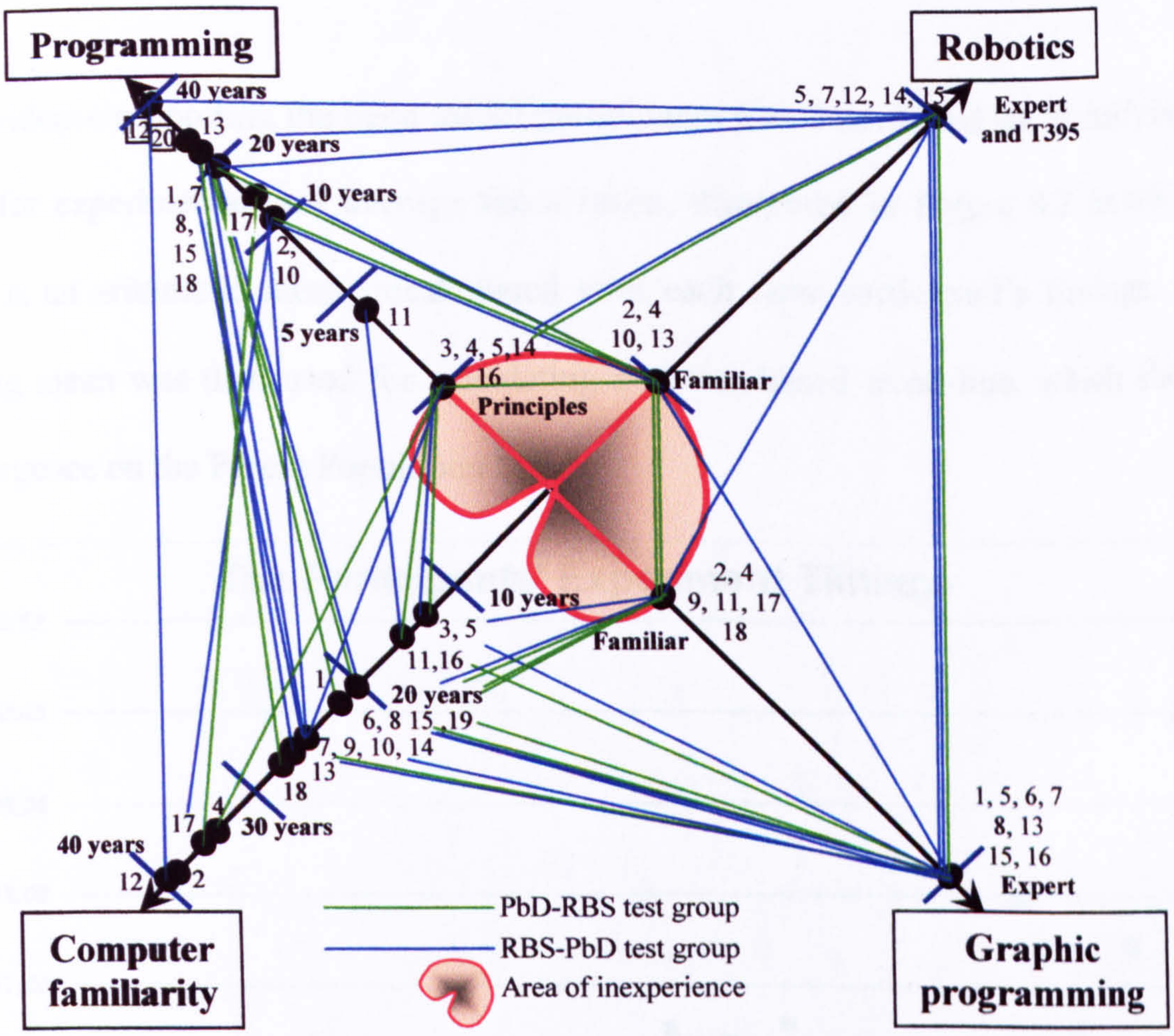



Figure 8.1 The distribution of the sampled participants

The sample of participants used for the experimentation was considered representative of the parent population. Familiarity with the T395 Open University Mechatronics Course was considered a measure of robotic expertise for the experimentation, as the Open University's T395 instructions were used as a premise for the Rule Based System's experimentation instructions. Two participants, 7 and 12, were respectively a graduate who studied T395 and a T395 course tutor, and both were considered to have expertise in using the system through prior familiarity.

There was a lack of 25+ years computer users who were experts in graphic programming, This is considered due to the development of home computing about 25 years ago, with computer users prior to 25 years ago being generally text-based programmers.

The evidence supporting the need for 20 participants was determined by identifying the trend for experimental task average times taken, illustrated in Figure 8.2 below. The mean is an arithmetic mean, recalculated with each new participant's timings. This moving mean was then used for calculating a power-based trend-line, which shows a convergence on the Parent Population Mean.

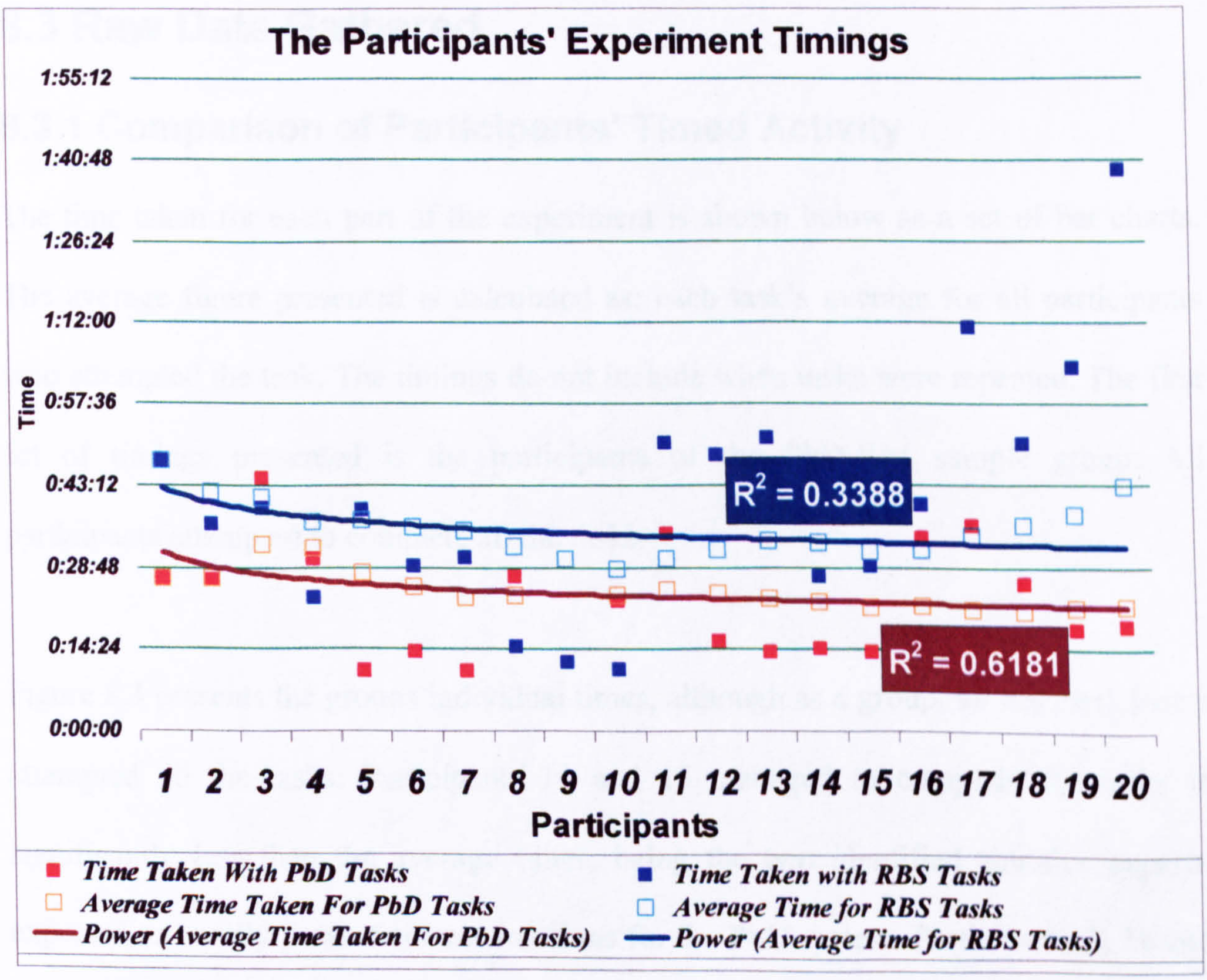



Figure 8.2 The participants' experiment timings

The $R^2$ value, is the '*co-efficient of determination*', calculated as the proportion of the variation in results, which can be explained with the calculation of the trend-line. The co-efficient is $R^2 = \frac{\text{Explained Variation}}{\text{Total Variation}}$ with the closer $R^2$ approaches 1 the closer the trend-line corresponds to the data. Although the $R^2$ value for the RBS is below 0.5, without participant 20, the $R^2$ value is 0.551, due to participant 20 taking a long time working on RBS task 5. Participant 20's results are discussed later. As the purpose of the experiment sample was to represent the parent population, this was considered achieved. Despite the individual scatter of results the conclusion is that 20 participants were sufficient for the experiment.

## 8.3 Raw Data Gathered

### 8.3.1 Comparison of Participants' Timed Activity

The time taken for each part of the experiment is shown below as a set of bar charts. The average figure presented is calculated as: each task's average for all participants who attempted the task. The timings do not include when tasks were repeated. The first set of timings presented is the participants of the PbD-first sample group. All participants attempted to complete all the tasks.

Figure 8.3 presents the groups individual times, although as a group, all the participants attempted all the tasks. Participants 14 and 15 managed to complete the tasks in significantly less than the average times, being the two identified robotics experts, expected to rapidly present correct solutions for the PbD system. Participants 3, 16 and 17 who took the longest time did not have robotics experience. Most noticeable is participant 16 who had graphical systems programming experience, but was significantly slower than the average to complete the tasks.

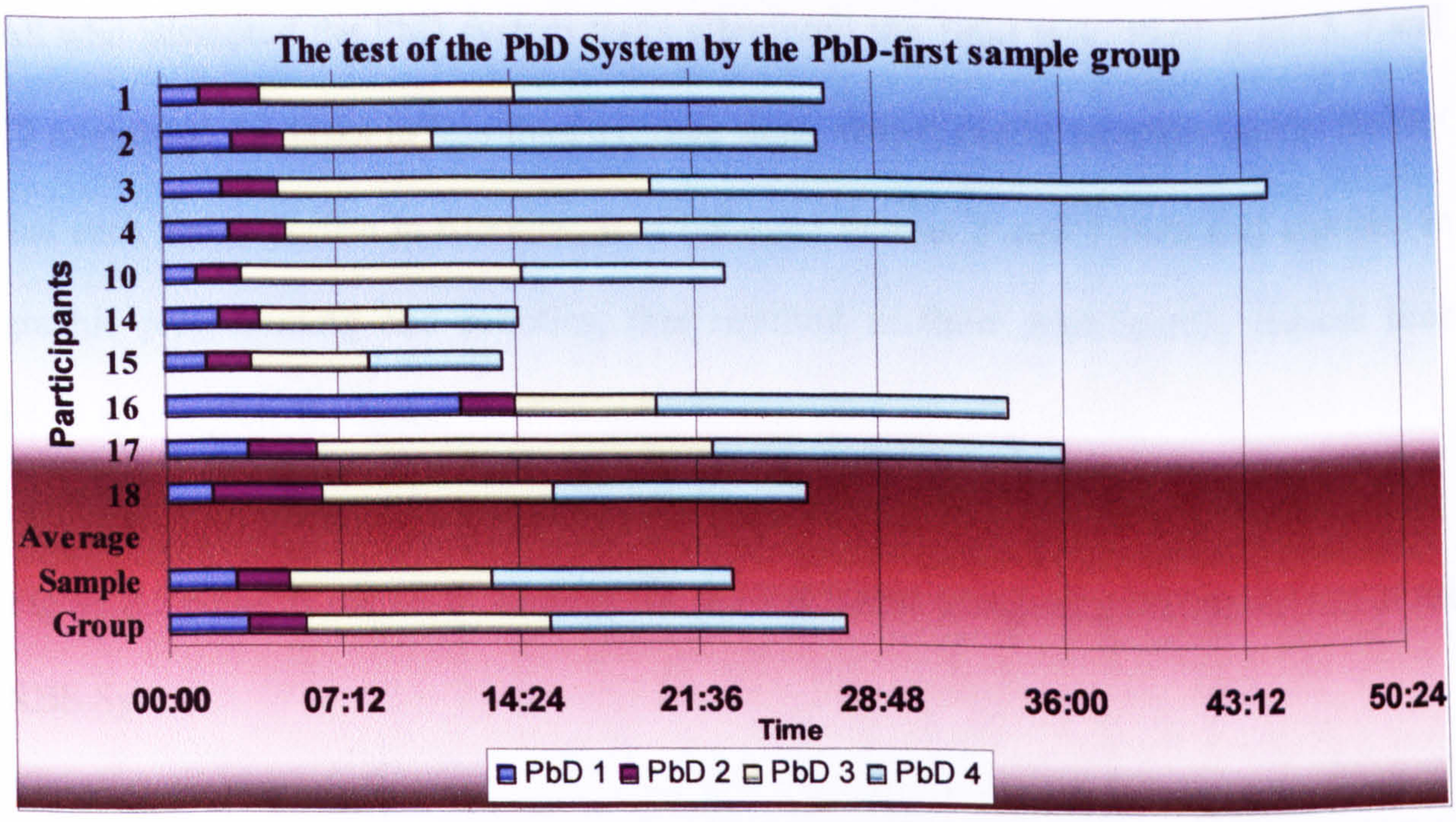



Figure 8.3 The timings for testing the PbD System by the PbD-first sample group

The second sample group were the RBS-first sample group participants. Participant 9 quit the experimentation during the Rule Base System tasks, otherwise all the remaining participants attempted all the tasks.

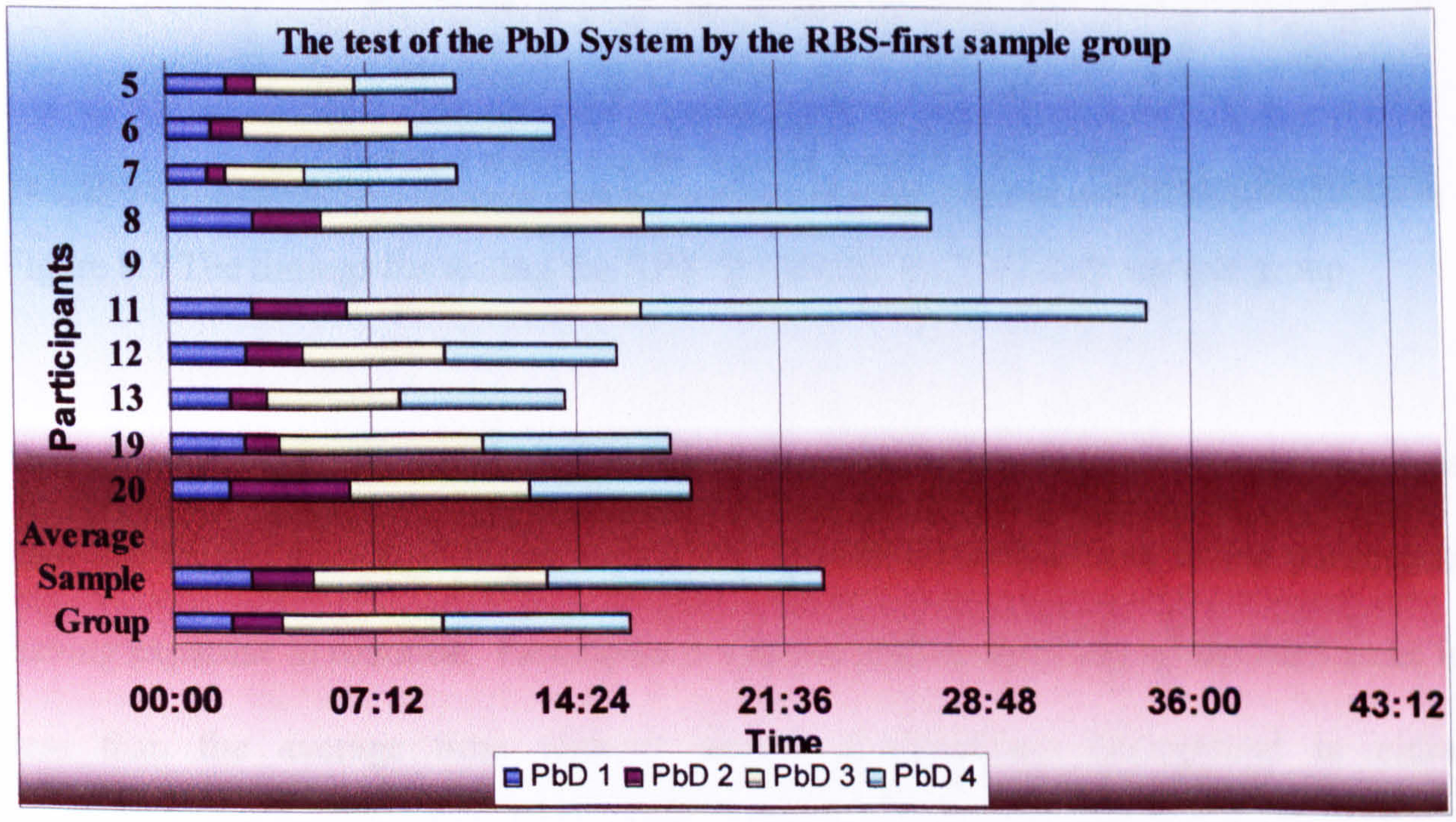



Figure 8.4 The timings for testing the PbD System by the RBS-first sample group

All who attempted the PbD system tasks attempted the final task. Participants 5, 7 and 12 completed the tasks in less than average time, which could be explained by the fact that each had expertise in robotics. It is believed that as 5 and 7 both had expertise in graphic programming and robotics, this resulted in their significantly reduced time taken to complete the tasks.

Figure 8.5 presents the set of timings for the RBS-first group of participants' test of the RBS System.

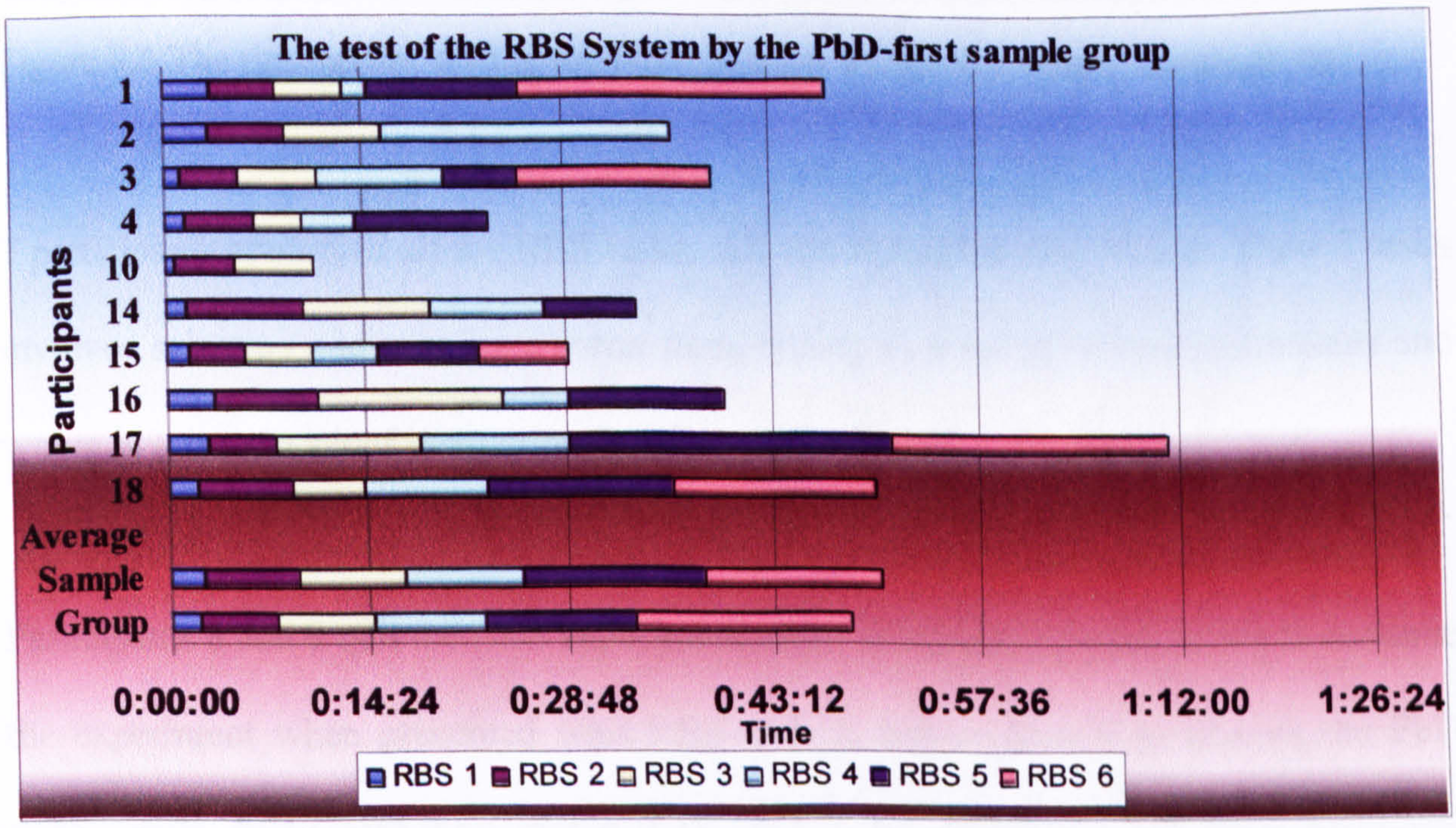



Figure 8.5 The timings for testing the RBS System by the PbD-first sample group

Only 7 participants attempted all the tasks. The only person to finish the tasks in less than the sample's average is participant 15. This is considered due to the participant having expertise in robotics. Participant 3 was the outlier, finishing all the RBS tasks in less than the average time without having a significant background in either programming or robotics.

Figure 8.6 presents the timings of the RBS-first participants' test of the RBS System.

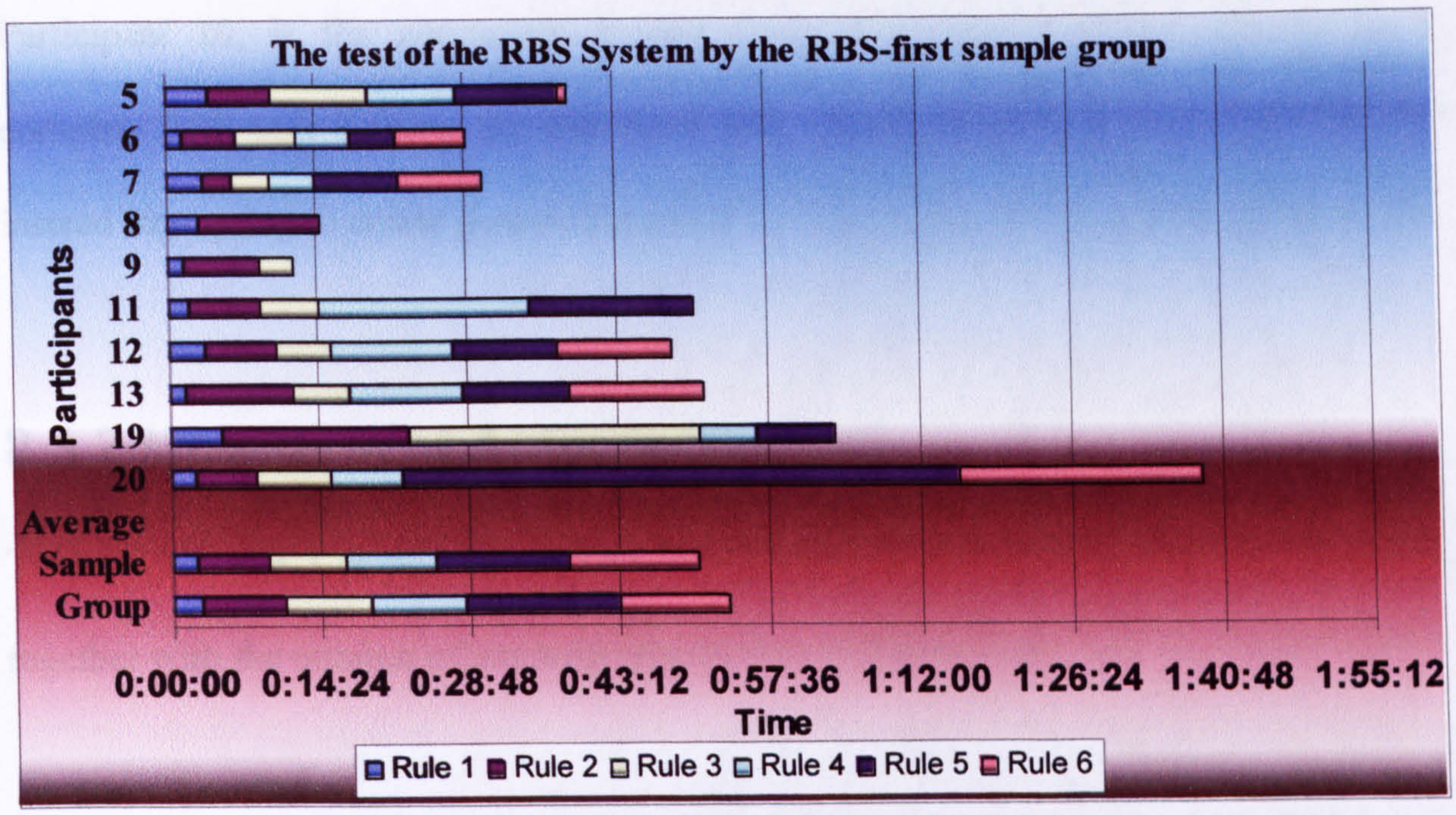



Figure 8.6 The timings for testing the RBS System by the RBS-first sample group

7 participants attempted all the RBS tasks. All attempted the first 3 tasks. These 3 tasks involved selecting and running a menu item, typing in a set of written instructions and typing in a second set of instructions similar to the first set, with 2 amendments.

Participants 8 and 9 quit the RBS experiment early. Participant 8 was unable to continue the experiment when presented with RBS task 3, but continued to attempt the PbD experiments. Participant 9 attempted RBS Task 3, but decided to halt the experiments and left the laboratory.

The outlier result is participant 6, who expressed no previous experience in robotics or programming, but who attempted all the tasks and succeeded in the shortest time. Further, there was an apparent disparity between the two sets of timings, with the PbD-first participants appearing to be quicker in completing the tasks. The possibility is that the group had experienced similar tasks when using the PbD system previously, while it has to be acknowledged that 5, 7 and 12 had a background in robotics.

Participant 20, is the anomalous result, however, deciding against attempting to complete RBS task 3 using the instructed task as the purpose of the experiment, but instead attempting to create generic behaviours which exceeded the task's expectations.

## 8.3.2 Measures of Activity Success

Tables 8.2 and 8.3 show who accomplished the tasks in the timescales given above, together with the number of attempts taken to complete the task.

Table 8.2 The PbD-first group's attempts to achieve success

| Experiment / Participant | Test order | PbD System Experiment Tasks | | | | RBS Experiment Tasks | | | | | | Stated Preference |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 5 | 6 | |
| 1 | PbD – RBS | 1 | 1 | 3 | 2 | 1 | 1 | 2 | 1 | 5 | 1 | PbD |
| 2 | PbD – RBS | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 5 | Quit | Quit | PbD |
| 3 | PbD – RBS | 1 | 1 | 3 | 1 | 1 | 1 | 3 | 9 | 6 | 1 | Rule Base |
| 4 | PbD – RBS | 1 | 1 | 5 | 8 | 1 | 1 | 0 | 0 | 11 | Quit | PbD |
| 10 | PbD – RBS | 1 | 1 | 2 | 1 | 1 | 1 | 4 | Quit | Quit | Quit | Rule Base |
| 14 | PbD – RBS | 1 | 1 | 2 | 1 | 1 | 1 | 4 | 2 | Quit | Quit | PbD |
| 15 | PbD – RBS | 1 | 1 | 3 | 2 | 1 | 1 | 2 | 2 | 3 | 4 | PbD |
| 16 | PbD – RBS | 2 | 1 | 3 | 5 | 1 | 2 | 5 | 3 | 2 | Quit | Rule Base |
| 17 | PbD – RBS | 1 | 2 | 5 | 3 | 2 | 2 | 5 | 2 | 11 | 6 | Rule Base |
| 18 | PbD – RBS | 1 | 2 | 2 | 3 | 1 | 2 | 1 | 4 | 1 | 3 | Unknown |

Key: Accomplished Task | Failed Task | Not Attempted

This showed that while the sampled group attempted but may not have completed the PbD tasks, there were 5 participants who were unable to attempt all the RBS tasks. This also shows that participants 3, 17 and 18 successfully finished the final task in the Rule Based System despite being unable to complete the tasks using the PbD system.

Participant 15 did not need guidance during PbD system task 3 on how to design the obstacle avoidance behaviour; however, the PbD simulator system failed to work properly. With the final task for the PbD system, participant 15 completed it with the minimum of help.

Table 8.3 The RBS-first group's attempts to achieve success

| Experiment Participant | Test order | RBS Experiment Tasks | | | | | | PbD System Experiment Tasks | | | | Stated Preference |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 | 1 | 2 | 3 | 4 | |
| 5 | RBS – PbD | 1 | 1 | 2 | 3 | 2 | 1 | 1 | 1 | 1 | 4 | PbD |
| 6 | RBS – PbD | 1 | 1 | 2 | 3 | 2 | 2 | 1 | 1 | 3 | 2 | Rule Base |
| 7 | RBS – PbD | 3 | 1 | 1 | 1 | 2 | 3 | 1 | 1 | 1 | 3 | Rule Base |
| 8 | RBS – PbD | 1 | Quit | Quit | Quit | Quit | Quit | 1 | 2 | 2 | 2 | PbD |
| 9 | RBS – PbD | 1 | 1 | Quit | Quit | Quit | Quit | Quit | Quit | Quit | Quit | **Unknown** |
| 11 | RBS – PbD | 1 | 1 | 2 | 6 | 1 | Quit | 1 | 1 | 1 | 1 | PbD |
| 12 | RBS – PbD | 1 | 3 | 1 | 4 | 2 | 4 | 1 | 1 | 3 | 2 | PbD |
| 13 | RBS – PbD | 1 | 3 | 3 | 7 | 4 | 14 | 1 | 1 | 3 | 2 | PbD |
| 19 | RBS – PbD | 1 | 1 | 3 | 1 | 2 | Quit | 1 | 1 | 3 | 1 | PbD |
| 20 | RBS – PbD | 1 | 2 | 1 | 2 | 19 | 8 | 1 | 1 | 1 | 1 | Rule Base |

Key: Accomplished Task | Failed Task | Not Attempted

## Catacreses

Catacreses are discussed in 3.2.1.3 as the modification of a 'tool' from its intended purpose, and were observed occurring during the PbD tasks. The PbD system had 2 tasks which comprised the design of a mechatronic vehicle's behaviours. Task 3, to negotiate an obstacle, and task 4, for the mechatronic vehicle to park itself in a corner.

For task 3, the majority of participants ignored the task instructions, which included the figure below.

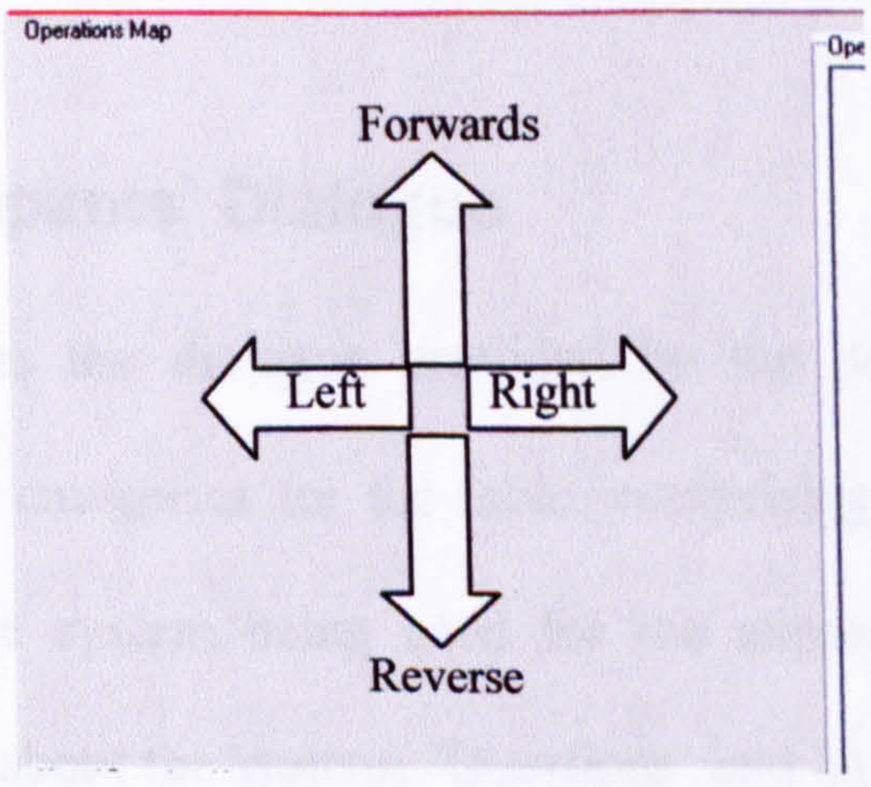



Figure 8.7 The mechatronic vehicle orientation for design of behaviours

The participants attempted to design a specific obstacle avoidance behaviour for the obstacle presented. During the design stage of the interface, this method of behaviour design was considered and rejected. The principle was that the designer could design a generic behaviour, using a simple format for the design map. The use of a specific circumstance could lead to a behaviour specification which would not be effective or appropriate for all circumstances. Further, when the prototyped Artificial Intelligence is tested in the Remote Access Laboratory, the obstacle(s) presented may not be known in advance. The principle for a generic behaviour is that the behaviour can respond to all circumstances.

Task 4 involved designing a behaviour or a set of behaviours for the vehicle to park in the corner, as described in Chapter 5. Participant 5 designed a set of obstacles to guide the mechatronic vehicle into a corner, and used the obstacle avoidance behaviours designed to direct the simulated vehicle to a 'corner'. Although, this has been marked as a failure, as it did not complete the task by the expected means, this was a 'success', as the vehicle was 'in a corner', and the participant introduced an innovative method to obtain the result.

### 8.3.3 The Participants' Dialogue

Table 8.4 summarises the dialogue provided by the participants. The dialogue is reduced to 5 general categories for the table, comprising 'Likes', which are positive expressions about the system being used for the experiment task, 'Dislikes' being negative expressions about the system; 'Questions' and 'Answers', 'Questions' identify when the participant requires assistance. 'Answers' are when there has been a necessary response in some way to the participant, either by activity or by a literal question, or a response to 'Comments'.

'Comments' are remarks by the participant which may neither be described as 'Positive', 'Negative', or as a 'Question'. During the introduction to the experiment the participants were asked to supply a running commentary effectively creating a documentary of the experiment. This led to general text of up to 1600 lines of transcribable dialogue per experiment.

Table 8.4 Participant dialogue during the experiments

| Participant | Rule Based System | | | | | Programming by Demonstration | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Like | Dislike | Question | Response | Comment | Like | Dislike | Question | Response | Comment |
| 1 | 360 | 58 | 122 | 255 | 431 | 265 | 54 | 65 | 261 | 167 |
| 2 | 103 | 18 | 84 | 116 | 94 | 324 | 61 | 164 | 287 | 130 |
| 3 | 231 | 12 | 45 | 272 | 145 | 164 | 19 | 33 | 183 | 111 |
| 4 | 45 | 4 | 19 | 51 | 117 | 159 | 19 | 31 | 159 | 124 |
| 5 | 39 | 30 | 52 | 128 | 73 | 5 | 3 | 12 | 35 | 41 |
| 6 | 5 | 19 | 23 | 67 | 65 | 10 | 12 | 17 | 87 | 80 |
| 7 | 324 | 31 | 64 | 94 | 68 | 87 | 12 | 30 | 128 | 57 |
| 8 | 74 | 35 | 42 | 123 | 98 | 198 | 23 | 56 | 140 | 193 |
| 9 | 46 | 80 | 14 | 100 | 94 | 0 | 0 | 0 | 0 | 0 |
| 10 | 68 | 21 | 25 | 76 | 80 | 70 | 39 | 41 | 119 | 111 |
| 11 | 42 | 5 | 30 | 74 | 61 | 12 | 3 | 6 | 21 | 67 |
| 12 | 448 | 39 | 140 | 313 | 320 | 190 | 13 | 77 | 159 | 136 |
| 13 | 121 | 21 | 13 | 60 | 72 | 65 | 6 | 14 | 98 | 40 |
| 14 | 96 | 19 | 38 | 80 | 87 | 41 | 7 | 19 | 33 | 12 |
| 15 | 281 | 32 | 81 | 210 | 270 | 142 | 11 | 181 | 241 | 247 |
| 16 | 71 | 10 | 25 | 105 | 58 | 45 | 13 | 24 | 87 | 62 |
| 17 | 242 | 38 | 66 | 53 | 122 | 183 | 27 | 26 | 23 | 63 |
| 18 | 311 | 23 | 15 | 50 | 71 | 226 | 10 | 33 | 70 | 156 |
| 19 | 160 | 11 | 113 | 79 | 54 | 48 | 14 | 43 | 29 | 32 |
| 20 | 252 | 130 | 184 | 403 | 478 | 140 | 49 | 90 | 275 | 239 |

Participants17 and 19 asked rhetorical questions. The comments for all the experiments have a particular element of the participant reading aloud the experiment's instructions. The experiments did not solely comprise completing task 1, start task 2... but had an element of dialogue between tasks including question and answer routines. For example participants 17 and 18 over-analysed the RBS system, expressing how they would park the vehicle, using a metaphor of how they would drive their car.

A determinant of whether a person liked or disliked a system was undertaken by examination of the ratio of expressed likes to dislikes, which has the problem that the two systems are very different. The number of questions asked was an indication of how much assistance a person needed while attempting the tasks. On its own, this would be misleading, as the assumption would be that the participants were only asking one form of question. However, many of the participants were interested in how the systems worked. As a result, table 8.9 the ration of likes/dislikes, includes the success in the final task for both systems.

## 8.3.4 The Usability Questionnaire Results

### 8.3.4.1 The PbD System Questions

After completing the tasks, each participant was requested to fill-in a Usability Questionnaire, which comprised the following questions on the PbD system, referred to as 'the Simulator' as task results were provided by use of a simulation.

1. Do you think that the Simulator system provides a sufficient degree of detail about what is happening on the screen?
2. Do you understand the Simulator's systems information on the screen?
3. How intuitive did you find the Simulator system?
4. Did you find it hard to remember how to do anything with the Simulator System?
5. How did you find the Simulator interface prompted you to particular actions?
6. Do you believe that any part of the Simulator interface was unnecessary?
7. Are you satisfied with the Interface's names?
8. What would you like as further dialogue from the Simulator System's Interface?
9. What are your thoughts about the methods of determining the Vehicle Behaviours using the Robot's Behaviour View?
10. Do you believe you needed a Help System with the Simulator system's Interface?

11. Did you find the Simulator system's Interface easy to use?

Table 8.5, below, illustrates what the participants considered were positive and negative aspects of the PbD based Interface.

Table 8.5 Usability questionnaire responses for the PbD system

| Tester | Qn 1 Detail | Qn 2 Info | Qn 3 Intuitive | Qn 4 Recall | Qn 5 Prompt | Qn 6 Surplus | Qn 7 Names | Qn 8 Dialogue | Qn 9 Methods | Qn 10 Help | Qn 11 Easy |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Yes | Yes, Had to ask | | Not Huge | Easy | Measurements Scale 5:1 | Yes | | Not intuitive, Hierarchical | Yes | Yes |
| 2 | Yes | Mostly, esp. visual behaviour | Very | Not obvious | Very little | Jargon at bottom right hand corner | OK, some could be Improved | Line drawing is Sufficient | Useful and Expressive | Prompt based | Reasonably easy |
| 3 | Fairly easy | User-friendly | Very | Parking | Yes | No | Yes, satisfied | Guidance | Efficient | Beneficial | Yes |
| 4 | Yes | Yes | Became familiar | Too many buttons | No | When Familiar | Yes, satisfied | Contextual Help | Not intuitive | Yes | No, easier than RBS |
| 5 | Depends | Well named but difficult | Quite | No | Fairly clear | Not aware of any | Yes, satisfied | Keep Goals Visible | Had to look elsewhere | Possibly | Yes |
| 6 | Not activity related | Yes | OK | Not obvious | Fairly well | Hard to say | OK | Don't know | | Yes | OK |
| 7 | Too abstract | Yes | Not instantly | | No | All at once | | | Orientation | Yes | Moderate |
| 8 | Needed clarifying | Needs learning | Partially | Yes | No prompts | No | No | | | Yes | |
| 9 | | | | | | | | | | | |
| 10 | Very busy screen | A little, | Not really | Yes | Not well | All at once | No | Interface animation | Confusing | Tutorial | No |
| 11 | Yes | Some of it unclear | Quite | No | No | Yes | Yes, satisfied | Goal set/view | Can be understood | No | No |
| 12 | Yes | Not fully yet | Fairly | At first | OK | No | Yes, satisfied | Can improve | Interesting | Yes | Yes, with practice |
| 13 | Yes | Yes | Fairly | Orientated | No | Yes | No Situational | Direction headings | Quick | No | Yes, max info |
| 14 | Visuals Useful | OK, but some duplicity | OK, but can improve | Too much to recall | When familiar | Don't know | Yes, better with more familiarity | Need block diagram | Not intuitive but useful | Need to see block diagram | Yes,better with familiarity |
| 15 | | | | | | | | | | | |
| 16 | No | No | Not intuitive | Yes | | Measurements | No | | | Yes | No |
| 17 | Need more feedback | adequate | Some aspects | Needed to refer | fairly | No – but a lot to take in | No | No more! | Had to be nudged | Yes | Improved with time |
| 18 | | | | | | | | | | | |
| 19 | Probably enough | Some of it | Needed sheet | No | Very little | No – it was all useful | Yes, satisfied | | Orientation | Usually useful | Fairly |
| 20 | No, difficult | No - Confused | When familiar | Second run easier | Didn't really | Measurements | Not really | Visual Cues | Orientation confusing | Depends | When familiar |

Key: Positive statements; Negative statements; Familiarity issues; Not answered

No participant provided only positive answers, although participants 10 and 16 provided only negative answers. There were 19 familiarity issue answers provided by participants, 4, 5, 6, 7, 8, 11, 12 14 and 20. Participants 2, 3, 4, 5, 6, 8, 11, 12, 14, and 17 stated that the PbD Interface was intuitive; only participants 10 and 16 stated they found the PbD Interface was not intuitive.

7 participants 1, 2, 3, 4, 13, 14, 19 found the simulator sufficiently detailed in its instructions and could understand the screen information, the main criticisms being that the screen was too abstract or too much information was provided, with participants 5, 8, 11 and 12 stating familiarity issues.

Only participants 1 and 5 found that the Interface prompted their next action or could easily remember how to use it. The main criticisms were that there was too much to recall or that functions were not obvious. Participant 13 expressed that a significant problem was that the 'Operations Map' mapped the laboratory when creating Goal-Based Behaviours, and used a fixed orientation premised on a mechatronic device, as shown in Figure 8.7 above, for creating Sensor-Activated and Goal-Activated Behaviours. The overall impression was that the interface could be initially intimidating.

Participants 7 and 10 both criticised the interface for presenting everything all at once. Participant 7 wanted parts of the interface to appear when necessary, while participant 10 wanted buttons hidden when not used.

When attempting to find out if there should be more dialogue presented on the interface, suggestions were mostly related to the 'Use of Laboratory Map' for locating goals, and for the design of behaviours independent of the laboratory setting. This was summarised by participant 13 who suggested a directional headings for the vehicle actions. Of the two positive comments, participant 17's reply was a veiled criticism about the amount of data already presented on the screen.

When considering the creation of behaviours on the Use of Operations Map, participants 1, 4 and 14 found the method not intuitive. The positive comments were that this method was '*a useful and expressive form of output*' by participant 2, '*efficient way of monitoring the vehicle's behaviour*' by participant 3, and '*seemed quick to assemble behaviours*' by participant 13.

Question 10 enquired about the provision of a help function. 2 participants considered that there was no need for help, participant 13 arguing that the system is '*assisted by limited button operations*' that users can interact with. There were suggestions of what type of help system would be most useful – '*prompt system*' from participant 2, '*contextual help*' from participant 4, '*a tutorial*' from participant 10, '*a block diagram*' from participant 14.

Questioning the 'ease of use' allowed participants to express a final verdict. Only 3 participants, 10, 11, and 16, found that the Interface was not easy to use, and participant 10 had a familiarity issue with the ease of use.

Eight participants thought there were parts of the interface that were unnecessary, six who thought that all parts of the interface were necessary, three who were unable to decide. Participant 8 gave a caveat of '*maybe it is used at a later stage*'.

Question 7 was the issue of the buttons names. 10 Participants expressed that they were happy with the names, with 6 participants expressing their criticisms of the existing names; of these 6 only 3 suggested alternatives. Participants, 8, 13 and 20, with both 13 and 20 suggested that the names should be context or situational sensitive. Table 8.6 lists the suggested alternative names.

Table 8.6 The suggested names for buttons on the PbD interface

| Original Names | Participant | | | | | | |
|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 8 | 13 | 14 | 20 |
| Use of Operations Map | | View of Laboratory | | Vehicle Movement Area | Operations Map (changes name) | | View of Operations Map |
| Set Goal Locations | | Position Goals | Set Vehicle Goals | | | | Goals |
| Set laboratory Map | | Fix Goal Locations | Set use of Vehicle | | | 'Routing' | Actions |
| Set Obstacles | | Place Obstacle | | | | | Obstacles |
| Clear Obstacles | | Remove Obstacle | | | | | Clear |
| Vehicle Actions | | Vehicle Route | | | | Vehicle Behaviour | Goal Achieved |
| Clear Operations Map | Reset | Delete Operations | | | | | Clear |
| Specify Vehicle Behaviour | | Specify Vehicle Route | | | | | Behaviour |
| Resulting Vehicle Behaviour | | | | | | | Show Action(s) |
| Robot's Behaviour View | Program Running | Robot's Behaviour | | | | | Behaviour Map Tree? Set? Hierarchy |

Context Sensitive

For the suggested alternative to '*Use of Operations Map*', the suggestions of '*View of...*' is not favoured, as the '*Use of Operations Map*' is used for both viewing the simulation and for creating the mechatronic vehicle's behaviours. The suggestion of changing the name for the context of use, is considered impractical, as in either case the alternative name is hidden. Participant 1 suggested renaming '*Clear Operations Map*' as '*Reset*'. This is accepted as a useful suggestion, along with '*Delete Operations*' from participant 3, and '*Clear*'.

Participant 20's suggestions are mostly questionable, as the use of single words becomes too abstract, not descriptive enough, which could prove to be problematic when there is already a potential problem of the Interface not providing sufficient prompts. Likewise there are reservations about most of participant 2's suggestions, with the suggestion '*Robot Behaviour*' being considered. The suggestion of '*Robot Behaviour View*' (the view of the program(s) being develop or tested) as '*Program Running*' by participant 1 is questionable, as there could be alternative prototyped Artificial Intelligences.

### 8.3.4.2 The RBS System Questions

The questionnaire also had the following questions on the Rule Based System (alternative system).

1. Do you think that the Rule Based system provides a sufficient degree of detail about the what is happening to the screen?
2. Do you understand the Rule Based systems information on the screen?
3. How intuitive did you find the Rule Based system?
4. Did you find it hard to remember how to do anything with the Rule Based system?
5. How did you find the Rule Based system interface prompted you to particular actions?
6. Do you believe that any part of the Rule Based Interface was unnecessary?
7. What would you like as further Dialogue from the Rule Based system's Interface?
8. What are your thoughts about the methods of determining the Vehicle Behaviours using the Rule Based system?
9. Did you find the Interface easy to use?

There were only 9 questions, as the study was not trying to validate the naming of the various elements of the Rule Base System, or enquire if the system would require a help function. Table 8.7 below, illustrates the participants' range of replies to the questions.

Table 8.7 Usability questionnaire responses for the RBS system

| Tester | Qn 1 Detail | Qn 2 Info | Qn 3 Intuitive | Qn 4 Recall | Qn 5 Prompt | Qn 6 Surplus | Qn 7 Dialogue | Qn 8 Methods | Qn 9 Easy |
|---|---|---|---|---|---|---|---|---|---|
| 1 | | | | | | | | | |
| 2 | No - Minimal Feedback, observe buggy | Yes | Not really Odd concept | Easy, familiarised in ~30mins | By leaving blanks | No | Vehicle activity and fault action | Restricted in actions | Reasonably, not intuitive |
| 3 | Yes | Yes, Enjoyable | Very intuitive | Easy | Easy to follow | No | Demonstration of commands | Obvious, easier control | Yes |
| 4 | Probably if Understood | No | Not intuitive | Yes | Followed Instructions | Can't tell | | | No |
| 5 | Not for me | Not particularly | Not particularly | Yes | Got stuck regularly | Difficult for me | Mind set required | Mind set required | No |
| 6 | Once grasped it is OK | Yes, but have to think about it | Logical & simple, when familiar | | Needed experimenter prompts | No | Feedback about Vehicle activity | Don't know | OK, once grasped |
| 7 | Yes | Yes | Very | No | Reasonably | The Display | | | Yes |
| 8 | | | reasonably | | | | | | |
| 9 | | | | | | | | | |
| 10 | Yes | Yes | Fairly | Not v. hard … a bit | | No | | Fairly intuitive | Easier, less buttons |
| 11 | Yes | Think so | Quite | No | | No | Clarify where rule starts | Quite useful if visual | No |
| 12 | Yes | Largely yes | Quite intuitive | Need reminder | Quite well | No | | Limited capabilities | Yes |
| 13 | Requires programming experience | Yes | Once accustomed not too bad | Looked at previous rules | Only help from user guide | Display box | Requires De-bugger | Math skills to zone Sensor values | No help, no debug, little unclear |
| 14 | Secondary Visual | Understood principles, but not fluent | When familiar | Remember structures when familiar | Mismatch facts-tasks | No way of judging | Visual of resulting program | Needs additional info – "data dictionary" | Needs Structure Diagram |
| 15 | | | | | | | | | |
| 16 | No | Yes | Not intuitive | No | Didn't use Variable Database | | | | Yes, Sometimes |
| 17 | Yes | Yes | Fairly intuitive | Easy, need reminder | Informed progress | Display | Pop-up dialogs | Appears complete | Yes, easier |
| 18 | | | | | | | | | |
| 19 | Yes | Difficult to follow | Not at all | Copied instructions | Dialogue box provided | Variables Display | Mind set required | Mind set required | No |
| 20 | Yes - almost | No | Initial confusion | No | Drop down hints handy | No | Highlights prompts on acitivity | Powerful, when understood | OK |

Key: Positive statements; Negative statements; Familiarity issues; Not answered

The PbD based interface received 63 positive remarks, 55 negative remarks, and 6 stated familiarity issues. The familiarity issues are more significant as the participants had a greater time exposure to the RBS system than the PbD system. Participants 1, 9, 15 and 18 supplied no answers.

Participant 5 supplied no positive statements about the Rule Based System, while participant 10 supplied no negative statements about it. The physical issue with the Rule Based System was that it did not supply sufficient dialogue. The principle dialogue criticism was that it does not provide visual feedback, with participants 2, 3, 6, 11 and 14 requesting some form of visualisation of how the program worked. The RBS system problem was, as expressed by participants 5 and 19, that it required a '*mind set*' and participant 13 stated that it requires '*programming experience*'.

Participants 2, 4, 5, 13 and 16 expressed that the RBS system was not intuitive, with participants 4, 5, 6, and 13 expressing that they found the RBS system did not provide any prompts on how to progress, with participants 19 and 20 expressing that they did not find the system provided sufficient information, and participants 4, 5, 12, 13, 14 and 19 stating that the Rule Base system did not provide any help in remembering how to do any particular operation.

When questioned about the system methods, participants 2, 12 and 13 all expressed that the it had limited capabilities.

## 8.4 Comparative Usability of the Two Interfaces

This includes an examination of how the participants responded to the two different systems tasks. The principle measure of analysis was the extent to which the participants repeated the tasks or task results.

### 8.4.1 Repetitions of the Experiment

A number of the participants, after achieving a task decided to repeat it. These are shown in Table 8.8 below. The importance of task repetition is that the participant showed an ease with the system being tested, and this can interpreted as the participant enjoyed the experiment.

Table 8.8 Repetitions of the experiment

| Participant | Rule Base System | | | | | | Programming by Demonstration | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Task 1 | Task 2 | Task 3 | Task 4 | Task 5 | Task 6 | Task 1 | Task 2 | Task 3 | Task 4 |
| 1 | Yes | Yes | | Yes | | | Yes | Yes | Yes | Yes |
| 2 | | | | | Quit | Quit | Yes | Yes | Yes | |
| 3 | | | Yes | Yes | Yes | | Yes | Yes | Yes | |
| 4 | Yes | Yes | | | | | | Yes | Yes | |
| 5 | | Yes | | | | | Yes | | | |
| 6 | | | | | | | Yes | Yes | Yes | |
| 7 | | | | | | | | | | |
| 8 | | Quit | Quit | Quit | Quit | Quit | | | | |
| 9 | | | Quit | Quit | Quit | Quit | Quit | Quit | Quit | Quit |
| 10 | | | | Quit | Quit | Quit | Yes | Yes | Yes | |
| 11 | | | | | | Quit | | | | |
| 12 | | | | | | | Yes | Yes | | |
| 13 | | | | | | | Yes | Yes | Yes | Yes |
| 14 | | | | | Quit | Quit | Yes | | | |
| 15 | | | Yes | | | | Yes | Yes | Yes | |
| 16 | | | | | | Quit | Yes | Yes | | |
| 17 | | | | | | | | | Yes | |
| 18 | | | | | | | | | Yes | |
| 19 | | Yes | | Yes | | Quit | | | | |
| 20 | | | | | Yes | Yes | | Yes | Yes | |

Key: Repeated run (for vindication)
Repeated programming (from enthusiasm)

Participant 13 successfully repeated the entire PbD tasks after accidently deleting the experimental data, in 3½ minutes, compared with the average time for completing the PbD tasks of ~23 minutes. This repetition included running the simulator for each task.

As can be observed from Table 8.8, with the PbD based interface more people repeated the task programming from enthusiasm, because it was found to be easy, while with the RBS system 8 participants quit. The only person to quit the PbD System was participant 9 who had already walked out.

## 8.4.2 The Like/Dislike Ratio from the Experiment Dialogue

Table 8.9 below comprises the ratio of likes to dislikes obtained from 8.3.3 above. This is combined with the participants' success in the final task, '*Parking in a corner*'.

Table 8.9 The ratio of likes/dislikes

| System | Participants | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| RBS | | | | | | | | | | | | | | | | | | | | |
| Likes | 360 | 103 | 231 | 45 | 39 | 5 | 324 | 74 | 46 | 68 | 42 | 448 | 121 | 96 | 281 | 71 | 242 | 311 | 160 | 252 |
| Dislikes | 58 | 18 | 12 | 4 | 30 | 19 | 31 | 35 | 80 | 21 | 5 | 39 | 21 | 19 | 32 | 10 | 38 | 23 | 11 | 130 |
| Ratio | 6.2 | 5.7 | 19.3 | 11.3 | 1.3 | 0.26 | 10.5 | 2.1 | 0.58 | 3.2 | 8.4 | 11.5 | 5.8 | 5.1 | 8.78 | 7.1 | 6.4 | 13.5 | 14.5 | 1.9 |
| Success | 1 | Quit | 1 | Quit | 1 | 2 | 3 | Quit | Quit | Quit | Quit | 4 | 14 | Quit | 4 | Quit | 6 | 3 | Quit | 8 |
| PbD | | | | | | | | | | | | | | | | | | | | |
| Likes | 265 | 324 | 164 | 159 | 5 | 10 | 87 | 198 | 0 | 70 | 12 | 190 | 65 | 41 | 142 | 45 | 183 | 226 | 48 | 140 |
| Dislikes | 54 | 61 | 19 | 19 | 3 | 12 | 12 | 23 | 0 | 39 | 3 | 13 | 6 | 7 | 11 | 13 | 27 | 10 | 14 | 49 |
| Ratio | 4.91 | 5.31 | 8.6 | 8.3 | 1.67 | 0.87 | 7.25 | 8.61 | | 1.79 | 4 | 14.6 | 10.8 | 5.9 | 12.9 | 3.46 | 6.78 | 22.6 | 3.4 | 2.86 |
| Success | 2 | 2 | 1 | 8 | 4 | 2 | 3 | 2 | Quit | 1 | 1 | 2 | 2 | 1 | 2 | 5 | 3 | 4 | 1 | 1 |

This table appears counter-intuitive. Participants 2, 4, 10, 11, 16 and 19 all had a greater like- to-dislike ratio in favour of the RBS system, but then quit the last task. For the participants who completed all the PbD tasks, participants 1, 2, 7 and 16 all had a greater like-to-dislike ratio in favour of the RBS system.

## 8.5 General Conclusions about the Experimentation

In section 8.3 above, participant 14 achieved an unexpected result of completing the tasks for the PbD system in less than overall average times, which would initially appear to be anomalous. However, participant 19 claimed in the usability questionnaire that this experimentation required the ability to think logically. Participant 14 provided the feedback of mapping 'thoughts as diagrams', which is the fundamental principle underlying PbD.

Table 8.10 The evidential and stated preference of the participants

| Participant | Evidential preference based on experimental data | | Stated Preference |
|---|---|---|---|
| | Timings | Success | |
| 1 | PbD | PbD | PbD |
| 2 | PbD | PbD | PbD |
| 3 | Rule Base | Rule Base | Rule Base |
| 4 | PbD | PbD | PbD |
| 5 | PbD | Rule Base | PbD |
| 6 | Rule Base | Rule Base | Rule Base |
| 7 | PbD | PbD | Rule Base |
| 8 | PbD | PbD | PbD |
| 9 | Unknown | Unknown | Unknown |
| 10 | PbD | PbD | Rule Base |
| 11 | PbD | PbD | PbD |
| 12 | PbD | PbD | PbD |
| 13 | PbD | Rule Base | PbD |
| 14 | PbD | PbD | PbD |
| 15 | PbD | PbD | PbD |
| 16 | Rule Base | PbD | Rule Base |
| 17 | PbD | Rule Base | Rule Base |
| 18 | PbD | Rule Base | Unknown |
| 19 | PbD | PbD | PbD |
| 20 | Rule Base | Rule Base | Rule Base |

Nearly twice as many preferred the PbD system to the RBS system. There was no distinguishing variable such as age, gender, education which influenced the determination of preference. Evidence from timing and success rates was mostly, but not always coincidental with the stated preference. There was complete agreement with participants 1, 2, 3, 4, 6, 8, 11, 12, 14, 15, 16, 19 and 20, partial agreement with 5, 13, 16, and 17 and complete disagreement with participants 7 and 10.

*The Rule Base System*: The first two tasks were found to be easy, but they became progressively more difficult. There was a high drop-out rate for tasks 4 to 6. Only one person showed enthusiasm for using the system. Participants took longer to achieve the tasks. For detailed statistics see appendix D.

*The PbD System*: The first two tasks were found to be easy. Tasks 3 and 4 were found to be more problematic, because with task 3, there was a problem with map orientation and map scaling and with task 4, there was a problem of abstraction in understanding how to design the complex behaviours. There was a zero dropout rate for all who attempted the tasks, and approximately half the sample found the system a pleasure to use. Overall timings were a lot less than for the RBS. For detailed statistics see appendix D.

In conclusion the PbD system shows great promise, but could be improved by some form of on-screen help to resolve the PbD design orientation problem, highlighted in the questionnaire responses. The interface could be improved by some renaming of the buttons, or using icon-based buttons.

# Chapter 9

# Conclusions

This chapter draws conclusions on the entirety of the research, with proposals for further research, consideration of reflective practice, and puts forward implications of the research.

## 9.1 Conclusions to the Research Questions

1 *What are the criteria for designing a remote access laboratory for prototyping Artificial Intelligence, as part of a distance learning organisation's available tools?*

- A critical criterion is how to present experiments which accurately represent the physical world, applying the associated theory being learned. The emphasis is on the laboratory's design as a tool for appropriate student learning, only achieved by a physical, remote access laboratory, allowing 'any-time, any-place' access.
- The laboratory is to be flexible, facilitating students carrying out individual work or group activities, and adaptable to course contents developments, while invoking enthusiasm to facilitate a successful course and continued learning.
- Experimental problem solving is expected to provide the same solutions to problems as occurs in good engineering practice.

These criteria were established in chapters 2 and 4, which drew upon the literature and the practical experience of using the facilities of an existing Mechatronics course.

2 *What fusion of technologies should be used to develop a Mechatronics prototyping laboratory?*

This was established in chapter 4, which drew upon the recommendations from the research in collaboratories to propose a multi-agent system architecture, which is a network of autonomous agents. The fusion envisaged comprises:

- A multi-user interactive environment architecture with the expectation of adaptability to rapid developments, best served by a distributed software architecture, and developed as a multi-agent system, comprising :

(i) a Blackboard agent architecture developed as a Mechatronic Device Operating agent to test a received prototype Artificial Intelligence safely,

(ii) a Knowledge-base agent architecture to store the system's knowledge.

- a Mechatronic Device with both sensors and actuators to be accessed during experiments,
- a Programming by Demonstration Interface as a programming tool. The experimentation demonstrated its potential to engender enthusiasm, which is important for the laboratory's success.

3 *What methodology and technologies could assist in rapid prototyping Artificial Intelligence in a distance learning mechatronics course?*

- Programming by Demonstration method. This involves a directness of designing robot behaviours, with an immediate visualisation of the behaviour being designed. The experimental results described in Chapter 8.3.1 show this is easy to learn, and from 8.3.4, that it is demonstrably intuitive, with 8.4.1 showing that it engenders enthusiasm.

- An Object Orientated approach to create a hierarchy of robots behaviours. The experimentation established that this allows the assembly of a prototype Artificial Intelligence using minimal of effort, with chapter 8.3.4 demonstrating its efficiency and usefulness, though in its current implementation it needs modification to aid orientation while designing the demonstrated robot path.

4 *What design of interface to such a laboratory would allow appropriate analysis and demonstration of prototype Artificial Intelligence?*

- The current design of the interface, which focuses on designing, analysing and demonstrating a prototype Artificial Intelligence, was demonstrated to be effective, in section 8.3.2. Section 8.4.1 shows that some participants had problems with the scaling and orientation of the robot's path.
- The use of Distributed Cognition methods for the interface design can help prompt the next part of the programming process as shown in 8.4.1, by limiting the interactivity of the interface to appropriate predetermined choices, although there is an issue of presenting too much information to the user all at once.
- The use of a pop-up based help system to guide the design and development of the prototype Artificial Intelligence would help overcome the issues of interface navigation.
- The use of simulation to help rapidly test the prototype Artificial Intelligence before use in a laboratory was found to be effective. It allowed an immediacy of recognition of robotic behaviours designed and their activity, as shown in section 5.3.4 to section 5.4.3, and in section 8.3.4.
- The use of a video-linked image to the interface from the laboratory to show the mechatronic device while under test, described in section 5.4.4.

## 9.2 Conclusions to the Aims and Hypothesis

The aims of the research were:

1. *To establish the viability of remote access facilities to augment distance learning.*

This was established in chapter 4, with the design proposal for a laboratory. Particular attributes taken into consideration were that it must be:

- easy to assimilate new mechatronic technologies for the courses offered, to keep it relevant,
- easy to scale the laboratory software with a distributed multi-agent system, within the constraints of the laboratory's physical size, number of computers, and the available bandwidth for internet use.

2. *To design and evaluate technology which can provide an environment for students to learn to rapidly develop prototype Artificial Intelligence for a mechatronic device.*

This was achieved through experimentation with a novel prototyping interface, which included the use of Programming of Demonstration.

- Chapter 8 described how seven participants managed to complete the Programming by Demonstration final task of 'parking in a corner', with minimal tutoring, a sample average of 23 minutes, compared to the Rule Base System, where only 4 people finished the final task with a sample average time for using the interface of 50 minutes.

3. *To test the hypothesis*:

**Programming by Demonstration could prove a more intuitive approach to the complexity of developing an emergent intelligent behaviour than text-based programming.**

The hypothesis is supported by strong experimental evidence described in section 8.5, which establishes that 11 participants of a sample of 20 stated a preference for using the Programming by Demonstration based interface compared with an established text-based programming system. While using the text-based programming system, one participant left the laboratory and 7 other participants quit the experimentation. No participants quit while using the Programming by Demonstration system, and 7 showed enthusiasm for the system.

The particular feature of the PbD was the ease of developing and assimilating behaviours. It was found easy to repeat and modify the behaviours designed. The text-based system was found to be slow to implement, and difficult to debug and modify; furthermore it was found difficult to relate the programming to the mechatronic device's actions.

## 9.3 Conclusions for Remote Access to a Prototyping Laboratory

This research sought to increase the knowledge of theories relating to Remote Access Laboratories. The first issue to be determined was that the laboratory can only operate in an education environment, as explained by Coventry's [1995] (Re)conceptualisation Cycle, which approves the teaching of theory and emphasises the necessity for presenting authentic problems during the experimental process.

Current research in distance learning laboratories was considered, with the expectations and objectives for an engineering laboratory, the crucial point being that experimentation had to be an accurate representation of the physical world for applying the associated theory. This led to rejecting a simulated laboratory. The proposal of a physical Remote Access Laboratory maintains the 'any-where, any-time' principle of distance learning.

With the requirements for an engineering instructional laboratory to support teamwork, the principles of a collaboratory were considered essential for the 'access architecture' to the laboratory. An established need for data sharing, with simplicity of use when adopting new technologies, necessitated a form of easily updated distributed software architecture, to support a variable number of users working in groups, or individually in the laboratory. The proposed use of a Multi-Agent System was intended to achieve optimal design of a Remote Access Laboratory.

The design of the laboratory interface could not be satisfied by the principles of Intelligent Training Systems which are the current design principles applied to hypermedia-based learning systems. The laboratory interface design had to draw on the theory of Human-Computer Interfacing, which is fragmented, and problematic due to the disconnection between theory and practice. Also the application of theory requires experimentation, which may provide spurious results described by the 'Hawthorne Effect'.

The question of how to rapidly prototype Artificial Intelligence was addressed. The problem was that using text-based languages resulted in students having to learn a computer language before programming a prototype Artificial Intelligence, which can be a time-consuming process.

The solution was the use of Programming by Demonstration, which allowed the designer to directly prototype Artificial Intelligence, without the demands of a structured language forcing its constraints and requirements on the prototyping process. Further, the problem of translating the intended prototype Artificial Intelligence program into a programming language is compounded by the '*Gulfs of Execution and Evaluation*'.

## 9.4 Further Research

There is a need for further research to bring the remote access laboratory closer to implementation.

**1 Further development of the interface**

to increase the intuitive nature of the Programming by Demonstration interface, it will be essential to examine further the design, with additional work in:

- developing appropriate and intuitive buttons for rapid understanding and use of a Programming by Demonstration system for prototype Artificial Intelligence development,
- developing a more intuitive method of assimilating the Robotic Behaviours into an integrated prototype Artificial Intelligence.

## 2 Development of a prototyping laboratory

Experimentation is proposed to:

- establish the viability of a multi-agent system which can support a multi-user interactive environment architecture which operates a physical laboratory,
- connect the laboratory interface design to the laboratory allowing remote experimentation of the prototype Artificial Intelligence.

## 3 Development of the communications architecture

Before further testing, there is a need to develop the communications infrastructure. This will allow deeper exploration of the facilities needed for mechatronics learning, including:

- a method of transmitting the prototype Artificial Intelligence to the laboratory for testing,
- a connection of the Interface to a mechatronic device with actuators and sensors, allowing the responses to be relayed to the interface in real-time,
- a means for the users to collaborate, sharing real-time data and results. This will involve an architecture to access the laboratory, for the communication of data, and combining multiple streams of real-time user data with the laboratory's data. Finally, the knowledge-base agents will need to be developed to support the combination of multiple real-time data streams with previous knowledge.

## 4 Further analysis of the applicability to Mechatronics

Mechatronics is the fusion of three engineering subjects: mechanics, electronics and intelligent control systems. For greater applicability to Mechatronics, the laboratory could be reasonably expected to allow experimental examination of:

- other physical behaviours of intelligent control systems. Robots are not just expected to move forwards and backwards. Intelligent control systems are being used in aircraft, helicopters and submersibles to enable autonomous, unmanned control.
- other AI software methods such as Neural Networks, Fuzzy Logic systems and computer vision.
- co-operation between robots, and the design of co-operative robotics. This is a current sphere of growing research interest, from robo-football to swarm-based robotics, relevant to remote, harsh environments such as required by a Mars explorer, for which co-operative robotics could provide resilience.
- expanded testing of a complete course, or sections of a course. The current state of developments gives encouragement that this is now worth considering as a next piece of research

## 9.5 Future Open-Learning Access to Online Laboratories

Open access to learning material is currently limited to a few applications and organisations. However, with the development of web-based technologies which contribute to the globalisation of information transfer, the issue of whether and how online laboratories would be widely accessible becomes of increasing significance. Research into how such facilities as web 2.0/semantic web, open source and wiki models could influence future developments in engineering design learning and its commercial development.

The semantic web is broadly but vaguely understood as an extension to the worldwide web, in which the contents are no longer just freestanding items comprising text, images and software that require humans to understand and utilise them. The development would involve a search engine to access information available on the internet, using a set of inference rules for automated reasoning. The current solution provided is the use of XML, eXtensible Markup Language, a set of tags for a document providing details of its contents, without providing details of its structure. Additionally the contents would be in a form that could be utilised by autonomous intelligent agents, allowing integration and sharing of information in a substantially automated form of engineering design. This suggests that the Programming by Demonstration approach of this research could lead to PbD at a higher level, in which demonstrable functions would be web accessed and be of wider application than vehicle navigation.

Similarly, open source could allow a wider community of researchers, students and professional engineers to use an online laboratory. At a basic level, software components could be freely obtained as 'objects'. For example: the laboratory Bookkeeping agent could provide an open access library of programmes for students to use as part of their learning how to design prototype Artificial Intelligence. This would be a library of previously designed and tested prototype Artificial Intelligences which achieve similar objectives to a learner's intended prototype Artificial Intelligence. The learner could examine the provided source to obtain an insight to solutions of the posed problem. Further, this would allow development of the laboratory by establishing potentially more efficient or innovative solutions to problems.

The use of a Wiki model concept with an open access library, could be useful as a means of collaboration amongst groups of learners or designers. There is currently a problem with vandalism such as the creation of deliberately inaccurate data in a Wiki entry. By the deliberate design of behaviours which visibly display the program operation (circumventing the Gulf of Execution), this problem could be reduced. Adopting Wikipedia's new concept of a 'trusted' user, would further promote trustworthiness based on creating a hierarchy of contributors, with their rankings based on verification of the material supplied. Any new prototype Artificial Intelligence placed as an open resource could be moderated and assessed for trustworthiness.

## 9.6 Reflective Practice

My aim was to develop my research skills to the extent of being able to continue from this research to working without the need for supervision.

Before beginning the PhD I completed an MSc in Digital and Opto Electronics, where the research for the final project (in real-time video colour restoration-based computer vision) had been extensive, but mainly comprised surveying background knowledge, and providing a context for the development. I employed this form of background research in the beginning of the PhD, providing an original 80-page state-of-the-art report which did not initially survey the literature in a critical manner.

From the original research proposal, I outlined a plan to research and test a remotely accessed laboratory. The initial research was directed at the laboratory development, which was an over-ambitious development of a laboratory which allowed a user to program nearly every aspect it, with a catalogue of about 7 different agents, to test prototype Artificial Intelligence including prototype Computer Vision AI. I started with the Programming by Demonstration system, but had difficulty formulating the 'end product', as I was unaware of any comparable software system to investigate for design and development guidance. As the system developed, the ideas for the system started to emerge. This was achieved by considering what would be expected from a prototype Artificial Intelligence's behaviour.

My first learning step was to critique the initial background research to a greater depth, compared with the standards attained previously for my Masters degree. My next step was to consider carefully, what I wanted to achieve and how to achieve it. I had to learn to work to a greater degree to a structured outline, to plan the progress, as much as possible. Part of this lesson was as much the importance of structuring software. *Ad-hoc* software development is notoriously 'buggy', and without structure it quickly becomes too complex and hard to manage.

I started to work on a focused piece of software, the PbD system. I had an understanding of what was expected from the system, although the development was laborious. The focused development was useful; this also guided my research, as I was now able to clearly determine what I was attempting to achieve. Although, from the outset, I was able to do my own research, and report on it, I subsequently learned the difference between summarising and critical assessment of research papers.

While working on the research I gained experience and skills that would enable me to continue with further research independently, having learnt a number of lessons from this experience. These are:

- The importance of structuring research. I have learnt that there is a diversity of opinions for solutions to a problem, and there is a need to assemble and critique the variety of opinions.
- To progress the research, I had to identify key issues, and ultimately place them within a wider context.
- To develop the experimental software system, I had to become adept at structuring my approach to problem solving. As I needed to develop a Programming by Demonstration system, without any equivalent software as a guide, I had to learn to develop it in distinct progressive stages, towards a prototyped system.
- As a part of the experiments, I had to work with people, and I learned the importance of ethical behaviour in the conduct of experimentation. During the experimentation, I perceived that I was also a participant, because my comments affected the outcome of the participants' efforts. It became clear that although I intended to be an independent, objective observer, I was inevitably also a stakeholder in the research.

## 9.7 Implications of the Research

The significance of the work reported is that it is a response to the growing need for distance learning, particularly for an adult population that is seeking to progress in an environment of life-long learning. Currently resources and efforts are being ploughed into the development and maintenance of conventional laboratories, as part of an expansion in higher education, led by Government policies. There is a continuing need for modern engineering techniques and laboratories to achieve competitiveness in a globalised market.

Continuation of this research should be viewed not merely as an interesting academic exercise, but as an urgently needed tool for the development of skills for engineering as a whole.

# References


Agarwal D., Sachs S.R., Johnston W.E., **1998**, *The Reality of Collaboratories*, Computer Physics Communications, Vol.110(1), pp.134-141

Alberta Government, **2002**, *Understanding Design and Analysis of Research Experiments*, The Alberta Government, Department of Agriculture and Food, Website
www1.agric.gov.ab.ca/$department/deptdocs.nsf/all/webdoc3033
accessed: 22 August 2007

Angehrn A., and Nabeth T., **1997**, *Leveraging Emerging Technologies in Management Education: Research and Experiences*, European Management Journal, Vol.15(3), pp.275-285

Archer W., Garrison R., Anderson T., **1999**, *Adopting Disruptive Technologies in Traditional Universities: Continuing Education as an Incubator for Innovation*, Canadian Journal of University Continuing Education Vol.25(1), Spring, pp.13-30

Asami K., Takeuchi A., Otsuki S., **1998**, *A Dialogue Method for Assisting Students in Understanding Causalities in Physical Systems*, Systems and Computers in Japan, Vol.29, Issue 6, pp. 1-15

Baker R. S. J., **2007**, Modelling and Understanding Students' Off-Task Behaviour in Intelligent Tutoring Systems, Proceedings of the SIGCHI conference on Human Factors in Computing Systems, pp.1059-1068.

Barab S., and Plucker J., **2002**, *Smart Contexts: Smart People or Smart Contexts? Cognition, Ability, and Talent Developments in an Age of Situated Approaches to Knowing and Learning*, Educational Psychologist Vol.37(3), pp.165-182.

Bardeen M., Gilbert E., Jordan T., Nepywoda P., Quigg E., Wilde M., Yong Z., **2006**, *The QuarkNet/Grid Collaborative Learning e-Lab*, Future Generation Computer Systems, Vol.22 Issue6, pp.700-708.

Barnard P., May, J., Duke, D., Duce, D., **2000**, *Systems Interactions and Macrotheory. Transactions On Computer Human Interaction*, Vol.(7), pp.222-262.

Béguin P., and Rabardel, P., **2001**, *Designing for Instrument Mediated Activity*, Scandinavian Journal of Information Systems, Vol(12), pp.173-190.

Bellotti V., **1988**, *Implications of Current Design Practice for the Use of HCI Techniques*, People and Computers IV pp.13-34, Cambridge: Cambridge Univ. Press.

Bergin S., and Reilly R., **2005**, *The Influence of Motivation and Comfort-Level on Learning to Program*, 17th Workshop of the Psychology of Programming Interest Group, Sussex University, pp.293-304.

Biggs G., and MacDonald B., **2003**, *A Survey of Robot of Programming Systems*, Proceedings of the 2003 Australian Conference on Robotics and Automation CSIRO
www.araa.asn.au/acra/acra2003/papers/27.pdf
accessed: 22 August 2007

Bonk C., and Cunningham D., **1998**, *Searching for Learner-Centred, Constructivist, and Sociocultural Components of Collaborative Educational Learning*, Journal of Electronic Collaborators: Learner-Centred Technologies for Literacy, Apprenticeship, and Discourse, pp.25-50.

Bourguin G., Derycke A., Tarby J., **2001**, *Beyond the interfaces, Co-evolution inside Interactive Systems: A proposal founded on the Activity Theory*, People and Computers, pp.297-310.

Bourne J., Harris D., Mayadas F., **2005**, *Online Engineering Education: Learning Anywhere, Anytime*, Journal of Online Engineering Education, Vol.94(1), pp.131-146.

Brooks R., **1986**, *A Robust Layered Control System for a Mobile Robot*, IEEE Journal of Robotics and Automation 2(1), pp.14-23.

Brooks R., Connell J., Ning P., **1988**, *HERBERT: A Second Generation Mobile Robot*, Technical Report AIM 1016, MIT AI Lab, Cambridge, MA.

Bryson J., **2000**, *Cross-Paradigm Analysis of Autonomous Agent Architecture*, Journal of Experimental and Theoretical Artificial Intelligence, Vol.12(2), pp.165-190

Butz B., Duarte M., Miller S., **2006**, *An Intelligent Tutoring System for Circuit Analysis*, IEEE Transactions on Education, Vol.49(2), pp.216-223

Calinon S., and Billard A., **2007**, *Incremental Learning of gestures by Imitation in a Humanoid Robot*, Proceedings of the ACM/ICCC International Conference on Human Robot Interaction (HRI)

Campbell J., Bourne J., Mosterman P., Brodersen A., **2002**, *The Effectiveness of Learning Simulations for Electronic Laboratories*, Journal of Engineering Education 91 (1) pp.81-87

Card S., Moran, T., Newell, A., **1983**, *The Psychology of Human-Computer Interaction*. Lawrence Erlbaum Assoc. Inc., Hillsdale, NJ.

Carlson S., **2000**, *Campus Survey Finds that Adding Technology to Teaching is a Top Issue*. The Chronicle of Higher Education, 47, A46

Carroll J., **1987**. *Interfacing Thought: Cognitive Aspects of Human-Computer Interaction*. Cambridge, Mass. MIT Press

Carver N., and Lesser V., **1992**, *The Evolution of Blackboard Based Architectures*, CPMSCI Technical Report, Dept of Computer Science, University of Massachusetts, pp.92-71

Castel, F., **2002**, *Theory, Theory On The Wall....*, CACM, Vol.45, 25-26, Cognitive Science, 18, pp.87-122.

Central Texas Science and Engineering Fair, **2007**, *The Scientific Method*, Website ctsef.org/misc/scientific_method.htm accessed: 22 August 2007

Chavez A., Moukas A., Maes P., **1997**, *Challenger: a Multi-Agent System for Distributed Resource Allocation*, Proceedings of the First International Conference on Autonomous Agents '97, Marina Del Ray, California, pp.323-331

Chen J., **2005**, *Constructing Task-Level Assembly Strategies in Robot Programming by Demonstration*, International Journal of Robotics Research. Vol.24(12), pp.1073-1085.

Chen J., and McCarragher B, **1998**, *Robot Programming by Demonstration-Selecting Optimal Event Paths*, Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on, Vol.1, pp.518-523,

Chen J., and McCarragher B., **2000**, *Programming by demonstration - constructing task level plans in hybrid dynamic framework*, Proceedings of the IEEE Intl. Conf. on Robotics and Automation (ICRA '00), Vol.2, pp.1402-1407,

Chen J., and Zelinsky A., **2001**, *Programming by demonstration: removing suboptimal actions in a partially known configuration space*, Proceedings of the IEEE Intl. Conf. on Robotics and Automation (ICRA '01), Vol.4, pp.4096-4103,

Chickering A., and Ehrmann, S., **1996**, *Implementing the Seven Principles: Technology as a Lever*. American Association for Higher Education Bulletin, Vol.49(2), pp.3-6

Christensen C., **1997**, *The Innovator's Dilemma: When New Technologies Cause Great Firms to Fail*, Boston: Harvard Business School Press

Clark R., **1994**, *Media Will Never Influence Learning*, Educational Technology Research & Development, 42, pp.21-29

Cole, M., and Engeström, Y., **1993**. *A Cultural-Historical Approach to Distributed Cognition*. Distributed cognitions: Psychological and educational considerations (pp.1-46). New York: Cambridge University Press

Corkill D., **1991**, *Blackboard Systems*, AI Expert, 6(9) pp.40-47,

Corkill D., **2003**, *Collaborating Software: Blackboard and Multi-Agent Systems and the Future*, Proceedings of the International Lisp Conference, New York,

Corkill D., **2005**, *Representation and Contribution-Integration Challenges in Collaborative Situation Assessment*, Proceedings of the 8th International Conference on Information Fusion (Fusion 2005)

Coventry L., **1995**, *Video Conferencing in Higher Education and Business* www.agocg.ac.uk/reports/mmedia/video3/contents.htm accessed: 22 August 2007

Crowley S., and Medvedeva O., **2006**, An Intelligent Tutoring System for Visual Classification Problem Solving, Artificial Intelligence in Medicine, Vol.36(1), pp.85-117

Davidsson P., **1996**, *A Linear Quasi-Anticipatory Agent Architecture for Adaptive Intelligent Systems: Some Preliminary Experiments*, Distributed Artificial Intelligence Architecture and Modelling (Lecture notes in Artificial Intelligence 1087), C. Zhang and D. Lukose (ed) Springer Verlag, pp.189-203

Davies B., McDonald M., Harrigan R., **1994**, *Virtual Collaborative Environments: Programming and Controlling Robotic Devices Remotely*, Proceedings of the SPIE – The International Society for Optical Engineering 2351 pp.34-43

de Paula J., **2001**, *Experimental Errors and Data Analysis*, Haverford College, Department of Chemistry, Laboratory in Chemical Structure and Reactivity haverford.edu/chem/302/data.pdf accessed: 22 August 2007

Dewan P., Rajiv C., Honghai S., **1994**, *An Editing-Based Characterisation of the Design Space of Collaborative Applications*, Journal of Organizational Computing, Vol.4 (3), pp.1-19

Dewan P., and Shen H., **1998**, *Controlling Access in Multiuser Interfaces*, ACM Transactions on Computer-Human Interaction,Vol.5(1), pp.34-62

di Iorio V., Coura D., Reis L., Oilawa M., Junior C., **2007**, A Visual Language for Animated Simulation, Journal of Universal Computer Science, Vol.13(6) pp.767-785

Dimitracopoulou A., and Petrou A., **2003**, *Advanced Collaborative Distance Learning Systems for young students: Design Issues and current trends on new cognitive and metacognitive tools*, In Themes in Education International Journal, Special Issue, Issues and Trends Regarding the Application of Information and Communication Technologies to Distance Learning, pp.1-83

Doran J., Franklin S., Jennings N., Norman T., **1997**, *On Cooperation in Multi-Agent Systems*, The Knowledge Engineering Review, Vol.12(3), pp.309-314

Downes S., **2007**, *Models for Sustainable Open Educational Resources*, Interdisciplinary Journal of Knowledge and Learning Objects, Vol.3, pp.29-44

Edwards J., **2005**, *Subtext: Uncovering the Simplicity of Programming*, Proceedings of the 20th Annual ACM SIGPLAN Conference on Object Oriented Programming Systems, languages, and applications, San Diego, CA, USA, pp.505-518

Ehrenmann M., Rogalla O. Zöllner R. and Dillmann R., **2001**, *Teaching Service Robots Complex Tasks: Programming by Demonstration for Workshop and Household Environments*, Proceedings of the IEEE International Conference on field and Service Robotics 2001 (FRS), pp. 397-402

Ehrenmann M., Zollner R., Rogalla O., and Dillmann R., **2002**. *Programming Service Tasks in Household Environments by Human Demonstration*, Robot and Human Interactive Communication, Proceedings. 11th IEEE International Workshop on, pp.460-467

Engeström Y., **1987**, *Learning by Expanding*. Helsinki, Finland: Orienta-konsultit

Engeström Y., **1993**, *Developmental Studies of Work as a Testbench of Activity Theory: The Case of Primary Care Medical Practice*, Understanding practice: Perspectives on activity and context pp.64-103, Cambridge, MA: Cambridge University Press

Erman L., Hayes-Roth F., Lesser V., Reddy R., **1980**, *The Hearsay II Speech Understanding System: Integrating Knowledge to Resolve Uncertainty*, Computing Surveys, 12(2) pp.213-253

ESRC, **2007**, Research Ethics Framework.
www.ESRC.ac.uk/ESRCInfoCentre/Images/ESRC_RE_Ethics_FRAME_tcm6-11291.pdf
Accessed: 23rd August 2007

Feigenbaum E., **1977**, *The Art of Artificial Intelligence: Themes and Case Studies of Knowledge Engineering'*, 5th International Joint Conference Artificial Intellignence, pp.1049-1029

Feisel L., and Rosa A., **2005**, *The Role of the Laboratory in Undergraduate Engineering Education*, Journal of Engineering Education, pp.121-130,

Fernando T., and Dew P., **1998**, *A Distributed Virtual Environment for Collaborative Engineering*, Presence: Teleoperators and Virtual Environments, Vol.7(3), pp.241-261

Festing M., **2001**, *Guidelines for the Design and Statistical Analysis of Experiments*, in Papers Submitted to ATLA (Alternatives to Laboratory Animals), ATLA 29, pp.427-446,

Fjuk A., **1995**, *Towards an Analytical framework for CSCdistanceL*, Department of Informatics University of Oslo, pp.1-9
www.telenor.no/fou/program/nomadiske/articles/CSCL95.pdf
accessed: 22 August 2007

Forbus K., Whalley P., Everett J., Ureel L., Brokowski M., Baher J., Kuehne S., **1999**, *CyclePad: An Articulate Virtual Laboratory for Engineering Thermodynamics*, Artificial Intelligence 114, pp.297-347,

Forinash K., and Wisman R., **2005**, *Building Real Laboratories on the Internet*, International Journal of Continuing Engineering Education and Lifelong Learning, Vol.1(1), pp.56-66

Friedman A., **1989**, *Computer Systems Development: History, Organization and Implementation*, Chichester: John Wiley

Friedrich H., and Dillmann R., **1995**, *Robot Programming Based on a Single Demonstration and User Intentions*, 3rd European Workshop on Learning Robots (ECML95)

Friedrich H., and Kaiser M., **1995**, *What can Robots Learn from Humans*, IFC Workshop on Human-Orientated Design of Advanced Robotics Systems, (DARS '95), Vienna, Austria, pp.47-74

Friedrich H., Münch S., Dillmann R., Bocionek S., Sassin M., **1996**, *Robot programming by demonstration: Supporting the induction by human interaction*, Machine Learning, Seiten pp.163-189

Garrison, **1990**, *An Analysis and Evaluation of Audio Teleconferencing to Facilitate Education at a Distance*, The American Journal of Distance Education, Vol.4(3), pp.16-23

Gat E., **1998**, *On Three-Layer Architectures*, Artificial Intelligence and Mobile Robotics, AAAI Press, pp.195-210

Gleixner S., Young G., Vanasupa L., Dessouky Y., Allen E., Parent D., **2002**, *Teaching Design of Experiments and Statistical Analysis of Data Through Laboratory Experiments*, Frontiers in Education Conference, Vol.1, pT2D – 1

Golab L., and Özsu M., **2003**, *Issues in Data Stream Management*, SIGMOD Record, Vol.32(2), pp.5-14.

Gordon V., and Bieman J., **1995**, *Rapid Prototyping: Lesson Learned*, IEEE Software, 12(1) pp.85-95

Graham C., Cagiltay, K., Craner, J., Lim B., Duffy, T., **2000**, *Teaching in a Web-Based Distance Learning Environment: An Evaluation Summary Based on Four Courses*. Center for Research on Learning and Technology Technical Report No. 13-00. Bloomington, IN: Indiana University

Graham R., **2006**, *Experiment Design, The Engineers Companion Website*, College of New Jersey
www.tcnj.edu/~rgraham/rhetoric/experiment-design.html
accessed: 22 August 2007

Green, T., Schiele, F. and Payne, S., **1988**, *Formalisable models of user knowledge in human-computer interaction*, Working with Computers: Theory versus outcome, London: Academic Press, pp. 3-46

Green T., Davies, S., Gilmore, D., **1996**, *Delivering Cognitive Psychology to HCI: the Problems of Common Language and of Knowledge Transfer*, Interacting with Computers, 8 (1), pp.89-111

Gröber S., Vetter M., Eckert B., Jodl H. J., 2007, *Experimenting from a Distance-Remotely Controlled Laboratory,* European Journal of Physics, Vol.28 pp. s127-s141.

Grudin, J., **1990**, *The Computer Reaches Out: The Historical Continuity of User Interface Design*, Proceedings of CHI '90, ACM SIGCHI Conference, Seattle, Wash., USA

Hashemi J., Kholamkar S., Chandrashekar N.Anderson E., **2006**, *Web-Based Delivery of Laboratory Experiments and Its Effectiveness Based on Student Learning Style*, ASEE Annual Conference and Exposition: Excellence in Education, USA, June

Hayes G., Demiris J., **1994**, *A Robot Controller Using Learning by Imitation*, Proceedings of the Symposium on Intelligent Robotic Systems, Grenoble, France, pp.198-204

Hayes-Roth B., Pfleger K., Lalanda K., Morignot P., **1995**, *A Domain Specific Software Architecture for Adaptive Intelligent Systems*, Knowledge Systems Laboratory Report No. 94-11
ftp://ksl.stanford.edu/pub/KSL_Reports/KSL-94-11/ps.gz
accessed: 22 August 2007

Hayes-Roth B., **1995**, *Agents on Stage, Advancing the State of the Art of AI*, Knowledge Systems Laboratory Report No. KSL 95-50
ftp://ksl.stanford.edu/pub/KSL-Reports/KSL-95-50.ps.gz
Accessed: 22 August 2007

Helsinger A., and Wright T., **2005**, *Cougaar: A Robust Configurable Multi Agent Platform*, IEEE Aerospace Conference pp.1-10

Hoare C., **1969**, *An Axiomatic Basis for Computer Programming*, Communications of the ACM Vol.12,(10), pp.576-581

Hoare C., **1973**, *Hints on Programming Language Design*, Stanford Artificial Intelligence Laboratory, Memo AIM 224, Computer Science Department Report No. CS-403

Hollan J., Hutchins E., Kirsh D., **2000**, *Distributed Cognition: Toward a New Foundation for Human-Computer Interaction Research*, Computer-Human Interaction, Vol.7, No. 2, pp. 174 -196

Horton W., **1994**, *How We Communicate*. Paper Presented at the Meeting of the Rocky Mountain Chapter of the Society for Technical Communication, Denver, CO

Hundhausen C., Douglas S., Stasko J., **2002**, *A Meta-Study of Algorithm Visualization Effectiveness*, Journal of Visual Languages and Computing 13 pp.259-290

Hylén, **2006**, *Open Educational resources: Opportunities and Challenges*, Open Education, pp.49-63

Jayakumar S., Squires R. G., Reklaitis G. V., Andersen P. K., Dietrich B. K., **1995**, *The Purdue-Dow Styrene-utadiene Polymerization Simulation*, Journal of Engineering Education, Vol.84(3), pp.271-277

Jennings N., Corera J., Laresgoiti I., **1995**, *Developing Industrial Multi-Agent Systems*, Proceedings of the First International Conference on Multi-Agent Systems, pp.423-430

Johansson M., and Astrorn K., **1996**, *Virtual Interactive Systems for Control Education*, Proceedings of the IEEE Conference of Decision and Control, pp.3888-3889

Johansson M., Gafvert M., Astrorn K., **1998**, *Interactive Tools for Education in Automatic Control*, IEEE Control Systems Magazine, 18(3), pp.33-40

Johnson P., **1992**, *Human-Computer Interaction: Psychology, Task Analysis and Software Engineering*, New York: McGraw Hill

Johnson S., Aragon S., Shaik N., Palma-Rivas N., **2000**, *Compararive Analysis of Learning Satisfaction and Learning Outcomes in Online and Face-to-Face Learning Environments*, Journal of Interactive Learning Research, Vol.11(1), pp.29-50

Johnson S., Benson A., Duncan J., Shinkareva O., Taylor G., Treat T., **2003**, *Distance Learning in Postsecondary Career and Technical Education*, University of Minnesota, St Paul MN: National Research Center for Career and Technical Education,

Johnstone S. M., **2005**, *Open Educational Resources Serve the World*, Educause Quarterly, Vol.28(3), p 15-19

Kaiser M., Friedrich H., Dillmann R., **1995**, *Obtaining good performance from a bad teacher*, In Workshop: Programming by Demonstration vs Learning from Examples, International Conference on Machine Learning, California, USA,

Kaptelinin V., **1996**, *Activity Theory : Implications for Human-Computer Interaction*, Context and Consciousness; Activity Theory and Human-Computer Interaction, MIT, Massachusetts, pp.103-116

Kicinger R., and Wiegard R., **2003**, *Experiment Design and Methodology: Basic Lessons in Empricism*, 2003 Summer Lecture Series www.kicinger.com/presentations/pdf/MethodologyLectureGAG2003.pdf Accessed: 22 August 2007

Kies J., **1997**, *Empirical Methods for Evaluating Video-Mediated Collaborative Work*, PhD Thesis, Virginia Polytechnic Institute and State University

Kim J., and Kalb J., **1996**, *Design of Experiments: An Overview and Application Example*, Medical Device and Diagnostic Industry Magazine, pp.78-88

Kolb D., **1984,** *Experiential Learning*, Englewood Cliffs, NJ: Prentice-Hall

Koper R., and Olivier B., **2004,** *Representing the Learning Design of Units of Learning*, Educational Technology and Society, 7(3), pp.97-111

Kuniyoshi Y., Inaba M., Inoue H., **1994,** *Learning by Watching: Extracting Reusable Task Knowledge from Visual Observation of Human Performance*, IEEE Transaction on Robotics and Automation 10(6), pp.799-822

Kuutti K., **1996,** *Activity Theory as a Potential Framework for Human-Computer Interaction Research*, Context and Consciousness; Activity Theory and Human-Computer Interaction, MIT, Massachusetts, pp.18–44

Kypuros J., and Connolly T., **2005,** *Collaborative Experimentation and Simulation: A Pathway to Improving Student Conceptualization of the Essentials of System Dynamics and Control Theory*, Proceedings of the American Society for Engineering Education Annual Conference and Exposition: The Changing Landscape of Engineering and Technology Education in a Global World

Lau T., and Weld D., **1998,** *Programming by Demonstration: An Inductive Learning Formulation*, Proceedings of the Fourth International Conference on Intelligent User Interfaces, pp.145 – 152

Lesser V., and Erman, L., **1980,** *Distributed Interpretation: A Model and Experiment*, IEEE Transactions on Computers, C-29(12) pp.1144 -1163

Lieberman H., **1993,** *Tinker: A Programming by Demonstration System for Beginning Programmers*, Watch What I Do: Programming by Demonstration, Allen Cypher, MIT Press

Lieberman H., **1994,** *A User Interface for Knowledge Acquisition from Video*, Proceedings of the 12th National Conference of the American Association for Artificial Intelligence, Seattle, pp.527-534,

Luck M., and d'Inverno M., **1995,** *Structuring a Z Specification to Provide a Formal Framework for Autonomous Agent Systems*, Computer Science, 967, Springer-Verlag pp.47-62 Heidelberg

Maes P., **1994,** *Modeling Adaptive Autonomous Agents*, Artificial Life, Vol.1(1), pp.135-162, MIT Press

Mamone S., **1992,** *Empirical Study of Motivation in an Entry Level Programming Course*. ACM SIGPLAN Notices Vol.27(3), pp.54-60

Manzone J., and Angehrn A., **1997,** *Understanding Organizational Dynamics of IT-Enabled Change: A Multi-Media Simulation Approach*, Journal of Management Systems, Vol.14(3) 1997, pp.109-140

Massy W., and Zemsky R., **1995,** *Using Information Technology to Enhance Academic Productivity*, Washington DC: Educom
www.educause.edu/ir/library/html/nli004.html
Accessed: 22 August 2007

Matarić M., **1997,** *Behaviour-Based Control: Examples from Navigation, Learning and Group Behaviour*, Journal of Experimental and Theoretical Artificial Intelligence 9(2-3), pp.323-336

McKay D., Presor J. and M$^{c}$Entire R., Finin T., **1996,** *An Architecture for Information Agents* Proceedings of the 3$^{rd}$ International Conference of Artificial Intelligence Planning Systems, (ARPI Supplement) (AIPS 96) AAAI Press, pp.1-8

McNabb J., **1994,** *Telecourse Effectiveness: Findings in the Current Literature.* Tech Trends, 39(4), pp.39-40

Menges R., and Austin, A., **2001,** *Teaching in Higher Education*, Handbook of research on teaching, 4 th Edition, Washington, DC: American Educational Research Association, pp.1122-1156

Merrill M., **1994,** *Instructional Design Theory*, Englewood Cliffs: Educational Technology Publications

Mizell A., **1994,** *Graduate Education through Telecommunications: The Computer and You*, AECT National Convention, Annual Meeting of the Association for Educational Communication and Technology, Nashville, Tennessee, pp.1-13

Molnar L., Omerdic E., van de Ven P., Toal D., Flanagan C., **2004,** *On the Development of Tethra Submersible Vehicle for Autonomous and Remotely Operated Modes*, WSEAS Transactions on Systems, Vol.3(6), pp.2454-2459

Moore M., and Kearsky, G., **1996,** *Distance Education: A Systems View*, Belmont, CA: Wadsworth

Moritz S., Wei F., Blank G., **2005,** From Objects-First to Design-First with Multimedia and Intelligent Tutoring, ACM Association for Computing Machinary, Vol.37(3), pp.99-103

Morse L., and Truman B., **1996,** *A Distance Education Infrastructure*, MC Journal: The Journal of Academic Media Librarianship, Vol.4(1)

Mulholland P., **1998,** *A Principled Approach to the Evaluation of SV: A Case Study in Prolog, Software Visualization* – Programming as a Multimedia Experience, The MIT Press pp.453-480

Müller D. and Ferreira J., **2005,** *Online Labs and the MARVEL Experience*, International Journal of Online Engineering, Vol.1(1)
www.i-joe.org/ojs/include/getdoc.php?id=41&article=4&mode=pdf
Accessed: 22 August 2007

Mwanza D., **2001,** *Where Theory meets Practice: A Case for an Activity Theory based Methodology to guide Computer System Design*, Proceedings of INTERACT 2001: 8$^{th}$ IFIP TC 13 Conference on Human-Computer Interaction
kmi.open.ac.uk/publications/pdf/kmi-01-7.pdf
Accessed: 22 August 2007

Myers C., Bennett, D., Brown G., Henderson T., **2004**, *Emerging Online Learning Environments and Student Learning: An Analysis of Faculty Perceptions.* Educational Technology & Society, 7(1), pp.78-86.

Nabeth T., Razmerita L., Anghem A., Roda C., **2005**, *InCA: A Cognitive Multi-Agents Architecture for Designing Intelligent & Adaptive Learning Systems*, ComSIS journal Vol.2(2) www.Comsis.fon.bg.ac.yu/ComSIS/vol2No2/RegularPapers/pdf/paper5.pdf Accessed: 22 August 2007

Nakakoji and Yamamoto, **2003**, *Toward a Taxonomy of Interaction Design Techniques for Externalizing in Creative Work*, Proceedings of the 10th International Conference on Human Computer Interaction (HCII 2005) Vol.2, Theory and Practice (Part II), pp.1258-1262

Nardi B., **1996**, *Context and Consciousness; Activity Theory and Human-Computer Interaction,* MIT, Massachusetts, pp.235-246

Nardi B., **2002**, *Coda and response to Christine Halverson.* CSCW, pp.269-275

Nevalainen S., and Sajaniemi J., **2005**, *Short-Term Effects of Graphical versus Textual Visualisations of Variables on Program Perception*, 17th Workshop of the Psychology of Programming interest Group, Sussex University, pp.77-91

Nevill-Manning C., and Witten I., **1995**, *Detecting Sequential Structure*, Proceedings of the Workshop on Programming by Demonstration, pp.49-56

Nevill-Manning C., and Witten I., **1997**, *Identifying Hierarchical Structure in Sequences: A linear-time algorithm.* Journal of Artificial Intelligence Research, 7 pp.67-82

Nicolescu M., **2003**, *A Framework for Learning from Demonstration, Generalization and Practice in Human-Robot Domains*, PhD Thesis, Faculty of the Graduate School, University of Southern California

Nicolescu M., and Matarić M., **2002**, *A Hierarchical Architecture for Behaviour-Based Robots*, Proceedings of the First International Joint conference on Autonomous Agents and Multi-Agent Systems, Bologna, Italy, pp.227-233

Nicolescu M. Matarić M., **2003**, *Linking Perception and Action in a Control Architecture for Human-Robot Domains*, Proceedings of the 36th Annual Hawaii International Conference on System Sciences, IEEE Computer Society, Hawaii, USA

NIST/SEMATECH, **2006**, *eHandbook of Statistical Methods* www.itl.nist.gov/div898/handbook/ Accessed: 22 August 2007

Norman D., **1988**, *The Design of Everyday Things*, Basic Books, New York

Norman T., and Jennings N., **2002**, *Constructing a Virtual Training Laboratory using Intelligent Agents*, International Journal of Continuing Engineering Education and Lifelong Learning, Vol.12, Numbers 1-4, pp.201-213

Odell J., Parunak H, Bauer B., **2000**, *Extending UML for Agents*, Proceedings of the 2nd International Bi-Confernece Workshop on Agent-Orientated Information Systems, AOIS'00, pp.3-17

Olson J., Olson, G., **1991**, *The Growth of Cognitive Modeling since GOMS*. Human Computer Interaction, Vol.5, pp.221-266

Onda H., Suehiro T., Kitagaki K., **2002**, *Teaching by Demonstration of Assembly Motion in VR - Non-deterministic Search-Type Motion in the Teaching Stage*, Intelligent Robots and System, 2002. IEEE/RSJ International Conference on, Vol.3, pp.3066-3072

Page E., Buss A., Fishwick P., Healy K, Nance R., Paul R., **2000**, *Web-Based Simulation: Revolution or Evolution?*, ACM Transactions on Modelling and Computer simulation, Vol.10(1), pp.3-17

Patrick J., **1992**, *Training: Research and Practice*, Academic Press, San Diego, CA

Pearson D., Huffman S. Willis M. Laird J. and Jones R., **1993**, *A Symbolic Solution to Intelligent Real-Time Control*, Robotics and Autonomous Systems, Vol.11, pp.279-291

Perkins D., **1993**, *Person-Plus: A Distributed View of Thinking and Learning*, Distributed cognitions: Psychological and educational considerations pp.47-87, New York: Cambridge University Press

Petre M., Blackwell A., **1999**, *Mental Imagery in Program Design and Visual Programming*, Intenational Journal of Human-Computer Studies, Vol.1, pp.7-30

Ramadhan H., **1997**, *Improving the Engineering of Model Tracing based Intelligent Programming Diagnosis,* IEE Proceedings - Software Engineering, Vol.144(3), pp.149-161

Ravitch D., **1987**, *Technology and the Curriculum, What curriculum for the Information age?*, Hillsdale, NJ: Lawrence Erlbaum Associates, Inc

Rauwerda H., Roos M., Hertzberger B., Breit T., **2006**, *The Promise of a Virtual Lab in Drug Discovery*, Drug Discovery Today, Vol.11(5-6), pp.228-236

Razmerita L., Nabeth T., Angehrn A., Roda C., **2004**, *INCA: An Intelligent Cognitive Agent Based Framework for Adaptive and Interactive Learning*, Proceedings of the IEEE International Conference on Cognition and Explratory Learning in the Digital Age (IADIS CELDA 2004) pp.373-380.

Rich E., Knight K., **1991**, *Artificial Intelligence*, Mc Graw-Hill,

Roberts J., **1964,** *The Self-Management of Culture. In Explorations in Cultural Anthropology: Essays in Honor of George Peter Murdoc*, W. Goodenough, Ed. McGraw-Hill, London, UK

Roda C., Angehrn A., Nabeth T., **2001,** *Matching Competencies to Enhance Organisational Knowledge Sharing: An Inelligent Agents Approach*, Proceedings of the 7th International Netties Conference, pp.931-938

Rogers E., **1995,** *Diffusion of Innovation*, 4th Edition, Free Press, NY.

Rogers Y., **2004,** *New Theoretical Approaches For HCI*, ARIST: Annual Review of Information Science and Technology, Vol.38, pp.38-144

Russell B., **1946,** *A History of Western Philosophy,* George Allen and Unwin (Publishers) Ltd.

Sadeh N., Hildum D., Kjenstad D., Tseng A., **2001,** *MASCOT: An Agent-Based Architecture for Dynamic Supply Chain Creation and Coordination in the Internet Economy*, Production Planning and Control, Vol.12(3), pp.212-223

Saettler P., **1990,** *A History of Instructional Technology,* Englewood, Co: Libraries Unlimited

Setchi R., **2007,** *Personalisation in professional e-Learning and Performance Support*, Challenges in Higher Education and Research in 21st Century, CHER21'07, Sozopol, Bulgaria, 5-9 June

Salomon G., **1993,** *Distributed Cognitions: Psychological and Educational Considerations. Learning in Doing: Social, Cognitive, and Computational Perspectives*, Cambridge University Press, New York, NY

Scaife M., and Rogers Y., **1996,** *External Cognition: How Do Graphical Representations Work?*, International Journal of Human-Computer Studies, Vol.45, pp.185-213

Schamber L., **1988,** *Delivery Systems for Distance Education*, Eric Clearinghouse on Information Resources, Syracuse University, School of Education, School of Information Studies

Schaude H., and Dillmann R., **1995,** *Integration of World creation and World model Adaptation for Planning and Simulation in Telerobotics*, Institute for Real-Time Computing and Robotics, University of Karlsruhe, Proceedings of the International Conference on Modelling and Simulation, Colombo, Sri Lanka
wwwipr.ira.uka.de/en/Publications/detailed_publication.htm?id=975588905
Accessed: 22 August 2007

Sherry L., **1996,** *Issues in Distance Learning*, International Journal of Educational Telecommunications Vol.1(4), pp.337-365

Shneiderman B., **2002,** *HCI Theory is Like the Public Library*, Posting to CHIplace online discussion forum, Oct 15th
www.chiplace.org/index.php?name=PNphpBB2&file=viewtopic&t=227

Shortliffe E., **1976**, *Computer-Based Medical Consultation MYCiN*, Elsevier, New York

Siemens G., **2005**, *Connectivism: A learning Theory for the Digital Age*, International Journal of Instructional Technology and Distance Learning, Vol.2(1), pp.521-528

Siemer J., and Angelides M., **1998**, *A Comprehensive Method for the Evaluation of Complete Intelligent Tutoring Systems,* Decision Support Systems, Vol.22(1), pp.85-102

Simanek D., **1996**, *Error Analysis (Non-Calculus)*, Donald Simanek's Pp., Lock Haven University of Pennsylvania, Website.
www.lhup.edu/~dsimanek/errrors.htm
Accessed: 22 August 2007

Simmons R., **1994**, *Structured Control for Autonomous Robots*, IEEE Transactions on Robotics and Automation 10(1), pp.34-43

Sison R., Numao M., Shimura M., **2000**, *Multistrategy Discovery and Detection of Novice Programmer Errors*, Machine Learning 38, Klewer Academic Publishers, Vol.38(1-2), pp.157-180

Statsoft, **2003**, Statsoft Experiment Design (Industrial DOE), Statsoft Inc, Website:
www.statsoft.com/textbook/stexdes.html
Accessed: 22 August 2007

Suchman L., **1987**, *Plans and Situated Actions: The Problem of Human-Machine Communication*, Cambridge University Press, New York, NY

Sutcliffe A., **2000**, *On the Effective Use and Reuse of HCI Knowledge*, Transactions on Computer-Human Interaction, Vol.7(2), pp.197-221

Tait G., and Chao N., **2003**, *Hands-On Remote Laboratory for Freshman Engineering Education*, 33rd ASEE/IEEE Frontiers in Education Conference T3E-7

Tambe M., and Rosenbloom P., **1996**, *Architectures for Agents that Track other Agents in Multi-Agent Worlds*, Intelligent Agents, Vol.II, Springer Verlag Lecture Notes in Artificial Intelligence, (LNAI 1037), Issue 1037, pp.156-170

Thielscher M., **2005**, *FLUX: A Logic Programming Method for Reasoning Agents, Theory and Practice of Logic Programming*, Vol.5(4-5), pp.533-565

Trella M., Carmona C., Conejo R., **2005**, *MEDEA: An Open Service-Based Learning Platform for Developing Intelligent Educational System for the Web*, Proceedings of Workshop on Adaptive Systems for Web-based Education as 12th International Conference on Artificial Intelligence, Vol.12, pp.27-34

Van Wijngaarden A., **1963**, *Generalised ALGOL*, Annotated Revision in Automated Programming 3, pp.17-26

Voyles R., and Khosla P., **2001**, *A Multi-Agent System for Programming Robots by Human Demonstration*, Integrated Computer-Aided Engineering, Vol.8(1), pp.59-67

Vygotsky L., **1978**, *Mind in Society – The Development of Higher Psychological Processes*, Harvard University Press

Waters J., **2005**, *Experimental Research: Research Design*, Research Guidelines, Department of Psychology, Capilano College, North Vancouver, Website: www.capcollege.bc.ca/programs/psychology/students/research/experiment.html
Accessed 22 August 2007

Wegner P., **1976**, *Programming Languages – The First 25 Years*, IEEE Transactions on Computers, pp.1207-1225

White M., **1987**, *Information and imagery education, What Curriculum for the Information Age?*, Hillsdale, NJ: Lawrence Erlbaum Associates, Inc

Willis B., **1992**, *Instructional Development for Distance Education*, ERIC Clearing House on Information Resources, Document Reproduction Service, No ED 351 007, pp.1-5

Wirth N., **1974**, *On The Design of Programming Languages*, Proceedings of IFIP Congress 74, pp.386-393

Witten I., Nevill-Manning C., Maulsby D., **1996**, *Interacting with Learning Agents: Implications for ml from HCI*, Workshop on Machine Learning meets Human-Computer Interaction, ML'-6, pp.51-58

Wooldridge M., Jennings N., Kinny D., **2000**, *The Gaia Methodology for Agent-Orientated Analysis and Design,* Autonomous Agents and Multi-Agent Systems, Vol.3, pp.285-312

Wright P., Fields R. Harrison M., **2000**, *Analysing Human-Computer Interaction as Distributed Cognition; The Resources Model*, Human-Computer interaction Vol.15(1), pp.1-42

Wright T., and Cockburn A., **2003**, *A Language and Task-Based Taxonomy of Programming Environments*, IEEE Symposia on Human-Centric Languages and Environments, Auckland, New Zealand, pp.192-194

Wright T., and Cockburn A., **2005**, *Evaluation of Two Textual Programming Notations for Children*, Proceedings of the Sixth Australasian Conference on User Interface, Vol.40, pp.55-62

Wudka J., **1998**, *The Scientific Method*, Department of Physics and Astronomy at the University of California, Riverside, Website
physics.ucr.edu/~wudka/Physics7/Notes_www/Node5.html
Accessed: 22 August 2007

Young R., Howes A., Whittington J., **1990**, *A Knowledge Analysis of Interactivity.* Proceedings of INTERACT `90 International Conference on Human-Computer Interaction, pp.115-120. Amsterdam: North Holland

Zary N., Johnson G., Boberg J., Uno F., **2006**, *Development, Implementation and Pilot Evaluation of a Web-based Virtual patient Case Simulation Enviroment – Web-SP*, BMC Medical Education 6:10, doi:10.1186/1472-6920-6-10 www.biomedcentral.com/1472-6920/6/10 Accessed: 22 August 2007

Zhang J., **1996**, *A Representational Analysis of Relational Information Displays*, International Journal of Human-Computer Studies, Vol.45, pp.59-74

Zhang J., **1997**, *The Nature of External Representations in Problem Solving.* Cognitive Science, Vol.21(2), pp.179-217

Zhang J., and Norman D., **1994**, *Representations in Distributed Cognitive Tasks.* Cognitive Science, Vol.18, pp.87-122

Zollner R., Rogalla O., Dillmann R., Zollner M., **2002**. *Understanding Users Intention: Programming Fine Manipulation Tasks by Demonstration*, Intelligent Robots and System, IEEE/RSJ International Conference on, Vol.2, pp.1114-1119

# Appendix A

# The Interface Design

## The Interface Design

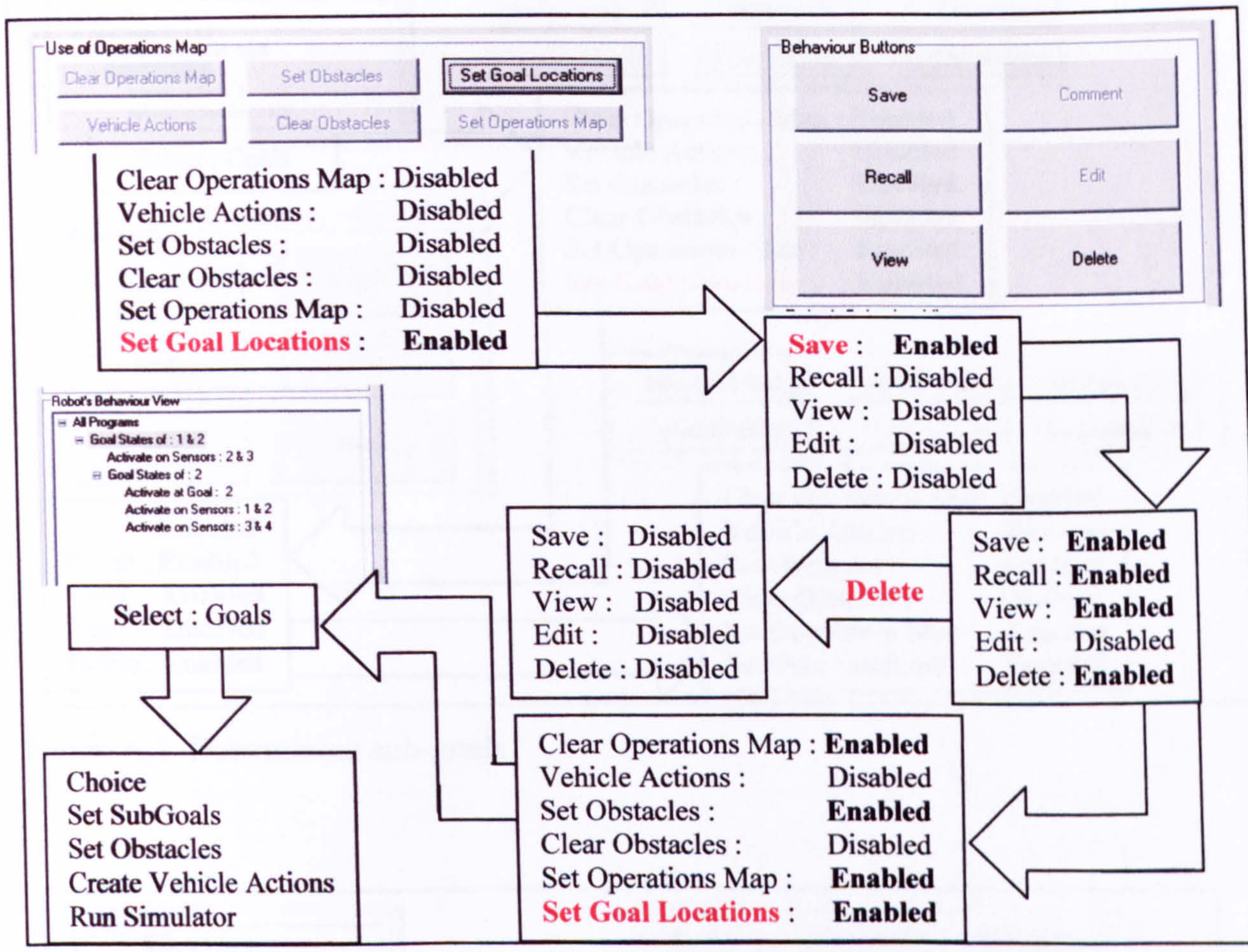



Figure A.1 Goal-Based Behaviour

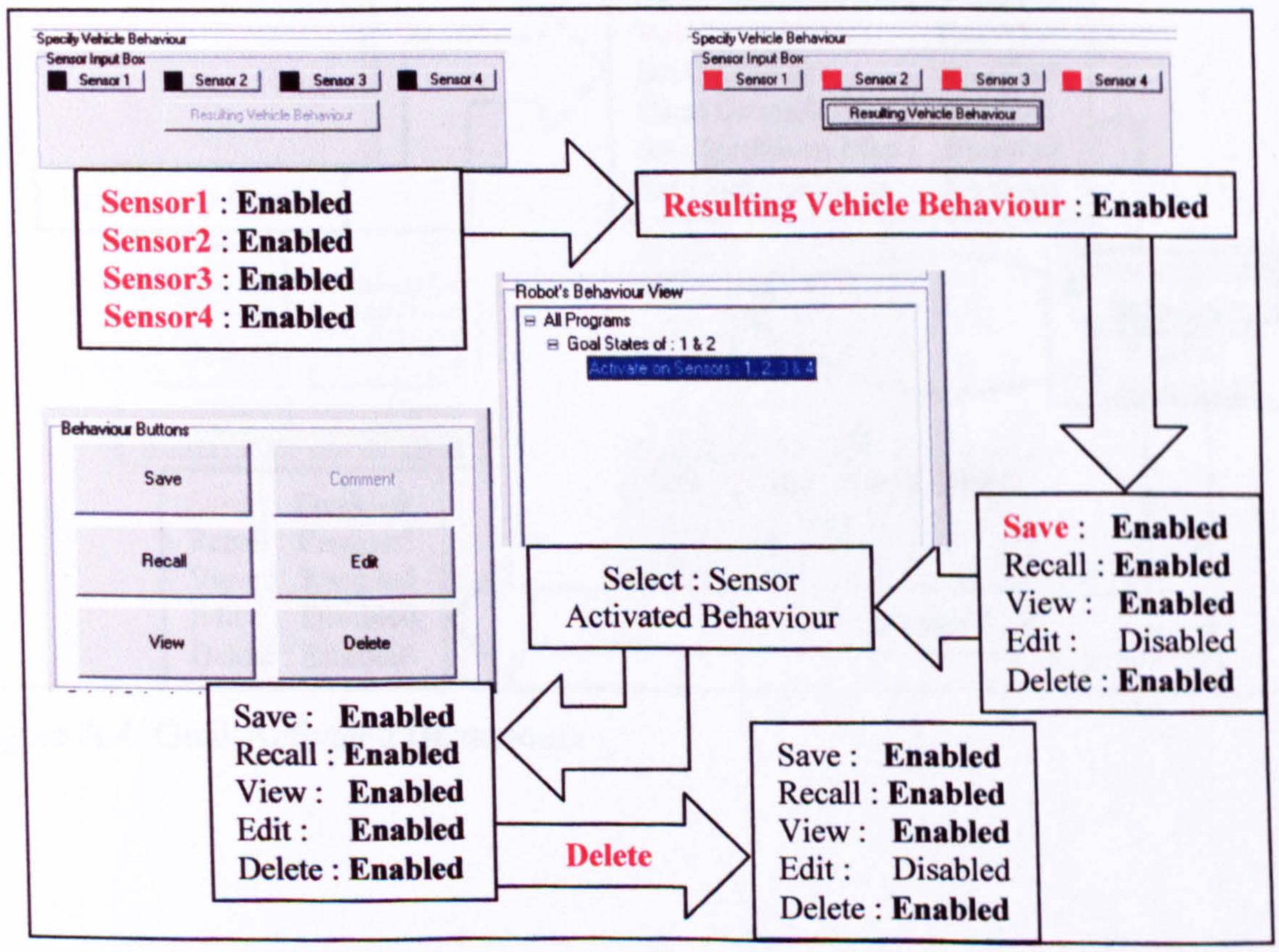



Figure A.2 Sensor-Activated Behaviour

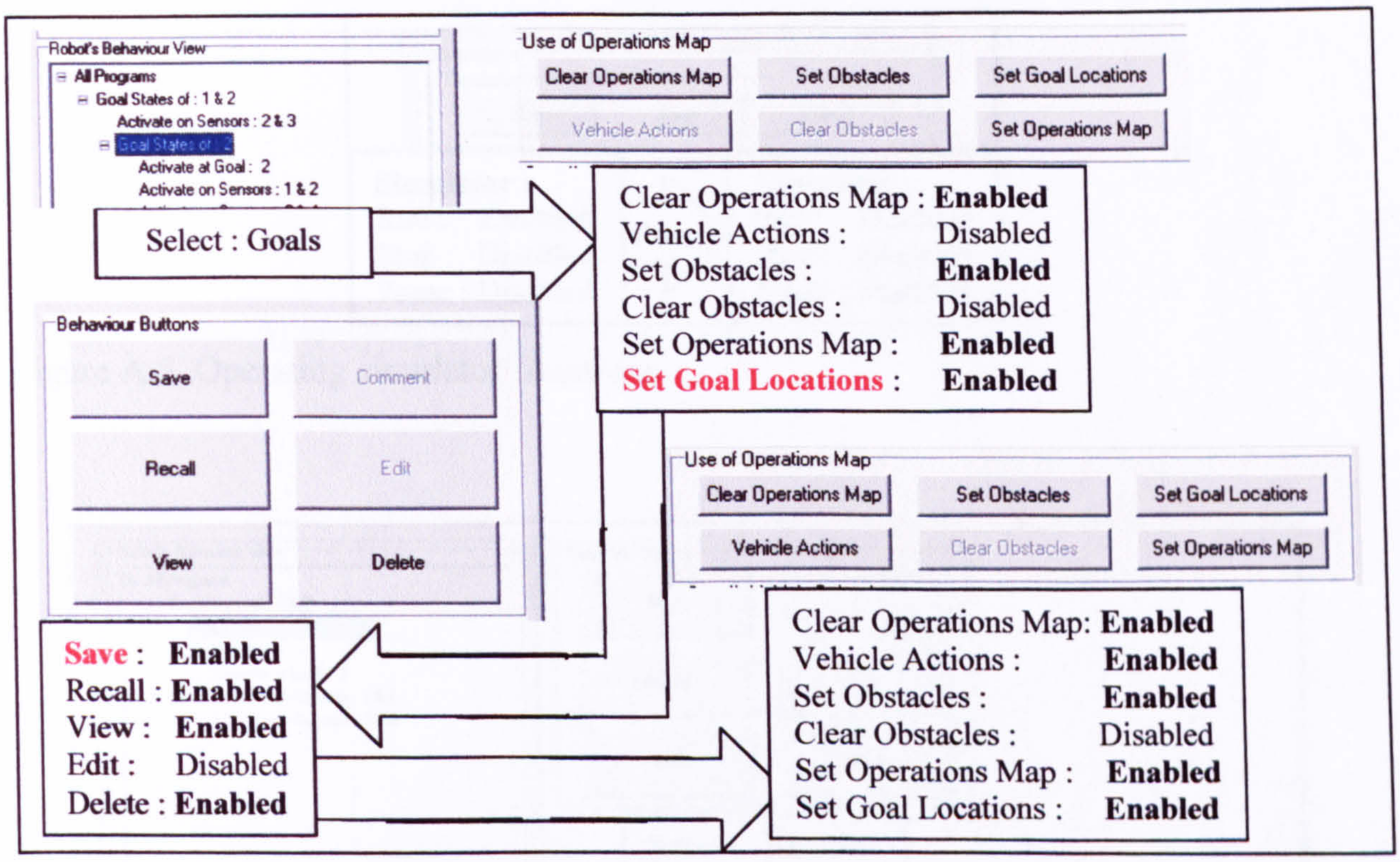



Figure A.3 Determining sub-goals

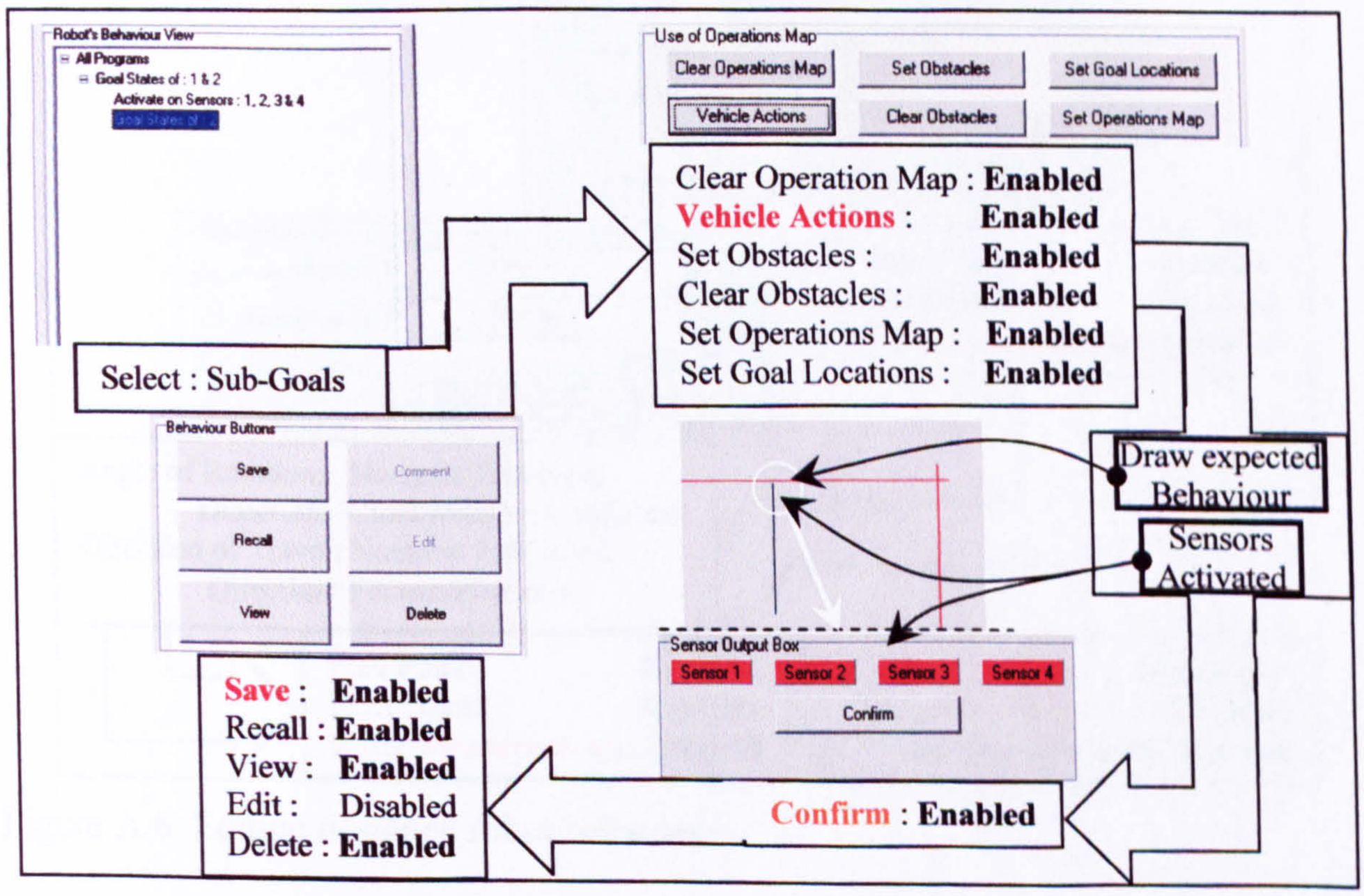



Figure A.4 Goal-Activated Behaviours

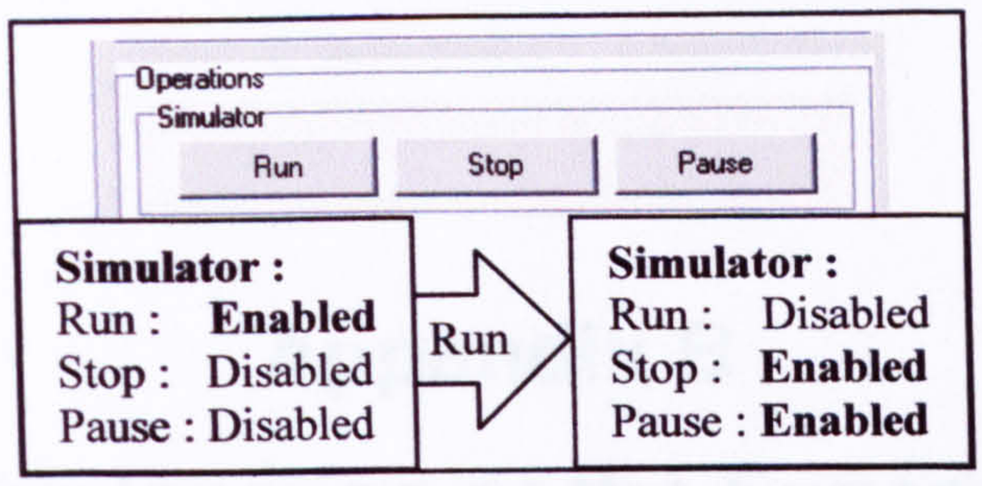



Figure A.5 Operating simulator functions

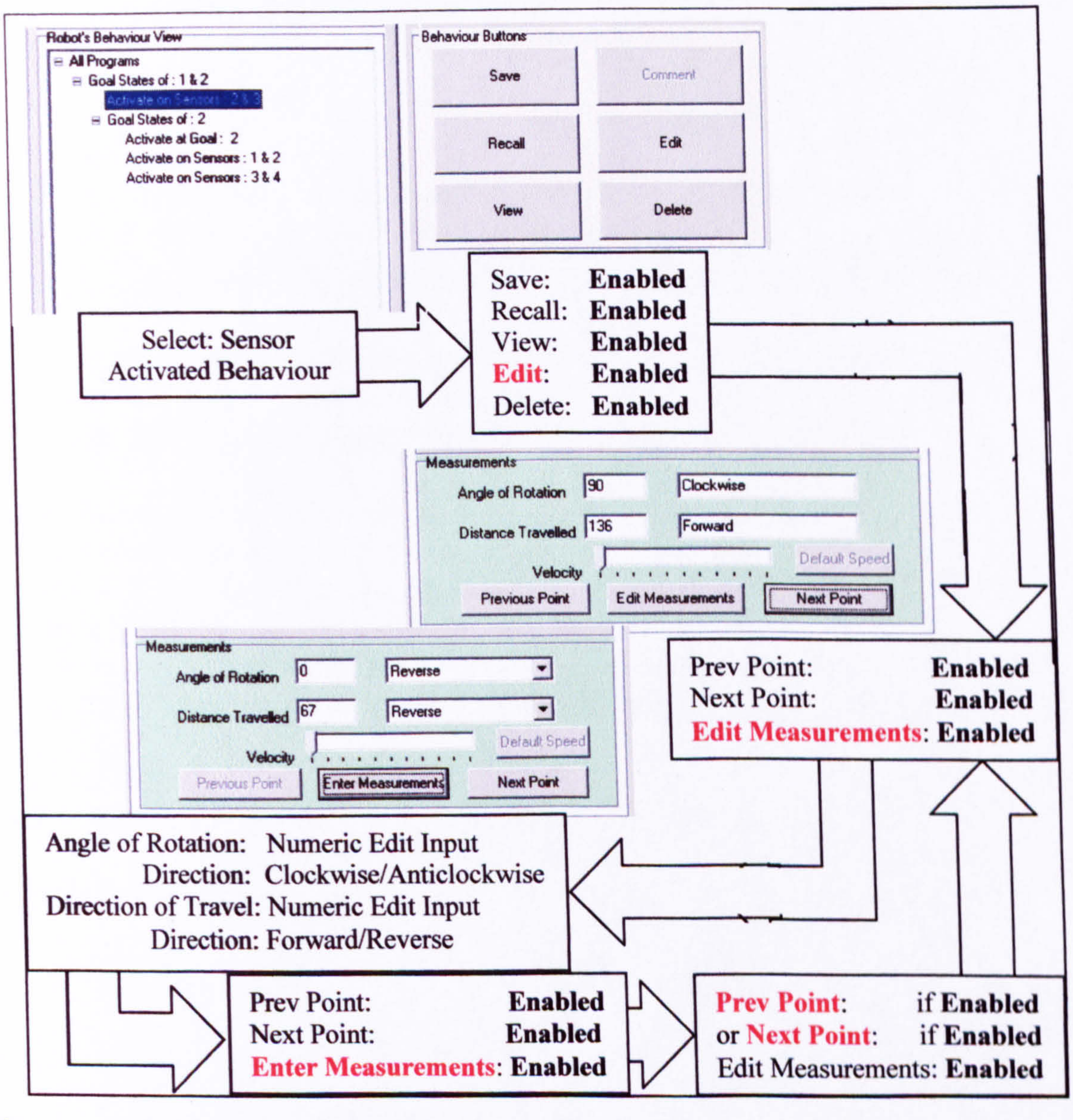


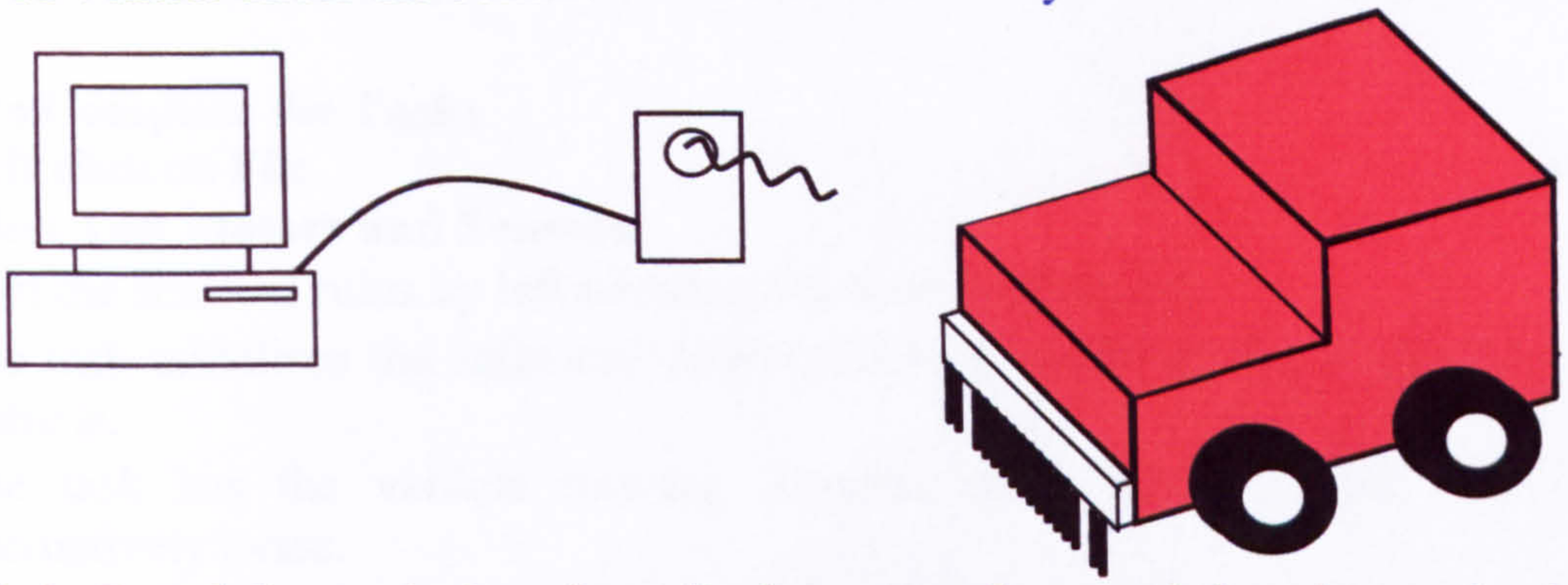

Figure A.6 Editing designed robot behaviour

# Appendix B

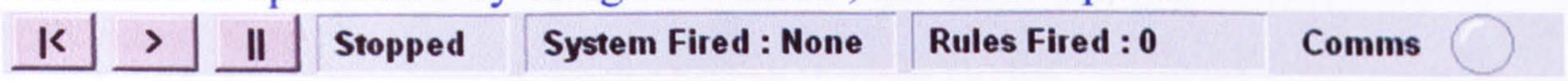
# An Explanation of the Experiment

# An Introduction to the Rule Base System

- This Rule Based system is based on:
  - A Computer with the Rule Base System
  - An Infra-Red Communications Link
  - A Vehicle under the control of the Rule Based System via the Infra-Red Link.



This Rule Based System's interface displays a toolbar, and four internal windows. The means of operation is by using the Toolbar, which comprises:




[|<] **Stop and Reset:** To stop the system and re-initialises the variables.
[>] **Run:** To run the system's current data.
[||] **Step Through:** Run the system one rule at a time.

Further, on the Toolbar there are:

1. **State Indicator:** this can be Stopped, Running or Stepping.
2. **System Fired:** Conflict Resolution Mode Indicator.
3. **Rules Fired:** Number of times the rules have been examined.
4. **Comms:** The communications light. This indicates the current status of communication between the rulebase and the Vehicle.

**Additional Notes:**

**[Esc] The 'Panic Button':** When the vehicle runs out of control pressing the 'panic button', (usually at the top left of the keyboard) will stop the vehicle's motors.

**The Communications light:** This is on the Toolbar at the top right of the screen.

- Communications have not been initiated:


- The communications is working properly.


- If a red cross is imposed then stop the program. [|<]

# TASK 1: Testing Motors and Sensors

This Task is intended to:

- Test the communication between the vehicle and the computer,
- Demonstrate that the vehicle is under the computer's control.
- Allow initial interaction

**Help to Complete the Task:**

1. **Left click** on **File**
2. select **Test Motors and Sensors**.
3. Start the selected rules by left clicking the **Run Button [>]**

- The task initialises the infra-red communications between the computer and the Vehicle.
- The task has the vehicle moving forward using the left and right motors alternatively twice.
- The task displays the current sensor values in the bottom left of the screen.

**Additional Notes**

1. **The Motors:** The motors on both sides of the vehicle activate alternately twice. After this the sensors can be tested.
2. **The Sensors:** Look at the value of the variable all sensors in the Variable Database window (Bottom left of the screen). It should be zero. If not check that one or more of the sensors (springs) is not touching the common earth rail.
   - Press any of the sensors springs on the Vehicle to see the sensor v alues change.

The sensors have the following values:

| *Your Vehicle* | Left Right | | | | | Right Left | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| 4096 | 2048 | 1024 | 512 | 256 | 128 | 64 | 32 | 16 | 8 | 4 | 2 |

- To stop the experiment from running: press the stop button. **[|<]**
- The emergency stop button is the **[ESC]** button on the keyboard

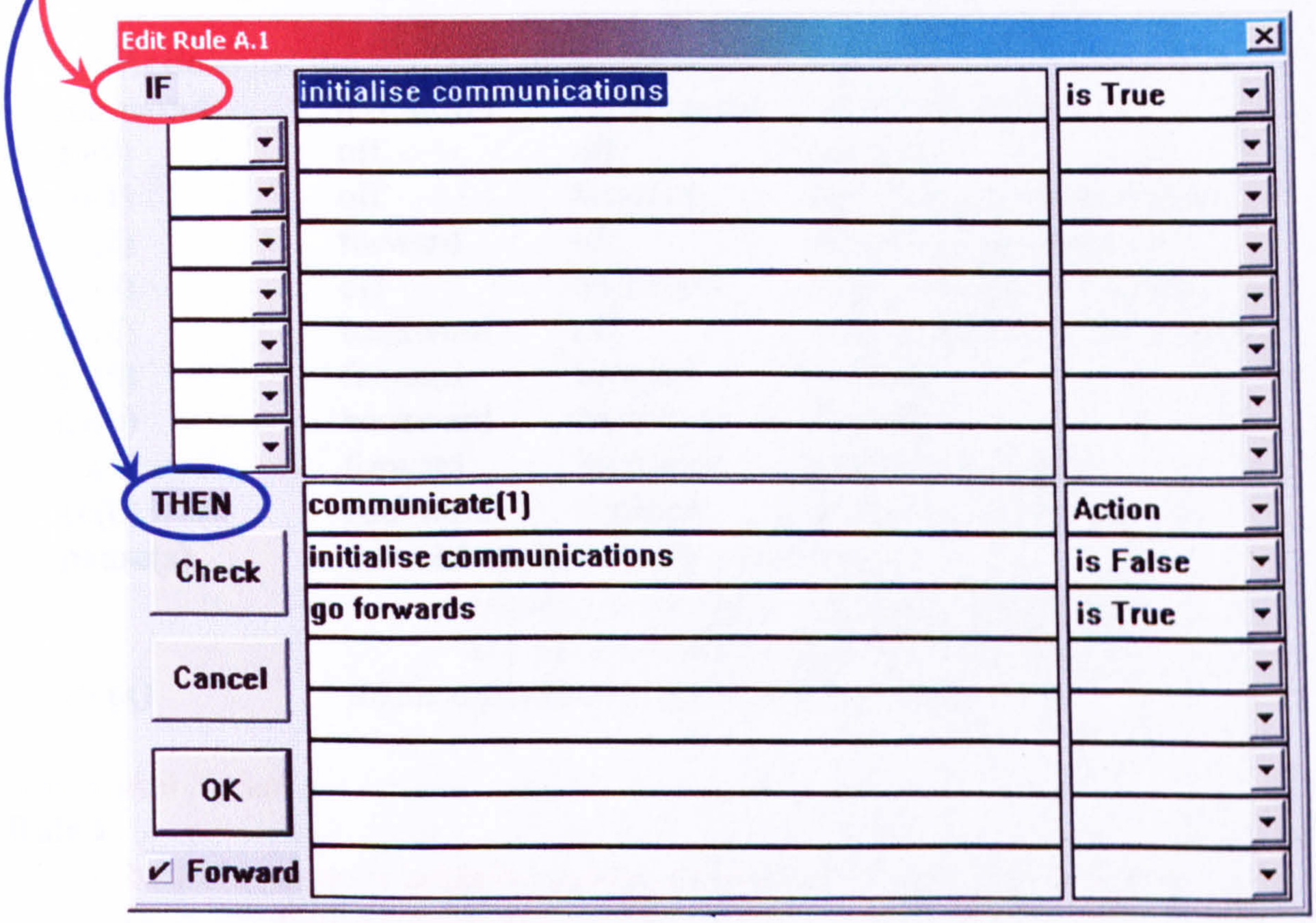
# Task 2: Make Vehicle Move Forwards

This task is to edit existing rules 1 and 2 to match the rules below:

**Rule 1**

**IF** initialise communications _____ is **True**

**THEN**
communicate(1) _____ **Action**
initialise communications _____ is **False**
go forwards _____ is **True**




1. Edit **Rule 2** to below

**IF** go forwards _____ is **True**
**THEN** go(5) _____ **Action**
pause(2.1) _____ **Action**
stop() _____ **Action**
go forwards _____ is **False**

2. In the **top left hand screen**, there is a **Fact Database**.

Set go forwards _____ is **False**

Start the selected rules by left clicking the **Run Button [>]**

**Additional Notes:**

1. **Modifying Rules:** double left-click a rule to produce the rule's edit window.
2. **Adding New Rules:** double left-click the Rules Database top-most clear line to bring up a window for inputting new rules.

# Task 3: Make the Vehicle Move Backwards

This task is to Design and edit a new rule, which allows the vehicle to move backwards.

**Help to complete the task:**
An explanation of rule construction

| | | |
|---|---|---|
| **IF** | RULE NAME_____ | is **True** |
| **THEN** | **… command**_____ | **Action** |
| | … RULE NAME _____ | is **False** |
| | … NEXT RULE | is **True** |

The **Actions** use the available **commands** given below.

| **command** | **left motor** | **right motor** | **vehicle moves:** |
|---|---|---|---|
| **go(0)** | off | off | stopped |
| **go(1)** | off | forward | turn forward anticlockwise |
| **go(2)** | forward | off | turn forward clockwise |
| **go(3)** | off | backward | turn backward clockwise |
| **go(4)** | backward | off | turn backwards anticlockwise |
| **go(5)** | forward | forward | forward |
| **go(6)** | backward | backward | backward |
| **go(7)** | forward | backward | rotate clockwise |
| **go(8)** | backward | forward | rotate anticlockwise |

**pause(s)** the rulebase pauses for s seconds,
pause(s) has a range of 0.0 and 65.0 seconds,
as **s** has a resolution of 0.1 is 0.055seconds.

**stop()** this is equivalent to go(0) and stops the vehicle

**Additional Notes:**

**Rule 1**

- As the **Rule Name** is **initialise communications**, Rule 1 starts with
  **IF** Initialise Communications is **True.**
- The **commands** in Rule 1 are:
- **communicate(1)** initialises the infra-red communications.
- **initialise communications** is **False** **Rule 1** is not to be performed again.
- **go forwards** is **True** execute **go forward** as a set of rules.

**Rule 2**

- The **Rule Name** is **go forwards**, Rule 2 starts with
  **IF** go forwards is **True**
  The Rule will only execute when go forwards as a rule set is **True**
  **Rule 1** has: **go forwards** is **True**.
- The **commands** in Rule 2 are:

| | | | |
|---|---|---|---|
| **THEN** | **go(5)** | **Action** | both motors set forwards |
| | **pause(2.1)** | **Action** | Pause for 1.165s before |
| | **stop()** | **Action** | stop the vehicle. |

- The **Rule Name** is **False** to prevent repetition
  go forwards is **False**

# Task 4: Detecting Obstacles

Amend rules to receive and react to sensor input.
The vehicle should move forwards, detect an obstacle and move backwards.

**Help to complete the task:**

- The previous rules drove the Vehicle backwards and forwards irrespective of external conditions.
- The rules will be modified to use sensors (the set of springs on the front of the Vehicle) to control in different circumstances.

**Rule 2**

| | | |
|---|---|---|
| **IF** | go forwards | is **True** |
| **THEN** | go(5) | **Action** |
| | go forwards | is **False** |
| | test sensors | is **True** |

**Rule 4**

| | | |
|---|---|---|
| **IF** | test sensors | is **True** |
| **THEN** | all_sensors = allSensors() | **Assign** |

**Rule 5**

| | | |
|---|---|---|
| **IF** | test sensors | is **True** |
| **and** | all_sensors > 0 | is **True** |
| **THEN** | stop() | **Action** |
| | test sensors | is **False** |
| | go backwards | is **True** |

**Additional Notes:**

**Rule 4**

- **all_Sensors = allSensors()** **Assign**
  **all_Sensors** is a variable with the value given by the **allSensors()** command

**Rule 5**

- The **Rule Name** is

  **IF** **test sensors** is **True**
  **and** **all_sensors > 0** is **True**

  The Rule will only execute when both **test sensors** as a rule set is **True**
  and **all_sensors** is a positive value.

  **Rule 2** has: **test sensors** is **True**
  **Rule 4** provides **all_sensors** with its value.

# Task 5: Obstacle Navigation

The task is creating rules to react to specific sensor values and negotiate an obstacle.

**Help to complete the task:**

1. The sensors have the following values:

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Left | | | | | | Right | | |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| 2 | 4 | 8 | 16 | 32 | 64 | 128 | 256 | 512 | 1024 | 2048 | 4096 |
| | | | 126 | | | | | | 8064 | | |

2. **all_Sensors** has the sensors value provided by the **allSensors()** command.
   If some of the sensors on the left have been activated, but none on the right, the following would be true:

- **bitAnd(all_sensors, 126) > 0**
- **bitAnd(all_sensors, 8064) = 0**

3. These conditions can be used to test if a detected object is on the right or the left. The following rule outline can be used for determining the vehicles behaviour.

**Rule 5**

| | | |
|---|---|---|
| **IF** | bitAnd(all_sensors, 126) > 0 _____ | **TRUE** |
| **and** | bitAnd(all_sensors, 8064) = 0 _____ | **TRUE** |
| **THEN** | all_sensors = 0 _____ | **Action** |

4. To rotate the vehicle 90° either

- **clockwise**

go(7)_____ **Action**
pause(2.1)_____ **Action**
stop() _____ **Action**

**or**

- **anticlockwise**

go(8)_____ **Action**
pause(2.1)_____ **Action**
stop() _____ **Action**

# Task 6: Park in a Corner

Create rules for the vehicle to park in a corner parallel to one of the walls.

**Additional Notes:**
The Commands for the Rule-Based System are:
**allSensors()** This returns the status of the sensors as a number between 0 and 2 + 4 + 8 + 16 + 32 + 64 + 128 + 256 + 512 + 1024 + 2048 + 4096 = 8190

**collisionMode()** After this command, the moving vehicle will stop when any of the sensors are activated. Once the vehicle has stopped, or the motors have timed out, this mode is cancelled.

**communicate(1)** This command initialises the infra-red communications.
**communicate(0)** This command closes down communications

**go(n)** This determines the motor directions.

| command | left motor | right motor | vehicle moves: |
|---|---|---|---|
| go(0) | off | off | stopped |
| go(1) | off | forward | turn forward anticlockwise |
| go(2) | forward | off | turn forward clockwise |
| go(3) | off | backward | turn backward clockwise |
| go(4) | backward | off | turn backward anticlockwise |
| go(5) | forward | forward | forward |
| go(6) | backward | backward | backward |
| go(7) | forward | backward | rotate clockwise |
| go(8) | backward | forward | rotate anticlockwise |

**neverStop()** This is the equivalent to stopAfter(0).

**pause(s)** The rulebase pauses for s seconds, where s is in the range of 0.0 and 65.0 seconds, with a resolution of 0.1 is 0.055seconds.

**stop()** This is equivalent to go(0) and stops the vehicle
**stopAfter(time)** The Vehicle is instructed how long to obey the last go(n) command. The time can be between 0.1 and 25.4 seconds, with a resolution of 0.1 seconds. The default is 5.0 seconds. Setting stopAfter(0) sets the command as continuous.

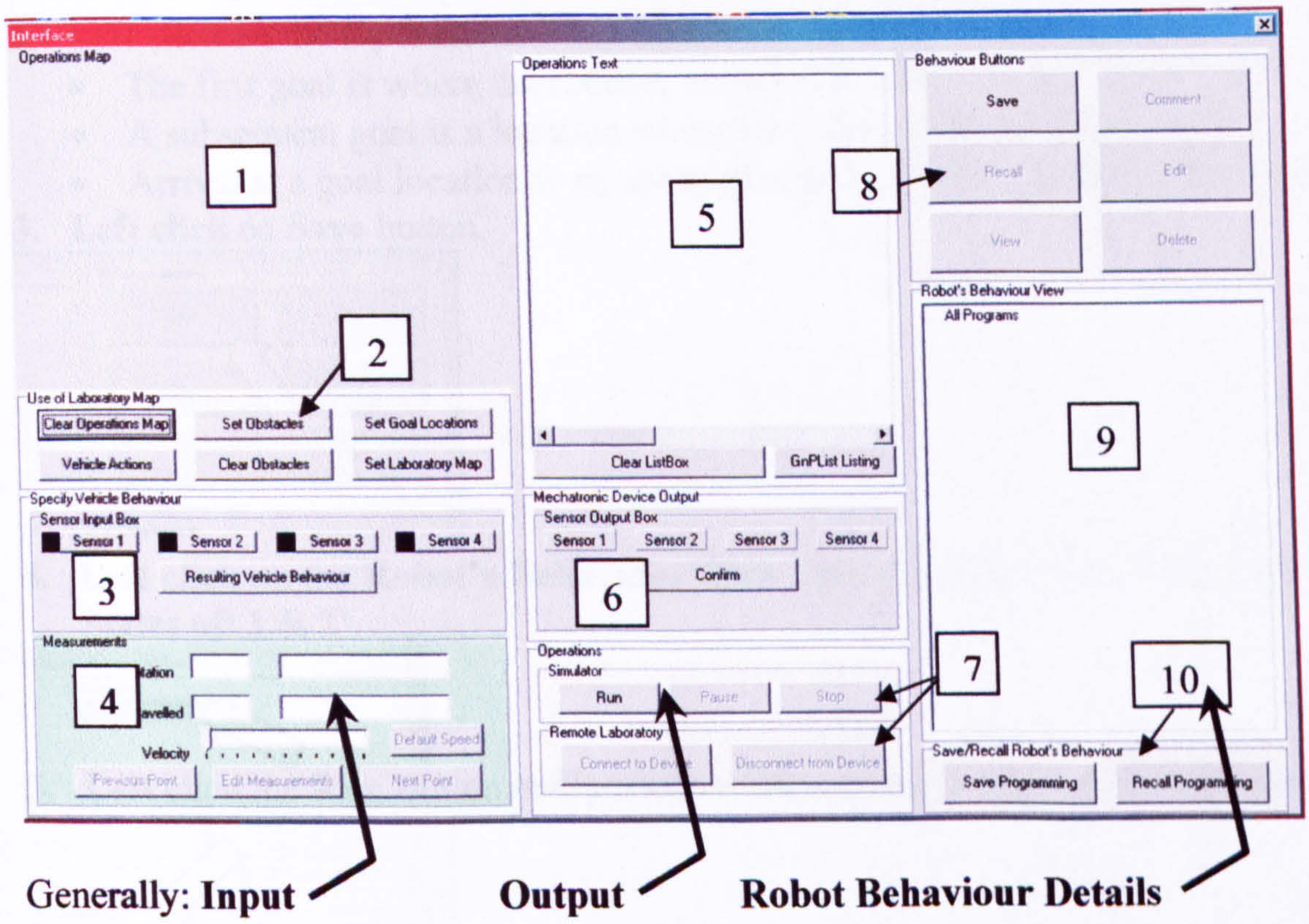
**The Sensors** have the following values:
The sensors have the following values:

| *Your Vehicle* | Left Right | | | | | | | Right Left | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| 4096 | 2048 | 1024 | 512 | 256 | 128 | 64 | 32 | 16 | 8 | 4 | 2 |

# An introduction to the Programming by Demonstration System

This Programming by Demonstration System is based on:

- **Graphically designed instructions**, allowing a designer:
  - To design the robot's program in a visual format.
  - To check the robot's program by inspection.




This systems interface has:

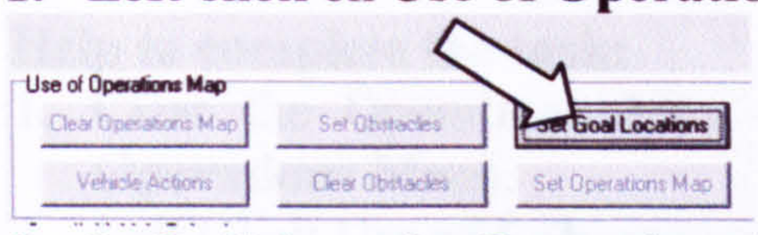
1. **Operations Map:** This is where the robot is graphically programmed, and where the simulation of the robotic programming is observed.
2. **Use of Operations Map:** This set of Buttons affect the operation of the Operations Map.
3. **Specify Vehicle Behaviour:** This allows design of robot behaviours related to sensor activation values.
4. **Measurements:** As the program is designed the details of program length, and direction of travel, angle and direction of rotation is given.
5. **Operations Text:** This is where the interface provides information about what the interface is doing.
6. **Mechatronic Device Output:** This is the robot's sensors values.
7. **Operations:** The robots program is tested, by using simulation, or connection to a laboratory.
8. **Control Buttons:** These buttons manipulate the elements of the robot's intended designed behaviour.
9. **Robot's Behaviour View:** This displays the elements of the robot's programmed behaviour.
10. **Save/Recall Behaviours:** These two buttons save the Robots behaviours to a file, and recall a robot's behaviours from a previously saved file.

# Task 1: Initial Goal Setting

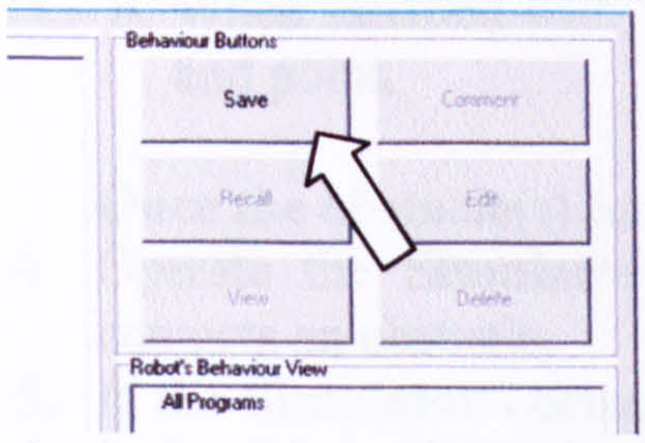
Set some initial goals for the interface to simulate a robot's behaviour.

**Help to complete the task:**

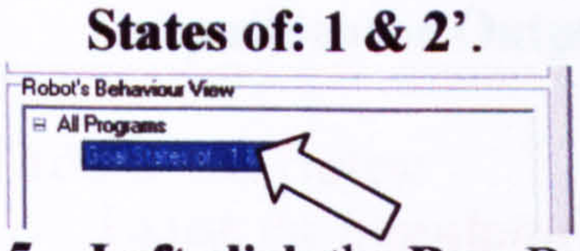
1. **Left click** on **Use of Operations Map|Set Goal Locations** button.



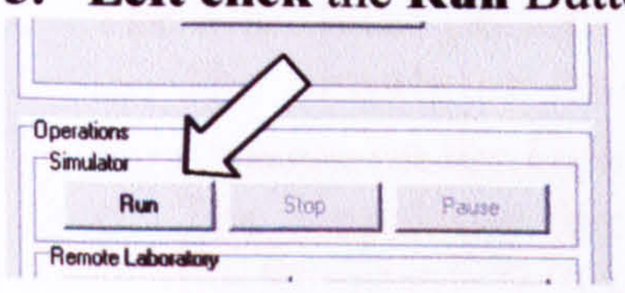

2. **Left click** on the **Operations Map** area to locate the two 'Goals'.
   - The first goal is where the robot is intended to start.
   - A subsequent goal is a location where the robot is intended to go to.
   - Arrival at a goal location is an accomplishment of the task set for the robot.
3. **Left click** on **Save** button.




4. **Left click** on the **Robot's Behaviour View** area, locating and highlighting '**Goal States of: 1 & 2**'.




5. **Left click** the **Run** Button in **Operations|Simulator** to operate the **Simulator**.




**Additional Notes:**

- The **Simulator** will begin by placing on the **Operations Map** the First Goal Location, and subsequently performs a path following cycle
  - The **Simulator** determines if there is another location to 'go to', then places this location on the **Operations Map**.
  - The **Simulator** then 'follows' the path which is displayed as a black line.

1. **To pause and restart a** simulation: press the **Pause** button in **Operations|Simulator**.
2. **To stop a simulation**: press the **Stop** button in **Operations|Simulator**.

# Task 2: Creating Obstacles

**The Task's Intentions:**
Create a simple obstacle which the simulator is expected to interact with, and a robot may encounter.

**Help to complete the task:**

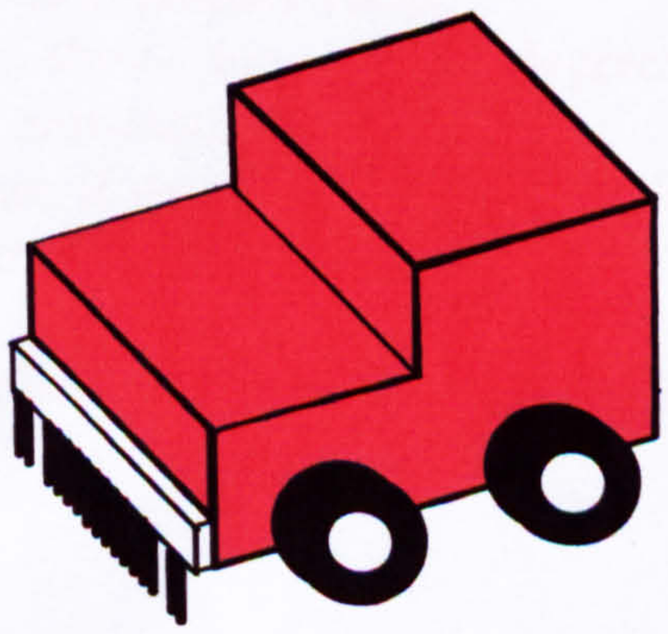
1. Clear the Operations Map (**Left Click** the **Use of Laboratory Map|Clear Operations Map**).
2. Left Click **Use of Laboratory Map|Set Operations Map**.
   The goals currently being identified in **Robot's Behaviour View** are presented on the **Operations Map**.
3. To outline an obstacle,
   a. **Single left click** to define an start point for drawing an obstacle.
   b. When satisfied with an obstacle's length and direction, **single left click** for the end point.

   **Once the obstacle(s) have been illustrated**
4. Operate the **Simulator**, and examine the robot's behaviour when the robot contacts an obstacle.
5. If the **Simulator** contacts an obstacle, the simulator determines what the sensor outputs are and Displays the sensor outputs in **Mechatronic Device Output|Sensor Output Box**.

**Additional Notes:**

- To use the **Simulator, Left Click** the **Run** Button in **Operations|Simulator**.
- The **Simulator** performs the following path following cycle.
  - The **Simulator** determines if there is a location to 'go to', then places this location on the **Operations Map**.
  - The **Simulator** then 'follows' the path which is displayed as a black line, as the Simulator follows the line, it performs this cycle
    1. Determine what the sensor outputs are and Displays the sensor outputs in the **Sensor Output Box**.
    2. Use the Sensor Data to determine the robot's obstacle avoidance behaviour.
    3. If the **Simulator** does not find an appropriate behaviour to avoid an obstacle, the simulator will stop.

- The sensors are intended to simulate the sensors on the robot shown below, they are at the front and are activated by touching an object.



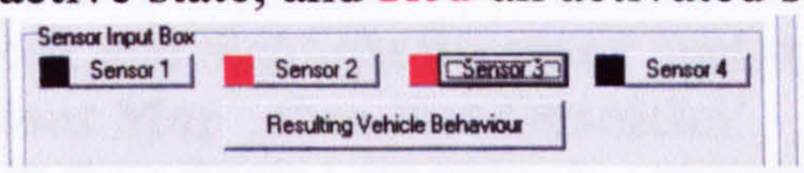
# Task 3: Obstacle Avoidance

Create a simple generic behaviour for the simulated robot to avoid the simulated obstacle, and observe the simulated behaviour results.

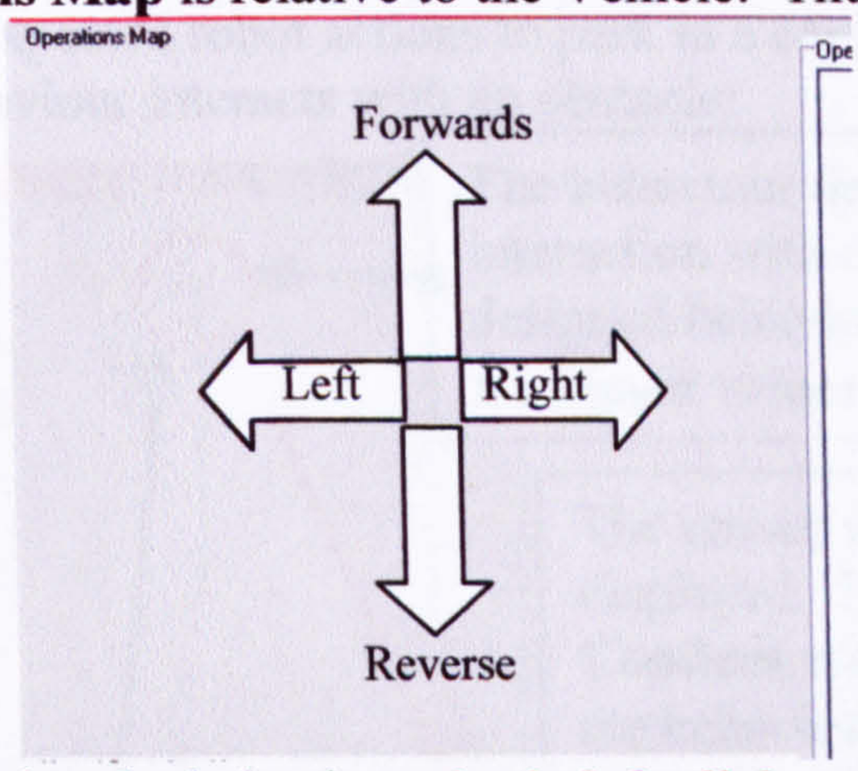
The behaviour is not situation specific, but is intended to operate in all circumstances where the sensors have been activated regardless of orientation.

**Help to complete the task:**

1. **Single left click** the **Specify Vehicle Behaviour|Sensor** buttons to specifiy which sensors activate the robot's behaviour.
   - As each **Sensor Button** is pressed, an indicator to the left switches colour, **Black** indicates an inactive state, and **Red** an activated state.




2. **Single left click** **Specify Vehicle Behaviour|Resulting Vehicle Behaviour**
3. Use the **Operations Map** to design the behaviour
   - The **Operations Map** is relative to the Vehicle. Thus:




   - To start designing the behaviour **single left click** on the **Operations Map**.
   - As each part of the Robot's behaviour is determined, **single left click**.
   - To finish illustrating the behaviour, use a **double left click**.
   - During design, the behaviour is magnified, at a scale of 5:1.
4. When the behaviour has been demonstrated: **double left click** to finish.
5. Click on '**Goal States of : 1 & 2**', within the **Robot's Behaviour View** area.
6. Click the **Save** Button in the Behaviour Buttons panel.

**Additional Notes:**

While designing the Behaviour, the interface displays the measurements detailed by the behaviour in the **Measurements** Panel.

- **Angle of Rotation**: shows the angle in degrees and automatically determines if this is clockwise or anti-clockwise rotation.
- **Distance:** determines if the vehicle is moving forwards or backwards, and shows the distance being measured during the design.

This information is duplicated to the right in the **Operations Text** box.

# Task 4: Park in a Corner

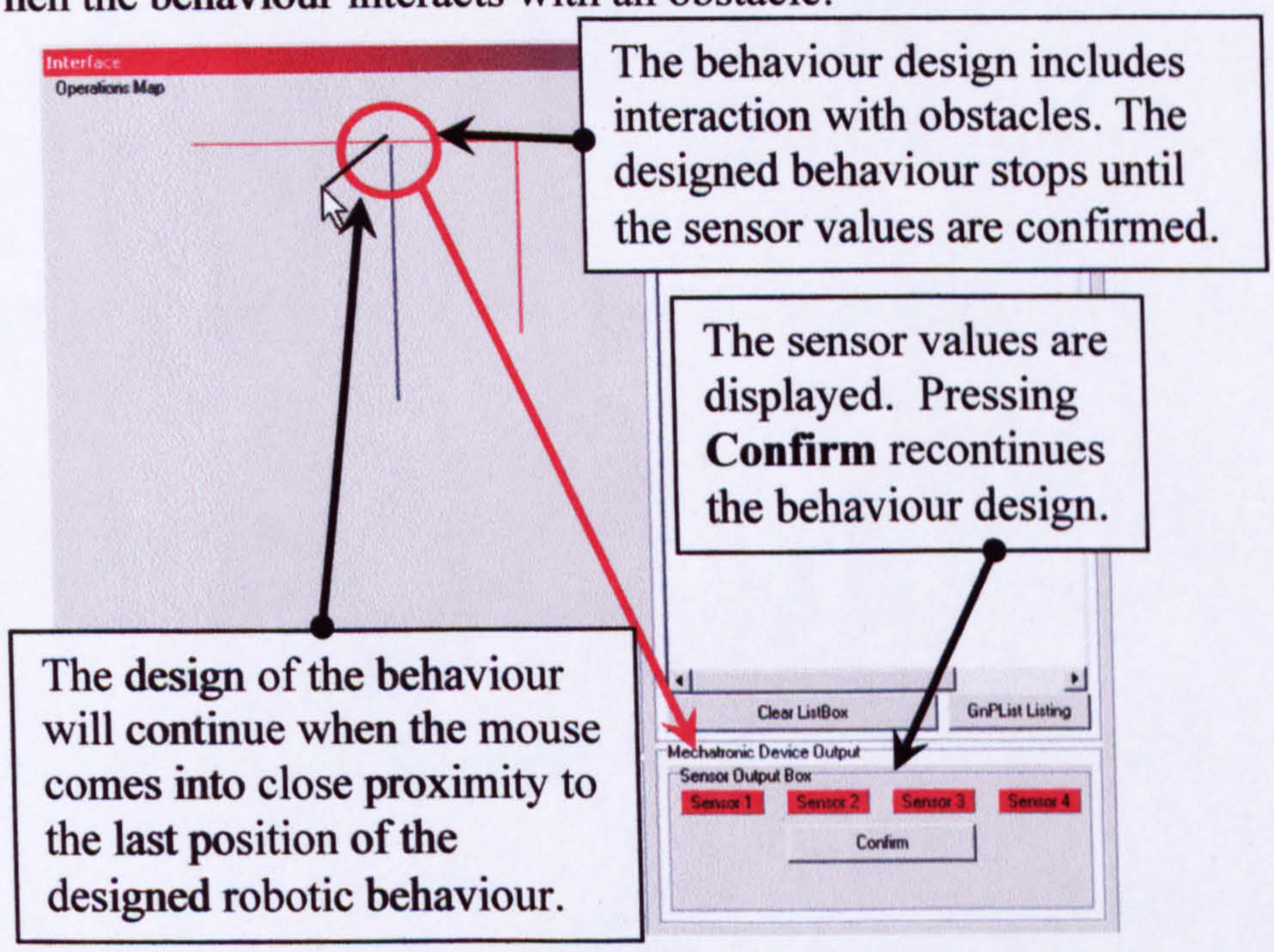
**The Task's Intentions:**

- Create a rule for the rulebase to detect the edges of the robot's work area and park in a corner parallel to one of the walls.

**Help to complete the task:**

- The Behaviour designed to park in a corner is generic

1. Highlight **Goal States 1 & 2**, in the **Robot's Behaviour View**.
2. Press the **Set Goals Locations** button. When the mouse is on the **Operations Map**, the nearest goal is displayed.
3. Select the Goal the behaviour is expected to be activated at.
4. Save the Goal state selected, and highlight the saved goal state.
5. Press the '**Clear Operations Map**', then '**Set Obstacles**'.
6. Create the obstacles which the robot's behaviour is expected to interact with.
7. Press **Use of laboratory Map|Vehicle Actions** button to design the behaviour.
8. **Single left click** on **Operations Map** initialising the **Goal Activated Behaviour**.
9. Demonstrate the expected robot actions to park in a corner noting:
   - When the behaviour interacts with an obstacle:




   - The behaviour is built up from a series of independent (and relatively co-ordinated) actions that result in an overall behaviour which achieves the objective designed.
10. **Double left click** to Finish
11. Select '**Goal States of : 2**' in **Robot Behaviour View**, and **Save** the behaviour.
12. Each Goal Activated Behaviour is saved with '**Goal States of : 2**' selected
13. Select '**Goal States of : 1 & 2**' and run the simulator.

**Additional Notes:**

**Sensor Activated Behaviours** have to be created to compensate for any mis-matches between the 'physical results' compared to the designed **Goal Activated Behaviour**.

# Appendix C

# Usability Questionnaire

Usability Questionaire.

Do you think that the Simulator system provides a sufficient degree of detail about what is happening on the screen?

Do you understand the Simulator's systems information on the screen?

How intuitive did you find the Simulator system?

Did you find it hard to remember how to do anything with the Simulator System?

How did you find the Simulator interface prompted you to particular actions?

Do you believe that any part of the Simulator interface was unnecessary?

Are you satisfied with the Interface's Names?

Use of Laboratory Map? _____
What you consider a better alternative?

_____

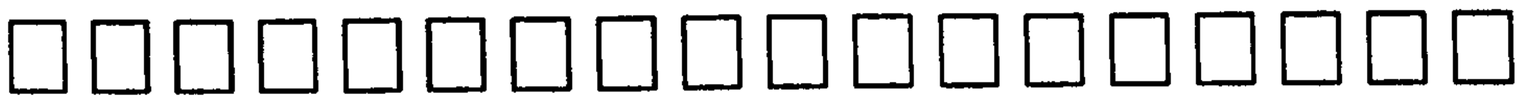
_____

_____

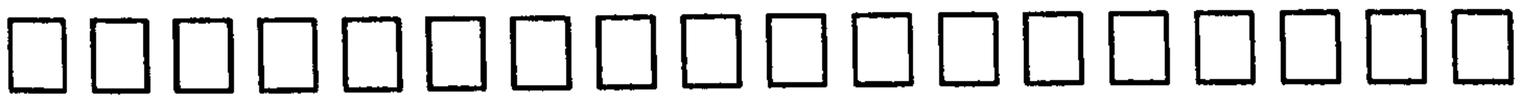
Set Goal Locations?

Set Laboratory Map?

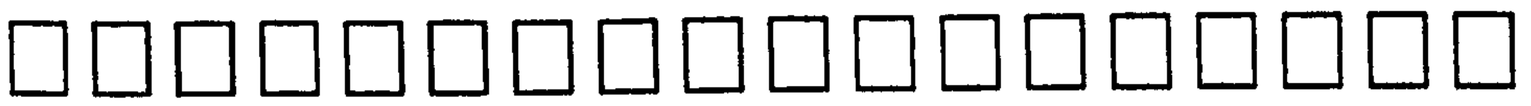
Set Obstacles?



Clear Obstacles?

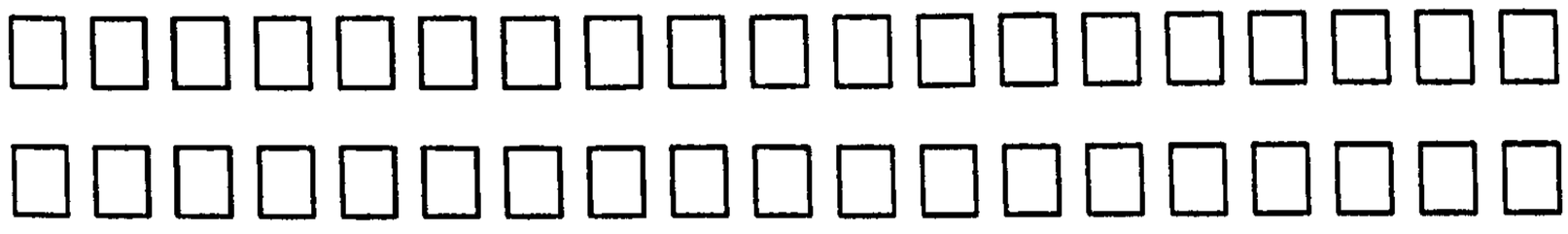


Vehicle Actions?

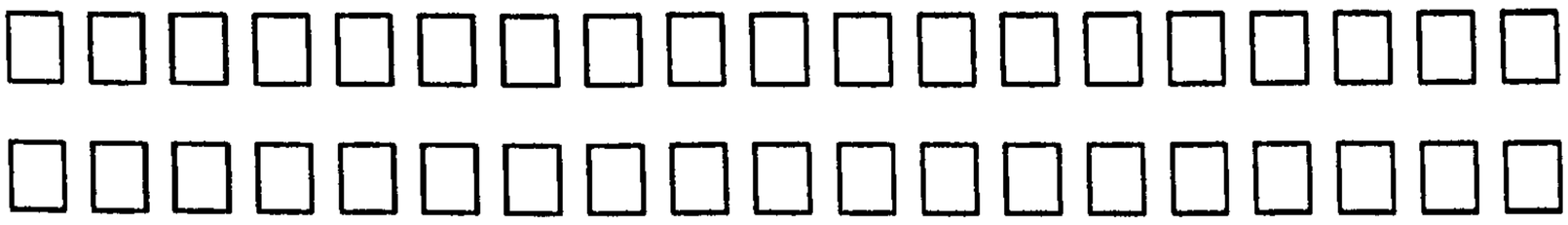


Clear Operations Map?

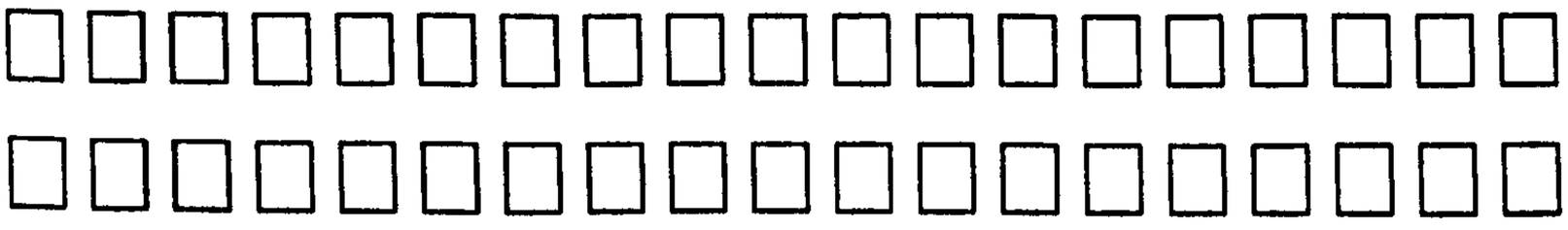
Specify Vehicle Behaviour ?



Resulting Vehicle Behaviour?



Robot's Behaviour View



What would you like as further dialog from the Simulator System's Interface?

What are your thoughts about the methods of determining the Vehicle Behaviours using the Robot's Behaviour View?

Do you believe you needed a Help System with the Simulator system's Interface?

Did you find the Simulator system's Interface easy to use?

Do you think that the Rule Based system provides a sufficient degree of detail about what is happening on the screen?

Do you understand the Rule Based systems information on the screen?

How intuitive did you find the Rule Based system?

Did you find it hard to remember how to do anything with the Rule Based system?

How did you find the Rule Based system interface prompted you to particular actions?

Do you believe that any part of the Rule Based interface was unnecessary?

What would you like as further dialog from the Rule Based system's Interface?

What are your thoughts about the methods of determining the Vehicle Behaviours using the Rule Based system?

Did you find the Interface easy to use?

Which System did you prefer and why?

# Appendix D

# Participant Timings

# Participant Timings

Table D.1 The test of the PbD system by the PbD first sample group

| PbD - RBS | Averagess | | Participants | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Group | Sample | 18 | 17 | 16 | 15 | 14 | 10 | 4 | 3 | 2 | 1 |
| PbD 1 | 03:19 | 02:51 | 01:59 | 03:27 | 12:12 | 01:46 | 02:20 | 01:26 | 02:49 | 02:31 | 03:00 | 01:42 |
| PbD 2 | 02:24 | 02:11 | 04:28 | 02:48 | 02:07 | 01:51 | 01:40 | 01:47 | 02:16 | 02:22 | 02:10 | 02:27 |
| PbD 3 | 09:59 | 08:18 | 09:26 | 16:03 | 05:43 | 04:49 | 06:05 | 11:32 | 14:31 | 15:04 | 06:08 | 10:28 |
| PbD 4 | 11:50 | 09:43 | 10:05 | 13:42 | 13:49 | 05:27 | 04:28 | 08:06 | 10:42 | 24:23 | 15:16 | 12:19 |
| | 27:31 | 23:02 | | | | | | | | | | |

Table D.2 The test of the RBS system by the PbD-first sample group

| PbD - RBS | Averages | | Participants | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Group | Sample | 18 | 17 | 16 | 15 | 14 | 10 | 4 | 3 | 2 | 1 |
| RBS 1 | 02:10 | 02:30 | 02:08 | 03:01 | 03:26 | 01:36 | 01:33 | 00:39 | 01:30 | 01:09 | 03:15 | 03:22 |
| RBS 2 | 05:35 | 06:52 | 06:48 | 04:54 | 07:26 | 04:10 | 08:25 | 04:26 | 05:01 | 04:16 | 05:33 | 04:46 |
| RBS 3 | 06:59 | 07:33 | 05:14 | 10:25 | 13:21 | 04:55 | 09:08 | 05:40 | 03:28 | 05:43 | 06:59 | 04:59 |
| RBS 4 | 08:01 | 08:36 | 08:48 | 10:37 | 04:41 | 04:37 | 08:08 | 00:00 | 03:50 | 09:04 | 20:40 | 01:42 |
| RBS 5 | 10:48 | 12:49 | 13:14 | 22:42 | 10:55 | 07:16 | 06:37 | 00:00 | 09:33 | 05:18 | 00:00 | 10:53 |
| RBS 6 | 15:04 | 12:31 | 14:13 | 19:30 | 00:00 | 06:16 | 00:00 | 00:00 | 00:00 | 13:41 | 00:00 | 21:40 |
| | 48:37 | 50:51 | | | | | | | | | | |

Table D.3 The test of the PbD system by the RBS-first sample group

| RBS - PbD | Averages | | Participants | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Group | Sample | 20 | 19 | 13 | 12 | 11 | 9 | 8 | 7 | 6 | 5 |
| PbD 1 | 02:05 | 02:51 | 02:06 | 02:40 | 02:09 | 02:46 | 02:58 | 00:00 | 03:02 | 01:25 | 01:37 | 02:11 |
| PbD 2 | 01:44 | 02:11 | 04:18 | 01:13 | 01:17 | 01:57 | 03:22 | 00:00 | 02:29 | 00:41 | 01:06 | 01:02 |
| PbD 3 | 05:47 | 08:18 | 06:22 | 07:12 | 04:49 | 05:07 | 10:25 | 00:00 | 11:24 | 02:51 | 06:04 | 03:33 |
| PbD 4 | 06:38 | 09:43 | 05:43 | 06:40 | 05:45 | 06:01 | 17:47 | 00:00 | 10:07 | 05:29 | 05:07 | 03:38 |
| | 16:14 | 23:02 | | | | | | | | | | |

Table D.4 The test of the RBS system by the RBS-first sample group

| RBS - PbD | Averages | | Participants | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Group | Sample | 20 | 19 | 13 | 12 | 11 | 9 | 8 | 7 | 6 | 5 |
| Rule 1 | 02:50 | 02:30 | 02:29 | 05:05 | 01:30 | 03:25 | 02:00 | 01:37 | 03:08 | 03:23 | 01:37 | 04:08 |
| Rule 2 | 08:09 | 06:52 | 05:51 | 18:01 | 10:26 | 07:08 | 06:51 | 07:25 | 11:40 | 02:58 | 05:06 | 06:05 |
| Rule 3 | 08:10 | 07:33 | 07:04 | 27:59 | 05:32 | 05:11 | 05:42 | 03:08 | 00:00 | 03:35 | 05:52 | 09:25 |
| Rule 4 | 09:15 | 08:36 | 07:09 | 05:24 | 10:51 | 11:50 | 20:24 | 00:00 | 00:00 | 04:29 | 05:13 | 08:37 |
| Rule 5 | 14:50 | 12:49 | 53:03 | 07:28 | 10:16 | 10:00 | 15:39 | 00:00 | 00:00 | 08:07 | 04:21 | 09:46 |
| Rule 6 | 10:24 | 12:31 | 23:08 | 00:00 | 12:49 | 10:48 | 00:00 | 00:00 | 00:00 | 07:58 | 06:54 | 00:49 |
| | 53:38 | 50:51 | | | | | | | | | | |